# Contents


*Foreword* ix
*Preface* xi

**PART I DESCRIPTION AND SPECIFICATION** 1
*David Lorge Parnas, P.Eng*

**1 Introduction** *John McLean* 7

**Using Assertions About Traces to Write Abstract Specifications for Software Modules** 9
*Wolfram Bartussek and David L. Parnas*

1.1 Introduction 9
1.2 A Formal Notation for Specification Based on Traces 12
1.3 Some Simple Examples 15
1.4 Discussion of the Simple Examples 17
1.5 A Compressed History of the Development of an Abstract Specification 19
1.6 Conclusions 26

**2 Introduction** *William Wadge* 29

**Less Restrictive Constructs for Structured Programs** 31
*David L. Parnas and William Wadge*

2.1 Abstract 31
2.2 Introduction 31
2.3 The State of a Computing Machine 32
2.4 Programs 32
2.5 Program Specifications 32
2.6 Primitive Programs 33
2.7 Control Constructs and Constructed Programs 34
2.8 Defining the Semantics of Constructed Programs 34
2.9 The Value of a Program 34
2.10 The Syntax of the Constructs 34
2.11 Notation 35
2.12 Guard Semantics 35
2.13 The Semantics of a Limited Component 36
2.14 The Semantics of Limited Component Lists 36
2.15 The Semantics of "`;`" 36
2.16 The Semantics of "`stop`", "`go`" and "`init`" 36

2.17 Semantics of the Iterative Construct (`it ti`) 37
2.18 The Semantics of Parentheses 38
2.19 The Value of "#" 38
2.20 The Value Stack 39
2.21 Exits and Entrances 39
2.22 A Very Simple Example Done Three Ways 40
2.23 The DEED Problem 41
2.24 Conclusions 42

**3** **Introduction** *Martin van Emden* 49

**Predicate Logic for Software Engineering** 51
*David Lorge Parnas*
3.1 Abstract 51
3.2 Introduction 51
3.3 The Structure of This Paper 52
3.4 Comparison with Other Work 53
3.5 Basic Definitions 55
3.6 The Syntax of Logical Expressions 57
3.7 The Meaning of Logical Expressions 58
3.8 Examples of the Use of This Logic in Software Documentation 60
3.9 Conclusions 63

**4** **Introduction** *Joanne Atlee* 67

**Tabular Representations in Relational Documents** 71
*Ryszard Janicki, David Lorge Parnas, Jeffery Zucker*
4.1 Abstract 71
4.2 A Relational Model of Documentation 71
4.3 Industrial Experience with Relational Documentation 73
4.4 Why Use Tabular Representations of Relations? 75
4.5 Formalisation of a Wide Class of Tables 77
4.6 Transformations of Tables of One Kind to Another 82
4.7 Conclusions 85

**5** **Introduction** *Ali Mili* 89

**Precise Description and Specification of Software** 93
*D.L. Parnas*
5.1 Abstract 93
5.2 On Foundational Research 93
5.3 Language Is Not the Issue 94
5.4 A Polemic About Four Words 95
5.5 Four Types of Software Products 97
5.6 Programs and Executions 98
5.7 A Mathematical Interlude: LD-Relations 99
5.8 Program Construction Tools 99
5.9 Describing Programs 100

5.10 Specifying Programs 102
5.11 Objects Versus Programs 104
5.12 Descriptions and Specifications of Objects 104
5.13 Conclusions 105

**6** Introduction *Kathyrn Heninger Britton* 107

**Specifying Software Requirements for Complex Systems: New Techniques and Their Application** 111
*Kathryn L. Heninger*
6.1 Abstract 111
6.2 Introduction 111
6.3 A-7 Program Characteristics 112
6.4 Requirements Document Objectives 113
6.5 Requirements Document Design Principles 114
6.6 Techniques for Describing Hardware Interfaces 116
6.7 Techniques For Describing Software Functions 121
6.8 Techniques for Specifying Undesired Events 130
6.9 Techniques for Characterizing Types of Changes 131
6.10 Discussion 131
6.11 Conclusions 132

**PART II SOFTWARE DESIGN** 137
*David Lorge Parnas, P.Eng*

**7** Introduction *David M. Weiss* 143

**On the Criteria to Be Used in Decomposing Systems into Modules** 145
*D.L. Parnas*
7.1 Abstract 145
7.2 Introduction 145
7.3 A Brief Status Report 146
7.4 Expected Benefits of Modular Programming 146
7.5 What Is Modularization? 146
7.6 Example System 1: A KWIC Index Production System 146
7.7 Hierarchical Structure 153
7.8 Conclusions 154

