



Ron Patton

# Software Testing

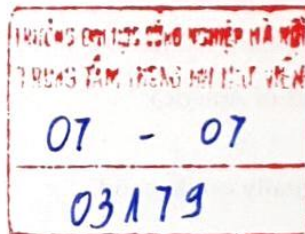
Second Edition

**SAMS**



Ron Patton

# Software Testing



**SAMS**

800 East 96th Street, Indianapolis, Indiana 46240

# Contents at a Glance

	Introduction .....	1
<b>Part I</b>	<b>The Big Picture</b>	
1	Software Testing Background .....	9
2	The Software Development Process .....	23
3	The Realities of Software Testing .....	37
<b>Part II</b>	<b>Testing Fundamentals</b>	
4	Examining the Specification .....	53
5	Testing the Software with Blinders On .....	63
6	Examining the Code .....	91
7	Testing the Software with X-Ray Glasses .....	105
<b>Part III</b>	<b>Applying Your Testing Skills</b>	
8	Configuration Testing .....	125
9	Compatibility Testing .....	141
10	Foreign-Language Testing .....	153
11	Usability Testing .....	169
12	Testing the Documentation .....	183
13	Testing for Software Security .....	193
14	Website Testing .....	211
<b>Part IV</b>	<b>Supplementing Your Testing</b>	
15	Automated Testing and Test Tools .....	231
16	Bug Bashes and Beta Testing .....	253
<b>Part V</b>	<b>Working with Test Documentation</b>	
17	Planning Your Test Effort .....	263
18	Writing and Tracking Test Cases .....	277
19	Reporting What You Find .....	291
20	Measuring Your Success .....	313



**Part VI The Future**

21	Software Quality Assurance	329
22	Your Career as a Software Tester	343

**Appendix**

A	Answers to Quiz Questions	355
---	---------------------------	-----

	Index	377
--	-------	-----

# Table of Contents

<b>Introduction</b>	<b>1</b>
About the Second Edition	1
Who Should Use This Book?	2
What This Book Will Do for You	2
Software Necessary to Use This Book	3
How This Book Is Organized	3
Part I: The Big Picture	3
Part II: Testing Fundamentals	4
Part III: Applying Your Testing Skills	4
Part IV: Supplementing Your Testing	5
Part V: Working with Test Documentation	5
Part VI: The Future	5
Appendix	6
Conventions Used in This Book	6
 <b>Part I The Big Picture</b>	
<b>1 Software Testing Background</b>	<b>9</b>
Infamous Software Error Case Studies	9
Disney's <i>Lion King</i> , 1994–1995	10
Intel Pentium Floating-Point Division Bug, 1994	10
NASA Mars Polar Lander, 1999	11
Patriot Missile Defense System, 1991	12
The Y2K (Year 2000) Bug, circa 1974	12
Dangerous Viewing Ahead, 2004	13
What Is a Bug?	13
Terms for Software Failures	13
Software Bug: A Formal Definition	14
Why Do Bugs Occur?	16
The Cost of Bugs	18
What Exactly Does a Software Tester Do?	19
What Makes a Good Software Tester?	20
Summary	21
Quiz	22



