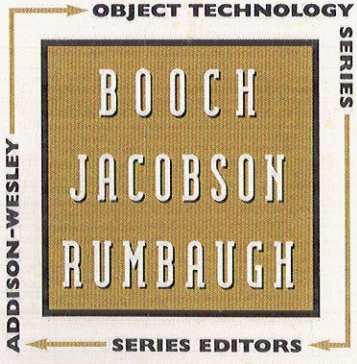
TESTING OBJECT-ORIENTED SYSTEMS

MODELS, PATTERNS, AND TOOLS

VOLUME 1

ROBERT V. BINDER

Foreword by Boris Beizer



Contents

List of Figures xxv
List of Tables xxxi
List of Procedures xxxvii
Foreword xxxix
Preface xli
Acknowledgments xlv

Part I Preliminaries 1

Chapter 1 A Small Challenge 3

Chapter 2 How to Use This Book 11

2.1 Reader Guidance 11
2.2 Conventions 14
2.2.1 Chapter Elements 14
2.2.2 Degree of Difficulty 15
2.2.3 Standards 17
2.2.4 Object-oriented Terminology 17
2.2.5 Programming Languages and Code Examples 19
2.2.6 Testing Tools 21
2.2.7 Humility Department, or Bug Reports Cheerfully Accepted 22
2.3 FAQs for Object-oriented Testing 23
2.3.1 Why Test Objects? 23
2.3.2 Test Design 25
2.3.3 Test Design for Methods and Classes 27
2.3.4 Testing for Reuse 29
2.3.5 Test Design for Subsystems and Application Systems 29
2.3.6 Integration Testing and Order of Development 30
2.3.7 *Regression Testing and Iterative, Incremental* Development 32

2.3.8 Testing with UML Models 33
2.3.9 Test Automation 36
2.4 Test Process 40

Chapter 3 Testing: A Brief Introduction 41
3.1 What Is Software Testing? 41
3.2 Definitions 44
3.3 The Limits of Testing 53
3.3.1 The Input/State Space 53
3.3.2 Execution Sequences 54
3.3.3 Fault Sensitivity and Coincidental Correctness 56
3.3.4 Absolute Limitations 57
3.4 What Can Testing Accomplish? 58
3.5 Bibliographic Notes 61

Chapter 4 With the Necessary Changes: Testing and Object-oriented Software 63
4.1 The Dismal Science of Software Testing 63
4.1.1 I'm Okay, You're Okay, Objects Are Okay 64
4.1.2 The Role of a Fault Model 65
4.1.3 Fault Models for Object-oriented Programming 67
4.2 Side Effects of the Paradigm 69
4.2.1 What Goes Wrong? 69
4.2.2 Encapsulation 71
4.2.3 Inheritance 72
Incorrect Initialization and Forgotten Methods 73
Testing Axioms 73
Inheritance Structure 74
Multiple Inheritance 75
Abstract and Generic Classes 76
4.2.4 Polymorphism 77
Dynamic Binding 80
The Yo-Yo Problem 81
4.2.5 Message Sequence and State-related Bugs 83
Message Sequence Model 83
Equivalent Sequences 84
Implications of Cooperative Design 84
Observability of State Faults 85
Nonspecific State Faults 85
4.2.6 Built-in Low-level Services 86
4.2.7 Bug Lists 86

4.3 Language-specific Hazards 91
4.3.1 C++ 92
4.3.2 Java 94
4.3.3 Smalltalk 95
4.4 Coverage Models for Object-oriented Testing 97
4.4.1 Code Coverage and Fault Models 97
Optimistic Scope 98
Method Scope Branch Coverage 98
Coverage for a Classless Language 99
ICpak Test Suite 99
Coverage Checklist 100
Incremental Class Testing 100
Class Interface Data Flow Coverage 101
Polymorphic Bindings 102
4.5 An OO Testing Manifesto 103
Unique Bug Hazards 103
Object-oriented Test Automation 105
Test Process 106
4.6 Bibliographic Notes 107

