

WHY PROGRAMS FAIL

A Guide to Systematic Debugging

ANDREAS ZELLER



ELSEVIER



MORGAN KAUFMANN PUBLISHERS



dpunkt.verlag

AMSTERDAM • BOSTON • HEIDELBERG • LONDON • NEW YORK • OXFORD
PARIS • SAN DIEGO • SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

MORGAN KAUFMANN PUBLISHERS IS AN IMPRINT OF ELSEVIER

HEIDELBERG

CONTENTS

Foreword xix

Preface xxi

1 HOW FAILURES COME TO BE 1

- 1.1 My Program Does Not Work! 1
- 1.2 From Defects to Failures 3
- 1.3 Lost in Time and Space 6
- 1.4 From Failures to Fixes 10
- 1.5 Automated Debugging Techniques 15
- 1.6 Bugs, Faults, or Defects? 19
- 1.7 Concepts 21
- 1.8 Tools 23
- 1.9 Further Reading 23
- 1.10 Exercises 24

2 TRACKING PROBLEMS 27

- 2.1 Oh! All These Problems 27
- 2.2 Reporting Problems 28
- 2.3 Managing Problems 32
- 2.4 Classifying Problems 34
 - 2.4.1 Severity 34
 - 2.4.2 Priority 35
 - 2.4.3 Identifier 35
 - 2.4.4 Comments 35
 - 2.4.5 Notification 35
- 2.5 Processing Problems 35

2.6	Managing Problem Tracking	38
2.7	Requirements as Problems	39
2.8	Managing Duplicates	40
2.9	Relating Problems and Fixes	42
2.10	Relating Problems and Tests	44
2.11	Concepts	46
2.12	Tools	48
	BUGZILLA	48
	PHPBUGTRACKER	49
	ISSUETRACKER	49
	TRAC	49
	SOURCEFORGE	49
	GFORGE	50
2.13	Further Reading	50
2.14	Exercises	51
3	MAKING PROGRAMS FAIL	53
3.1	Testing for Debugging	53
3.2	Controlling the Program	55
3.3	Testing at the Presentation Layer	58
	3.3.1 Low-level Interaction	58
	3.3.2 System-level Interaction	60
	3.3.3 Higher-level Interaction	61
	3.3.4 Assessing Test Results	62
3.4	Testing at the Functionality Layer	63
3.5	Testing at the Unit Layer	66
3.6	Isolating Units	71
3.7	Designing for Debugging	74
3.8	Preventing Unknown Problems	77
3.9	Concepts	79
3.10	Tools	80
	JUNIT	80
	ANDROID	81
	APPLESCRIPT	81
	VBSCRIPT	81
	Other scripting languages	81
	FAU	81
	VMWare	82
	Virtual PC	82

- 3.11 Further Reading 82
- 3.12 Exercises 83

4 REPRODUCING PROBLEMS 85

- 4.1 The First Task in Debugging 85
- 4.2 Reproducing the Problem Environment 86
- 4.3 Reproducing Program Execution 89
 - 4.3.1 Reproducing Data 91
 - 4.3.2 Reproducing User Interaction 92
 - 4.3.3 Reproducing Communications 94
 - 4.3.4 Reproducing Time 95
 - 4.3.5 Reproducing Randomness 96
 - 4.3.6 Reproducing Operating Environments 96
 - 4.3.7 Reproducing Schedules 99
 - 4.3.8 Physical Influences 101
 - 4.3.9 Effects of Debugging Tools 102
- 4.4 Reproducing System Interaction 104
- 4.5 Focusing on Units 105
 - 4.5.1 Setting Up a Control Layer 105
 - 4.5.2 A Control Example 106
 - 4.5.3 Mock Objects 109
 - 4.5.4 Controlling More Interaction 112
- 4.6 Concepts 112
- 4.7 Tools 114
 - WINRUNNER 114
 - ANDROID 114
 - REVRT 114
 - Checkpointing Tools 114
- 4.8 Further Reading 114
- 4.9 Exercises 115

5 SIMPLIFYING PROBLEMS 117

- 5.1 Simplifying the Problem 117
- 5.2 The Gecko BugATHon 118
- 5.3 Manual Simplification 121
- 5.4 Automatic Simplification 123
- 5.5 A Simplification Algorithm 125
- 5.6 Simplifying User Interaction 132
- 5.7 Random Input Simplified 133

