Preface

As I was writing the final chapters in this book in the summer of 2009, I realized that software engineering was forty years old. The name 'software engineering' was proposed in 1969 at a NATO conference to discuss software development problems: large software systems were late, did not deliver the functionality needed by their users, cost more than expected, and were unreliable. I did not attend that conference but, a year later, I wrote my first program and started my professional life in software.

Progress in software engineering has been remarkable over my professional lifetime. Our societies could not function without large professional software systems. For building business systems, there is an alphabet soup of technologies – J2EE, .NET, SaaS, SAP, BPEL4WS, SOAP, CBSE, etc. – that support the development and deployment of large enterprise applications. National utilities and infrastructure – energy, communications and transport – all rely on complex and mostly reliable computer systems. Software has allowed us to explore space and to create the World Wide Web – the most significant information system in the history of mankind.

Humanity is now faced with a new set of challenges – climate change and extreme weather, declining natural resources, an increasing world population to be fed and housed, international terrorism, and the need to help elderly people lead satisfying and fulfilled lives. We need new technologies to help us address these problems and, for sure, software will play a central role in these technologies.

Software engineering is, therefore, a critically important technology for the future of mankind. We must continue to educate software engineers and develop the discipline so that we can create more complex software systems. Of course, there are still problems with software projects. Software is still sometimes late and costs more than expected. However, we should not let these problems conceal the real successes in software engineering and the impressive software engineering methods and technologies that have been developed.

Software engineering is now such a huge area that it is impossible to cover the whole subject in one book. My focus, therefore, is on key topics that are fundamental to all software development processes and topics concerned with the development of reliable, distributed systems. There is an increased emphasis on agile methods and software reuse. I strongly believe that agile methods have their place but so too does 'traditional' plan-driven software engineering. We need to combine the best of these approaches to build better software systems.

Books inevitably reflect the opinions and prejudices of their authors. Some readers will inevitably disagree with my opinions and 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.

Integration with the Web

There is an incredible amount of information on software engineering available on the Web and some people have questioned if textbooks like this one are still needed. However, the quality of available information is very patchy, and information is sometimes presented badly or can be hard to find. Consequently, I believe that textbooks still have an important role to play in learning. They serve as a roadmap to the subject and allow information on method and techniques to be organized and presented in a coherent and readable way. They also provide a starting point for deeper exploration of the research literature and material available on the web.

I strongly believe that textbooks have a future but only if they are integrated with and add value to material on the web. This book has therefore been designed as a hybrid print/web text in which core information in the printed edition is linked to supplementary material on the web. Almost all chapters include specially written 'web sections' that add to the information in that chapter. There are also four 'web chapters' on topics that I have not covered in the print version of the book.

The web site that is associated with the book is:

http://www.SoftwareEngineering-9.com

The book's web has four principal components:

1. *Web sections* These are extra sections that add to the content presented in each chapter. These web sections are linked from breakout boxes in each chapter.
2. *Web chapters* There are four web chapters that cover formal methods, interaction design, documentation and application architectures. I may add other chapters on new topics during the lifetime of the book.
3. *Material for instructors* The material in this section is intended to support people who are teaching software engineering. See the 'Support Materials' section in this Preface.
4. *Case studies* These provide additional information about the case studies used in the book (insulin pump, mental health care system, wilderness weather system), as well as information about further case studies, such as the failure of the Ariane 5 launcher.

As well as these sections, there are also links to other sites with useful material on software engineering, further reading, blogs, newsletters, etc.

I welcome your constructive comments and suggestions about the book and the web site. You can contact me at ian@SoftwareEngineering-9.com. Please include [SE9]

in the subject of your message. Otherwise, my spam filters will probably reject your mail and you will not receive a reply.

Readership

The book is primarily aimed at university and college students taking introductory and advanced courses in software and systems engineering. Software engineers in industry may find the book useful as general reading and as a means of updating their knowledge on topics such as software reuse, architectural design, dependability and security and process improvement. I assume that readers have completed an introductory programming course and are familiar with programming terminology.

Changes from previous editions

This edition has retained the fundamental material on software engineering covered in previous editions but I have revised and updated all chapters and have included new material on many different topics. The most important changes are:

1. The move from a print-only book to a hybrid print/web book, with web material tightly integrated with sections in the book. This has allowed me to reduce the number of chapters in the book and focus on core material in each chapter.
2. Complete restructuring to make it easier to use the book for teaching. The book now has four rather than eight parts and each part may be used on its own or in combination with other parts as the basis of a software engineering course. The four parts are an introduction to software engineering, dependability and security, advanced software engineering and software engineering management.
3. Several topics from previous editions are presented more concisely in a single chapter, with extra material moved onto the Web.
4. Additional web chapters, based on chapters from previous editions that I have not included here, are available on the Web.
5. I have updated and revised the content in all chapters. I estimate that between 30% and 40% of the text has been completely rewritten.
6. I have added new chapters on agile software development and embedded systems.

7. As well as these new chapters, there is new material on model-driven engineering, open source development, test-driven development, Reason's Swiss Cheese model, dependable systems architectures, static analysis and model checking, COTS reuse, software as a service, and agile planning.
8. A new case study on a patient record system for patients who are undergoing treatment for mental health problems has been used in several chapters.

Using the book for teaching

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

1. *General introductory courses in software engineering*. The first part of the book has been designed explicitly to support a 1-semester course in introductory software engineering.
1. *Introductory or intermediate courses on specific software engineering topics.* You can create a range of more advanced courses using the chapters in parts 2-4. For example, I have taught a course in critical systems engineering using the chapters in Part 2 plus chapters on quality management and configuration management.
2. *More advanced courses in specific software engineering topics*. In this case, the chapters in the book form a foundation for the course. These are then supplemented with further reading that explores the topic in more detail. For example, a course on software reuse could be based around Chapters 16, 17, 18 and 19.

More information about using the book for teaching, including a comparison with previous editions, is available on the book's web site.

Support materials

A wide range of support material is available to help people using the book for teaching software engineering courses. This includes:

- PowerPoint presentations for all of the chapters in the book.
- Figures in PowerPoint.
- An instructor's guide that gives advice on how to use the book in different courses and explains the relationship between the chapters in this edition and previous editions.

- Further information on the book's case studies.
- Additional case studies that may be used in software engineering courses.
- Additional PowerPoint presentations on systems engineering.
- Four web chapters covering formal methods, interaction design, application architectures and documentation.

All of this material is available freely to readers of the book from the book's website or from the Pearson support site below. Additional information is available on a restricted basis to accredited instructors only:

- Model answers to selected end of chapter exercises.
- Quiz questions and answers for each chapter.

All support material, including restricted material, is available from:

http://www.pearsonhighered.com/sommerville

Instructors using the book for teaching can obtain a password to access restricted material by registering at the Pearson website by contacting their local Pearson representative or by requesting a password by email from computing@aw.com. Passwords are not available from the author.

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 commented on previous editions and made constructive suggestions for change.

I'd particularly like to thank my family (Anne, Ali and Jane) for their help and support while the book was being written. A big thank-you especially to my daughter Jane, who has discovered a talent for proofreading and editing. She was tremendously helpful in reading the entire book and did a great job spotting and fixing a large number of typos and grammatical errors.

Ian Sommerville,
October 2009.