PREFACE TO THE THIRD EDITION

He threw down his book, stretched his legs towards the embers in the grate, and clasped his hands at the back of his head, in that agreeable afterglow of excitement when thought lapses from examination of a specific object into a suffusive sense of its connexions with all the rest of our existence.

George Eliot (*Middlemarch*)

When the first edition of this book appeared in 1991, there was a great deal of excitement and interest in object-oriented techniques in the IT industry and in academia, with public seminars held almost weekly. New journals and regular conferences were being established, and the membership of special interest groups was growing rapidly. By the time the second edition came out interest had peaked and many commercial organizations had already had their first experiences of the technology for better or worse. Still the fuss continues and, while the languages have stabilized a little, the furore over methods and life-cycle issues has, if anything intensified even further – even though the emergence of the Unified Modelling Language (UML) has largely settled disputes over notation. Current concern now focuses on cross-enterprise application integration and component based development. Since the second edition appeared the field of object technology (OT) has, perhaps, trebled in its size and scope: published work and topics covered. It was a daunting prospect to begin to expand this volume, whose original aim had been to provide a comprehensive survey, without making the book excessively large. On the other hand the character of many of the new developments has been that of variations on themes that were around in 1994: the popularity of Jim Coplien's C++ idioms were the first inkling of the huge explosion of interest in design patterns; early object request brokers based on the Object Management Group (OMG) architecture model have matured and been put into production; new, better object-oriented programming languages have emerged; object-oriented databases are now in everyday commercial use – though still on a limited scale. Beside all this there has been a complete shake-out in the area of analysis and design methods. *Plus ça change, plus c'est la même chose*. The result has had to be a very substantial rewrite.

SUBJECT MATTER

This book is essentially a survey of the whole area of object technology. It covers object-oriented programming, object-oriented design, object-oriented analysis, object-oriented databases and concerns several related technologies. There are a number of good books on object technology covering specific languages and methods. More general coverage is provided in these books only incidentally. They give a high level overview of the philosophy and benefits of object-orientation in general but trouble the reader with a great deal of material specifically about programming and are dependent on the syntax of particular languages. At the other extreme there are now some very good 'management surveys' available, but these are not generally of sufficient technical depth for practitioners or students. The reader seeking a reasonably detailed understanding of those aspects of object-orientation not related to programming, has to turn to the research literature, conference proceedings, massive monographs or collections of highly technical papers. If such a reader wants to gain a general understanding of the whole field rapidly or evaluate the future rôle of object technology, there few coherent and comprehensive sources. My aims are therefore to address a gap in the literature in the following ways.

- Providing a single source, comprehensive, language-independent introduction covering all aspects of object technology from the perspectives of both the developer and management.
- Placing much more emphasis on the viewpoint of the conceptual modeller, compared to that of the programmer or designer; and upon the practical issues surrounding the use of object-oriented techniques in commercial environments.
- Propagating the view that object-orientation, artificial intelligence and data modelling together (rather than separately) are required to address the central issues of IT.
- Providing an introduction to and evaluation of object-oriented languages, middleware, databases and methods; and relating them each to conventional technology. In particular, providing the first concise explanation of the powerful Catalysis™ method for component based development (D'Souza and Wills, 1999)
- Attempting to explode some of the myths surrounding object technology while retaining a genuinely optimistic evaluation of its practical use.
- Supplying sufficient depth and reference material to guide students and practitioners entering the field.

The further objective of this book, which it shares with both previous editions, is to state in a clear manner the answers to the following questions.

- What are object-oriented methods?
- What are the benefits, pitfalls and likely costs?
- What languages, methods and tools are available, and how may they be evaluated?
- What has to be done to get started with adoption?

- What is the rôle of object-oriented analysis and design methods?
- How does one capture requirements for OO systems?
- How can object-orientation be managed?
- What special skills are required?
- What are the links to other areas of Information Technology (IT)?
- What are suitable applications?

Incidentally to the above aims the book exposes some of my own original work on object-oriented conceptual modelling using the idea of rulesets, requirements engineering and development process – collectively referred to as SOMA

Catalysis and SOMA. A new appendix summarises the UML notation. Chapter 7 is new and covers software architecture, patterns and component based development. Chapter 8 describes the SOMA approach to requirements engineering in detail. Chapter 9, on management, is substantially reorganized for greater clarity of exposition and to give a far more definite prescription of the recommended development process. It now includes guidance on user interface design. All other chapters and Appendix A have undergone slight revision and improvement to reflect new developments in the field and correct any errors which remained in the Second Edition and which I was aware of.

