IAN SOMMERVILLE

Software Engineering

6th Edition

Software Engineering

Sixth Edition

Ian Sommerville

Addison-Wesley

An imprint of PEARSON EDUCATION

Harlow, England · London · New York · Reading, Massachusetts · San Francisco · Toronto · Don Mills, Ontario · Sydney
Tokyo · Singapore · Hong Kong · Seoul · Taipei · Cape Town · Madrid · Mexico City · Amsterdam · Munich · Paris · Milan

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies around the world

Visit us on the World Wide Web at:
www.pearsoneduc.com

First published 1982
Second Edition 1984
Third Edition 1989
Fourth Edition 1992
Fifth Edition 1995
Sixth Edition 2001

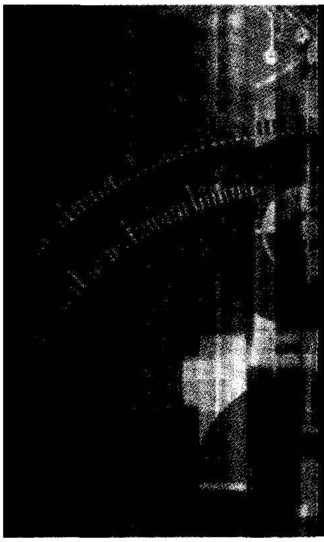
ISBN 0 201 39815 X

British Library Cataloguing-in-Publication Data
A catalogue record for this book can be obtained from the British Library

Library of Congress Cataloging-in-Publication Data
Sommerville, Ian, 1951–
Software engineering / Ian Sommerville. – 6th ed.
p. cm. – (International computer science series)
Includes bibliographical references and index.
ISBN 0-201-39815-X
1. Software engineering. I. Title. II. Series.
QA76.758. S.657 2000
005.1–dc21 00-033197

10 9 8 7 6 5 4 3 2
05 04 03 02 01

Typeset by 35 in 10/12.5pt Times
Printed and bound in the United States of America



Preface

Software systems are now ubiquitous. Virtually all electrical equipment now includes some kind of software; software is used to help run manufacturing industry, schools and universities, health care, finance and government; many people use software of different kinds for entertainment and education. The specification, development, management and evolution of these software systems make up the discipline of *software engineering*.

Even simple software systems have a high inherent complexity, so engineering principles have to be used in their development. Software engineering is therefore an engineering discipline where software engineers use methods and theory from computer science and apply this cost-effectively to solve difficult problems. These difficult problems have meant that many software development projects have not been successful. However, most modern software provides good service to its users; we should not let high-profile failures obscure the real successes of software engineers over the past 30 years.

Software engineering was developed in response to the problems of building large, custom software systems for defence, government and industrial applications. We now develop a much wider range of software, from games on specialised consoles through personal PC products and web-based systems to very large-scale distributed systems. Although some techniques that are appropriate for custom systems, such as object-oriented development, are universal, new software engineering techniques are evolving for different types of software. It is not possible to cover everything in one book, so I have concentrated on universal techniques and techniques for developing large-scale systems rather than individual software products.

Although the book is intended as a general introduction to software engineering, it is oriented towards my own interests in system requirements engineering and critical systems. I think these are particularly important for software engineering in the 21st century where the challenge we face is to ensure that our software meets the real needs of its users without causing damage to them or to the environment.

The approach that I take in this book is to present a broad perspective on software engineering and I don't concentrate on any specific methods or tools. I dislike zealots of any kind whether they are academics preaching the benefits of formal methods or salesmen trying to convince me that some tool or method is the answer to software development problems. There are no simple solutions to the problems of software engineering and we need a wide spectrum of tools and techniques to solve software engineering problems.

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

Changes from the fifth edition

Like many software systems, this book has grown and changed since its first edition was published in 1982. One of my goals in preparing this edition was to reduce rather than increase the size of the book and this has entailed some reorganisation and difficult decisions on what to cut out while still including important new material. The end result is a book that is about 10% shorter than the fifth edition.

- The book has been restructured into seven rather than eight parts covering an introduction to software engineering, specification, design, critical systems development, verification and validation, management, and software evolution.
- There are new chapters covering software processes, distributed systems architectures, dependability and legacy systems. The section on formal specification has been cut to a single chapter and material on CASE has been reduced and distributed to different chapters. Coverage of functional design is now included in the new chapter on legacy systems. Chapters on verification and validation have been amalgamated.
- All chapters have been updated and several chapters have been extensively rewritten. Reuse now focuses on development with reuse, with material on patterns and component-based development; object-oriented design has more of a process focus; the chapters on requirements have been separated into chapters on the requirements themselves and chapters on the requirements engineering process; cost estimation has been updated to COCOMO 2.
- The introductory part now includes four chapters. I have taken introductory material that was distributed throughout the book in the fifth edition and covered

it all in this part. Chapter 1 has been completely rewritten as a set of frequently asked questions about software engineering.

