Contents

List of Contributors xv

I Introduction 1

1 The Object Paradigm 3
- 1.1 The Ubiquitous Object . . . 3
  - 1.1.1 Active Objects . . . 4
- 1.2 Defining the Object Concept . . . 5
- 1.3 Objects in Software . . . 6
- 1.4 Object Classes . . . 7
  - 1.4.1 Class Hierarchies and Inheritance of Properties . . . 9
  - 1.4.2 Multiple Inheritance . . . 10
- 1.5 Clientship . . . 11
  - 1.5.1 Polymorphic Substitutability . . . 12
  - 1.5.2 Abstract Classes and Dynamic Binding . . . 13
  - 1.5.3 Abstract Interfaces . . . 14
  - 1.5.4 Aggregation . . . 15
  - 1.5.5 Reuse . . . 15
- 1.6 Conclusion . . . 16

2 Formality in Object Technology 17
- 2.1 Formal Methods Concepts . . . 17
- 2.2 Formal Methods in Object Technology . . . 21
- 2.3 Object Technology in Formal Methods . . . 24
- 2.4 Formal Underpinnings . . . 26
- 2.5 Concurrency . . . 28
- 2.6 Conclusions . . . 29

II Formal Methods in Object Technology 31

3 LOTOS in the Object-Oriented Analysis Process 33
- 3.1 Introduction . . . 33
- 3.2 Reasons for Choosing LOTOS . . . 34

3.3 The ROOA Method 35
3.4 Structuring Specifications 41
3.4.1 Aggregation 41
3.4.2 Behavioural Inheritance 43
3.5 Conclusions 46

4 The Impact of Inheritance on Software Structure 47
4.1 Introduction 47
4.2 Multiple Inheritance Hierarchies 49
4.3 Type-Module Hierarchies 50
4.4 A Model Of Multiple Inheritance Hierarchies 51
4.4.1 Basic Definitions 51
4.4.2 Discussion: Assumptions in the Model 53
4.5 A Model of Type-Module Hierarchies 53
4.5.1 Basic Definitions 53
4.5.2 Discussion: From Inheritance to Module Hierarchies . . 55
4.5.3 Discussion: Language Issues 56
4.6 Relating Multiple Inheritance and Module Hierarchies 57
4.7 Summary & Conclusions 58

III Object Technology in Formal Methods 61

5 D_Parlog^{++}: Object-Oriented Logic Programming with Distributed Active Classes 63
5.1 Introduction 63
5.2 Informal Computation Model of D_Parlog^{++} 64
5.3 An Introductory Example 66
5.3.1 State Declaration and State Variables 68
5.3.2 Method Declaration 70
5.4 Communication Between Objects 73
5.4.1 Atomic Transactions 73
5.4.2 Self Communications 74
5.5 Inheritance and Delegation of Classes 77
5.5.1 Super Communication 81
5.5.2 Self Communication in an Inheritance Hierarchy 82
5.6 Concluding Remarks 84

6 Concurrency and Real-time in VDM^{++} 86
6.1 Concurrency and Object Structuring 86
6.1.1 Why Concurrency 86
6.2 The world model 87
6.3 Synchronisation and threads in VDM^{++} 88
6.4 Describing the behaviour of active objects 88
6.4.1 A new look at the Dining Philosophers 90
6.5 Concurrency and Contention 91

6.5.1 Declarative synchronization control (permission predicates) 92
6.5.2 Example of a Passive Class 95
6.5.3 A return to the Dining Philosophers 97
6.6 Other Ways of Expressing Synchronisation 99
6.6.1 Traces 99
6.7 Feasibility and the periodic obligation 100
6.8 Specifying Real-time systems 101
6.8.1 Real-time requirements engineering 101
6.9 Language facilities 102
6.9.1 The notion of time 102
6.10 Continuous models 104
6.10.1 Continuous time variables 104
6.11 Continuous classes 105
6.11.1 An example of a continuous model. 106
6.11.2 Subclasses of Component 107
6.12 The principle of discretising 109
6.12.1 Reasoning about sampling 112
6.13 Synchronising the components 112

7 Integrating Formal and Structured Methods in Object-Oriented System Development 113
7.1 B Abstract Machine Notation 114
7.2 Representation of Data Models 117
7.3 Representation of Dynamic Models 120
7.3.1 Process Model for Formalisation 120
7.4 Alternative State Representation 122
7.4.1 Comparison 122
7.4.2 Alarmclocks 123
7.5 Ship Load Planning 126
7.5.1 Results 145
7.6 Formalisation of Dynamic Models: A Lift System 146
7.6.1 From Analysis to Specification 149
7.6.2 Applications of Formalisation 156