Part II Models 109

Chapter 5 Test Models 111

5.1 Test Design and Test Models 111
5.1.1 Why Testing Must Be Model-Based 111
5.1.2 What Is a Model? 112
5.1.3 The Role of Models in Testing 113
5.1.4 Cartoons or Test-ready Models? 114
5.1.5 Consequences 118
5.2 Bibliographic Notes 119

Chapter 6 Combinational Models 121

6.1 How Combinational Models Support Testing 121
6.2 How to Develop a Decision Table 123
6.2.1 Basic Approach 123
6.2.2 Components and Structure 124
6.2.3 The Auto Insurance Renewal Model 125
6.2.4 Don't Care, Don't Know, Can't Happen 127
6.2.5 Decision Tables in OO Development 131
6.3 Deriving the Logic Function 133
6.3.1 Boolean Expressions 133

6.3.2 Truth Tables versus Decision Tables 136
6.3.3 Elements of Boolean Expressions 136
6.3.4 Karnaugh-Veitch Matrix 141
6.3.5 Cause–Effect Graphs 144
6.4 Decision Table Validation 150
6.5 Test Generation 152
6.5.1 Fault Model 152
6.5.2 All-Explicit Variants 154
6.5.3 All-Variants, All-True, All-False, All-Primes 154
6.5.4 Each-Condition/All-Conditions 155
6.5.5 Binary Decision Diagram Determinants 157
6.5.6 Variable Negation 161
6.5.7 Nonbinary Variable Domain Analysis 164
6.5.8 Additional Heuristics 169
6.6 Choosing a Combinational Test Strategy 169
6.7 Bibliographic Notes 173

Chapter 7 State Machines 175

7.1 Motivation 176
7.2 The Basic Model 177
7.2.1 What Is a State Machine? 177
7.2.2 State Transition Diagrams 181
7.2.3 Some Properties of Finite State Automata 182
7.2.4 Guarded Transitions 183
7.2.5 Mealy and Moore 185
7.2.6 State Transition Tables 189
7.2.7 Limitations of the Basic Model 193
7.2.8 Statecharts 194
7.2.9 State Machines and Object-oriented Development 201
7.3 The FREE State Model 204
7.3.1 Limitations of OOA/D Behavior Models 204
7.3.2 States 205
A State Is a Value Set 205
State Invariants 208
Scope and Granularity 209
No Hybrids Allowed 212
7.3.3 Transitions 212
7.3.4 Alpha and Omega States 213
7.3.5 Inheritance and Class Flattening 215
Flattening the Class Hierarchy 215
Expanding the Statechart 220

7.3.6 Unspecified Event/State Pairs 223
The Response Matrix 225
Designing Responses to Illegal Events 228
7.4 State-based Test Design 229
7.4.1 How State Machines Fail 229
Control Faults 229
Incorrect Composite Behavior 234
7.4.2 Developing a Testable Model 236
When and How 236
State Model Validation 237
7.4.3 The N+ Test Strategy 242
The ThreePlayerGame Example 243
Generate the Round-trip Path Tree 243
Sneak Paths: Illegal Transitions and Evading the Guard 253
Event Path Sensitization 255
Built-in Test Support 256
Checking the Resultant State 256
7.4.4 Relative Power and Limitations 259
Piecewise 259
All Transitions 260
All *n*-Transition Sequences 261
All Round-trip Paths 261
M-Length Signature 261
7.4.5 Choosing a State-based Test Strategy 262
7.5 Bibliographic Notes 266

Chapter 8 A Tester's Guide to the UML 269

8.1 Introduction 269
8.1.1 The UML as a Test Model 269
8.1.2 Relational Testing Strategy 270
8.2 General-purpose Elements 271
8.2.1 Organization and Annotation 271
Package Diagrams and Packages 271
Keywords and Stereotypes 274
Expressions, Constraints, and Comments 274
Notes 274
Element Properties 274
8.2.2 Object Constraint Language 275
8.3 Use Case Diagram 276
8.3.1 Notation and Semantics 276
8.3.2 Generic Test Model 278
8.3.3 Testability Extensions 279
Operational Variables 281
The Operational Relation 282

