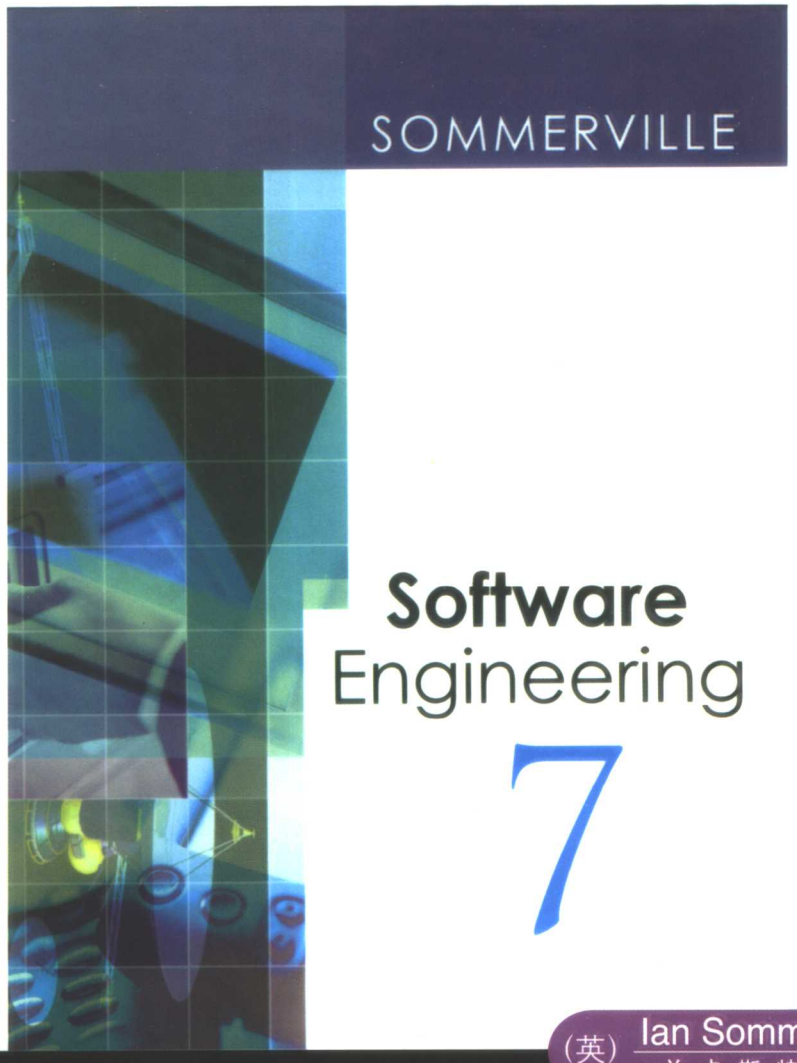
HZ BOOKS

经 典 原 版 书 库

PEARSON Addison Wesley

软件工程

（英文版 · 第7版）



(英) Ian Sommerville 著
兰卡斯特大学

机械工业出版社
China Machine Press

经 典 原 版 书 库

软件工程

（英文版 · 第7版）

Software Engineering

(Seventh Edition)

(英) Ian Sommerville
兰卡斯特大学 著

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2004-5731

图书在版编目（CIP）数据

软件工程（英文版·第7版）/（英）萨默维尔（Sommerville, I.）著. -北京：机械工业出版社，2004.11

（经典原版书库）

书名原文：Software Engineering, Seventh Edition

ISBN 7-111-15309-X

Ⅰ.软… Ⅱ.萨… Ⅲ.软件工程-英文 Ⅳ.TP311.5

中国版本图书馆CIP数据核字（2004）第098879号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：迟振春

北京中兴印刷有限公司印刷·新华书店北京发行所发行

2004年11月第1版第1次印刷

787mm×1092mm 1/16·49.25印张

印数：0 001-3 000册

定价：75.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专诚为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国

家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzedu@hzbook.com
联系电话：（010）68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

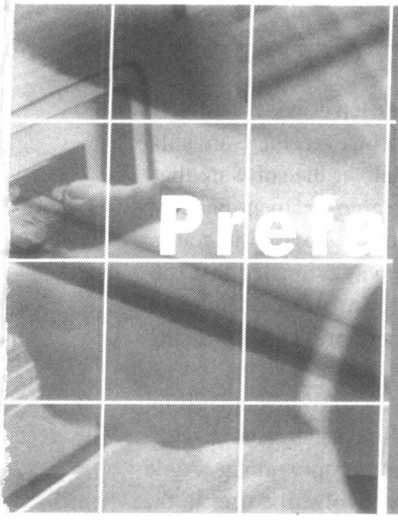
专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周立柱	周克定	周傲英	孟小峰	岳丽华
范　明	郑国梁	施伯乐	钟玉琢	唐世渭
袁崇义	高传善	梅　宏	程　旭	程时端
谢希仁	裘宗燕	戴　葵		

秘　书　组

武卫东　　温莉芳　　刘　江　　杨海玲



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.

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.

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.

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.

The case study material is indicated by margin icons. 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.

Table 1 Chapter revisons

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)
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)

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.

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.

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-

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, Daniel Rausch, Carol Noble, and Sharon Burkhardt 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