

# SOFTWARE TESTING TECHNIQUES

Second Edition

**Boris Beizer**

Technische Universität Darmstadt	
FACHBEREICH INFORMATIK	
<u>B I B L I O T H E K</u>	
Inventar-Nr.:	<u>101-00838</u>
Sachgebiete:	_____
Standort:	_____



**VAN NOSTRAND REINHOLD**  
New York

# CONTENTS

**PREFACE TO THE SECOND EDITION** ix

**PREFACE TO THE FIRST EDITION** xiii

**ACKNOWLEDGMENTS** xv

## **1. INTRODUCTION** 1

1. The Purpose of Testing 1
  - 1.1. What We Do 1
  - 1.2. Productivity and Quality in Software 2
  - 1.3. Goals for Testing 3
  - 1.4. Phases in a Tester's Mental Life 4
  - 1.5. Test Design 7
  - 1.6. Testing Isn't Everything 7
  - 1.7. The Pesticide Paradox and the Complexity Barrier 9
2. Some Dichotomies 9
  - 2.1. Testing Versus Debugging 9
  - 2.2. Function Versus Structure 10
  - 2.3. The Designer Versus the Tester 11
  - 2.4. Modularity Versus Efficiency 13
  - 2.5. Small Versus Large 13
  - 2.6. The Builder Versus the Buyer 14
3. A Model for Testing 15
  - 3.1. The Project 15
  - 3.2. Overview 17
  - 3.3. The Environment 17
  - 3.4. The Program 18
  - 3.5. Bugs 18
  - 3.6. Tests 20
  - 3.7. Testing and Levels 20
  - 3.8. The Role of Models 22
4. Playing Pool and Consulting Oracles 22
  - 4.1. Playing Pool 22
  - 4.2. Oracles 23
  - 4.3. Sources of Oracles 23
5. Is Complete Testing Possible? 24

## **xviii CONTENTS**

### **2. THE TAXONOMY OF BUGS 27**

1. Synopsis 27
2. The Consequences of Bugs 27
  - 2.1. The Importance of Bugs 27
  - 2.2. How Bugs Affect Us—Consequences 28
  - 2.3. Flexible Severity Rather Than Absolutes 29
  - 2.4. The Nightmare List and When to Stop Testing 30
3. A Taxonomy for Bugs 33
  - 3.1. General 33
  - 3.2. Requirements, Features, and Functionality Bugs 34
  - 3.3. Structural Bugs 36
  - 3.4. Data Bugs 39
  - 3.5. Coding Bugs 47
  - 3.6. Interface, Integration, and System Bugs 48
  - 3.7. Test and Test Design Bugs 54
  - 3.8. Testing and Design Style 55
4. Some Bug Statistics 56
5. Summary 58

### **3. FLOWGRAPHS AND PATH TESTING 59**

1. Synopsis 59
2. Path-Testing Basics 59
  - 2.1. Motivation and Assumptions 59
  - 2.2. Control Flowgraphs 60
  - 2.3. Path Testing 70
  - 2.4. Loops 80
  - 2.5. More on Testing Multi-Entry/Multi-Exit Routines 85
  - 2.6. Effectiveness of Path Testing 90
  - 2.7. Variations 91
3. Predicates, Path Predicates, and Achievable Paths 92
  - 3.1. General 92
  - 3.2. Predicates 93
  - 3.3. Predicate Expressions 94
  - 3.4. Predicate Coverage 98
  - 3.5. Testing Blindness 99
4. Path Sensitizing 101
  - 4.1. Review; Achievable and Unachievable Paths 101
  - 4.2. Pragmatic Observations 102
  - 4.3. Heuristic Procedures for Sensitizing Paths 103
  - 4.4. Examples 104
5. Path Instrumentation 109
  - 5.1. The Problem 109
  - 5.2. General Strategy 110