8.4 Class Diagram 283
8.4.1 Notation and Semantics 283
Association 284
Aggregation 285
Generalization 286
8.4.2 Generic Test Model 286
8.5 Sequence Diagram 286
8.5.1 Notation and Semantics 286
8.5.2 Generic Test Requirements 290
8.5.3 Testability Extensions 292
8.6 Activity Diagram 293
8.6.1 Notation and Semantics 293
8.6.2 Generic Test Model 296
8.7 Statechart Diagram 297
8.8 Collaboration Diagram 297
8.8.1 Notation and Semantics 297
8.8.2 Generic Test Model 300
8.8.3 Testability Extensions 300
8.9 Component Diagram 303
8.9.1 Notation and Semantics 303
8.9.2 Generic Test Model 304
8.10 Deployment Diagram 305
8.10.1 Notation and Semantics 305
8.10.2 Generic Test Model 306
8.11 Graphs, Relations, and Testing 307
8.12 Bibliographic Notes 313

Part III Patterns 315

Chapter 9 Results-oriented Test Strategy 317

9.1 Results-oriented Testing 317
9.1.1 The Role of Responsibility-based Test Design 322
Class Responsibilities 322
Why Test from Responsibility Models? 323
9.1.2 The Role of Implementation-based Test Design 324
9.1.3 Integration in Object-oriented Development 326
9.1.4 Harnessing Responsibility and Implementation 327
9.2 Test Design Patterns 328
9.2.1 What Is a Pattern? 328

9.2.2 Patterns and Testing 330
9.2.3 Test Design Pattern Template 330
Test Design Template 334
9.2.4 Test Patterns in This Book 338
9.2.5 Using Test Design Patterns 338
9.3 Documenting Test Cases, Suites, and Plans 340
9.3.1 IEEE 829 Documentation 340
9.3.2 Traceability 344
9.4 Bibliographic Notes 346

Chapter 10 Classes 347

10.1 Class Test and Integration 347
10.1.1 What Is Class Scope Testing? 347
10.1.2 Why Test at Class Scope? 350
10.1.3 Who and When 353
10.1.4 A Case Study: MPR Teltech 353
10.2 Preliminaries 355
10.2.1 Class Scope Integration 355
Small Pop 355
Alpha-Omega Cycle 356
10.2.2 Implementation-based Test Models 357
The Role of Code Coverage 357
A Code Coverage FAQ 360
Method Scope Code Coverage Models 362
The Class Flow Graph: A Class Scope Coverage Model 389
Once More with Feeling: The Dark Side of Code Coverage 396
10.2.3 Path Sensitization 399
10.2.4 Domain Testing Models 401
Domain Analysis 403
On, Off, In, and Out 405
Boundary Conditions with Two or More Variables 406
Modeling the Domain of Objects 407
The One-by-One Selection Criteria 410
The Domain Test Matrix 412
10.3 Method Scope Test Design Patterns 416
10.3.1 Functional Cohesion 416
10.3.2 Method Scope Integration 416
10.3.3 The Patterns 417
Category-Partition *419*
Combinational Function Test *427*

Recursive Function Test 433
Polymorphic Message Test 438

10.4 Class Scope Test Design Patterns 444
10.4.1 Class Modalities 444
10.4.2 The Patterns 446
Invariant Boundaries 447
Nonmodal Class Test 455
Quasi-modal Class Test 466
Modal Class Test 481

10.5 Flattened Class Scope Test Design Patterns 497
10.5.1 The Trouble with Superclasses 497
Some Definitions 497
Inheritance-related Bugs 501
10.5.2 Test Strategy for Flattened Classes 504
Testing Axioms 504
Some Subclass Development Scenarios 506
Inheriting Class Test Suites 510
10.5.3 The Patterns 510
Polymorphic Server Test 513
Modal Hierarchy Test 518

10.6 Bibliographic Notes 522

Chapter 11 Reusable Components 525

11.1 Testing and Reuse 525
11.1.1 Reuse Mechanisms 525
11.1.2 The Role of Testing in Reuse 528
11.1.3 Reusing Test Suites 532