<b>2</b>	<b>The Software Development Process</b>	<b>23</b>
	Product Components	23
	What Effort Goes Into a Software Product?	24
	What Parts Make Up a Software Product?	28
	Software Project Staff	29
	Software Development Lifecycle Models	30
	Big-Bang Model	31
	Code-and-Fix Model	32
	Waterfall Model	33
	Spiral Model	34
	Summary	36
	Quiz	36
<b>3</b>	<b>The Realities of Software Testing</b>	<b>37</b>
	Testing Axioms	38
	It's Impossible to Test a Program Completely	38
	Software Testing Is a Risk-Based Exercise	39
	Testing Can't Show That Bugs Don't Exist	40
	The More Bugs You Find, the More Bugs There Are	41
	The Pesticide Paradox	41
	Not All the Bugs You Find Will Be Fixed	42
	When a Bug's a Bug Is Difficult to Say	43
	Product Specifications Are Never Final	44
	Software Testers Aren't the Most Popular Members of a Project Team	45
	Software Testing Is a Disciplined Technical Profession	45
	Software Testing Terms and Definitions	46
	Precision and Accuracy	46
	Verification and Validation	47
	Quality and Reliability	48
	Testing and Quality Assurance (QA)	48
	Summary	49
	Quiz	49
<b>Part II</b>	<b>Testing Fundamentals</b>	
<b>4</b>	<b>Examining the Specification</b>	<b>53</b>
	Getting Started	53
	Black-Box and White-Box Testing	55
	Static and Dynamic Testing	56
	Static Black-Box Testing: Testing the Specification	56

Performing a High-Level Review of the Specification	57
Pretend to Be the Customer	57
Research Existing Standards and Guidelines	58
Review and Test Similar Software	59
Low-Level Specification Test Techniques	60
Specification Attributes Checklist	60
Specification Terminology Checklist	61
Summary	61
Quiz	62
<b>5 Testing the Software with Blinders On</b>	<b>63</b>
Dynamic Black-Box Testing: Testing the Software While Blindfolded	64
Test-to-Pass and Test-to-Fail	66
Equivalence Partitioning	67
Data Testing	70
Boundary Conditions	70
Sub-Boundary Conditions	75
Default, Empty, Blank, Null, Zero, and None	77
Invalid, Wrong, Incorrect, and Garbage Data	78
State Testing	79
Testing the Software's Logic Flow	80
Testing States to Fail	84
Other Black-Box Test Techniques	87
Behave Like a Dumb User	87
Look for Bugs Where You've Already Found Them	88
Think like a Hacker	88
Follow Experience, Intuition, and Hunches	88
Summary	89
Quiz	89
<b>6 Examining the Code</b>	<b>91</b>
Static White-Box Testing: Examining the Design and Code	91
Formal Reviews	92
Peer Reviews	94
Walkthroughs	95
Inspections	95
Coding Standards and Guidelines	96
Examples of Programming Standards and Guidelines	96
Obtaining Standards	98



	Generic Code Review Checklist	99
	Data Reference Errors	99
	Data Declaration Errors	100
	Computation Errors	101
	Comparison Errors	101
	Control Flow Errors	102
	Subroutine Parameter Errors	102
	Input/Output Errors	102
	Other Checks	103
	Summary	103
	Quiz	104
<b>7</b>	<b>Testing the Software with X-Ray Glasses</b>	<b>105</b>
	Dynamic White-Box Testing	106
	Dynamic White-Box Testing Versus Debugging	107
	Testing the Pieces	108
	Unit and Integration Testing	109
	An Example of Module Testing	111
	Data Coverage	113
	Data Flow	114
	Sub-Boundaries	115
	Formulas and Equations	115
	Error Forcing	116
	Code Coverage	117
	Program Statement and Line Coverage	118
	Branch Coverage	119
	Condition Coverage	120
	Summary	121
	Quiz	122
<b>Part III</b>	<b>Applying Your Testing Skills</b>	
<b>8</b>	<b>Configuration Testing</b>	<b>125</b>
	An Overview of Configuration Testing	126
	Isolating Configuration Bugs	129
	Sizing Up the Job	131
	Approaching the Task	132
	Decide the Types of Hardware You'll Need	133
	Decide What Hardware Brands, Models, and Device Drivers Are Available	133



