

Contents

<i>List of Figures</i>	<i>page ix</i>
<i>List of Tables</i>	xiii
<i>Preface</i>	xv
Part 1 Overview	1
1 Introduction	3
1.1 Activities of a Test Engineer	4
1.1.1 Testing Levels Based on Software Activity	5
1.1.2 Beizer's Testing Levels Based on Test Process Maturity	8
1.1.3 Automation of Test Activities	10
1.2 Software Testing Limitations and Terminology	11
1.3 Coverage Criteria for Testing	16
1.3.1 Infeasibility and Subsumption	20
1.3.2 Characteristics of a Good Coverage Criterion	20
1.4 Older Software Testing Terminology	21
1.5 Bibliographic Notes	22
Part 2 Coverage Criteria	25
2 Graph Coverage	27
2.1 Overview	27
2.2 Graph Coverage Criteria	32
2.2.1 Structural Coverage Criteria	33
2.2.2 Data Flow Criteria	44
2.2.3 Subsumption Relationships among Graph Coverage Criteria	50
2.3 Graph Coverage for Source Code	52

vi **Contents**

2.3.1	Structural Graph Coverage for Source Code	52
2.3.2	Data Flow Graph Coverage for Source Code	54
2.4	Graph Coverage for Design Elements	65
2.4.1	Structural Graph Coverage for Design Elements	65
2.4.2	Data Flow Graph Coverage for Design Elements	67
2.5	Graph Coverage for Specifications	75
2.5.1	Testing Sequencing Constraints	75
2.5.2	Testing State Behavior of Software	77
2.6	Graph Coverage for Use Cases	87
2.6.1	Use Case Scenarios	90
2.7	Representing Graphs Algebraically	91
2.7.1	Reducing Graphs to Path Expressions	94
2.7.2	Applications of Path Expressions	96
2.7.3	Deriving Test Inputs	96
2.7.4	Counting Paths in a Flow Graph and Determining Max Path Length	97
2.7.5	Minimum Number of Paths to Reach All Edges	98
2.7.6	Complementary Operations Analysis	98
2.8	Bibliographic Notes	100
3	Logic Coverage	104
3.1	Overview: Logic Predicates and Clauses	104
3.2	Logic Expression Coverage Criteria	106
3.2.1	Active Clause Coverage	107
3.2.2	Inactive Clause Coverage	111
3.2.3	Infeasibility and Subsumption	112
3.2.4	Making a Clause Determine a Predicate	113
3.2.5	Finding Satisfying Values	115
3.3	Structural Logic Coverage of Programs	120
3.3.1	Predicate Transformation Issues	127
3.4	Specification-Based Logic Coverage	131
3.5	Logic Coverage of Finite State Machines	134
3.6	Disjunctive Normal Form Criteria	138
3.7	Bibliographic Notes	147
4	Input Space Partitioning	150
4.1	Input Domain Modeling	152
4.1.1	Interface-Based Input Domain Modeling	153
4.1.2	Functionality-Based Input Domain Modeling	154
4.1.3	Identifying Characteristics	154
4.1.4	Choosing Blocks and Values	156
4.1.5	Using More than One Input Domain Model	158
4.1.6	Checking the Input Domain Model	158
4.2	Combination Strategies Criteria	160
4.3	Constraints among Partitions	165
4.4	Bibliographic Notes	166

	Contents	vii
5 Syntax-Based Testing	170	
5.1 Syntax-Based Coverage Criteria	170	
5.1.1 BNF Coverage Criteria	170	
5.1.2 Mutation Testing	173	
5.2 Program-Based Grammars	176	
5.2.1 BNF Grammars for Languages	176	
5.2.2 Program-Based Mutation	176	
5.3 Integration and Object-Oriented Testing	191	
5.3.1 BNF Integration Testing	192	
5.3.2 Integration Mutation	192	
5.4 Specification-Based Grammars	197	
5.4.1 BNF Grammars	198	
5.4.2 Specification-Based Mutation	198	
5.5 Input Space Grammars	201	
5.5.1 BNF Grammars	201	
5.5.2 Mutation for Input Grammars	204	
5.6 Bibliographic Notes	210	
Part 3 Applying Criteria in Practice	213	
6 Practical Considerations	215	
6.1 Regression Testing	215	
6.2 Integration and Testing	217	
6.2.1 Stubs and Drivers	218	
6.2.2 Class Integration Test Order	218	
6.3 Test Process	219	
6.3.1 Requirements Analysis and Specification	220	
6.3.2 System and Software Design	221	
6.3.3 Intermediate Design	222	
6.3.4 Detailed Design	223	
6.3.5 Implementation	223	
6.3.6 Integration	224	
6.3.7 System Deployment	224	
6.3.8 Operation and Maintenance	224	
6.3.9 Summary	225	
6.4 Test Plans	225	
6.5 Identifying Correct Outputs	230	
6.5.1 Direct Verification of Outputs	230	
6.5.2 Redundant Computations	231	
6.5.3 Consistency Checks	231	
6.5.4 Data Redundancy	232	
6.6 Bibliographic Notes	233	
7 Engineering Criteria for Technologies	235	
7.1 Testing Object-Oriented Software	236	
7.1.1 Unique Issues with Testing OO Software	237	

viii **Contents**

7.1.2	Types of Object-Oriented Faults	237
7.2	Testing Web Applications and Web Services	256
7.2.1	Testing Static Hyper Text Web Sites	257
7.2.2	Testing Dynamic Web Applications	257
7.2.3	Testing Web Services	260
7.3	Testing Graphical User Interfaces	260
7.3.1	Testing GUIs	261
7.4	Real-Time Software and Embedded Software	262
7.5	Bibliographic Notes	265
8	Building Testing Tools	268
8.1	Instrumentation for Graph and Logical Expression Criteria	268
8.1.1	Node and Edge Coverage	268
8.1.2	Data Flow Coverage	271
8.1.3	Logic Coverage	272
8.2	Building Mutation Testing Tools	272
8.2.1	The Interpretation Approach	274
8.2.2	The Separate Compilation Approach	274
8.2.3	The Schema-Based Approach	275
8.2.4	Using Java Reflection	276
8.2.5	Implementing a Modern Mutation System	277
8.3	Bibliographic Notes	277
9	Challenges in Testing Software	280
9.1	Testing for Emergent Properties: Safety and Security	280
9.1.1	Classes of Test Cases for Emergent Properties	283
9.2	Software Testability	284
9.2.1	Testability for Common Technologies	285
9.3	Test Criteria and the Future of Software Testing	286
9.3.1	Going Forward with Testing Research	288
9.4	Bibliographic Notes	290
	<i>List of Criteria</i>	293
	<i>Bibliography</i>	295
	<i>Index</i>	319