- The material on critical systems has been restructured and integrated so that reliability, safety and availability are not covered as separate topics. I have introduced some material on security as an attribute of a critical system.
- Program examples are now in Java and object models are described in the UML. Ada and C++ examples have been removed from the text but are available from my web site.

The further reading associated with each chapter has been updated from previous editions. However, in many cases, articles written in the 1980s are still the best introduction to some topics.

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, software design or management. Practitioners 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 which software engineers must develop.

I assume that readers have a basic familiarity with programming and modern computer systems and knowledge of basic data structures such as stacks, lists and queues.

Using the book as a course text

There are three main types of software engineering courses where this book can be used:

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 the chapters from the different 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.

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 of the parts in the book can serve as a text in its own right for an introductory or intermediate course on that topic. Some additional reading is suggested for these courses.

3. *More advanced courses in specific software engineering topics* In this case, the chapters in the book form a foundation for the course which must be supplemented with further reading which explores the topic in more detail. All chapters include my suggestions for further reading and additional reading is suggested on my web site.

The benefit of a general text like this is that it can be used in several different related courses. At Lancaster, we use the text in an introductory software engineering course, in courses on specification, design and critical systems and in a software management course where it is supplemented with further reading. With a single text, students are presented with a consistent view of the subject. They also like the extensive coverage because they don't have to buy several different books.

This book covers all suggested material in the SE Software Engineering component of the draft computer science body of knowledge proposed by the ACM/IEEE in the Computing Curricula 2001 document. The book is also consistent with the forthcoming IEEE/ACM 'Software Engineering Body of Knowledge' document which is due for publication sometime in 2000 or 2001.

Web site

My web site is http://www.software-engin.com and this includes links to material to support the use of this book in teaching and personal study. The following downloadable supplements are available:

- An instructor's guide including hints on teaching using the book, class and term project suggestions, case studies and examples and some solutions to the exercises. This is available in Adobe PDF format.
- A set of overhead projector transparencies for each chapter. These are available in Adobe PDF and in Microsoft PowerPoint format. Instructors may adapt and modify the presentations as they wish.
- Source code in Java for most of the individual program examples, including supplementary code required for compilation.

- Additional material based on chapters from previous editions on algebraic specification, Z and function-oriented design. Ada and C++ examples as used in the fifth edition are also available.

This page also includes links to copies of slides and papers on systems engineering, links to other software engineering sites, information on other books and suggestions for additional further reading.

I am always pleased to receive feedback on my books and you can contact me by e-mail at ian@software-engin.com. However, I regret that I don't have time to give advice to individual students on their homework.

Acknowledgements

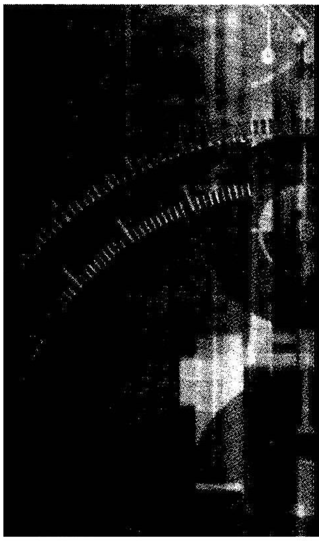
A large number of people have contributed over the years to the evolution of this book and I'd first like to thank everyone who has commented on previous editions and made suggestions for change. I am grateful to the reviewers of initial drafts of this text for their helpful comments and suggestions which helped me a great deal when completing the final version.

The reviewers of the first draft were Andy Gillies and Lindsey Gillies of the University of the West of England, Joe Lambert of Penn. State University, Frank Maddix of the University of the West of England, Nancy Mead of the Software Engineering Institute, Pittsburgh, Chris Price of the University of Wales, Aberystwyth, Gregg Rothermel of Oregon State University and Guus Schreiber of the University of Amsterdam. I'd particularly like to thank my friends Ron Morrison of St Andrews University and Ray Welland of Glasgow University who have reviewed previous editions and again volunteered to review this text.

Finally, my family has put up with my absence for more evenings than I like to think while I finished this book. Thanks to my wife Anne and my daughters Ali and Jane for their coffee and tolerance.

Ian Sommerville
Lancaster, February 2000

Contents at a glance



Contents