- 5.3. Link Markers 111
- 5.4. Link Counters 112
- 5.5. Other Instrumentation Methods 113
- 5.6. Implementation 113
- 6. Implement and Application of Path Testing 115
  - 6.1. Integration, Coverage, and Paths in Called Components 115
  - 6.2. New Code 116
  - 6.3. Maintenance 117
  - 6.4. Rehosting 117
- 7. Testability Tips 118
- 8. Summary 119

#### 4. TRANSACTION-FLOW TESTING 121

- 1. Synopsis 121
- 2. Generalizations 121
- 3. Transaction Flows 122
  - 3.1. Definitions 122
  - 3.2. Example 123
  - 3.3. Usage 125
  - 3.4. Implementation 125
  - 3.5. Perspective 128
  - 3.6. Complications 129
  - 3.7. Transaction-Flow Structure 131
- 4. Transaction-Flow Testing Techniques 133
  - 4.1. Get the Transaction Flows 133
  - 4.2. Inspections, Reviews, Walkthroughs 134
  - 4.3. Path Selection 135
  - 4.4. Sensitization 136
  - 4.5. Instrumentation 138
  - 4.6. Test Databases 139
  - 4.7. Execution 139
- 5. Implementation Comments 140
  - 5.1. Transaction-Based Systems 140
  - 5.2. Hidden Languages 141
- 6. Testability Tips 143
- 7. Summary 143

#### 5. DATA-FLOW TESTING 145

- 1. Synopsis 145
- 2. Data-Flow Testing Basics 145
  - 2.1. Motivation and Assumptions 145
  - 2.2. Data Flowgraphs 150
  - 2.3. The Data-Flow Model 157

## xx CONTENTS

- 3. Data-Flow Testing Strategies 161
    - 3.1. General 161
    - 3.2. Terminology 162
    - 3.3. The Strategies 163
    - 3.4. Slicing, Dicing, Data Flow, and Debugging 167
  - 4. Application, Tools, Effectiveness 168
  - 5. Testability Tips 171
  - 6. Summary 172
- 
- 6. DOMAIN TESTING 173**
    - 1. Synopsis 173
    - 2. Domains and Paths 173
      - 2.1. The Model 173
      - 2.2. A Domain Is a Set 174
      - 2.3. Domains, Paths, and Predicates 175
      - 2.4. Domain Closure 176
      - 2.5. Domain Dimensionality 177
      - 2.6. The Bug Assumptions 177
      - 2.7. Restrictions 179
    - 3. Nice Domains and Ugly Domains 182
      - 3.1. Where Do Domains Come From? 182
      - 3.2. Specified Versus Implemented Domains 183
      - 3.3. Nice Domains 183
      - 3.4. Ugly Domains and How Programmers and Testers Treat Them 188
    - 4. Domain Testing 192
      - 4.1. Overview 192
      - 4.2. Domain Bugs and How to Test for Them 192
      - 4.3. Procedure 200
      - 4.4. Variations, Tools, Effectiveness 201
    - 5. Domains and Interface Testing 202
      - 5.1. General 202
      - 5.2. Domains and Range 203
      - 5.3. Closure Compatibility 203
      - 5.4. Span Compatibility 204
      - 5.5. Interface Range/Domain Compatibility Testing 206
      - 5.6. Finding the Values 206
    - 6. Domains and Testability 207
      - 6.1. General 207
      - 6.2. Linearizing Transformations 207
      - 6.3. Coordinate Transformations 208
      - 6.4. A Canonical Program Form 209
      - 6.5. Great Insights? 210
    - 7. Summary 211

