# CONTENTS

## Preface

1 Introduction to UML

1.1 Models and modelling 1
1.2 Methodologies 5
1.3 The Unified Modeling Language 7
1.4 Design models and code 11
1.5 The software development process 12
1.6 Summary 12
1.7 Exercises 13

2 Modelling with Objects

2.1 The object model 14
2.2 Classes and objects 16
2.3 Object properties 19
2.4 Avoiding data replication 21
2.5 Links 21
2.6 Associations 24
2.7 Message passing 25
2.8 Polymorphism 27
2.9 Dynamic binding 32
2.10 The applicability of the object model 33
2.11 Summary 35
2.12 Exercises 36
### 3 Software Development Processes

3.1 The waterfall model 39
3.2 Alternatives to the waterfall model 43
3.3 The Unified Process 46
3.4 The role of models in development 47
3.5 The use of UML in the Unified Process 49
3.6 Summary 51
3.7 Exercises 52

### 4 Restaurant System: Business Modelling

4.1 Informal requirements 53
4.2 Use case modelling 55
4.3 Describing use cases 58
4.4 Structuring the use case model 61
4.5 Completing the use case model 65
4.6 Domain modelling 67
4.7 Glossaries 71
4.8 Summary 71
4.9 Exercises 72

### 5 Restaurant System: Analysis

5.1 The purpose of analysis 75
5.2 Object design 77
5.3 Software architecture 78
5.4 Use case realization 82
5.5 Recording new bookings 88
5.6 Cancelling bookings 90
5.7 Updating bookings 92
5.8 Completing the analysis model 94
5.9 Summary 94
5.10 Exercises 96

### 6 Restaurant System: Design

6.1 Receiving input from the user 98
6.2 Producing output 101
6.3 Persistent data storage 104
6.4 The design model 109
6.5 Detailed class design 109
6.6 Dynamic modelling of behaviour 111
6.7 A statechart for the booking system 113
6.8 A statechart for reservations 117
6.9 Summary 119
6.10 Exercises 119
7 Restaurant System: Implementation

7.1 Implementation diagrams
7.2 Implementation strategies
7.3 Application frameworks
7.4 The Java AWT framework
7.5 Implementation of classes
7.6 Implementation of associations
7.7 Implementation of operations
7.8 Summary
7.9 Exercises

8 Class and Object Diagrams

8.1 Data types
8.2 Classes
8.3 Describing objects with classes
8.4 Associations
8.5 Generalization and specialization
8.6 Inheritance of attributes and operations
8.7 Aggregation
8.8 Composition
8.9 Association classes
8.10 N-ary associations
8.11 Qualified associations
8.12 Interfaces
8.13 Templates
8.14 Summary
8.15 Exercises

9 Interaction Diagrams

9.1 Collaborations
9.2 Classifier roles
9.3 Association roles
9.4 Interaction diagrams
9.5 Object creation
9.6 Object destruction
9.7 Role multiplicity and iterated messages
9.8 Multiobjects
9.9 Conditional messages
9.10 Messages to self
9.11 Summary
9.12 Exercises
10  Statecharts  208
10.1 State-dependent behaviour  209
10.2 States, events and transitions  210
10.3 Initial and final states  211
10.4 Guard conditions  212
10.5 Actions  214
10.6 Activities  216
10.7 Composite states  218
10.8 History states  221
10.9 Summary of the CD player  222
10.10 Dynamic modelling in practice  223
10.11 Time events  228
10.12 Activity states  228
10.13 Summary of the ticket machine  229
10.14 Summary  229
10.15 Exercises  231

11  Component Diagrams  235
11.1 Dependencies  236
11.2 Components and artefacts  237
11.3 Component diagrams  239
11.4 Some common physical relationships  239
11.5 Compilation dependencies  241
11.6 Components and interfaces  245
11.7 Summary  246
11.8 Exercises  246

12  Constraints  248
12.1 Standard constraints  249
12.2 The Object Constraint Language  250
12.3 The context of a constraint  251
12.4 Navigation expressions  253
12.5 OCL data types and operations  256
12.6 Constraints  260
12.7 Stereotyped constraints  263
12.8 Constraints and generalization  266
12.9 Summary  267
12.10 Exercises  268

13  Implementation Strategies  270
13.1 Implementing associations  271
13.2 Unidirectional implementations  272
x CONTENTS

B.4 Model Types 344
B.5 Collections 344

Appendix C A Template for use case descriptions 348

References and Bibliography 350

Index 353
Mr Palomar’s rule had gradually altered: now he needed a great variety of models, perhaps interchangeable, in a combining process, in order to find the one that would best fit a reality that, for its own part, was always made of many different realities, in time and in space.

