## Contents

*Foreword* ix  
*Preface* xi

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

1. **Introduction**  
   *John McLean*  

#### Introduction

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*  

#### Introduction

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
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.17</td>
<td>2.17 Semantics of the Iterative Construct (it ti)</td>
<td>37</td>
</tr>
<tr>
<td>2.18</td>
<td>2.18 The Semantics of Parentheses</td>
<td>38</td>
</tr>
<tr>
<td>2.19</td>
<td>2.19 The Value of “#”</td>
<td>38</td>
</tr>
<tr>
<td>2.20</td>
<td>2.20 The Value Stack</td>
<td>39</td>
</tr>
<tr>
<td>2.21</td>
<td>2.21 Exits and Entrances</td>
<td>39</td>
</tr>
<tr>
<td>2.22</td>
<td>2.22 A Very Simple Example Done Three Ways</td>
<td>40</td>
</tr>
<tr>
<td>2.23</td>
<td>2.23 The DEED Problem</td>
<td>41</td>
</tr>
<tr>
<td>2.24</td>
<td>2.24 Conclusions</td>
<td>42</td>
</tr>
<tr>
<td>3</td>
<td>Introduction  Martin van Emden</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>3.1 Abstract</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>3.2 Introduction</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>3.3 The Structure of This Paper</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>3.4 Comparison with Other Work</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>3.5 Basic Definitions</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>3.6 The Syntax of Logical Expressions</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>3.7 The Meaning of Logical Expressions</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>3.8 Examples of the Use of This Logic in Software Documentation</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>3.9 Conclusions</td>
<td>63</td>
</tr>
<tr>
<td>4</td>
<td>Introduction  Joanne Atlee</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>4.1 Abstract</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>4.2 A Relational Model of Documentation</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>4.3 Industrial Experience with Relational Documentation</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>4.4 Why Use Tabular Representations of Relations?</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>4.5 Formalisation of a Wide Class of Tables</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>4.6 Transformations of Tables of One Kind to Another</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>4.7 Conclusions</td>
<td>85</td>
</tr>
<tr>
<td>5</td>
<td>Introduction  Ali Mili</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td>5.1 Abstract</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>5.2 On Foundational Research</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>5.3 Language Is Not the Issue</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>5.4 A Polemic About Four Words</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>5.5 Four Types of Software Products</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>5.6 Programs and Executions</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>5.7 A Mathematical Interlude: LD-Relations</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>5.8 Program Construction Tools</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>5.9 Describing Programs</td>
<td>100</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>5.10 Specifying Programs</td>
<td>102</td>
<td></td>
</tr>
<tr>
<td>5.11 Objects Versus Programs</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>5.12 Descriptions and Specifications of Objects</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>5.13 Conclusions</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td><strong>Introduction</strong> Kathyrn Heninger Britton</td>
<td>107</td>
<td></td>
</tr>
<tr>
<td>Specifying Software Requirements for Complex Systems:</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>New Techniques and Their Application</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Kathryn L. Heninger</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.1 Abstract</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>6.2 Introduction</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>6.3 A-7 Program Characteristics</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>6.4 Requirements Document Objectives</td>
<td>113</td>
<td></td>
</tr>
<tr>
<td>6.5 Requirements Document Design Principles</td>
<td>114</td>
<td></td>
</tr>
<tr>
<td>6.6 Techniques for Describing Hardware Interfaces</td>
<td>116</td>
<td></td>
</tr>
<tr>
<td>6.7 Techniques For Describing Software Functions</td>
<td>121</td>
<td></td>
</tr>
<tr>
<td>6.8 Techniques for Specifying Undesired Events</td>
<td>130</td>
<td></td>
</tr>
<tr>
<td>6.9 Techniques for Characterizing Types of Changes</td>
<td>131</td>
<td></td>
</tr>
<tr>
<td>6.10 Discussion</td>
<td>131</td>
<td></td>
</tr>
<tr>
<td>6.11 Conclusions</td>
<td>132</td>
<td></td>
</tr>
<tr>
<td><strong>PART II</strong> SOFTWARE DESIGN</td>
<td>137</td>
<td></td>
</tr>
<tr>
<td><em>David Lorge Parnas, P.Eng</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.1 Abstract</td>
<td>143</td>
<td></td>
</tr>
<tr>
<td>7.2 Introduction</td>
<td>143</td>
<td></td>
</tr>
<tr>
<td>7.3 A Brief Status Report</td>
<td>145</td>
<td></td>
</tr>
<tr>
<td>7.4 Expected Benefits of Modular Programming</td>
<td>146</td>
<td></td>
</tr>
<tr>
<td>7.5 What Is Modularization?</td>
<td>146</td>
<td></td>
</tr>
<tr>
<td>7.6 Example System 1: A KWIC Index Production System</td>
<td>146</td>
<td></td>
</tr>
<tr>
<td>7.7 Hierarchical Structure</td>
<td>153</td>
<td></td>
</tr>
<tr>
<td>7.8 Conclusions</td>
<td>154</td>
<td></td>
</tr>
<tr>
<td>8.1 Abstract</td>
<td>157</td>
<td></td>
</tr>
<tr>
<td>8.2 Introduction</td>
<td>157</td>
<td></td>
</tr>
<tr>
<td>8.3 General Properties of All Uses of the Phrase</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>“Hierarchical Structure”</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>8.4 Summary</td>
<td>168</td>
<td></td>
</tr>
</tbody>
</table>
9  Introduction  Daniel Siewiorek  
Use of the Concept of Transparency in the Design of Hierarchically Structured Systems  
D.L. Parnas and D.P. Siewiorek  
9.1 Abstract  
9.2 Introduction  
9.3 The “Top Down” or “Outside In” Approach  
9.4 “Transparency” of an Abstraction  
9.5 Preliminary Example  
9.6 “Register” for Markov Algorithm Machine  
9.7 A Hardware Example  
9.8 An Unsolved Transparency Problem from the Operating System Area  
9.9 “Suggestive Transparency”  
9.10 “Misleading Transparency”  
9.11 Outside In and Bottom Up Procedures in Combination  
10  Introduction  Ralph Johnson  
On the Design and Development of Program Families  
David L. Parnas  
10.1 Abstract  
10.2 Introduction  
10.3 Motivation for Interest in Families  
10.4 Classical Method of Producing Program Families  
10.5 New Techniques  
10.6 Representing the Intermediate Stages  
10.7 Programming by Stepwise Refinement  
10.8 Technique of Module Specification  
10.9 Comparison Based on the KWIC Example  
10.10 Comparative Remarks Based on Dijkstra’s Prime Program  
10.11 Comparative Remarks Based on an Operating System Problem  
10.12 Design Decisions in Stage 1  
10.13 Stage 3  
10.14 How the Module Specifications Define a Family  
10.15 Which Method to Use  
10.16 Relation of the Question of Program Families to Program Generators  
10.17 Conclusions  
10.18 Historical Note  
11  Introduction  John Shore  
Abstract Types Defined as Classes of Variables  
D.L. Parnas, J.E. Shore, and D.M. Weiss  
11.1 Introduction  
11.2 Previous Approaches  
11.3 Motivations for Type Extensions
### Contents