<b>7. METRICS AND COMPLEXITY</b>	<b>213</b>
1. Synopsis	213
2. Metrics, What and Why	213
2.1. Philosophy	213
2.2. Historical Perspective	214
2.3. Objectives	215
3. Linguistic Metrics	217
3.1. General	217
3.2. Lines of Code, Statements Counts, and Related Metrics	217
3.3. Halstead's Metrics	220
3.4. Token Count	226
4. Structural Metrics	227
4.1. General	227
4.2. Cyclomatic Complexity (McCabe's Metric)	228
4.3. Other Structural Metrics	236
5. Hybrid Metrics	237
6. Metrics Implementation	237
6.1. The Pattern of Metrics Research	237
6.2. A Pattern for Successful Applications	239
7. Testability Tips	240
8. Summary	242
<b>8. PATHS, PATH PRODUCTS, AND REGULAR EXPRESSIONS</b>	<b>243</b>
1. Synopsis	243
2. Motivation	243
3. Path Products and Path Expressions	244
3.1. Overview	244
3.2. Basic Concepts	244
3.3. Path Products	246
3.4. Path Sums	247
3.5. Distributive Laws	248
3.6. Absorption Rule	248
3.7. Loops	249
3.8. Identity Elements	250
4. A Reduction Procedure	251
4.1. Overview	251
4.2. Cross-Term Step	252
4.3. Parallel Term	254
4.4. Loop Term	254
4.5. Comments, Identities, and Node-Removal Order	256
5. Applications	257
5.1. General	257
5.2. How Many Paths in a Flowgraph?	258

## xxii CONTENTS

5.3.	Approximate Minimum Number of Paths	262
5.4.	The Probability of Getting There	266
5.5.	The Mean Processing Time of a Routine	271
5.6.	Push/Pop, Get/Return	274
5.7.	Limitations and Solutions	277
6.	Regular Expressions and Flow-Anomaly Detection	278
6.1.	The Problem	278
6.2.	The Method	279
6.3.	A Data-Flow Testing Example	280
6.4.	Generalizations, Limitations, and Comments	281
7.	Summary	282

## 9. SYNTAX TESTING 284

1.	Synopsis	284
2.	Why, What, and How	284
2.1.	Garbage	284
2.2.	Casual and Malicious Users	285
2.3.	Operators	285
2.4.	The Internal World	286
2.5.	What to Do	286
2.6.	Applications and Hidden Languages	287
2.7.	The Graph We Cover	288
2.8.	Overview	290
3.	A Grammar for Formats	291
3.1.	Objectives	291
3.2.	BNF Notation	291
4.	Test Case Generation	295
4.1.	Generators, Recognizers, and Approach	295
4.2.	Test Case Design	296
4.3.	Sources of Syntax	303
4.4.	Ambiguities and Contradictions	306
4.5.	Where Did the Good Guys Go?	307
5.	Implementation and Application	309
5.1.	Execution Automation	309
5.2.	Design Automation	311
5.3.	Productivity, Training, and Effectiveness	313
5.4.	Ad-Lib Tests	314
6.	Testability Tips	315
6.1.	The Tip	315
6.2.	Compiler Overview	315
6.3.	Typical Software	317
6.4.	Separation of Phases	317
6.5.	Prerequisites	318
7.	Summary	318

<b>10. LOGIC-BASED TESTING</b>	<b>320</b>
1. Synopsis	320
2. Motivational Overview	320
2.1. Programmers and Logic	320
2.2. Hardware Logic Testing	320
2.3. Specification Systems and Languages	321
2.4. Knowledge-Based Systems	321
2.5. Overview	322
3. Decision Tables	322
3.1. Definitions and Notation	322
3.2. Decision-Table Processors	324
3.3. Decision Tables as a Basis for Test Case Design	325
3.4. Expansion of Immaterial Cases	326
3.5. Test Case Design	328
3.6. Decision Tables and Structure	329
4. Path Expressions Again	332
4.1. General	332
4.2. Boolean Algebra	334
4.3. Boolean Equations	341
5. KV Charts	343
5.1. The Problem	343
5.2. Simple Forms	343
5.3. Three Variables	347
5.4. Four Variables and More	350
5.5. Even More Testing Strategies?	352
6. Specifications	352
6.1. General	352
6.2. Finding and Translating the Logic	353
6.3. Ambiguities and Contradictions	355
6.4. Don't-Care and Impossible Terms	357
7. Testability Tips	359
8. Summary	361
<b>11. STATES, STATE GRAPHS, AND TRANSITION TESTING</b>	<b>363</b>
1. Synopsis	363
2. Motivational Overview	363
3. State Graphs	364
3.1. States	364
3.2. Inputs and Transitions	365
3.3. Outputs	366
3.4. State Tables	367
3.5. Time Versus Sequence	368
3.6. Software Implementation	369