**8** Introduction *Paul C. Clements* 157

**On a "Buzzword": Hierarchical Structure** 161
*David Parnas*
8.1 Abstract 161
8.2 Introduction 161
8.3 General Properties of All Uses of the Phrase "Hierarchical Structure" 161
8.4 Summary 168

**9** Introduction *Daniel Siewiorek* 171

Use of the Concept of Transparency in the Design of Hierarchically Structured Systems 173
*D.L. Parnas and D.P. Siewiorek*

9.1 Abstract 173
9.2 Introduction 173
9.3 The "Top Down" or "Outside In" Approach 173
9.4 "Transparency" of an Abstraction 175
9.5 Preliminary Example 175
9.6 "Register" for Markov Algorithm Machine 178
9.7 A Hardware Example 182
9.8 An Unsolved Transparency Problem from the Operating System Area 186
9.9 "Suggestive Transparency" 188
9.10 "Misleading Transparency" 188
9.11 Outside In and Bottom Up Procedures in Combination 189

**10** Introduction *Ralph Johnson* 191

On the Design and Development of Program Families 193
*David L. Parnas*

10.1 Abstract 193
10.2 Introduction 193
10.3 Motivation for Interest in Families 194
10.4 Classical Method of Producing Program Families 194
10.5 New Techniques 196
10.6 Representing the Intermediate Stages 197
10.7 Programming by Stepwise Refinement 198
10.8 Technique of Module Specification 200
10.9 Comparison Based on the KWIC Example 201
10.10 Comparative Remarks Based on Dijkstra's Prime Program 202
10.11 Comparative Remarks Based on an Operating System Problem 202
10.12 Design Decisions in Stage 1 204
10.13 Stage 3 205
10.14 How the Module Specifications Define a Family 208
10.15 Which Method to Use 209
10.16 Relation of the Question of Program Families to Program Generators 210
10.17 Conclusions 210
10.18 Historical Note 211

**11** Introduction *John Shore* 215

Abstract Types Defined as Classes of Variables 217
*D.L. Parnas, J.E. Shore, and D.M. Weiss*

11.1 Introduction 217
11.2 Previous Approaches 217
11.3 Motivations for Type Extensions 218

11.4 A New Approach 220
11.5 Applying These Concepts to Designing a Language 226

**12 Introduction** *Stuart Faulk* 229

**Response to Undesired Events in Software Systems** 231
*D.L. Parnas and H. Würges*
12.1 Abstract 231
12.2 Introduction 231
12.3 Difficulties Introduced by a "Leveled Structure" 233
12.4 The Effect of Undesired Events on Code Complexity 233
12.5 Impossible Abstractions 234
12.6 Error Types and Direction of Propogation 235
12.7 Continuation After UE "Handling" 236
12.8 Specifying the Error Indications 237
12.9 Redundancy and Efficiency 240
12.10 Degrees of Undesired Events 241
12.11 Examples 244
12.12 Conclusions 244
Appendix 12.A Annotated Example of Module Design in Light of Errors 247

**13 Introduction** *James Horning* 255

**Some Software Engineering Principles** 257
*David L. Parnas*
13.1 Abstract 257
13.2 Introduction 257
13.3 What Is a Well-Structured Program? 258
13.4 What Is a Module? 259
13.5 Two Techniques for Controlling the Structure of Systems Programs 260
13.6 Results 261
13.7 Error Handling 262
13.8 Hierarchical Structure and Subsetable Systems 263
13.9 Designing Abstract Interfaces 263
13.10 Conclusions 264

**14 Introduction** *Barry Boehm* 267

**Designing Software for Ease of Extension and Contraction** 269
*David L. Parnas*
14.1 Abstract 269
14.2 Introduction 269
14.3 Software as a Family of Programs 270
14.4 How Does the Lack of Subsets and Extensions Manifest Itself? 271
14.5 Steps Toward a Better Structure 273
14.6 Example: An Address-Processing Subsystem 279

14.7 Some Remarks on Operating Systems: Why Generals Are Superior to Colonels 286
14.8 Summation 286

**15** Introduction *James Waldo* 291

**A Procedure for Designing Abstract Interfaces for Device Interface Modules** 295
*Kathryn Heninger Britton, R. Alan Parker, David L. Parnas*
15.1 Abstract 295
15.2 Introduction 295
15.3 Objectives 296
15.4 Definitions 299
15.5 Design Approach 301
15.6 Design Problems 307
15.7 Summary 313

**16** Introduction *David M. Weiss* 315

**The Modular Structure of Complex Systems** 319
*D.L. Parnas, P.C. Clements, and D.M. Weiss*
16.1 Abstract 319
16.2 Introduction 319
16.3 Background and Guiding Principles 321
16.4 A-7E Module Structure 325
16.5 Conclusions 335