Decide Which Hardware Features, Modes, and Options Are Possible	134
Pare Down the Identified Hardware Configurations to a Manageable Set	134
Identify Your Software's Unique Features That Work with the Hardware Configurations	135
Design the Test Cases to Run on Each Configuration	136
Execute the Tests on Each Configuration	137
Rerun the Tests Until the Results Satisfy Your Team	137
Obtaining the Hardware	137
Identifying Hardware Standards	139
Configuration Testing Other Hardware	139
Summary	140
Quiz	140
<b>9 Compatibility Testing</b>	<b>141</b>
Compatibility Testing Overview	142
Platform and Application Versions	143
Backward and Forward Compatibility	144
The Impact of Testing Multiple Versions	144
Standards and Guidelines	146
High-Level Standards and Guidelines	147
Low-Level Standards and Guidelines	148
Data Sharing Compatibility	148
Summary	150
Quiz	151
<b>10 Foreign-Language Testing</b>	<b>153</b>
Making the Words and Pictures Make Sense	154
Translation Issues	154
Text Expansion	155
ASCII, DBCS, and Unicode	155
Hot Keys and Shortcuts	156
Extended Characters	157
Computations on Characters	158
Reading Left to Right and Right to Left	158
Text in Graphics	159
Keep the Text out of the Code	159

	Localization Issues	160
	Content	160
	Data Formats	162
	Configuration and Compatibility Issues	163
	Foreign Platform Configurations	163
	Data Compatibility	165
	How Much Should You Test?	166
	Summary	167
	Quiz	168
<b>11</b>	<b>Usability Testing</b>	<b>169</b>
	User Interface Testing	170
	What Makes a Good UI?	170
	Follows Standards and Guidelines	171
	Intuitive	173
	Consistent	173
	Flexible	175
	Comfortable	176
	Correct	176
	Useful	178
	Testing for the Disabled: Accessibility Testing	178
	Legal Requirements	179
	Accessibility Features in Software	180
	Summary	182
	Quiz	182
<b>12</b>	<b>Testing the Documentation</b>	<b>183</b>
	Types of Software Documentation	183
	The Importance of Documentation Testing	187
	What to Look for When Reviewing Documentation	188
	The Realities of Documentation Testing	189
	Summary	190
	Quiz	190
<b>13</b>	<b>Testing for Software Security</b>	<b>193</b>
	WarGames—the Movie	194
	Understanding the Motivation	195
	Threat Modeling	197
	Is Software Security a Feature? Is Security Vulnerability a Bug?	200
	Understanding the Buffer Overrun	201



	Using Safe String Functions .....	203
	Computer Forensics .....	205
	Summary .....	208
	Quiz .....	209
<b>14</b>	<b>Website Testing .....</b>	<b>211</b>
	Web Page Fundamentals .....	212
	Black-Box Testing .....	213
	Text .....	215
	Hyperlinks .....	216
	Graphics .....	217
	Forms .....	217
	Objects and Other Simple Miscellaneous Functionality .....	218
	Gray-Box Testing .....	218
	White-Box Testing .....	220
	Configuration and Compatibility Testing .....	222
	Usability Testing .....	224
	Introducing Automation .....	226
	Summary .....	227
	Quiz .....	227
<b>Part IV</b>	<b>Supplementing Your Testing .....</b>	
<b>15</b>	<b>Automated Testing and Test Tools .....</b>	<b>231</b>
	The Benefits of Automation and Tools .....	231
	Test Tools .....	233
	Viewers and Monitors .....	234
	Drivers .....	235
	Stubs .....	236
	Stress and Load Tools .....	237
	Interference Injectors and Noise Generators .....	238
	Analysis Tools .....	239
	Software Test Automation .....	239
	Macro Recording and Playback .....	240
	Programmed Macros .....	242
	Fully Programmable Automated Testing Tools .....	243
	Random Testing: Monkeys and Gorillas .....	245
	Dumb Monkeys .....	246
	Semi-Smart Monkeys .....	248
	Smart Monkeys .....	248