Italo Calvino

This book aims to provide a practical and accessible introduction to object-oriented design. It assumes that readers have prior knowledge of an object-oriented programming language, ideally Java, and explains both the principles and application of UML. It is aimed principally at students in the later years of an undergraduate or Masters course in Computer Science or Software Engineering, although I hope that other audiences will find the book useful.

The overall strategy of the book is to emphasize the connections between design notations and code. There are many excellent books available which discuss systems analysis and design with UML, but fewer that pay detailed attention to the final product, the code of the system being developed. UML is at bottom a language for expressing the designs of object-oriented programs, however, and it seems natural to consider the notation and meaning of the language from this perspective. I have also, over the years, found this to be a good way of imparting to students a concrete sense of what the design notations actually mean.

The book has two main objectives related to this overall philosophy. The first is to provide a complete example of an object-oriented development using UML, starting with a statement of requirements and finishing with complete executable code which can be run, modified and extended.
This objective of course places limits on the size of the example that can be considered. To get round this, the book takes as its paradigm system architecture a typical stand-alone desktop application, supporting a graphical user interface and interfacing to a relational database. Within this framework, the text examines the development of some core functionality, and leaves extensions of the system to be worked out as exercises.

The second objective is to provide a tutorial introduction to those aspects of UML that are important in developing this kind of application. Much emphasis is placed on clearly explaining the constructs and notation of the design language, and demonstrating the close relationship between the design and the implementation of object-oriented programs. These issues are treated rather superficially in many books. If they are not clearly understood, however, it is difficult to make correct use of UML.

UML is a large and complex language, and when one is learning it there is a danger of being overwhelmed by details of the notation. In order to avoid this, the book uses a subset of UML that is sufficient to develop desktop applications. The most significant omissions are any coverage of concurrency, activity diagrams and anything other than a brief mention of deployment diagrams. These aspects of the language are obviously important for ‘industrial-strength’ applications of UML, but these lie somewhat outside the experience of the intended audience of this book.

**STRUCTURE OF THE BOOK**

Following an introductory chapter, Chapter 2 introduces the basic concepts of object modelling in the context of a simple programming example. Chapters 3 to 7 contain a more extended case study of the use of UML, while Chapters 8 to 12 present the most important UML notations systematically. These two sections are independent of each other, allowing different reading strategies as shown in Figure P.1. Chapter 13 discusses strategies for implementing UML designs and Chapter 14 provides a general discussion of some of the underlying principles of object-oriented design.

![Figure P.1](image-url)  
**Figure P.1** Chapter dependencies
CHANGES IN THE SECOND EDITION

The most significant change in the second edition is that the diagram editor example has been replaced by a new case study based on a simple booking system for a restaurant. This provides an application with more ‘real-world’ context than the diagram editor, and is one which many students find easier to relate to. It also allows concepts of different architectural layers to be introduced more naturally than before, and these topics are now covered explicitly in Chapters 4 to 7.

Although the focus of the book is on language rather than process, it is impossible to divorce the two entirely in any practical approach. In the new Chapter 3, the book now includes an explicit discussion of some issues in the development of software processes and provides an outline sketch of the Unified Process.

The remaining chapters are very much the same as in the previous edition, though minor changes have been made throughout the book, both in content and presentation. To make room for the new chapter and case study, some material has had to be omitted from this edition, most noticeably the second case study. All the material that has been omitted, including the diagram editor example, will be made available from book’s website.

FURTHER RESOURCES

A web page for the book provides access to the source code for the examples used in the book, solutions to all exercises and material from the first edition. It can be found at the following URL:

http://www.mcgraw-hill.co.uk/textbooks/priestley

An instructor’s manual, including slides, the diagrams used in the book and additional exercises, is available to bona fide academics who have adopted the book for classroom use. Information on how to obtain the manual can be found on the publisher’s website.

ACKNOWLEDGEMENTS

In the preparation of this new edition, my most significant debt is to my students who have made use of the earlier editions, and also sat through earlier presentations of the restaurant booking system. I am indebted to Michael Richards for the initial idea for this case study.