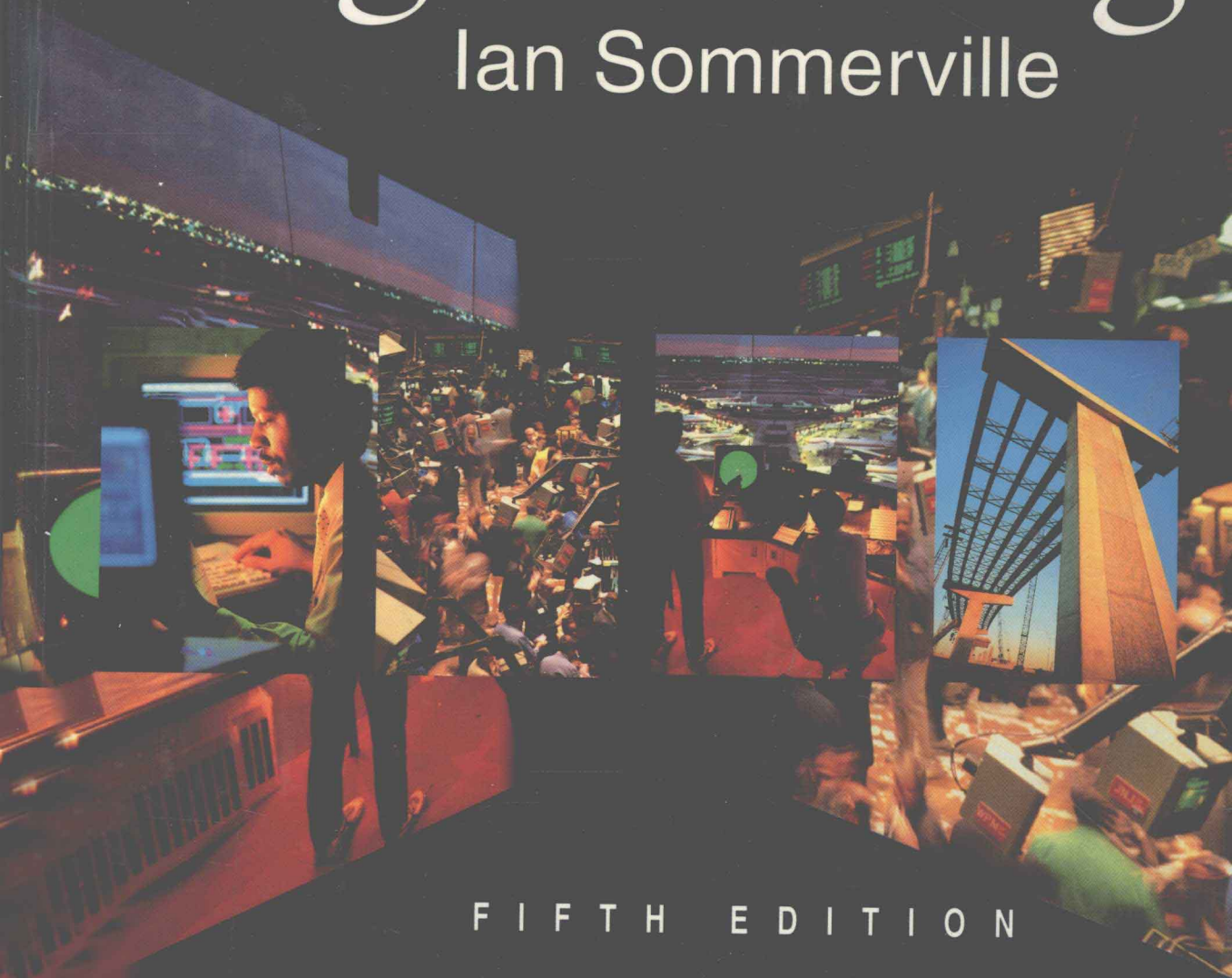
Software Engineering

Ian Sommerville



FIFTH EDITION

ADDISON-WESLEY

FIFTH EDITION

Software Engineering

Ian Sommerville
Lancaster University

ADDISON-WESLEY

Harlow, England • Reading, Massachusetts • Menlo Park, California • New York
Don Mills, Ontario • Amsterdam • Bonn • Sydney • Singapore
Tokyo • Madrid • San Juan • Milan • Mexico City • Seoul • Taipei

Addison Wesley Longman Limited
Edinburgh Gate
Harlow
Essex, CM20 2JE
England

Cover designed by Designers & Partners of Oxford
Typeset by Meridian Colour Repro Limited, Pangbourne
Printed in the United States of America

First edition published 1982. Reprinted 1983 and 1984.
Second edition published 1984. Reprinted 1985, 1986, 1987 and 1988.
Third edition published 1989. Reprinted 1989, 1990 (twice) and 1991.
Fourth edition published 1992. Reprinted 1993 and 1994.
Fifth edition printed 1995. Reprinted 1996 (twice). Reprinted 1997 and 1998 (twice).

ISBN 0-201-42765-6

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

Library of Congress Cataloging-in-Publication Data
Sommerville, Ian.
Software engineering / Ian Sommerville. -- 5th ed.
p. cm. -- (International computer science series)
Includes bibliographical references and index.
ISBN 0-201-42765-6 (alk. paper)
1. Software engineering. I. Title. II. Series.
QA76.758.S657 1996
005. 1--dc20 95-38788
CIP

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 now 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.

Books inevitably reflect the opinions and prejudices of their authors. Some readers will inevitably disagree with my opinions and with the choice of material which I include. 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.

Although the book is intended as a general introduction to software engineering, it is biased, to some extent, 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.

