



# Effective Software Testing

50 Specific  
Ways to  
Improve  
Your  
Testing

Elfriede Dustin

# Contents

---

Preface	xi
Acknowledgments	xv
<b>1. Requirements Phase</b>	<b>1</b>
<i>Item 1: Involve Testers from the Beginning</i>	3
<i>Item 2: Verify the Requirements</i>	5
<i>Item 3: Design Test Procedures As Soon As Requirements         Are Available</i>	11
<i>Item 4: Ensure That Requirement Changes Are         Communicated</i>	15
<i>Item 5: Beware of Developing and Testing Based on an         Existing System</i>	19
<b>2. Test Planning</b>	<b>23</b>
<i>Item 6: Understand the Task At Hand and the Related         Testing Goal</i>	25
<i>Item 7: Consider the Risks</i>	31
<i>Item 8: Base Testing Efforts on a Prioritized Feature         Schedule</i>	39
<i>Item 9: Keep Software Issues in Mind</i>	41
<i>Item 10: Acquire Effective Test Data</i>	43
<i>Item 11: Plan the Test Environment</i>	47
<i>Item 12: Estimate Test Preparation and Execution Time</i>	51

<b>3. The Testing Team</b>	63
Item 13: Define Roles and Responsibilities	65
Item 14: Require a Mixture of Testing Skills, Subject-Matter Expertise, and Experience	75
Item 15: Evaluate the Tester's Effectiveness	79
<b>4. The System Architecture</b>	91
Item 16: Understand the Architecture and Underlying Components	93
Item 17: Verify That the System Supports Testability	97
Item 18: Use Logging to Increase System Testability	99
Item 19: Verify That the System Supports Debug and Release Execution Modes	103
<b>5. Test Design and Documentation</b>	107
Item 20: Divide and Conquer	109
Item 21: Mandate the Use of a Test-Procedure Template and Other Test-Design Standards	115
Item 22: Derive Effective Test Cases from Requirements	121
Item 23: Treat Test Procedures As "Living" Documents	125
Item 24: Utilize System Design and Prototypes	127
Item 25: Use Proven Testing Techniques when Designing Test-Case Scenarios	129
Item 26: Avoid Including Constraints and Detailed Data Elements within Test Procedures	135
Item 27: Apply Exploratory Testing	139
<b>6. Unit Testing</b>	143
Item 28: Structure the Development Approach to Support Effective Unit Testing	145
Item 29: Develop Unit Tests in Parallel or Before the Implementation	151
Item 30: Make Unit-Test Execution Part of the Build Process	155
<b>7. Automated Testing Tools</b>	159
Item 31: Know the Different Types of Testing-Support Tools	161
Item 32: Consider Building a Tool Instead of Buying One	167

<i>Item 33: Know the Impact of Automated Tools on the Testing Effort</i>	171
<i>Item 34: Focus on the Needs of Your Organization</i>	177
<i>Item 35: Test the Tools on an Application Prototype</i>	183
<b>8. Automated Testing:</b>	
<b>Selected Best Practices</b>	185
<i>Item 36: Do Not Rely Solely on Capture/Playback</i>	187
<i>Item 37: Develop a Test Harness When Necessary</i>	191
<i>Item 38: Use Proven Test-Script Development Techniques</i>	197
<i>Item 39: Automate Regression Tests When Feasible</i>	201
<i>Item 40: Implement Automated Builds and Smoke Tests</i>	207
<b>9. Nonfunctional Testing</b>	211
<i>Item 41: Do Not Make Nonfunctional Testing an Afterthought</i>	213
<i>Item 42: Conduct Performance Testing with Production-Sized Databases</i>	217
<i>Item 43: Tailor Usability Tests to the Intended Audience</i>	221
<i>Item 44: Consider All Aspects of Security, for Specific Requirements and System-Wide</i>	225
<i>Item 45: Investigate the System's Implementation To Plan for Concurrency Tests</i>	229
<i>Item 46: Set Up an Efficient Environment for Compatibility Testing</i>	235
<b>10. Managing Test Execution</b>	239
<i>Item 47: Clearly Define the Beginning and End of the Test-Execution Cycle</i>	241
<i>Item 48: Isolate the Test Environment from the Development Environment</i>	245
<i>Item 49: Implement a Defect-Tracking Life Cycle</i>	247
<i>Item 50: Track the Execution of the Testing Program</i>	255
<b>Index</b>	259