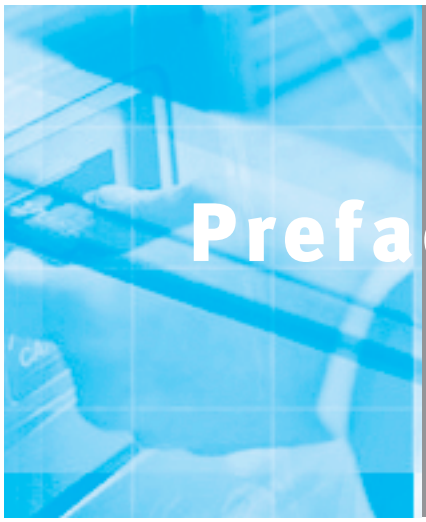

Preface

The first edition of this textbook on software engineering was published more than twenty years ago. That edition was written using a dumb terminal attached to an early minicomputer (a PDP-11) that probably cost about $50,000. I wrote this edition on a wireless laptop that cost less than $2,000 and is many times more powerful than that PDP-11. Software then was mostly mainframe software, but personal computers were just becoming available. None of us then realised how pervasive these would become and how much they would change the world.

Changes in hardware over the past twenty or so years have been absolutely remarkable, and it may appear that changes in software have been equally significant. Certainly, our ability to build large and complex systems has improved dramatically. Our national utilities and infrastructure—energy, communications and transport—rely on very complex and, largely, very reliable computer systems. For building business systems, there is an alphabet soup of technologies—J2EE, .NET, EJB, SAP, BPEL4WS, SOAP, CBSE—that allow large web-based applications to be deployed much more quickly than was possible in the past.

However, although much appears to have changed in the last two decades, when we look beyond the specific technologies to the fundamental processes of software engineering, much has stayed the same. We recognised twenty years ago that the waterfall model of the software process had serious problems, yet a survey published in December 2003 in *IEEE Software* showed that more than 40% of companies are still using this approach. Testing is still the dominant program validation technique, although other techniques such as inspections have been used more effectively since the mid 1970s. CASE tools, although now based around the UML, are still essentially diagram editors with some checking and code-generation functionality.

FM p.i-xxii 4/2/04 11:39 AM Page vi

Our current software engineering methods and techniques have made us much better at building large and complex systems than we were. However, there are still too many projects that are late, are over budget and do not deliver the software that meets the customer's needs. While I was writing this book, a government enquiry in the UK reported on the project to provide a national system to be used in courts that try relatively minor offenders. The cost of this system was estimated at £156 million and it was scheduled for delivery in 2001. In 2004, costs have escalated to £390 million and it is still not fully operational. There is, therefore, still a pressing need for software engineering education.

Over the past few years, the most significant developments in software engineering have been the emergence of the UML as a standard for object-oriented system description and the development of agile methods such as extreme programming. Agile methods are geared to rapid system development, explicitly involve the user in the development team, and reduce paperwork and bureaucracy in the software process. In spite of what some critics claim, I think these approaches embody good software engineering practice. They have a well-defined process, pay attention to system specification and user requirements, and have high quality standards.

However, this revision has not become a text on agile methods. Rather, I focus on the basic software engineering processes—specification, design, development, verification, and validation and management. You need to understand these processes and associated techniques to decide whether agile methods are the most appropriate development strategy for you and how to adapt and change methods to suit your particular situation. A pervasive theme of the book is critical systems—systems whose failure has severe consequences and where system dependability is critical. In each part of the book, I discuss specific software engineering techniques that are relevant to critical systems engineering.

Books inevitably reflect the opinions and prejudices of their authors. Some readers will disagree with my opinions and with my choice of material. Such disagreement is a healthy reflection of the diversity of the discipline and is essential for its evolution. Nevertheless, I hope that all software engineers and software engineering students can find something of interest here.