## xxiv CONTENTS

- 4. Good State Graphs and Bad 373
  - 4.1. General 373
  - 4.2. State Bugs 374
  - 4.3. Transition Bugs 380
  - 4.4. Output Errors 386
  - 4.5. Encoding Bugs 386
- 5. State Testing 387
  - 5.1. Impact of Bugs 387
  - 5.2. Principles 387
  - 5.3. Limitations and Extensions 388
  - 5.4. What to Model 389
  - 5.5. Getting the Data 390
  - 5.6. Tools 390
- 6. Testability Tips 391
  - 6.1. A Balm for Programmers 391
  - 6.2. How Big, How Small? 391
  - 6.3. Switches, Flags, and Unachievable Paths 391
  - 6.4. Essential and Inessential Finite-State Behavior 392
  - 6.5. Design Guidelines 394
- 7. Summary 395

## ~ 12. GRAPH MATRICES AND APPLICATIONS 397

- 1. Synopsis 397
- 2. Motivational Overview 397
  - 2.1. The Problem with Pictorial Graphs 397
  - 2.2. Tool Building 398
  - 2.3. Doing and Understanding Testing Theory 398
  - 2.4. The Basic Algorithms 398
- 3. The Matrix of a Graph 399
  - 3.1. Basic Principles 399
  - 3.2. A Simple Weight 401
  - 3.3. Further Notation 401
- 4. Relations 402
  - 4.1. General 402
  - 4.2. Properties of Relations 403
  - 4.3. Equivalence Relations 404
  - 4.4. Partial Ordering Relations 405
- 5. The Powers of a Matrix 405
  - 5.1. Principles 405
  - 5.2. Matrix Powers and Products 406
  - 5.3. The Set of All Paths 408
  - 5.4. Loops 410
  - 5.5. Partitioning Algorithm 411

5.6.	Breaking Loops and Applications	414
6.	Node-Reduction Algorithm	415
6.1.	General	415
6.2.	Some Matrix Properties	415
6.3.	The Algorithm	416
6.4.	Applications	419
6.5.	Some Hints	420
7.	Building Tools	421
7.1.	Matrix Representation in Software	421
7.2.	Matrix Operations	424
7.3.	Node-Reduction Optimization	426
8.	Summary	426
<b>13.</b>	<b>IMPLEMENTATION</b>	<b>428</b>
1.	Synopsis	428
2.	Overview	428
2.1.	General	428
2.2.	Who Does What Kind of Tests	428
2.3.	Adversarial Roles	431
3.	Strategies for Programmers	431
3.1.	Who's Responsible?	431
3.2.	Programming in Perspective	432
3.3.	Self-Inspection	435
3.4.	Test Case Design	437
3.5.	Test Type Priority	438
3.6.	Learn to "Cheat"	438
4.	Strategies for Independent Testers	439
4.1.	General	439
4.2.	Database Gold	439
4.3.	Reading Code and Other Design Documents	446
4.4.	Requirements	446
4.5.	Tests and the Threat of Tests	447
5.	Tests as Software Products	447
5.1.	Throwaway Tests?	447
5.2.	Tests, Subtests, Scripts, and Suites	447
5.3.	Test Configuration Control	448
6.	Tools	449
6.1.	General	449
6.2.	Fundamental Tools	450
6.3.	Test Execution Automation	451
6.4.	Test Design Automation	455
6.5.	Tool Usage and Training	459

**xxvi CONTENTS**

<b>APPENDIX BUG TAXONOMY AND STATISTICS</b>	<b>460</b>
<b>CITED REFERENCES/BIBLIOGRAPHY</b>	<b>477</b>
<b>GLOSSARY/INDEX</b>	<b>515</b>