PREFACE

> 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.

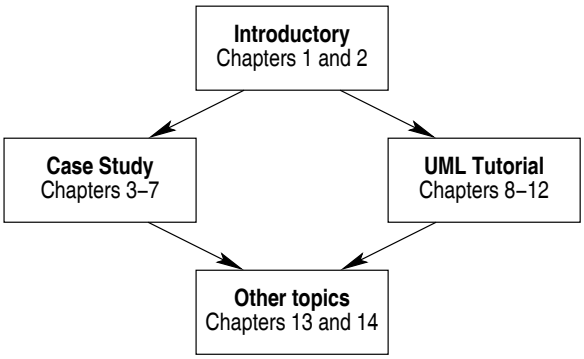

Figure P.1 Chapter dependencies

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.