The structure of the book

The structure of the book is based around the fundamental software engineering processes. It is organised into six parts with several chapters in each part:

Part 1: Introduces software engineering, places it in a broader systems context and presents the notions of software engineering processes and management.

FM p.i-xxii 4/2/04 11:39 AM Page vii

Part 2: Covers the processes, techniques and deliverables that are associated with requirements engineering. It includes a discussion of software requirements, system modelling, formal specification and techniques for specifying dependability.

Part 3: This part is devoted to software design and design processes. Three out of the six chapters focus on the important topic of software architectures. Other topics include object-oriented design, real-time systems design and user interface design.

Part 4: Describes a number of approaches to development, including agile methods, software reuse, CBSE and critical systems development. Because change is now such a large part of development, I have integrated material on software evolution and maintenance into this part.

Part 5: Focuses on techniques for software verification and validation. It includes chapters on static V & V, testing and critical systems validation.

Part 6: The final part covers a range of management topics: managing people, cost estimation, quality management, process improvement and configuration management.

In the introduction to each part, I discuss the structure and organisation in more detail.

New chapters

Chap. 13: Application architectures
Chap. 17: Rapid software development
Chap. 19: Component-based software engineering
Chap. 21: Software evolution

Chapters with significant new material and/or major structural revisions

Chap. 2: Socio-technical systems (2)
Chap. 4: Software processes (3)
Chap. 7: Requirements engineering processes (6)
Chap. 9: Critical systems specification (17)
Chap. 12: Distributed systems architectures (11)
Chap. 16: User interface design (15)
Chap. 18: Software reuse (14)
Chap. 23: Software testing (20)
Chap. 25: Managing people (22)
Chap. 24: Critical systems validation (21)
Chap. 28: Process improvement (25)

Updated chapters

Chap. 1: Introduction (1)
Chap. 3: Critical systems (16)
Chap. 5: Project management (4)
Chap. 6: Software requirements (5)
Chap. 8: System models (7)
Chap. 10: Formal specification (9)
Chap. 11: Architectural design (10)
Chap. 14: Object-oriented design (12)
Chap. 15: Real-time systems design (13)
Chap. 20: Critical systems development (18)
Chap. 22: Verification and validation (19)

FM p.i-xxii 4/2/04 11:39 AM Page viii

	Chap. 26: Software cost estimation (23)
	Chap 27: Quality management (24)
	Chap. 29: Configuration management (29)
Deleted chapters	Software prototyping (8)
	Legacy systems (26)
	Software change (27)
	Software re-engineering (28)

Changes from the 6th edition

There are significant changes to the organisation and content from the previous edition. I have included four new chapters and made major revisions of 11 other chapters. All other chapters have been updated and, where appropriate, new material has been added. More and more systems have high availability and reliability requirements, and I believe that we have to consider dependability as a basic driver for software engineering, so the chapters on critical systems have now been integrated into other sections. To avoid content creep, I have reduced the amount of material on software maintenance and have integrated material on maintenance and evolution with other chapters in the book. There are two running case studies—one on a document management system used in a library and the other on a medical system—that I draw on in several chapters.

Table 1 summarises the changes, with the number in parentheses indicating the corresponding chapter in the 6th edition. Further information on the changes is available on the book's web site.

Readership

The book is aimed at students taking undergraduate and graduate courses and at software engineers in commerce and industry. It may be used in general software engineering courses or in courses such as advanced programming, software specification, and software design or management. Software engineers in industry may find the book useful as general reading and as a means of updating their knowledge on particular topics such as requirements engineering, architectural design, dependable systems development and process improvement. Wherever practicable, the examples in the text have been given a practical bias to reflect the type of applications that software engineers must develop.

