HZ BOOKS
华章教育

计 算 机 科 学 丛 书

JONES & BARTLETT LEARNING

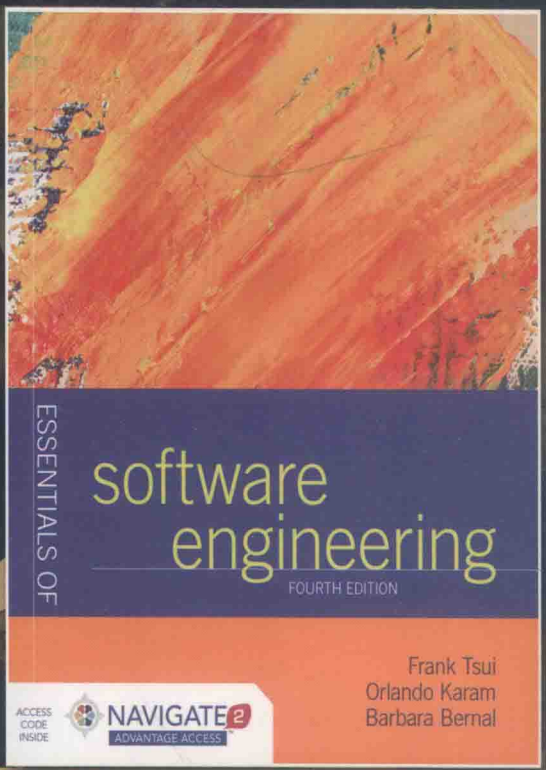
原书第4版

软件工程导论

弗兰克·徐（Frank Tsui）
[美] 奥兰多·卡拉姆（Orlando Karam）著
芭芭拉·博纳尔（Barbara Bernal）
崔展齐 潘敏学 王林章 译

Essentials of Software Engineering

Fourth Edition



机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

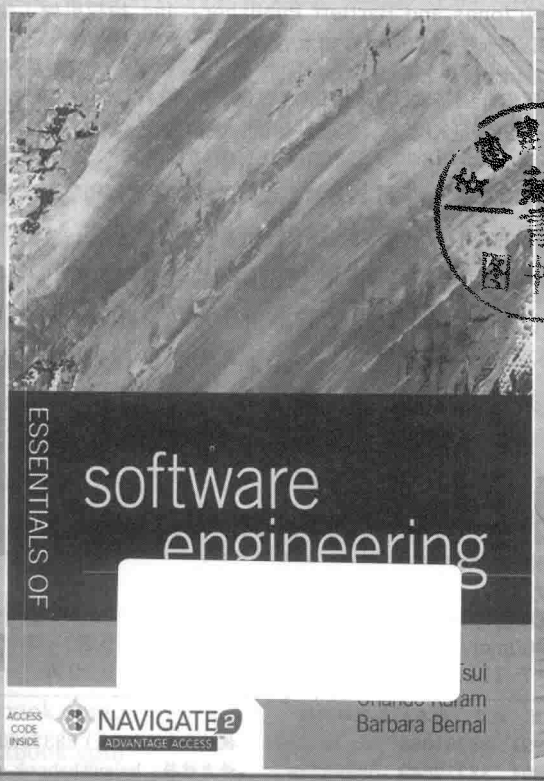
原书第4版

软件工程导论

弗兰克·徐（Frank Tsui）
[美] 奥兰多·卡拉姆（Orlando Karam）著
芭芭拉·博纳尔（Barbara Bernal）
崔展齐 潘敏学 王林章 译

Essentials of Software Engineering

Fourth Edition



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

软件工程导论（原书第4版）/（美）弗兰克·徐（Frank Tsui）等著；崔展齐，潘敏学，王林章译．—北京：机械工业出版社，2018.8
（计算机科学丛书）
书名原文：Essentials of Software Engineering, Fourth Edition

ISBN 978-7-111-60723-6

I. 软… II. ①弗… ②崔… ③潘… ④王… III. 软件工程－高等学校－教材 IV. TP311.5

中国版本图书馆CIP数据核字（2018）第192683号

本书版权登记号：图字 01-2017-4115

本书从基础入手，逐步介绍了软件工程的基本概念、软件过程模型、新兴过程方法、需求工程、设计的架构与方法论、设计的特征和度量、实现、测试、配置管理/集成/构建、软件支持和维护以及软件项目管理等内容。