**17** Introduction *Kathryn Heninger Britton* 337

**Active Design Reviews: Principles and Practices** 339
*David L. Parnas and David M. Weiss*
17.1 Abstract 339
17.2 Introduction 339
17.3 Objectives of Design Reviews 340
17.4 Conventional Design Reviews 341
17.5 A More Effective Review Process 343
17.6 Conclusions 350

**18** Introduction *Barry Boehm* 353

**A Rational Design Process: How and Why to Fake It** 355
*David Lorge Parnas and Paul C. Clements*
18.1 Abstract 355
18.2 The Search for the Philosopher's Stone: Why Do We Want a Rational Design Process? 355
18.3 Why Will a Software Design "Process" Always Be an Idealization? 356
18.4 Why Is a Description of a Rational Idealized Process Useful Nonetheless? 357

18.5 What Should the Description of the Development Process Tell Us? 358
18.6 What Is the Rational Design Process? 358
18.7 What Is the Role of Documentation in This Process? 364
18.8 Faking the Ideal Process 366
18.9 Conclusion 367

**19** Introduction *A. John van Schouwen* 369

Inspection of Safety-Critical Software Using Program-Function Tables 371
*David Lorge Parnas*
19.1 Abstract 371
19.2 Introduction 371
19.3 Safety-Critical Software in the Darlington Nuclear Power Generating Station 373
19.4 Why Is Software Inspection Difficult? 374
19.5 Functional Documentation 375
19.6 Program-Function Tables 376
19.7 The Inspection Process 378
19.8 Hazard Analysis Using Functional Documentation 380
19.9 Conclusions 380

**PART III CONCURRENCY AND SCHEDULING** 383
*David Lorge Parnas, P.Eng*

**20** Introduction *Pierre-Jacques Courtois* 387

Concurrent Control with "Readers" and "Writers" 389
*P.J. Courtois, F. Heymans, and D.L. Parnas*
20.1 Abstract 389
20.2 Introduction 389
20.3 Problem 1 389
20.4 Problem 2 390
20.5 Final Remarks 391

**21** Introduction *Stuart Faulk* 393

On a Solution to the Cigarette Smoker's Problem (without conditional statements) 395
*D.L. Parnas*
21.1 Abstract 395
21.2 Introduction 395
21.3 Comments 397
21.4 On Patil's Proof 397
21.5 Patil's Result 397
21.6 On a Complication Arising from the Introduction of Semaphore Arrays 398

21.7 On the Yet Unsolved Problem 398
21.8 On More Powerful Primitives 399

**22 Introduction** *Stuart Faulk* 403

**On Synchronization in Hard-Real-Time Systems** 407
*Stuart R. Faulk and David L. Parnas*
22.1 Abstract 407
22.2 Introduction 407
22.3 The Need for a Separation of Concerns 408
22.4 A Two-Level Approach to Synchronization 410
22.5 Considerations at the Lower Level 410
22.6 The Lower-Level Synchronization Primitives 411
22.7 Considerations at the Upper Level 413
22.8 The STE Synchronization Mechanisms 418
22.9 Implementation in Terms of the Lower-Level Mechanism 426
22.10 The Pre-Run-Time Scheduler 428
22.11 Why Another Synchronization Mechanism? 430
22.12 Experience and Results 430
22.13 Summary 432

**23 Introduction** *Aloysius Mok* 437

**Scheduling Processes with Release Times, Deadlines, Precedence, and Exclusion Relations** 439
*Jia Xu and David Lorge Parnas*
23.1 Abstract 439
23.2 Introduction 439
23.3 Overview of the Algorithm 442
23.4 Notation and Definitions 444
23.5 How to Improve on a Valid Initial Solution 447
23.6 Searching for an Optimal or Feasible Solution 449
23.7 Empirical Behavior of the Algorithm 451
23.8 Conclusions 452
Appendix 23.A An Implementation of the Procedure for Computing a Valid Initial Solution 455
Appendix 23.B An Implementation of the Main Algorithm 457
Appendix 23.C Examples 1–5 460

**PART IV COMMENTARY** 467
*David Lorge Parnas, P.Eng*

**24 Introduction** *James Horning* 471

**Building Reliable Software in BLOWHARD** 473
*David L. Parnas*
24.1 Introduction 473
24.2 On "Building In" 473

24.3 Four Views of a Programming Language 474
24.4 Resolving Conflicts of Viewpoint in the Design of BLOWHARD 474
24.5 What Is BLOWHARD? 475
24.6 Why This Farce? 475

