

CHAPMAN & HALL/CRC INNOVATIONS IN
SOFTWARE ENGINEERING AND SOFTWARE DEVELOPMENT

Fundamentals of Dependable Computing

for Software Engineers

John Knight

With Foreword by Brian Randell



CRC Press

Taylor & Francis Group

Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group an Informa business

A CHAPMAN & HALL BOOK

Table of Contents

Foreword	xiii
Preface	xv
CHAPTER 1 Introduction	1
1.1 The Elements of Dependability.....	1
1.1.1 A Cautionary Tale.....	1
1.1.2 Why Dependability?.....	4
1.2 The Role of the Software Engineer	5
1.3 Our Dependence on Computers	7
1.4 Some Regrettable Failures	9
1.4.1 The Ariane V	9
1.4.2 Korean Air Flight 801.....	10
1.4.3 The Mars Climate Orbiter.....	11
1.4.4 The Mars Polar Lander	11
1.4.5 Other Important Incidents.....	12
1.4.6 How to Think about Failures.....	12
1.5 Consequences of Failure	13
1.5.1 Non-Obvious Consequences of Failure.....	13
1.5.2 Unexpected Costs of Failure.....	14
1.5.3 Categories of Consequences	15
1.5.4 Determining the Consequences of Failure.....	16
1.6 The Need for Dependability	17
1.7 Systems and Their Dependability Requirements	18
1.7.1 Critical Systems.....	18
1.7.2 Systems That Help Build Systems.....	20
1.7.3 Systems That Interact with Other Systems.....	21
1.8 Where Do We Go from Here?.....	22
1.9 Organization of This Book	23
Exercises	25
CHAPTER 2 Dependability Requirements	27
2.1 Why We Need Dependability Requirements.....	27
2.2 The Evolution of Dependability Concepts	28
2.3 The Role of Terminology	30
2.4 What Is a System?	31
2.5 Requirements and Specification.....	34

2.6	Failure.....	35
2.6.1	The Notion of Service Failure.....	35
2.6.2	Sources of Service Failure.....	36
2.6.3	A Practical View of Requirements and Specification.....	38
2.6.4	Viewpoints of Service Failure.....	39
2.6.5	Informing the User about Failure.....	40
2.7	Dependability and Its Attributes.....	41
2.7.1	Reliability.....	43
2.7.2	Availability.....	44
2.7.3	Failure per Demand.....	48
2.7.4	Safety.....	48
2.7.5	Confidentiality.....	51
2.7.6	Integrity.....	52
2.7.7	Maintainability.....	53
2.7.8	A Word about Security.....	53
2.7.9	The Notion of Trust.....	54
2.8	Systems, Software, and Dependability.....	55
2.8.1	Computers Are neither Unsafe nor Insecure.....	55
2.8.2	Why Application System Dependability?.....	55
2.8.3	Application System Dependability and Computers.....	56
2.9	Defining Dependability Requirements.....	58
2.9.1	A First Example, an Automobile Cruise Control.....	60
2.9.2	A Second Example, a Pacemaker.....	61
2.10	As Low As is Reasonably Practicable ALARP.....	65
2.10.1	The Need for ALARP.....	65
2.10.2	The ALARP Concept.....	66
2.10.3	ALARP Carrot Diagrams.....	67
	Exercises.....	69
CHAPTER 3	Errors, Faults, and Hazards.....	73
3.1	Errors.....	73
3.2	The Complexity of Erroneous States.....	75
3.3	Faults and Dependability.....	76
3.3.1	Definition of Fault.....	76
3.3.2	Identifying Faults.....	78
3.3.3	Types of Fault.....	78
3.3.4	Achieving Dependability.....	78
3.4	The Manifestation of Faults.....	79
3.5	Degradation Faults.....	80
3.5.1	Degradation Fault Probabilities — The “Bathtub” Curve.....	80
3.5.2	An Example of Degradation Faults — Hard Disks.....	81
3.6	Design Faults.....	84
3.7	Byzantine Faults.....	85
3.7.1	The Concept.....	85
3.7.2	An Example Byzantine Fault.....	86