本书可作为软件工程相关专业本科生的教材，也可作为软件工程领域专业技术人员的参考书。

出版发行：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：郎亚妹　　责任校对：李秋荣
印　　刷：北京市荣盛彩色印刷有限公司　　版　　次：2018年9月第1版第1次印刷
开　　本：185mm×260mm 1/16　　印　　张：15.5
书　　号：ISBN 978-7-111-60723-6　　定　　价：69.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066　　投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259　　读者信箱：hzjsj@hzbook.com

文艺复兴以来，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；也正是这样的优势，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，还揭示了学术的源变，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和发展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到积极的推动作用，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公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brian W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 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、研究及珍藏。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相助，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，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，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。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

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。随着计算机科学与技术专业学科建设的不断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将步入一个新的阶段，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。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：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心

译者序

Essentials of Software Engineering, Fourth Edition

本书是一本概要介绍软件工程核心内容的教材。全书共14章，完整覆盖了从初始阶段到发布、支持阶段的软件系统生命周期，讨论了贯穿软件生命周期的过程、质量保障及项目管理等问题。软件工程是一门实践性很强的学科，本书的三位作者积累了丰富的软件工程课程教学经验，并拥有多年在IBM、微软和亚马逊等公司从事软件研发和管理的工作经历。本书中也融入了他们的丰富工程经验和职业背景，并用生动的示例贯穿全书，使读者更易理解相关概念。

在机械工业出版社引进该书版权后，我们应姚蕾编辑的邀请，于2017年4月开始了本书的翻译工作。本书译者包括崔展齐、潘敏学、王林章，并由崔展齐和王林章对译稿进行了审核。在翻译过程中，得到了南京大学计算机科学与技术系研究生潘秋红、陆申明、周风顺、朱亚伟等的帮助，在此对他们的辛勤工作表示感谢。

翻译工作虽前后历时近一年，但因是在繁忙的教学、科研等工作之余完成的，仍感时间紧张，有些内容的翻译表达仍不够理想。此外，本书翻译虽力求忠实于原著，但由于水平所限，译文表达难免有不当之处，敬请读者批评指正。

总之，这是一本适合于高校学生和软件从业人员的很好的软件工程导论性读物。在多年的软件工程相关课程的教学过程中，我们也认为需要这样一本内容精练但覆盖相对全面的教材。我们很高兴向读者推荐本书。相信通过本书的阅读，会使读者对软件工程有一个初步的总体认识。

译　者

2018年6月

本书源自我们软件工程导论的教学经验。尽管市面上已有不少同类图书，但很少有仅介绍核心内容，适用于一门为期一学期（授课 16 周，每周大约 3 学时）的课程的教材。随着小规模网络应用的激增，许多信息技术新人尚未完全理解软件工程必备知识就进入软件工程领域。本书适用于经验有限的新生和打算在软件工程领域开始新职业的经验丰富的信息技术专业人员。本书覆盖软件系统完整的生命周期，范围从初始阶段到发布阶段直至支持阶段。

本书内容的基础是我们的从业经验和职业背景——第一位作者曾在 IBM、Blue Cross Blue Shield、MARCAM 和 RCA 等公司从事构建、支持和管理大型复杂的关键业务软件等工作，有着超过 25 年的工作经验；第二位作者具有在微软和亚马逊等公司使用敏捷方法构建较小规模软件的大量经验；第三位作者具有双语能力，对美国大学生和非美国的西班牙语学生都有着丰富的软件工程教学经验。

尽管新思想和新技术会不断涌现，书中介绍的一些原理可能需要更新，但我们相信本书中介绍的底层的、基础的概念会保留下来。

第 4 版前言

第 3 版出版后，计算机行业更快地朝着服务应用和社交媒体的方向发展。虽然软件工程的基础仍然相对稳定，但我们还是决定做一些修改以反映软件工程领域的某些变化，包括增加一些教学辅助。我们的目标是保持本书的内容足够精练，以便用于为期一学期 16 周的课程教学。

下面是第 4 版主要增加的内容：

- 第 5 章中看板方法的讨论
- 第 7 章中 REST 分布式处理架构
- 第 7 章中数据设计、分析以及“大数据”
- 第 9 章中代码重用
- 第 9 章中云计算

此外，我们对书中部分语句的表达做了细微修改以突出重点和强化理解。我们也听取了来自不同大学，使用本书第 1 版、第 2 版、第 3 版读者的反馈，修改了一些语法和拼写错误。书中可能还会有一些错误，责任完全在我们。