	Realities of Using Test Tools and Automation	250
	Summary	251
	Quiz	252
<b>16</b>	<b>Bug Bashes and Beta Testing</b>	<b>253</b>
	Having Other People Test Your Software	253
	Test Sharing	255
	Beta Testing	256
	Outsourcing Your Testing	258
	Summary	259
	Quiz	259
<b>Part V</b>	<b>Working with Test Documentation</b>	
<b>17</b>	<b>Planning Your Test Effort</b>	<b>263</b>
	The Goal of Test Planning	264
	Test Planning Topics	265
	High-Level Expectations	265
	People, Places, and Things	266
	Definitions	267
	Inter-Group Responsibilities	268
	What Will and Won't Be Tested	270
	Test Phases	270
	Test Strategy	271
	Resource Requirements	271
	Tester Assignments	272
	Test Schedule	272
	Test Cases	274
	Bug Reporting	274
	Metrics and Statistics	274
	Risks and Issues	275
	Summary	275
	Quiz	275
<b>18</b>	<b>Writing and Tracking Test Cases</b>	<b>277</b>
	The Goals of Test Case Planning	277
	Test Case Planning Overview	279
	Test Design	281
	Test Cases	283
	Test Procedures	285



Test Case Organization and Tracking .....	287
Summary .....	289
Quiz .....	290
<b>19 Reporting What You Find .....</b>	<b>291</b>
Getting Your Bugs Fixed .....	292
Isolating and Reproducing Bugs .....	296
Not All Bugs Are Created Equal .....	298
A Bug's Life Cycle .....	300
Bug-Tracking Systems .....	303
The Standard: The Test Incident Report .....	303
Manual Bug Reporting and Tracking .....	304
Automated Bug Reporting and Tracking .....	306
Summary .....	310
Quiz .....	311
<b>20 Measuring Your Success .....</b>	<b>313</b>
Using the Information in the Bug Tracking Database .....	314
Metrics That You'll Use in Your Daily Testing .....	315
Common Project-Level Metrics .....	320
Summary .....	325
Quiz .....	326
<b>Part VI The Future .....</b>	
<b>21 Software Quality Assurance .....</b>	<b>329</b>
Quality Is Free .....	330
Testing and Quality Assurance in the Workplace .....	331
Software Testing .....	331
Quality Assurance .....	333
Other Names for Software Testing Groups .....	334
Test Management and Organizational Structures .....	335
Capability Maturity Model (CMM) .....	337
ISO 9000 .....	339
Summary .....	342
Quiz .....	342
<b>22 Your Career as a Software Tester .....</b>	<b>343</b>
Your Job as a Software Tester .....	344
Finding a Software Testing Position .....	345



# Introduction

It seems as though each day there's yet another news story about a computer software problem or security breach: a bank reporting incorrect account balances, a Mars lander lost in space, a grocery store scanner charging too much for bananas, or a hacker gaining access to millions of credit card numbers.

Why does this happen? Can't computer programmers figure out ways to make software just plain work? Unfortunately, no. As software gets more complex, gains more features, and is more interconnected, it becomes more and more difficult—actually, mathematically impossible—to create a glitch-free program. Despite how competent the programmers are and how much care is taken, there will always be software problems.

This is where software testing comes in. We've all found those little *Inspector 12* tags in the pockets of our new clothes. Well, software has *Inspector 12s*, too. Most large software companies are so committed to quality they have one or more testers for each programmer. These jobs span the software spectrum from computer games to factory automation to business applications.

This book, *Software Testing*, will introduce you to the basics of software testing, teaching you not just the fundamental technical skills but also the supporting skills necessary to become a successful software tester. You will learn how to immediately find problems in any computer program, how to plan an effective test approach, how to clearly report your findings, and how to tell when your software is ready for release.

## About the Second Edition

When I wrote the first edition of *Software Testing*, software security issues were just beginning to make the headlines. Hackers and security problems had always been a problem, but with the interconnectivity explosion that was about to occur, few in the industry could predict the impact that security bugs would have on developers and users of computer software.