I have added exercises at the end of most chapters to assist the substantial number of educational users of whom I became gradually aware over recent years. Model answers, where possible, are given as an appendix; where I know the answers that is. The Bibliography is substantially expanded, to reflect the general growth in the volume of the literature as well as the new material in this edition, and the Glossary has been updated and improved.

Despite these drastic changes the essential purpose of the book remains unchanged and I hope it is merely a more comprehensive, detailed, up to date and accurate survey of object-oriented methods than it would have been without the alterations.

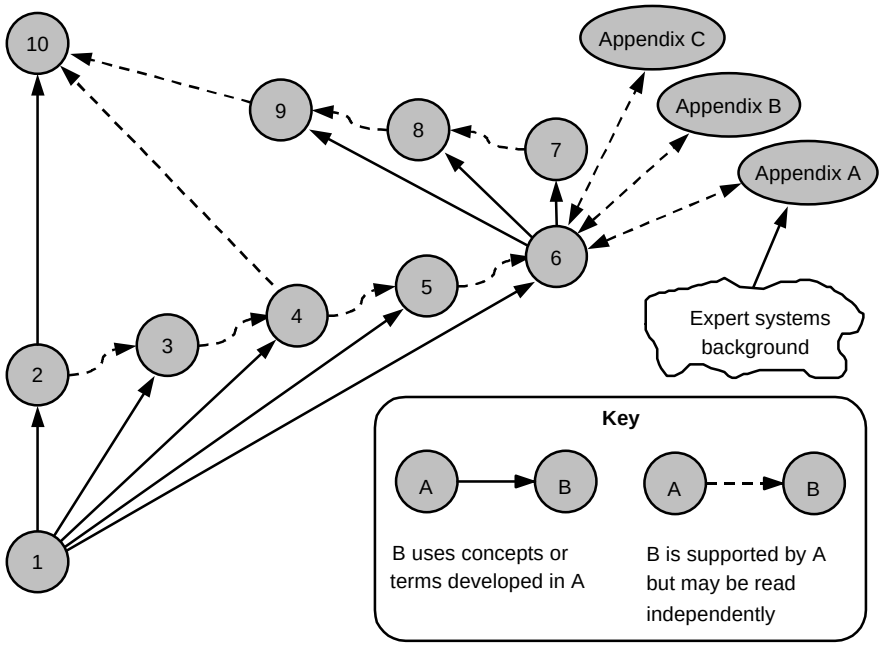
INTENDED READERSHIP

The book is intended to be accessible to readers who do not have deep specialist knowledge of theoretical computer science or advanced mathematics, but at the same time it attempts to treat the important issues accurately and in depth. It provides advice on how best to exploit object technology in practice.

The primary audience I had in mind for earlier editions was the IT and DP profession; software engineers and, generally, people who work with computers whether in user organizations, consultancies, universities or manufacturers. Although this is still the case with this edition, it has become clear that the book now has a loyal following in universities where the book is often used as a text for introductory undergraduate courses in object technology within an information technology or software engineering curriculum, perhaps complementing another course on object-oriented programming. It will therefore be of interest to teachers of Computer Science, Business Systems Analysis and possibly Artificial Intelligence. Researchers will be interested in the book as a survey and for the original contributions. They may also find some of the commentary scattered through the book thought provoking or even controversial. Managers and project planners will read it to gain an understanding of how the technology will affect their business practices and to be able to plan more effectively for change. Consultants, project managers, systems analysts and designers will read it to evaluate and stay abreast of the technology and, I hope, use the techniques explained in their day-to-day work. Programmers will read it to broaden their horizons in this area.

The material in this book, as it evolved, has been presented to very many audiences at various conferences, public seminars and in-house training courses.

READING MAP



There are a number of optional reading paths through the material presented in this book. The two prime routes through the text are illustrated in the reading map. Managers and those wishing for a high level view of the subject may take the high road through Chapters 1, 2 and 10. Project managers might also include Chapter 9. People interested in analysis methods must take the low road, as Chapters 6-8 build on all previous chapters. However, Chapter 1 should be read even by people familiar with object-oriented programming since it introduces terminology which may differ from that of other works, stressing – as I have indicated – a conceptual modelling viewpoint.

I hope the book tells a story if read sequentially. There are certainly some key themes to look out for. These are the differentiation of the viewpoints of the system designer from the conceptual modeller, the need for object-orientation to absorb techniques and ideas from other areas of computing and the need to add semantic richness to the more well-understood benefits of object-orientation: reusability and extensibility.

A NOTE ON LANGUAGE AND SPELLING

Since many of us are prone to strong opinions on language and spelling, and since some of my habitual usage has attracted comment from some referees of this and previous editions of this work, I feel that it is worthwhile clarifying the principles that I have applied in this respect in this text. Also, the conventions of spelling and the rules of transformational syntax vary between English, American English and other ‘Englishes’.