本书的第 1 版、第 2 版、第 3 版被许多高等院校使用，感谢他们的耐心和反馈，我们从中受益匪浅。我们希望对所有未来的读者而言，第 4 版是更好的一本书。

本书组织结构

第 1、2 章展示小型编程项目和构造关键业务软件系统所需工作之间的差异。我们特意用两章内容来说明这个概念，强调一个单人“车库作坊”和构建一个大型“专业”系统所需的项目团队之间的区别。这两章中的讨论给出了学习和理解软件工程的基本原理。第 3 章第

一次更加正式地讨论软件工程。这一章也介绍了软件工程职业的道德规范。

第 4、5 章涵盖软件过程、过程模型和方法学等传统主题。为了反映这个领域取得的大量进展，这两章非常详细地介绍了如何通过软件工程研究所（Software Engineering Institute，SEI）提出的能力成熟度模型来评价过程。

第 6、7、9、10、11 章在宏观层面上依次介绍了从需求到产品发布的开发活动。第 7 章用一个 HTML-Script-SQL 设计和实现示例，对 UI 设计展开讨论。在介绍软件设计的章节之后，第 8 章回顾并讨论了在评估概要设计和详细设计中使用的设计特征和度量指标。第 11 章不仅讨论了产品发布，还介绍了配置管理的基本概念。

第 12 章探讨了软件系统在发布给客户和用户后的相关支持和维护活动。涵盖的主题包括来电管理、问题修复和功能点发布等。在这一章中，进一步强调了配置管理的必要性。第 13 章总结了项目管理的不同阶段，同时介绍了一些具体的项目计划和监控技术。这章仅是一个概要总结，并没有包括团队建设、领导才能等主题。软件项目管理过程区别于开发和支持过程。第 14 章总结全书，介绍软件工程领域当前面临的问题，并展望了该领域未来的主题。

附录部分通过团队计划、软件开发计划、需求规格说明、设计计划和测试计划的“概要示例”让读者和学生深入理解软件开发主要活动可能产生的结果。一个常见的问题是需求文档或测试计划应该写成什么样。为帮助回答这个问题，并提供一个切入点，我们给出了计划、需求、设计和测试计划 4 项活动可能产生的文档样例格式。具体如下：

- 附录 A：软件开发计划概要
- 附录 B：软件需求规约概要
  - 示例 1：SRS 概要——描写
  - 示例 2：SRS 概要——面向对象
  - 示例 3：SRS 概要——IEEE 标准
  - 示例 4：SRS 概要——叙述法
- 附录 C：软件设计概要
  - 示例 1：软件设计概要——UML
  - 示例 2：软件设计概要——结构化
- 附录 D：测试计划概要

很多时候，对于新手软件工程师组成的团队项目，在开发过程中需要对如何使用文档描述过程进行特别指导。这 4 个附录用于给读者提供具体的概要大纲示例。每一个附录都提供大纲及相应的解释。这为教师开展课堂活动、分配团队项目和独立任务补充了具体材料。

本书涵盖的主题反映了 IEEE 计算机学会倡议的《软件工程知识体系》（SWEBOK）和《软件工程本科学位计划软件工程课程指南（2004）》强调的内容。有一个未突出强调但贯穿全书的主题是质量——一个需要集成到所有活动中去解决的主题。它不仅仅是测试人员的关注点，本书多个章节中都讨论了质量，这反映了其广泛的影响和跨越多个活动的特性。

建议教学方案

本书所有章节都可以在一个学期内学完。当然，不同教师的侧重点可能会有所不同：

- 希望聚焦于直接开发活动的教师应该在第 6～11 章多花时间。
- 希望关注间接和通用活动的教师应该在第 1、12 和 13 章多花时间。

应当指出，直接开发活动和间接支持活动都很重要，它们共同构成了软件工程学科。

每章后面都有针对本章的复习题和练习题。对于复习题，学生可以在书中直接找到答案。而练习题则用于课堂讨论、作业或小型项目。

致谢

首先要感谢我们的家庭，特别是我们的妻子 Lina Colli 和 Teresa Tsui。当我们把更多的时间用在编写本书而不是陪伴她们的时候，她们仍然给予了鼓励和理解。我们的孩子——Colleen 和 Nicholas、Orlando 和 Michelle 以及 Victoria、Liz 和 Alex——也积极地支持我们的工作。