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. I therefore don't

describe commercial design methods or CASE systems but paint a broad picture of software engineering methods and tools.

Software engineering research has made tremendous strides over the past 15 years but there has been a relatively slow diffusion of this research into industrial practice. The principal challenge which we now face is not the development of new techniques and methods but the transfer of advanced software engineering research into everyday use. I see this book as a contributor to this process. I therefore discuss some techniques, such as viewpoints for requirements engineering, which are reasonably well developed but which are not yet widely used in industry.

Finally, it is impossible to over-emphasize the importance of people in the software engineering process. People specify, design and implement systems which help other people with their work. Most of the difficulties of very large system engineering are not technical problems but are the problems of managing large numbers of people with diverse priorities, abilities and interests. Software engineering techniques and tools are only effective when applied in a context which respects these different skills and abilities.

Changes from the fourth edition

Like many software systems, this book has grown and changed since its first edition was published in 1982. This latest edition started as a relatively minor update of the fourth edition but, in the course of writing the book, I decided that more significant revision and re-engineering was necessary. Although much of the material in the fourth edition has been retained, the following changes have been made:

- There are five completely new chapters covering computer-based system engineering, requirements analysis, architectural design, process improvement and software re-engineering.
- The book has been restructured into eight parts covering an introduction to software engineering, requirements and specification, design, dependable systems development, verification and validation, CASE, management, and software evolution.
- There have been radical revisions of the material on requirements engineering, object-oriented and functional design, and CASE.
- Project management is introduced in the first part of the book then covered in more detail in a separate section which incorporates previous material on human factors. There is more emphasis on quality management.

In previous editions, I have presented program examples in Ada as I consider this an excellent language for large-scale software engineering. However, Ada has not become as widely used as was once predicted. C or C++ are the programming languages of choice for most personal computer and workstation applications. Because of this wide use, I have included C++ as well as Ada versions of most of the program examples in the book. For safety-critical systems, however, I think

it unwise to use a language which includes potentially unsafe constructs. Those examples are, therefore, only presented in Ada.

I considered for a long time whether it would be appropriate to include a new chapter on professional and ethical issues. I decided not to do so because the topic is so subjective that it is difficult to present in a balanced way in a single chapter. There are no absolutes in this area and it is best addressed in an interactive context rather than as a chapter of a book. However, I have included a brief discussion of these issues in the introduction to the book. I have also included possible ethical and professional topics for discussion as exercises in many chapters. Links to WWW pages on this topic are included in the Web page whose URL is given below.

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. As new articles which are useful become available, I will include them on the Web page. The author index in previous editions has been removed. Rather, each entry in the References section includes the page numbers where it has been referenced.

Readership

The book is aimed at students in 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. Some examples rely on knowledge of basic data structures such as stacks, lists and queues. The chapters on formal specification assume knowledge of very elementary set theory. No other mathematical background is required.

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 introductory 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 topics in software engineering such as software specification, design or dependable systems development. 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.

(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.

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 Units SE2 to SE5 in the ACM/IEEE 1991 Curriculum. It also includes material to supplement an introductory programming text which would normally cover Unit SE1 and all material in the suggested course entitled 'Advanced Software Engineering'.

Supplements

The following supplements are available:

- An instructor's guide including hints on teaching the material in each chapter, class and term project suggestions, and solutions to some of the exercises. This is available in Postscript or on paper from Addison-Wesley.
- A set of overhead projector transparencies for each chapter. These are available in Postscript and in Microsoft Powerpoint format.
- Source code for most of the individual program examples including supplementary code required for compilation.
- An introduction to the Ada programming language.
- Information on course presentation using electronically mediated communication and links to material for that approach to teaching.

These are available, free of charge, over the Internet at URL:

http://www.comp.lancs.ac.uk/computing/resources/ser/

This page also includes links to other software engineering resources which you may find useful. If you have any problems, you can contact me by E-mail (is@comp.lancs.ac.uk).

Acknowledgements

I am indebted to a number of reviewers who provided helpful and constructive criticism of early drafts of this book. Many thanks to Leonor Barocca of the Open University, Stewart Green of the University of the West of England, Andrew McGettrick of the University of Strathclyde, Philip Morrow of the University of Ulster and Ray Welland of the University of Glasgow. Thanks also to Rodney L. Bown, University of Houston-Clear Lake, Charles P. Howerton, Metropolitan State College of Denver, Josephine DeGuzman Mendoza of California State University, San Bernardino and David C. Rine of George Mason University.

Thanks also to all users of previous editions who have provided me with comments and constructive criticism and to my colleagues in the Cooperative Systems Engineering Group and Lancaster University.

Finally, a big thank-you to my wife Anne and daughters, Ali and Jay. They have provided coffee, encouragement and occasional inspiration during the long hours I spent writing this book.

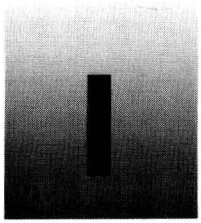
Ian Sommerville
August 1995

Contents

Part One
Introduction

The chapters in this introductory part introduce the topic of software engineering and place it in the context of a system engineering process. They emphasize that software engineering is a managed process by including discussions of software and system engineering process models and a short introduction to fundamentals of project management. Project management is also discussed in more detail later in Part 7.

Contents



Introduction

Objectives

- To define software engineering and to explain why it is important.
- To introduce the concept of a software product and the attributes of well-engineered software.
- To describe the basic activities of the software engineering process and to illustrate a number of generic software process models.
- To explain why software process visibility is essential for process management.
- To explain why software engineers must consider their responsibilities to the engineering profession.

Contents