3.7.3 Nuances of Byzantine Faults	88
3.8 Component Failure Semantics	89
3.8.1 Disk Drive Example	89
3.8.2 Achieving Predictable Failure Semantics.....	90
3.8.3 Software Failure Semantics.....	90
3.9 Fundamental Principle of Dependability.....	91
3.9.1 Fault Avoidance.....	92
3.9.2 Fault Elimination	92
3.9.3 Fault Tolerance	92
3.9.4 Fault Forecasting	93
3.10 Anticipated Faults	93
3.11 Hazards.....	94
3.11.1 The Hazard Concept	94
3.11.2 Hazard Identification	95
3.11.3 Hazards and Faults.....	96
3.12 Engineering Dependable Systems.....	97
Exercises	100
CHAPTER 4 Dependability Analysis	103
4.1 Anticipating Faults	103
4.2 Generalizing the Notion of Hazard	104
4.3 Fault Tree Analysis	105
4.3.1 Basic Concept of a Fault Tree	105
4.3.2 Basic and Compound Events.....	106
4.3.3 Inspection of Fault Trees	108
4.3.4 Probabilistic Fault Tree Analysis.....	108
4.3.5 Software and Fault Trees	109
4.3.6 An Example Fault Tree.....	111
4.3.7 Defense in Depth	113
4.3.8 Other Applications of Fault Trees	116
4.4 Failure Modes, Effects, and Criticality Analysis	117
4.4.1 FMECA Concept	117
4.5 Hazard and Operability Analysis	119
4.5.1 The Concept of HazOp.....	119
4.5.2 The Basic HazOp Process.....	120
4.5.3 HazOp and Computer Systems.....	120
Exercises	122
CHAPTER 5 Dealing with Faults.....	123
5.1 Faults and Their Treatment	123
5.2 Fault Avoidance.....	124
5.2.1 Degradation Faults.....	124
5.2.2 Design Faults	125
5.3 Fault Elimination.....	126
5.3.1 Degradation Faults.....	126

5.3.2	Design Faults	126
5.4	Fault Tolerance	127
5.4.1	Familiarity with Fault Tolerance.....	127
5.4.2	Definitions.....	127
5.4.3	Semantics of Fault Tolerance.....	129
5.4.4	Phases of Fault Tolerance	130
5.4.5	An Example Fault-Tolerant System.....	131
5.5	Fault Forecasting	133
5.5.1	Fault Forecasting Process	134
5.5.2	The Operating Environment	134
5.5.3	Degradation Faults	135
5.5.4	Design Faults	135
5.6	Applying the Four Approaches to Fault Treatment.....	137
5.7	Dealing with Byzantine Faults	137
5.7.1	The Byzantine Generals.....	138
5.7.2	The Byzantine Generals and Computers.....	139
5.7.3	The Impossibility Result.....	141
5.7.4	Solutions to the Byzantine Generals Problem	143
	Exercises	145
CHAPTER 6	Degradation Faults and Software.....	147
6.1	Impact on Software	147
6.2	Redundancy	148
6.2.1	Redundancy and Replication	148
6.2.2	Large vs. Small Component Redundancy.....	151
6.2.3	Static vs. Dynamic Redundancy	152
6.3	Redundant Architectures	153
6.3.1	Dual Redundancy.....	155
6.3.2	Switched Dual Redundancy.....	158
6.3.3	N-Modular Redundancy.....	164
6.3.4	Hybrid Redundancy	166
6.4	Quantifying the Benefits of Redundancy	168
6.4.1	Statistical Independence.....	168
6.4.2	Dual-Redundant Architecture	169
6.5	Distributed Systems and Fail-Stop Computers.....	170
6.5.1	Distributed Systems	170
6.5.2	Failure Semantics of Computers.....	171
6.5.3	Exploiting Distributed Systems	172
6.5.4	The Fail-Stop Concept	172
6.5.5	Implementing Fail-Stop Computers.....	174
6.5.6	Programming Fail-Stop Computers	175
	Exercises	178
CHAPTER 7	Software Dependability	181
7.1	Faults and the Software Lifecycle	181
7.1.1	Software and Its Fragility.....	182