In this second edition I've revisited every chapter to emphasize software security issues and point out how the basic testing techniques covered throughout the book can be used to prevent, find, and fix them. I've also added a chapter that specifically addresses how to test for software security bugs.



If you're a reader of the first edition, you know that no matter what you do, your software will still be released with bugs. As you'll learn in the second edition, this axiom still holds true—even for security problems. However, by applying the lessons taught in this book you'll go a long way towards assuring that the most important bugs don't slip through and that your team will create the highest quality and most secure software possible.

## Who Should Use This Book?

This book is written for three different groups of people:

- Students or computer hobbyists interested in software testing as a full-time job, internship, or co-op. Read this book before your interview or before your first day on the job to really impress your new boss.
- Career changers wanting to move from their field of expertise into the software industry. There are lots of opportunities for non-software experts to apply their knowledge to software testing. For example, a flight instructor could test a flight simulator game, an accountant could test tax preparation software, or a teacher could test a new child education program.
- Programmers, software project managers, and other people who make up a software development team who want to improve their knowledge and understanding of what software testing is all about.

## What This Book Will Do for You

In this book you will learn something about nearly every aspect of software testing:

- How software testing fits into the software development process
- Basic and advanced software testing techniques
- Applying testing skills to common testing tasks
- Improving test efficiency with automation
- Planning and documenting your test effort
- Effectively reporting the problems you find
- Measuring your test effort and your product's progress
- Knowing the difference between testing and quality assurance
- Finding a job as a software tester



## Software Necessary to Use This Book

The methods presented in this book are generic and can be applied to testing any type of computer software. But, to make the examples familiar and usable by most people, they are based on simple programs such as Calculator, Notepad, and WordPad included with Windows XP and Windows NT/2000.

Even if you're using a Mac or a PC running Linux or another operating system, you will likely have similar programs available on your computer that you can easily adapt to the text. Be creative! Creativity is one trait of a good software tester.

---

### NOTE

The examples used throughout this book of various applications, software bugs, and software test tools are in no way intended as an endorsement or a disparagement of the software. They're simply used to demonstrate the concepts of software testing.

---

## How This Book Is Organized

This book is designed to lead you through the essential knowledge and skills necessary to become a good software tester. Software testing is not about banging on the keyboard hoping you'll eventually crash the computer. A great deal of science and engineering is behind it, lots of discipline and planning, and there can be lots of fun, too—as you'll soon see.

### Part I: The Big Picture

The chapters in Part I lay the foundation for this book by showing you how software products are developed and how software testing fits into the overall development process. You'll see the importance of software testing and gain an appreciation for the magnitude of the job.

- Chapter 1, "Software Testing Background," helps you understand exactly what a software bug is, how serious they can be, and why they occur. You'll learn what your ultimate goal is as a software tester and what traits will help make you a good one.
- Chapter 2, "The Software Development Process," gives you an overview of how a software product is created in the corporate world. You'll learn what components typically go into software, what types of people contribute to it, and the different process models that can be used.
- Chapter 3, "The Realities of Software Testing," brings a reality check to how software is developed. You'll see why no matter how hard you try, software can never be perfect. You'll also learn a few fundamental terms and concepts used throughout the rest of this book.



## Part II: Testing Fundamentals

The chapters in Part II teach you the fundamental approaches to software testing. The work of testing software is divided into four basic areas, and you will see the techniques used for each one:

- Chapter 4, "Examining the Specification," teaches you how to find bugs by carefully inspecting the documentation that describes what the software is intended to do.
- Chapter 5, "Testing the Software with Blinders On," teaches you the techniques to use for testing software without having access to the code or even knowing how to program. This is the most common type of testing.
- Chapter 6, "Examining the Code," shows you how to perform detailed analysis of the program's source code to find bugs. You'll learn that you don't have to be an expert programmer to use these techniques.
- Chapter 7, "Testing the Software with X-Ray Glasses," teaches you how you can improve your testing by leveraging information you gain by reviewing the code or being able to see it execute while you run your tests.