The principle behind the spelling is etymological. Thus many words ending -IZE are spelt that way (as is the American fashion), rather than the more usual modern English -ISE, when the Greek or Latin original indicates that the former is correct. This has to be largely based on guesswork, since I have never formally learned any foreign languages other than Chinese and German. However, I have discovered that a useful rule of thumb is to ask whether there can be an -IZATION. If not, as with ADVERTISE, CIRCUMCISE or DEVISE, then the 's' is correct. The only exception I can think of at the moment is IMPROVISATION. CONNEXION is spelt with an 'x' for similar etymological reasons (and because it was good enoth for Jane Austen). However, when words from foreign languages or quotations are used the original spelling is retained. Thus, CONNECTIONISM is used for the school of thought on neural nets, due to its American origin. Latin and other imported words are consistently italicized. All such words should be found in a good English dictionary. Bold characters are used for emphasis and definitions.

The word DATA is the plural of DATUM and is used as such throughout. I fail to see why writers on Computing should get away with their almost universal error of treating DATA as a singular when writers in no other discipline do so. This is nothing to do with the differences between American and English as a brief glance a books on Geography, Statistics, Medicine, Management and so on from both sides of the Atlantic will soon demonstrate. Words with two dictionary definitions such as BIMONTHLY and BILLION are proscribed.

The word METHODS is used in three different ways in this book:

- methods for doing software engineering in general (in the title);
- particular methods for doing software engineering and modelling (as with the Catalysis and SOMA methods); and
- methods (programs) representing the behaviour of objects.

I have resisted the non-word methodologies on the same grounds that I would reject Physicses, or Chemistries. The OED has it that Methodology (and the capitalization is necessary) is 'the science of method' and Chambers adds 'within a science or discipline'. The plural could only be used if there had been a paradigm shift in the sense of Thomas Kuhn (a solecism in itself[1]) in our perception of the way of doing things. Since many people believe that object technology is such a paradigm shift, we could correctly write the sentence: 'There is a difference between the Structured and the Object-Oriented Methodologies.'; meaning the that whole approach to computer science has altered. It is pure barbarism to write: 'There is a difference between the Yourdon and Jackson methodologies.' I try to avoid words like PARADIGM too.

Sexist assumptions in writing – as in life – should be avoided. My preference is for a plural personal pronoun or s/he (pronounced 'she') instead of the ugly and intrusive 'he or she'. However, I distinguish between sexual and grammatical gender; so that MAN is merely short for MANKIND and means the same as

[1] PARADIGM really means a model or especially typical example.

HUMANITY. Thus the term MAN-

Florence Froidevaux, Stuart Frost, Tim Gee, Thomas Grotehen, Brian Henderson-Sellers, Chris Harris-Jones, Benedict Heal, Tim Lamb, David Harvey, Jane Hillston, Michael Jackson, Margaret James, Peter Jones, Fiona Kinnear, David Lee, Chris Lees, Mark Lewis, Nick Lukic, Margaret Lyall, Neil Maiden, Sonia Math, Clive Menhinick, Sally Mortimore, Derek Pearce, Colin Prosser; Rob Radmore, Al-Noor Ramji, Dan Rawsthorne, David Redmond-Pyle, Elaine Richardson, René Schwarb, Richard Seed, Tony Simons, Richard Smith, Gail Swaffield, John Taylor, Brian Thal; Ulrika Thyssen, Richard Walker, Rose Watson, John Welch, David Welton, Kerry Williams, Benoit Xhenseval, Jennifer Yates, Marcus Zemp, BIS Applied Systems, BIS Banking Systems, Chase Manhattan Bank, De Montfort University, Eric Leach Marketing, Equitable Life, Swiss Bank Corporation (now UBS), TriReme International and the Expert Systems, OOPS and Requirements Engineering Specialist Groups of the British Computer Society. All the staff at Pearson Education demonstrated their usual splendid professionalism and especial thanks are due to my editor, Alison Birtwell.

Thanks go too to the various audiences who endured my seminars and classes during the period when these ideas were being developed and refined and who provided valuable feedback, and occasionally – and most usefully – asked hard questions. I would also like to thank the thousands of people scattered throughout the world, living and dead, whose labour, by hand or by brain, created the word processing system I am now using. Without it no edition of this book would ever have been written.

Lastly, it's about time that I came clean and acknowledged the contribution of The Grove: the hostelry where much of all my books has been written and corrected. It is a Victorian pub with an interior that deserves to be listed and nearly always empty enough to guarantee me a table to work at; which perhaps why some of the locals – who come from all walks of life from car mechanics to criminologists – call it The Grave. The other writers that rely on it tend to refer to The Balham Reading