FM p.i-xxii 4/2/04 11:39 AM Page ix

Using the book for teaching

I have designed the book so that it can be used in three types of software engineering course:

1. *General introductory courses in software engineering* For students who have no previous software engineering experience, you can start with the introductory section then pick and choose chapters from the other sections of the book. This will give students a general overview of the subject with the opportunity of more detailed study for those students who are interested. If the course's approach is project-based, the early chapters provide enough material to allow students to get started on projects, consulting later chapters for reference and further information as their work progresses.
2. *Introductory or intermediate courses on specific software engineering topics* The book supports courses in software requirements specification, software design, software engineering management, dependable systems development and software evolution. Each part can serve as a text in its own right for an introductory or intermediate course on that topic. As well as further reading associated with each chapter, I have also included information on other relevant papers and books on the web site.
3. *More advanced courses in specific software engineering topics* The chapters can form a foundation for a specific software course, but they must be supplemented with further reading that explores the topic in greater detail. For example, I teach an MSc module in systems engineering that relies on material here. I have included details of this course and a course on critical systems engineering on the web site.

The benefit of a general text like this is that it can be used in several related courses. At Lancaster, we use the text in an introductory software engineering course and in courses on specification, design and critical systems. Courses on component-based software engineering and systems engineering use the book along with additional papers that are distributed to students. Having a single text presents students with a consistent view of the subject— and they don't have to buy several books.

To reinforce the student's learning experience, I have included a glossary of key terms, with additional definitions on the web site. Furthermore, each chapter has:

- a clearly defined set of objectives set out on the first page;
- a list of key points covered in the chapter;
- suggested further reading—either books that are currently in print or easily available papers (lists of other suggested readings and links can be found on my web site);
- exercises, including design exercises.

FM p.i-xxii 4/2/04 11:39 AM Page x

The Software Engineering Body of Knowledge project (http://www.swebok.org) was established to define the key technical knowledge areas that are relevant to professional software engineers. These are organised under 10 headings: requirements, design, construction, testing, maintenance, configuration management, management, process, tools and methods, and quality. While it would be impossible to cover all of the knowledge areas proposed by the SWEBOK project in a single textbook, all of the top-level areas are discussed in this book.

Web pages

The web site that is associated with the book is:

http://www.software-engin.com

This offers a wide range of supplementary material on software engineering. From there, you can access web pages dedicated to supporting both this and previous editions of *Software Engineering*.

It has been my policy, both in the previous edition and in this edition, to keep the number of web links in the book to an absolute minimum. The reason for this is that these links are subject to change and, once printed, it is impossible to update them. Consequently, the book's web page includes a large number of links to resources and related material on software engineering. If you use these and find problems, please let me know and I will update the links.

To support the use of this book in software engineering courses, I have included a wide range of supplementary material on the web site. If you follow the Material for Instructors links, you can find:

- lecture presentations (PowerPoint and PDF) for all chapters in the book;
- class quiz questions for each chapter;
- case studies;
- project suggestions;
- course structure descriptions;
- suggestions for further reading and links to web resources for each chapter;
- solutions for a selection of the exercises associated with each chapter and for the quiz questions (instructor's only).

I welcome your constructive comments and suggestions about the book and the web site. You can contact me at ian@software-engin.com. I recommend that you include [SE7] in the subject of the e-mail message to ensure that my spam filters do not accidentally reject your mail. I regret that I do not have time to help stu-

FM p.i-xxii 4/2/04 11:39 AM Page xi

dents with their homework, so please do not ask me how to solve any of the problems in the book.

Acknowledgements

A large number of people have contributed over the years to the evolution of this book and I'd like to thank everyone (reviewers, students and book users who have e-mailed me) who has commented on previous editions and made constructive suggestions for change. The editorial and production staff at Pearson Education in England and the US were supportive and helpful, and produced the book in record time. So thanks to Keith Mansfield, Patty Mahtani, **<names of others in Boston>** for their help and support.

Finally, I'd like to thank my family, who tolerated my absence when the book was being written and my frustration when the words were not flowing. A big thank-you to my wife Anne and daughters Ali and Jane for their help and support.

Ian Sommerville,
February 2004