5.8	Simplifying Faster	134
5.8.1	Caching	134
5.8.2	Stop Early	134
5.8.3	Syntactic Simplification	135
5.8.4	Isolate Differences, Not Circumstances	137
5.9	Concepts	138
5.10	Tools	139
	Delta Debugging	139
	Simplification Library	139
5.11	Further Reading	139
5.12	Exercises	140
6	SCIENTIFIC DEBUGGING	145
6.1	How to Become a Debugging Guru	145
6.2	The Scientific Method	146
6.3	Applying the Scientific Method	147
6.3.1	Debugging sample—Preparation	149
6.3.2	Debugging sample—Hypothesis 1	149
6.3.3	Debugging sample—Hypothesis 2	150
6.3.4	Debugging sample—Hypothesis 3	150
6.3.5	Debugging sample—Hypothesis 4	150
6.4	Explicit Debugging	151
6.5	Keeping a Logbook	153
6.6	Debugging Quick-and-Dirty	154
6.7	Algorithmic Debugging	155
6.8	Deriving a Hypothesis	158
6.9	Reasoning About Programs	161
6.10	Concepts	163
6.11	Further Reading	164
6.12	Exercises	165
7	DEDUCING ERRORS	167
7.1	Isolating Value Origins	167
7.2	Understanding Control Flow	168
7.3	Tracking Dependences	172
7.3.1	Effects of Statements	172
7.3.2	Affected Statements	174
7.3.3	Statement Dependences	175

7.3.4	Following Dependences	177
7.3.5	Leveraging Dependences	177
7.4	Slicing Programs	178
7.4.1	Forward Slices	179
7.4.2	Backward Slices	179
7.4.3	Slice Operations	180
7.4.4	Leveraging Slices	182
7.4.5	Executable Slices	182
7.5	Deducing Code Smells	183
7.6	Limits of Static Analysis	189
7.7	Concepts	193
7.8	Tools	194
	CODESURFER	194
	FINDBUGS	194
7.9	Further Reading	194
7.10	Exercises	195
8	OBSERVING FACTS	199
8.1	Observing State	199
8.2	Logging Execution	200
8.2.1	Logging Functions	202
8.2.2	Logging Frameworks	206
8.2.3	Logging with Aspects	208
8.2.4	Logging at the Binary Level	213
8.3	Using Debuggers	215
8.3.1	A Debugging Session	216
8.3.2	Controlling Execution	220
8.3.3	Postmortem Debugging	221
8.3.4	Logging Data	222
8.3.5	Invoking Functions	223
8.3.6	Fix and Continue	223
8.3.7	Embedded Debuggers	224
8.3.8	Debugger Caveats	225
8.4	Querying Events	225
8.4.1	Watchpoints	226
8.4.2	Uniform Event Queries	228
8.5	Visualizing State	230
8.6	Concepts	232
8.7	Tools	233
	LOG4J	233
	ASPECTJ	233

	PIN	233
	BCEL	234
	GDB	234
	DDD	234
	JAVA SPIDER	234
	eDOBS	235
8.8	Further Reading	235
8.9	Exercises	235
9	TRACKING ORIGINS	243
9.1	Reasoning Backwards	243
9.2	Exploring Execution History	244
9.3	Dynamic Slicing	246
9.4	Leveraging Origins	249
9.5	Tracking Down Infections	253
9.6	Concepts	254
9.7	Tools	254
	ODB	254
9.8	Further Reading	255
9.9	Exercises	255
10	ASSERTING EXPECTATIONS	257
10.1	Automating Observation	257
10.2	Basic Assertions	259
10.3	Asserting Invariants	261
10.4	Asserting Correctness	265
10.5	Assertions as Specifications	268
10.6	From Assertions to Verification	269
10.7	Reference Runs	272
10.8	System Assertions	275
	10.8.1 Validating the Heap with MALLOC_CHECK	276
	10.8.2 Avoiding Buffer Overflows with ELECTRICFENCE	277
	10.8.3 Detecting Memory Errors with VALGRIND	277
	10.8.4 Language Extensions	279
10.9	Checking Production Code	281
10.10	Concepts	283
10.11	Tools	284
	JML	284

