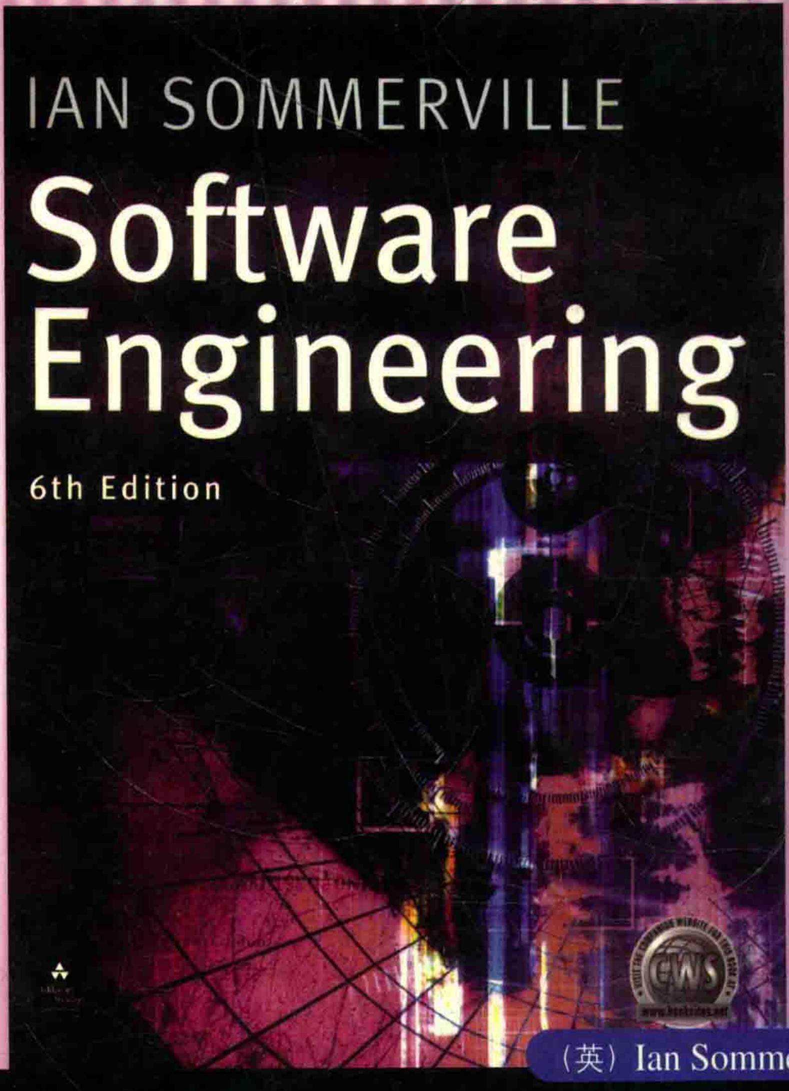
经 典 原 版 书 库

软件工程

（英文版 · 第6版）



IAN SOMMERVILLE

Software Engineering

6th Edition

（英）Ian Sommerville 著

机械工业出版社
China Machine Press

经 典 原 版 书 库

软件工程

（英文版 · 第6版）

Software Engineering

(6th Edition)

（英）Ian Sommerville 著

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2003-1010

图书在版编目（CIP）数据

软件工程（英文版·第6版）/（英）萨姆摩威尔（Sommerville, I.）著. –北京：机械工业出版社，2003.4

（经典原版书库）

书名原文：Software Engineering, 6th Edition

ISBN 7-111-11810-3

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（2003）第017227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：华　章

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2003年4月第1版第1次印刷

787mm × 1092mm 1/16 · 45.25印张

印数：0 001-3 000册

定价：69.00元

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，还揭櫫了学术的源变，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。

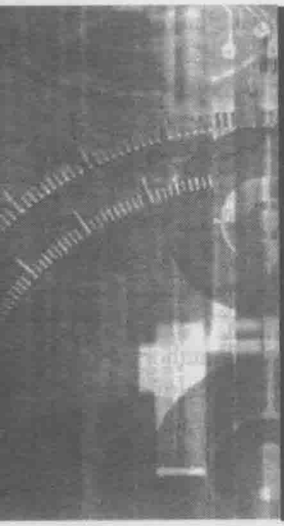
这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

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）



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