11.2 Test Design Patterns 533
Abstract Class Test 534
Generic Class Test 537
New Framework Test 549
Popular Framework Test 556

11.3 Bibliographic Notes 562

Chapter 12 Subsystems 563

12.1 Subsystems 563
12.1.1 What Is a Subsystem? 563
12.1.2 Why Test at Subsystem Scope? 565

12.2 Subsystem Test Design Patterns 566
Class Association Test 569
Round-trip Scenario Test 579

Controlled Exception Test 592
Mode Machine Test 599
12.3 Bibliographic Notes 624

Chapter 13 Integration **627**
13.1 Integration in Object-oriented Development 627
13.1.1 Definitions 627
13.1.2 Integration Testing Is Essential 629
13.1.3 Dependency Analysis 634
13.1.4 Integration Faults 640
13.2 Integration Patterns 642
13.2.1 Scope-specific Considerations 643
Classes 643
Clusters 644
Subsystem/System Scope 645
Big Bang Integration 649
Bottom-up Integration 652
Top-down Integration 662
Collaboration Integration 670
Backbone Integration 679
Layer Integration 687
Client/Server Integration 691
Distributed Services Integration 699
High-frequency Integration 706
13.3 Bibliographic Notes 714

Chapter 14 Application Systems **715**
14.1 Testing Application Systems 715
14.1.1 A Cautionary Tale 715
14.1.2 Testing Object-oriented Application Systems 716
14.1.3 Application System Test Strategy 718
14.2 Test Design Patterns 721
Extended Use Case Test 722
Covered in CRUD 732
Allocate Tests by Profile 736
14.3 Implementation-specific Capabilities 742
14.3.1 Configuration and Compatibility 742
14.3.2 Performance 743
14.3.3 Integrity and Fault Tolerance 744
Concurrency Testing 744
Stress Testing 745
Restart/Recovery Testing 746
14.3.4 Human–Computer Interaction 746

14.4 Post-development Testing 748
14.5 Notes on Testing Performance Objectives 749
14.5.1 Batch Systems 750
14.5.2 Interactive System 751
14.5.3 Real-time Systems 752
14.6 Bibliographic Notes 753

Chapter 15 Regression Testing **755**
15.1 Preliminaries 755
15.1.1 What and Why 755
15.1.2 When and How 760
15.1.3 Regression Faults 761
15.1.4 Test Automation 763
15.1.5 Test Suite Maintenance 767
15.1.6 Considerations for Reducing a Test Suite 770
Safe Reduction 770
Unsafe Reduction 771
15.2 Test Patterns 772
Retest All 774
Retest Risky Use Cases 777
Retest by Profile 780
Retest Changed Code 784
Retest Within Firewall 790
15.3 Bibliographic Notes 797

Part IV Tools **799**
Chapter 16 Test Automation **801**
16.1 Why Testing Must Be Automated 801
16.2 Limitations and Caveats 804

Chapter 17 Assertions **807**
17.1 Introduction 807
17.1.1 What Are Assertions? 809
17.1.2 Why Use Assertions? 810
17.1.3 Who Uses Assertions? 814
A Million Lines of Built-in Test: OS/400 814
17.2 Implementation-based Assertions 815
17.2.1 Assumption Checkers 815
17.2.2 Dark Alleys, Lurking Faults, and Sniffers 817