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**  
*D.L. Parnas and H. Würges* 231

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**  
*David L. Parnas* 257

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**  
*David L. Parnas* 269

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
19

Introduction  A. John van Schouwen

Inspection of Safety-Critical Software Using Program-Function Tables

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

David Lorge Parnas, P.Eng

20

Introduction  Pierre-Jacques Courtois

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

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
22 Introduction Stuart Faulk

On Synchronization in Hard-Real-Time Systems

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

Scheduling Processes with Release Times, Deadlines, Precedence, and Exclusion Relations

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

David Lorge Parnas, P.Eng

24 Introduction James Horning

Building Reliable Software in BLOWHARD

David L. Parnas

24.1 Introduction 473
24.2 On “Building In” 473
### 24.3 Four Views of a Programming Language

24.4 Resolving Conflicts of Viewpoint in the Design of BLOWHARD

24.5 What Is BLOWHARD?

24.6 Why This Farce?

### 25 Introduction  
*John Shore*

**The Impact of Money-Free Computer Assisted Barter Systems**  
*David L. Parnas*

25.1 Introduction

25.2 Money Versus Barter as a Mechanism for Exchanging Our Current Goods and Services

25.3 Money Versus Barter for Future Sales?

25.4 What Would Barter Mean for Foreign Trade?

25.5 Are CABS a Dream or Are They Current Technology?

25.6 Turning Theory into Practice

25.7 What Would Be the Net Effect of the Use of CABS?

25.8 Can a Materialistic, “Rational”, System Be Humane?

25.9 CABS and the Moral Illnesses in the Bishop’s Report

### 26 Introduction  
*David M. Weiss*

**Software Aspects of Strategic Defense Systems**  
*David Lorge Parnas*

26.1 Abstract

26.2 Introduction

26.3 Why Software Is Unreliable

26.4 Why the SDI Software System Will Be Untrustworthy

26.5 Why Conventional Software Development Does Not Produce Reliable Programs

26.6 The Limits of Software Engineering Methods

26.7 Artificial Intelligence and the Strategic Defense Initiative

26.8 Can Automatic Programming Solve the SDI Software Problem?

26.9 Can Program Verification Make the SDI Software Reliable?

26.10 Is SDIO an Efficient Way to Fund Worthwhile Research?

### 27 SDI: A Violation of Professional Responsibility  
*David Lorge Parnas*

27.1 Introduction

27.2 SDI Background

27.3 The Role of Computers

27.4 My Decision to Act

27.5 Critical Issues

27.6 Broader Questions
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