另外，还要感谢那些在多方面帮助我们改进本书的审阅人。我们要特别感谢对本书第 3 版有贡献的下列人员：

- Brent Auernheimer，加州州立大学弗雷斯诺分校（California State University，Fresno）
- Ayad Boudiab，乔治亚佩雷米特学院（Georgia Perimeter College）
- Kai Chang，奥本大学（Auburn University）
- David Gustafson，堪萨斯州立大学（Kansas State University）
- Theresa Jefferson，乔治华盛顿大学（George Washington University）
- Dar-Biau Liu，加州州立大学长滩分校（California State University，Long Beach）
- Bruce Logan，莱斯利大学（Lesley University）
- Jeanna Matthews，克拉克森大学（Clarkson University）
- Michael Oudshoorn，蒙大拿州立大学（Montana State University）
- Frank Ackerman，蒙大拿技术学院（Montana Tech）
- Mark Hall，哈斯汀学院（Hastings College）
- Dr. Dimitris Papamichail，新泽西学院（The College of New Jersey）
- Dr. Jody Paul，丹佛大都会州立大学（Metro State Denver）
- Dr. David A. Cook，斯蒂夫奥斯汀州立大学（Stephen F. Austin State University）
- Dr. Reza Eftekari，乔治华盛顿大学、马里兰大学帕克分校（George Washington University，University of Maryland at College Park）
- Dr. Joe Hoffert，印第安纳卫斯理大学（Indiana Wesleyan University）
- Dr. Sofya Poger，佛里森学院（Felician College）
- Dr. Stephen Hughes，寇伊学院（Coe College）
- Ian Cottingham，内布拉斯加大学林肯分校杰弗里 S. 莱克斯学院（Jeffrey S. Raikes School at The University of Nebraska，Lincoln）
- Dr. John Dalbey，加州理工州立大学（California Polytechnic State University）
- Dr. Michael Murphy，康考迪亚大学得克萨斯分校（Concordia University Texas）
- Dr. Edward G. Nava，新墨西哥大学（University of New Mexico）
- Dr. Yenumula B. Reddy，关柏林州立大学（Grambling State University）
- Alan C. Verbit，特拉华县社区学院（Delaware County Community College）
- Dr. David Burris，萨姆休斯顿州立大学（Sam Houston State University）

我们还要感谢对本书第 4 版有贡献的下列人员：

- Savador Almanza-Garcia，Vector CANtech 公司

- Dr. Ronand Finkbine，印第安纳大学东南分校（Indiana University Southeast）
- Dr. Christopher Fox，詹姆斯麦迪逊大学（James Madison University）
- Paul G. Garland，约翰霍普金斯大学（Johns Hopkins University）
- Dr. Emily Navarro，加州大学尔湾分校（University of California，Irvine）
- Benjamin Sweet，罗伦斯科技大学（Lawrence Technological University）
- Ben Geisle，威斯康星大学绿湾分校（University of Wisconsin，Green Bay）

我们还要感谢 Jones & Bartlett Learning 出版社 Taylor Ferracane、Laura Pagluica、Bharathi Sanjeev、Amy Rose、Mary Menzemer 以及其他员工对本书的帮助。书中还存在的问题都是作者的错误。

——Frank Tsui

——Orlando Karam

——Barbara Bernal

创建一个程序

目标

- 分析制作一个简单程序所涉及的一些问题：
  - 需求（功能、非功能）
  - 设计约束和设计决策
  - 测试
  - 工作量估算
  - 实现细节
- 理解编写一个简单的程序所涉及的活动。
- 概括介绍一些在后续章节中涉及的软件工程主题。

1.1 一个简单的问题

在本章中，我们将分析编写一个较为简单的程序所涉及的任务。这将与第2章中描述的开发大型系统所涉及的内容形成对照。

假设有以下简单问题：给定存储在一个文件中的多行文本行（字符串），将其按字母顺序排序，并写入另一个文件。这可能是你遇见的最简单的问题之一。在程序设计课程中你可能已经完成过类似的作业。

1.1.1 决策

上述简单问题的陈述并不完全明确。你需要明确需求才能制作出更满足实际问题的程序。你需要了解**程序需求**和客户对设计施加的**设计约束**，并做出重要的技术决策。一个完整的问题描述包括程序需求（说明和限定程序的作用）和设计约束（描述了设计和实现程序的方式）。

> **程序需求** 定义和限定程序需要做什么的声明。
>
> **设计约束** 限制软件设计和实现方式的声明。

最重要的是要认识到需求（requirement）这个词在这里的含义和日常英语中的不同。在许多商业交易中，需求是必须发生的。然而，在软件工程中，很多项都是可以协商的。鉴于每项需求都会有成本，客户可能会在了解相关费用后，决定不再需要它们。需求通常被分为“确实需要的（needed）”和“有了会更好的（nice to have）”。