17.3 Responsibility-based Assertions 820
17.3.1 Overview 820
17.3.2 Responsibilities, Contracts, and Subtypes 821
The Contract Metaphor 821
Assertions and the UML 823
Assertion Strength and Weakness 823
17.3.3 Method Scope 826
Preconditions 826
Loops 826
Postconditions 830
17.3.4 Class Scope 832
17.3.5 Sequential Constraints 834
State Invariant 834
Accepting and Resulting Conditions 836
Runtime Sequence Checking 838
17.3.6 Superclass/Subclass Scope 840
Subcontracts and Type Substitution 840
Buggy Inheritance 842
17.3.7 Client/Server Scope 842
Public and Private Contracts 842
Defensive Contracts? 845
Exceptions 846
17.4 Implementation 846
17.4.1 A Programmer's Assertion FAQ 846
17.4.2 Assertion Actions 850
17.4.3 Nonexecutable Assertions 853
17.4.4 Ada 95 854
`Assert` Pragma 854
Inline Assertion Procedure 857
17.4.5 C++ 860
`assert.h` Macro 860
Assert Template 862
The Coherence Idiom 863
Tips and Tricks 866
17.4.6 Eiffel 868
17.4.7 Java 870
Basic Checking Capabilities 870
Final Static/Short Circuit Idiom 871
17.4.8 Objective-C 875
OpenStep Foundation Kit Assertion Macros 876
17.4.9 Smalltalk 878
Context Assertion Method 878
Design-by-Contract Extensions 879
The Percolation Pattern 882

17.5 Deployment 897
17.5.1 Verification of Built-in Test 897
17.5.2 Using Assertions to Design Tests 898
Unarticulated Contracts 898
Practical Considerations 900
Test Suite Reduction 901
17.5.3 Prerelease Considerations 902
17.5.4 Post-release Considerations 903
17.6 Limitations and Caveats 907
17.7 Some Assertion Tools 911
17.8 Bibliographic Notes 913

Chapter 18 Oracles **917**

18.1 Introduction 917
18.2 Oracle Patterns 919
18.2.1 Introduction 919
Design Considerations 919
Choosing an Approach 923
Pattern Summary 924
18.2.2 Judging 926
18.2.3 Prespecification Oracles 927
Solved Example Oracle 927
Simulation Oracle 928
Approximation Oracle 929
Parametric Oracle 930
18.2.4 Gold Standard Oracles 931
Trusted System Oracle 931
Parallel Test Oracle 932
Regression Test Oracle 933
Voting Oracle 933
18.2.5 Organic Oracles 935
Smoke Test 935
Built-in Test Oracle 935
Built-in Check Oracle 936
Reversing Oracle 937
Executable Specification Oracle 939
Generated Implementation Oracle 940
Different But Equivalent Oracle 941
18.3 Comparators 943
18.3.1 Introduction 943
18.3.2 Determining Object Equality 944

18.3.3 Deep and Shallow Equality 946
18.3.4 Abstract Content Versus Concrete Content 950
18.3.5 Aliases 951
18.3.6 Collections 951
18.3.7 Type/Subtype Equality 952
18.3.8 Partial Equality Methods 954
18.4 Bibliographic Notes 954

Chapter 19 Test Harness Design 957

19.1 How to Develop a Test Harness 957
19.1.1 Requirements 959
19.1.2 Architecture 962
19.2 Test Case Patterns 962
19.2.1 Considerations 962
Test Case/Test Suite Method 966
Test Case/Test Suite Class 972
Catch All Exceptions 976
19.3 Test Control Patterns 979
19.3.1 Considerations 979
Server Stub 980
Server Proxy 986
19.4 Driver Patterns 988
19.4.1 Considerations 988
Design Goals 988
Inheriting Test Suites 988
Controllability, Observability, and Reuse 990
19.4.2 Driver Design Patterns 992
TestDriver Superclass 994
Percolate the Object Under Test 996
Symmetric Driver 999
Subclass Driver 1008
Private Access Driver 1014
Test Control Interface 1023
Drone 1026
Built-in Test Driver 1031
19.5 Test Execution Patterns 1039
Command Line Test Bundle 1040
Incremental Testing Framework 1043
Fresh Objects 1052
19.6 A Test Implementation Syntax 1059
19.7 Bibliographic Notes 1063

Appendix **BigFoot's Tootsie: A Case Study** **1065**
Requirements 1065
1. Effective Tester Interface 1066
2. Automate Test Generation 1066
3. Automate Test Setup and Execution 1067
OOA/D for Capability-driven Testing 1067
Implementation 1069
Test Control Strategy 1069
Some Test Automation Scenarios 1070
Glossary **1073**
References **1119**
Index **1143**