**25** Introduction *John Shore* 477

The Impact of Money-Free Computer Assisted Barter Systems 479
*David L. Parnas*
25.1 Introduction 479
25.2 Money Versus Barter as a Mechanism for Exchanging Our Current Goods and Services 480
25.3 Money Versus Barter for Future Sales? 484
25.4 What Would Barter Mean for Foreign Trade? 486
25.5 Are CABS a Dream or Are They Current Technology? 487
25.6 Turning Theory into Practice 488
25.7 What Would Be the Net Effect of the Use of CABS? 490
25.8 Can a Materialistic, "Rational", System Be Humane? 490
25.9 CABS and the Moral Illnesses in the Bishop's Report 491

**26** Introduction *David M. Weiss* 493

Software Aspects of Strategic Defense Systems 497
*David Lorge Parnas*
26.1 Abstract 497
26.2 Introduction 497
26.3 Why Software Is Unreliable 499
26.4 Why the SDI Software System Will Be Untrustworthy 501
26.5 Why Conventional Software Development Does Not Produce Reliable Programs 504
26.6 The Limits of Software Engineering Methods 506
26.7 Artificial Intelligence and the Strategic Defense Initiative 510
26.8 Can Automatic Programming Solve the SDI Software Problem? 512
26.9 Can Program Verification Make the SDI Software Reliable? 514
26.10 Is SDIO an Efficient Way to Fund Worthwhile Research? 516

**27** SDI: A Violation of Professional Responsibility 519
*David Lorge Parnas*
27.1 Introduction 519
27.2 SDI Background 520
27.3 The Role of Computers 522
27.4 My Decision to Act 523
27.5 Critical Issues 524
27.6 Broader Questions 528

**28** Introduction *Leonard L. Tripp* 533

The Professional Responsibilities of Software Engineers 537
*David Lorge Parnas*

28.1 Abstract 537
28.2 Personal Responsibility, Social Responsibility, and Professional Responsibility 537
28.3 The Social Responsibility of Scientists and Engineers 538
28.4 The Professional Responsibilities of Engineers 540
28.5 What Are the Obligations of the Engineer? 542
28.6 Professional Practice in Software Development 543
28.7 A Simple Example, Pacemakers 543
28.8 Other Concerns 545
28.9 The "Know How" Isn't There 546
28.10 How to Improve the Level of Professionalism in Software Development 546

**29** Introduction *Victor R. Basili* 549

Software Aging 551
*David Lorge Parnas*

29.1 Abstract 551
29.2 What Nonsense! 551
29.3 The Causes of Software Aging 552
29.4 Kidney Failure 553
29.5 The Costs of Software Aging 553
29.6 Reducing the Costs of Software Aging 554
29.7 Preventive Medicine 555
29.8 Software Geriatrics 559
29.9 Planning Ahead 562
29.10 Barriers to Progress 563
29.11 Conclusions for Our Profession 565

**30** Introduction *Richard Kemmerer* 569

On ICSE's "Most Influential" Papers 571
*David Lorge Parnas*

30.1 Background 571
30.2 What Are the Best Papers of Our Most Important Software Engineering Conference? 571
30.3 We Must Be Doing Something(s) Wrong! 572
30.4 We Need to Change Something 575
30.5 Conclusions 576

**31** Introduction *Daniel Hoffman* 577

Teaching Programming as Engineering 579
*David Lorge Parnas*
31.1 Introduction 579
31.2 Programming Courses and Engineering 579
31.3 The Important Characteristics of Programming Courses 580
31.4 The Role of Mathematics in Engineering 581
31.5 The Role of Programming in Engineering, Business, and Science 581
31.6 The Content of Most "Standard" Programming Courses 582
31.7 Programming Courses Are Not Science Courses 582
31.8 A New Approach to Teaching Programming 584
31.9 The Mathematics Needed for Professional Programming 584
31.10 Teaching Programming with This Mathematical Background 587
31.11 Experience 590
31.12 Conclusions 591

**32** Introduction *Victor R. Basili* 593

Software Engineering: An Unconsummated Marriage 595
*David Lorge Parnas*
32.1 Software Engineering Education 595

**33** Introduction *John Shore* 597

Who Taught Me About Software Engineering Research? 599
*David Lorge Parnas, P.Eng.*
33.1 Whom to Thank? 599
33.2 Everard M. Williams 599
33.3 Alan J. Perlis 601
33.4 Leo Aldo Finzi 602
33.5 Harlan D. Mills 603
33.6 Conclusions 605

**PART V BIBLIOGRAPHY** 607

Bibliography 609

Biographies 625

Credits 631

Index 635