区分**功能需求**（程序的作用）和**非功能需求**（程序应该具备的特性）也是有用的。在某种程度上，功能类似于语法上直接和间接宾语的作用。因此，上述问题的功能需求将描述它要做什么：对文件进行排序（以及需要的所有细节）；而非功能性需求则描述诸如性能、可用性和可维护性等内容。功能需求只有满足或不满足，可以用一个布尔值来度量。而非功能需求的（测量结果）会有很大的差异，可以用一个线性标度来度量。例如，性能和可维护性需求可以通过

> **功能需求** 一个程序需要做什么。
>
> **非功能需求** 实现功能需求需要达到的方式。

满意程度来度量。

非功能需求也被非正式地称为 ilities，因为描述它们的词语通常以 ility 结尾。一些典型的用于描述非功能需求的特征包括：性能、可修改性、可用性、可配置性、可靠性、实用性、安全性和可扩展性。

除需求之外，还会有设计约束，如编程语言的选择、系统的运行平台以及与其连接的其他系统。这些设计约束有时被认为是非功能需求。并不容易界定其区别（类似于何处结束需
2 求分析开始设计）；对于临界个案，主要通过协商界定。大多数开发人员将可用性作为非功能需求，并将特定用户界面的选择，如图形用户界面（GUI）或基于 Web 用户界面，作为设计约束。但是，它也可以被定义为如下的功能需求："该程序将显示一个 60×80 像素的对话框，然后……"

需求由客户在软件工程师的帮助下确立，而技术决策通常由软件工程师做出，无须太多的客户参与。通常，某些技术决策（例如使用哪种编程语言或工具）可以作为需求，因为程序需要与其他程序进行交互，或者客户组织对特定技术具有专业知识或战略投资。

在以下内容中，我们将分析软件工程师面临的各种问题，即使是简单的示例程序。我们将把这些决定分为功能和非功能需求、设计约束和设计决策。但请记住，其他软件工程师可能会对其中一些问题有不同的分类方式。我们将使用前面提出的简单排序问题作为示例。

1.1.2　功能需求

在设计和实现解决方案之前，我们必须考虑该问题的几个方面，并提出许多问题。以下是关于功能需求思考过程的非正式总结：

- *输入格式*：输入数据的格式是什么？数据应如何存储？字符是什么？在我们的例子中，需要定义文件中的行是用什么分隔的。这一点非常重要，因为不同的平台可能使用不同的分隔符。通常使用的是换行和回车的一些组合。为了知道边界的准确位置，我们还需要知道输入字符集。最常见的表示方式是每个字符 1 个字节，这对于英语和大多数拉丁语派生语言来说是足够的。但是有些表示方式，如中文或阿拉伯语，因为超过 256 个字符，所以每个字符需要 2 个字节。其他情况则需要两种类型的组合。对于单字节字符和双字节字符表示的组合，通常需要一个转义字符将模式从单字节转换为双字节，反之亦然。对于排序问题，我们将假设每个字符 1 个字节的简单情况。
- *排序*：虽然它似乎是一个定义明确的问题，但排序有很多略有不同的含义。对于初学者（假设我们只有英文字符），我们是按升序还是降序排序？如何处理非字母字符？数字在字母之前还是之后？如何处理小写和大写字母？为了简化问题，我们将字符之间的排序定义为数字顺序，并将文件升序排列。
- *特殊情况、边界和错误情况*：是否有特殊情况？我们应该如何处理诸如空行和空文件之类的边界情况？如何处理不同的错误情况？在详细设计甚至实现阶段之前，可
3 能并不能完全明确所有需求。这尽管不是很好的做法，但却很常见。对于我们的程序，我们不以任何特殊的方式处理空行，除了指定当输入文件为空时输出文件应该被创建但是为空。只要所有错误都向用户发出信号，并且输入文件没有被损坏，我们不指定任何特殊的错误处理机制。