7.1.2	Dealing with Software Faults	183
7.1.3	The Software Lifecycle	184
7.1.4	Verification and Validation	185
7.2	Formal Techniques	186
7.2.1	Analysis in Software Engineering	186
7.2.2	Formal Specification.....	189
7.2.3	Formal Verification.....	189
7.2.4	The Terminology of Correctness	190
7.3	Verification by Model Checking	190
7.3.1	The Role of Model Checking	190
7.3.2	Analyzing Models.....	191
7.3.3	Using a Model Checker	192
7.4	Correctness by Construction	193
7.5	Approaches to Correctness by Construction	194
7.6	Correctness by Construction — Synthesis.....	197
7.6.1	Generating Code from Formal Specifications	197
7.6.2	The Advantages of Model-Based Development.....	198
7.6.3	Examples of Model-Based Development Systems.....	199
7.6.4	Mathworks Simulink®	200
7.7	Correctness by Construction — Refinement.....	201
7.8	Software Fault Avoidance	203
7.8.1	Rigorous Development Processes	204
7.8.2	Appropriate Notations	205
7.8.3	Comprehensive Standards for All Artifacts.....	206
7.8.4	Support Tools.....	207
7.8.5	Properly Trained Personnel	207
7.8.6	Formal Techniques.....	207
7.9	Software Fault Elimination	207
7.9.1	Static Analysis	208
7.9.2	Dynamic Analysis.....	209
7.9.3	Eliminating a Fault — Root-Cause Analysis	210
7.10	Managing Software Fault Avoidance and Elimination	212
7.10.1	Fault Freedom as Properties	212
7.11	Misconceptions about Software Dependability.....	215
	Exercises	218
CHAPTER 8	Software Fault Avoidance in Specification	221
8.1	The Role of Specification.....	221
8.2	Difficulties with Natural Languages	222
8.3	Specification Difficulties.....	223
8.3.1	Specification Defects.....	223
8.3.2	Specification Evolution	224
8.4	Formal Languages.....	226
8.4.1	Formal Syntax and Semantics	226
8.4.2	Benefits of Formal Languages.....	228

8.4.3	Presentation of Formal Languages.....	230
8.4.4	Types of Formal Languages.....	231
8.4.5	Discrete Mathematics and Formal Specification.....	232
8.4.6	The Before and After State.....	232
8.4.7	A Simple Specification Example.....	233
8.5	Model-Based Specification.....	234
8.5.1	Using a Model-Based Specification.....	235
8.6	The Declarative Language Z.....	237
8.6.1	Sets.....	237
8.6.2	Propositions and Predicates.....	238
8.6.3	Quantifiers.....	240
8.6.4	Cross Products.....	241
8.6.5	Relations, Sequences, and Functions.....	241
8.6.6	Schemas.....	242
8.6.7	The Schema Calculus.....	243
8.7	A Simple Example.....	244
8.8	A Detailed Example.....	245
8.8.1	Version 1 of the Example.....	247
8.8.2	Version 2 of the Example.....	248
8.8.3	Version 3 of the Example.....	248
8.8.4	Version 4 of the Example.....	251
8.9	Overview of Formal Specification Development.....	252
	Exercises.....	254
CHAPTER 9	Software Fault Avoidance in Implementation.....	257
9.1	Implementing Software.....	257
9.1.1	Tool Support for Software Implementation.....	258
9.1.2	Developing an Implementation.....	258
9.1.3	What Goes Wrong with Software?.....	259
9.2	Programming Languages.....	261
9.2.1	The C Programming Language.....	262
9.3	An Overview of Ada.....	264
9.3.1	The Motivation for Ada.....	264
9.3.2	Basic Features.....	265
9.3.3	Packages.....	268
9.3.4	Concurrent and Real-Time Programming.....	268
9.3.5	Separate Compilation.....	269
9.3.6	Exceptions.....	270
9.4	Programming Standards.....	270
9.4.1	Programming Standards and Programming Languages.....	270
9.4.2	Programming Standards and Fault Avoidance.....	272
9.5	Correctness by Construction — SPARK.....	273
9.5.1	The SPARK Development Concept.....	274
9.5.2	The SPARK Ada Subset.....	276
9.5.3	The SPARK Annotations.....	278
9.5.4	Core Annotations.....	278