8 Introducing Object-Oriented Concepts into a Net-Based Hierarchical Software Development Process 158
8.1 Introduction 158
8.2 Channel/Agency Nets 160
8.2.1 General Presentation 160
8.2.2 Techniques for the Use of CA Nets 161
8.3 General Object Concepts and Characteristics 162
8.4 The O-CA Net Model 164
8.4.1 Integration Methods 165
8.4.2 Description of Object Concepts in O-CA Nets 165
8.4.3 Structure Rules and Transformation Rules of O-CA Nets 170
8.5 O-CA Nets and PROOFS Method 172

8.5.1 Use of O-CA Nets 173
8.5.2 Link from O-CA Nets to PNs and CPNs 174
8.6 Example 175
8.7 Conclusion 179

IV Formal Foundations of Object Technology 181

9 Design Structures for Object-Based Systems 183
9.1 Introduction 183
9.2 Temporal Specification of Objects 185
9.3 Interconnecting Objects 190
9.4 Interfacing: Action Calling 193
9.5 Abstraction: Object Calling 198
9.6 Concluding Remarks 203

10 Interconnection of Object Specifications 205
10.1 Some Preliminaries 208
10.1.1 Categories 208
10.1.2 Functors and Natural Transformations 210
10.1.3 Limits 211
10.1.4 Monoids 213
10.1.5 Right Actions of Monoids 214
10.1.6 Sheaves 215
10.2 Process Classes 217
10.3 Object Specification 219
10.4 System Specification 222
10.5 Some Speculations 225

11 Refinement of Concurrent Object-Oriented Programs 227
11.1 Overview of FOOPS 228
11.1.1 Functional Level 228
11.1.2 Object Level 230
11.2 Refinement for FOOPS 233
11.2.1 Simulations 235
11.2.2 Refinement of States 239
11.2.3 Refinement of Expressions 240
11.2.4 Refinement of Specifications 241
11.3 Proving Refinement 242
11.3.1 Memory Cells 242
11.3.2 Buffers 248
11.3.3 Laws of FOOPS Method Combiners 256
11.4 Aspects Considered by Refinement 258
11.5 Conclusions 259

12 Static Typing for Object-Oriented Languages 262
12.1 Introduction 262
12.2 Subclases and Subtypes 263
12.2.1 Type vs. Class 263
12.2.2 Subtype vs. Subclass 264
12.2.3 ST&T: An Example 267
12.3 Types in ST&T 270
12.3.1 Some Subclass Relationships in Smaltalk 271
12.3.2 Object Types 271
12.3.3 The Subtype Relationship 272
12.3.4 Method Signatures 273
12.3.5 Receiver Type, Argument Type, Application of a Signature 274
12.3.6 Method Types 275
12.4 Type Rules in ST&T 275
12.4.1 The Environment 275
12.4.2 Type Checking Classes 276
12.4.3 Types of Assignments 277
12.4.4 Types of Return Expressions 277
12.4.5 Types of Message Expressions 277
12.5 Discussion, Implementation, Soundness 282
12.6 Conclusions and Further Work 285

13 A Note on the Semantics of Inclusion Polymorphism 287
13.1 The Concept of Inclusion Polymorphism 287
13.2 Formalization of Inclusion Polymorphism 288
13.2.1 Object Types, Classes and Theories 288
13.2.2 Behavioural Subtyping - Simple Cases 290
13.2.3 Behavioural Subtyping - Contravariance 292
13.2.4 Behavioural Subtyping - General Case 294
13.3 Inclusion Polymorphism in Practice 298
13.3.1 C++ 298
13.3.2 Eiffel 298
13.3.3 Smalltalk-80 300
13.3.4 Objective-C 300
13.4 Related Work 300
13.5 Conclusions 301

14 Categorical Semantics for Object-Oriented Data-Specifications 302
14.1 A Categorical Specification Mechanism 303
14.1.1 Classes and Dependencies 303
14.1.2 An Extended Example 304
14.1.3 Constraints 305
14.1.4 Attributes 306
14.2 Canonical Forms for Specifications 307
14.3 Verification of Specifications 308
14.4 A Note on Terminology 309

14.5 Specifications 309
14.5.1 Specifications without Attributes 309
14.5.2 Specifications with Attributes 310
14.5.3 Elimination of Attributes 311
14.6 Canonical Forms for Specifications 311
14.7 Verification of Specifications 313
14.7.1 Verification of the Underlying Category 314
14.7.2 Verification of Specifications 315
14.8 Conclusion 315

15 A Type-Theoretic Basis for an Object-Oriented Refinement Calculus 317
15.1 Introduction 317
15.2 Statements as Predicate Transformers 318
15.3 Objects as Packed Records 323
15.4 Using Abstract and Concrete Classes 329
15.5 Related Work and Conclusions 331

Bibliography 336

Index 357