ESC	285
GUARD	285
VALGRIND	285
PURIFY	285
INSURE++	285
CYCLONE	286
CCURED	286
10.12 Further Reading	286
10.13 Exercises	288
11 DETECTING ANOMALIES	295
11.1 Capturing Normal Behavior	295
11.2 Comparing Coverage	297
11.3 Statistical Debugging	302
11.4 Collecting Data in the Field	303
11.5 Dynamic Invariants	305
11.6 Invariants on the Fly	309
11.7 From Anomalies to Defects	311
11.8 Concepts	312
11.9 Tools	313
DAIKON	313
DIDUCE	313
11.10 Further Reading	313
11.11 Exercises	314
12 CAUSES AND EFFECTS	317
12.1 Causes and Alternate Worlds	317
12.2 Verifying Causes	319
12.3 Causality in Practice	320
12.4 Finding Actual Causes	322
12.5 Narrowing Down Causes	323
12.6 A Narrowing Example	324
12.7 The Common Context	325
12.8 Causes in Debugging	325
12.9 Concepts	326
12.10 Further Reading	327
12.11 Exercises	328

13 ISOLATING FAILURE CAUSES 331

- 13.1 Isolating Causes Automatically 331
- 13.2 Isolating versus Simplifying 332
- 13.3 An Isolation Algorithm 335
- 13.4 Implementing Isolation 336
- 13.5 Isolating Failure-inducing Input 340
- 13.6 Isolating Failure-inducing Schedules 340
- 13.7 Isolating Failure-inducing Changes 343
- 13.8 Problems and Limitations 349
- 13.9 Concepts 351
- 13.10 Tools 352
 - Delta Debugging Plug-ins for ECLIPSE 352
 - CCACHE 352
- 13.11 Further Reading 353
- 13.12 Exercises 353

14 ISOLATING CAUSE-EFFECT CHAINS 357

- 14.1 Useless Causes 357
- 14.2 Capturing Program States 360
- 14.3 Comparing Program States 364
- 14.4 Isolating Relevant Program States 366
- 14.5 Isolating Cause-Effect Chains 370
- 14.6 Isolating Failure-inducing Code 375
- 14.7 Issues and Risks 379
- 14.8 Concepts 382
- 14.9 Tools 383
 - ASKIGOR 383
 - IGOR 383
- 14.10 Further Reading 383
- 14.11 Exercises 384

15 FIXING THE DEFECT 387

- 15.1 Locating the Defect 387
- 15.2 Focusing on the Most Likely Errors 389
- 15.3 Validating the Defect 391
 - 15.3.1 Does the Error Cause the Failure? 391
 - 15.3.2 Is the Cause Really an Error? 392

- 15.3.3 Think Before You Code 393
- 15.4 Correcting the Defect 395
 - 15.4.1 Does the Failure No Longer Occur? 395
 - 15.4.2 Did the Correction Introduce New Problems? 396
 - 15.4.3 Was the Same Mistake Made Elsewhere? 397
 - 15.4.4 Did I Do My Homework? 398
- 15.5 Workarounds 398
- 15.6 Learning from Mistakes 399
- 15.7 Concepts 402
- 15.8 Further Reading 403
- 15.9 Exercises 404

FORMAL DEFINITIONS 407

- A.1 Delta Debugging 407
 - A.1.1 Configurations 407
 - A.1.2 Passing and Failing Run 407
 - A.1.3 Tests 408
 - A.1.4 Minimality 408
 - A.1.5 Simplifying 408
 - A.1.6 Differences 409
 - A.1.7 Isolating 410
- A.2 Memory Graphs 411
 - A.2.1 Formal Structure 411
 - A.2.2 Unfolding Data Structures 412
 - A.2.3 Matching Vertices and Edges 414
 - A.2.4 Computing the Common Subgraph 415
 - A.2.5 Computing Graph Differences 415
 - A.2.6 Applying Partial State Changes 418
 - A.2.7 Capturing C State 418
- A.3 Cause-Effect Chains 420

GLOSSARY 423

BIBLIOGRAPHY 429

INDEX 439