9.5.5 Proof Annotations.....	281
9.5.6 Loop Invariants.....	283
9.5.7 The SPARK Tools.....	287
Exercises	289
CHAPTER 10 Software Fault Elimination.....	291
10.1 Why Fault Elimination?	291
10.2 Inspection	293
10.2.1 Artifacts and Defects	293
10.2.2 Fagan Inspections	295
10.2.3 Active Reviews.....	298
10.2.4 Phased Inspections.....	299
10.3 Testing	303
10.3.1 Exhaustive Testing.....	303
10.3.2 The Role of Testing	304
10.3.3 The Testing Process	305
10.3.4 Software Form	307
10.3.5 Output Checking.....	308
10.3.6 Test Adequacy	309
10.3.7 Modified Condition Decision Coverage.....	311
10.3.8 Test Automation	313
10.3.9 Real-Time Systems.....	314
Exercises	317
CHAPTER 11 Software Fault Tolerance.....	319
11.1 Components Subject to Design Faults	319
11.2 Issues with Design Fault Tolerance.....	321
11.2.1 The Difficulty of Tolerating Design Faults	321
11.2.2 Self-Healing Systems	323
11.2.3 Error Detection	324
11.2.4 Forward and Backward Error Recovery.....	324
11.3 Software Replication	326
11.4 Design Diversity.....	327
11.4.1 N-Version Systems	328
11.4.2 Recovery Blocks.....	331
11.4.3 Conversations and Dialogs	333
11.4.4 Measuring Design Diversity.....	334
11.4.5 Comparison Checking	335
11.4.6 The Consistent Comparison Problem	337
11.5 Data Diversity	339
11.5.1 Faults and Data	339
11.5.2 A Special Case of Data Diversity	340
11.5.3 Generalized Data Diversity	341
11.5.4 Data Reexpression	342
11.5.5 N-Copy Execution and Voting.....	343
11.6 Targeted Fault Tolerance	344

11.6.1 Safety Kernels.....	345
11.6.2 Application Isolation.....	347
11.6.3 Watchdog Timers	349
11.6.4 Exceptions.....	349
11.6.5 Execution Time Checking.....	351
Exercises	353
CHAPTER 12 Dependability Assessment.....	355
12.1 Approaches to Assessment.....	355
12.2 Quantitative Assessment	357
12.2.1 The Basic Approach.....	357
12.2.2 Life Testing	359
12.2.3 Compositional Modeling	360
12.2.4 Difficulties with Quantitative Assessment.....	361
12.3 Prescriptive Standards	362
12.3.1 The Goal of Prescriptive Standards	364
12.3.2 Example Prescriptive Standard — RTCA/DO-178B.....	364
12.3.3 The Advantages of Prescriptive Standards	369
12.3.4 The Disadvantages of Prescriptive Standards.....	370
12.4 Rigorous Arguments.....	371
12.4.1 The Concept of Argument.....	372
12.4.2 Safety Cases	373
12.4.3 Regulation Based on Safety Cases.....	375
12.4.4 Building a Safety Case.....	376
12.4.5 A Simple Example	377
12.4.6 The Goal Structuring Notation.....	380
12.4.7 Software and Arguments.....	382
12.4.8 Types of Evidence.....	385
12.4.9 Safety Case Patterns.....	387
12.5 Applicability of Argumentation	388
Exercises	391
Bibliography	393
Index	405