## Part III: Applying Your Testing Skills

The chapters in Part III take the techniques that you learned in Part II and apply them to some real-world scenarios that you'll encounter as a software tester:

- Chapter 8, "Configuration Testing," teaches you how to organize and perform software testing on different hardware configurations and platforms.
- Chapter 9, "Compatibility Testing," teaches you how to test for issues with different software applications and operating systems interacting with each other.
- Chapter 10, "Foreign-Language Testing," shows you that a whole world of software is out there and that it's important to test for the special problems that can arise when software is translated into other languages.
- Chapter 11, "Usability Testing," teaches you how to apply your testing skills when checking a software application's user interface and how to assure that your software is accessible to the disabled.
- Chapter 12, "Testing the Documentation," explains how to examine the software's documentation such as help files, user manuals, even the marketing material, for bugs.
- Chapter 13, "Testing for Software Security," shows you how to find bugs that allow hackers to gain access to (supposedly) secure computer systems and data.



- Chapter 14, “Website Testing,” takes everything you’ve learned so far and applies it to a present-day situation. You’ll see how something as simple as testing a website can encompass nearly all aspects of software testing.

## **Part IV: Supplementing Your Testing**

The chapters in Part IV show you how to improve your test coverage and capability by leveraging both technology and people to perform your testing more efficiently and effectively:

- Chapter 15, “Automated Testing and Test Tools,” explains how you can use computers and software to test other software. You’ll learn several different methods for automating your tests and using tools. You’ll also learn why using technology isn’t foolproof.
- Chapter 16, “Bug Bashes and Beta Testing,” shows you how to use other people to see the software differently and to find bugs that you completely overlooked.

## **Part V: Working with Test Documentation**

The chapters in Part V cover how software testing is documented so that its plans, bugs, and results can be seen and understood by everyone on the project team:

- Chapter 17, “Planning Your Test Effort,” shows you what goes into creating a test plan for your project. As a new software tester, you likely won’t write a test plan from scratch, but it’s important to know what’s in one and why.
- Chapter 18, “Writing and Tracking Test Cases,” teaches you how to properly document the test cases you develop so that you and other testers can use them.
- Chapter 19, “Reporting What You Find,” teaches you how to tell the world when you find a bug, how to isolate the steps necessary to make it recur, and how to describe it so that others will understand and want to fix it.
- Chapter 20, “Measuring Your Success,” describes various types of data, charts, and graphs used to gauge both your progress and success at testing and your software project’s steps toward release.

## **Part VI: The Future**

The chapters in Part VI explain where the future lies in software testing and set the stage for your career:



- Chapter 21, "Software Quality Assurance," teaches you the big difference between software testing and quality assurance. You'll learn about different software industry goals such as ISO 9000 and the Capabilities Maturity Model and what it takes to achieve them.
- Chapter 22, "Your Career as a Software Tester," gives you that kick in the behind to go out and be a software tester. You'll learn what types of jobs are available and where to look for them. You'll also find many pointers to more information.

## Appendix

Each chapter in this book ends with a short quiz where you can try out the testing concepts that you learn. The answers appear in Appendix A, "Answers to Quiz Questions."

## Conventions Used in This Book

This book uses several common conventions to help teach software testing topics. Here's a summary of those typographical conventions:

- New terms are emphasized in *italics* the first time they are used.
- Commands and computer output appear in a special monospaced font.
- Words you type appear in a **monospaced bold** font.

In addition to typographical conventions, the following special elements are included to set off different types of information to make them easily recognizable.

### NOTE

---

Special notes augment the material you read in each chapter. These notes clarify concepts and procedures.

---

### TIP

---

You'll find various tips that offer shortcuts and solutions to common problems.

---

### REMINDER

---

Reminders refer to concepts discussed in previous chapters to help refresh your memory and reinforce important concepts.

---