1.1.3 非功能需求

涉及非功能需求的思考过程可以非正式地总结如下：

- *性能需求*：虽然并不像大多数人所认为的那样重要，但性能始终是一个问题。程序需要在一定时间内处理完大部分或全部输入。对于排序问题，我们将性能要求定义为在 1 分钟内对含有 100 行、每行 100 个字符的文件进行排序。
- *实时性需求*：当程序需要实时执行时，即它必须在给定的时间内完成处理时，性能是一个问题，运行时间的变化也是一个大问题。我们可能需要选择一个性能低于平均值的算法，如果该算法在最坏情况下的性能更好。例如，快速排序被认为是最快的排序算法之一，然而，对于一些输入，它可能性能较差。在算法术语中，它的预期运行时间的阶是 $n \log(n)$，但是其最坏情况下的性能是 n^2。如果你有实时性需求，平均情况下可以接受，但最坏的情况不能，那么你可能需要选择变化较小的算法，例如堆排序或合并排序。在文献（Main et al., 2010）中进一步讨论了实时性能分析。
- *可修改性需求*：在编写程序之前，了解程序的预期寿命以及是否有计划修改程序很重要。如果程序只使用一次，那么可修改性不是一个大问题。但是，如果程序要使用 10 年以上，那么我们需要使其易于维护和修改。当然这个需求会在 10 年的时间内发生改变。如果我们知道有计划以某些方式扩展程序，或者需求将以特定方式发生变化，那么我们应该在程序设计和实现时为这些修改做准备。请注意，即使可修改性要求较低，这也不是编写糟糕代码的许可证，因为我们仍然需要能够理解程序以进行调试。对于排序程序示例，如果我们知道下一步的需求可能会将排序方式从降序变为升序，或者更改为同时包括升序和降序的排序，我们将如何设计和实现解决方案呢？
- *安全性需求*：客户组织和软件开发人员需要就安全定义达成一致，以防止数据和资源的丢失、不准确、变更、不可用或滥用，其中安全定义是从客户的业务应用目标、项目资产的潜在威胁和管理控制中获得的。安全性可以是功能需求或非功能需求。例如，软件开发人员可能会说，系统必须防止拒绝服务攻击，以完成其任务。文献（Mead et al., 2005）中讨论了安全质量需求工程（Security Quality Requirement Engineering, SQUARE）。
- *可用性需求*：在开发软件时需考虑到程序的最终用户具有的特定背景、教育、经验、需求和交互方式。收集并研究程序的用户、产品和环境特征，以便设计用户界面。这种非功能需求集中在程序和最终用户之间的交互中。最终用户对这种交互的有效性、效率和成功性进行评估。可用性需求的评估是不可直接度量的，因为它是通过最终用户在特定的可用性测试中报告的可用性属性进行限定的。

4

1.1.4 设计约束

与设计约束相关的思考过程可概括如下。

- *用户界面*：程序有什么样的**用户界面**？应该是命令行界面（CLI）还是图形用户界面（GUI）？我们应该使用基于 Web 的界面吗？对于排序问题，基于 Web 的界面听起来不合适，因为用户需要上传文件并下载已排序的文件。虽然 GUI 在过去十年左右已经成为常态，但是 CLI 更适合排序问题，特别是因为在脚本中调

> **用户界面** 用户从系统看见、感知和听到的内容。

用它更容易，可以将手动过程自动化，并将此程序在将来作为模块复用。这是涉及用户界面设计的注意事项之一。在1.4节中，我们将创建几个实现，一些基于CLI，一些基于GUI。第7章将详细讨论用户界面设计。

- *典型和最大输入规模*：根据典型的输入规模，我们可能希望在算法和性能优化上花费不同的时间。此外，特定算法对于某些类型的输入来说会特别好或特别坏，例如，几乎排好序的输入会使得快速排序需要更多的时间。请注意，系统有时会给出不准确的估计值，但是即使是大致的数字也可以帮助你预测问题或引导你采用适当的算法。在这个例子中，如果输入规模较小，你可以使用几乎所有的排序算法。所以你应该选择实现最简单的那种。如果输入规模较大，但内存（RAM）仍然够用，则需要使用高效的算法。如果输入规模超过了内存，则需要选择专门的磁盘排序算法。
- *平台*：程序需要运行在哪些平台？这是一个重要的业务决策，可能包括架构、操作系统和可用的库，而且几乎总是会在需求中表达出来。请记住，虽然跨平台开发变得更加容易，并且有许多语言被设计为可跨平台的，但并不是所有的库都可以在所有平台上使用。支持新平台总是有额外的成本。另一方面，即使不需要，丰富的编程经验也有助于实现可移植性。设计和实现程序时多考虑一点，能最大限度地减少移植到新平台所需的工作。即使不需要支持新平台，对是否支持其他平台以及是否使用最方便移植的技术和编程实践进行快速成本效益分析也是一个很好的做法。
- *进度需求*：完成项目的最后期限取决于客户，以及技术方面的可行性和在成本上的投入。例如，关于进度计划的对话可能有以下形式。你的客户可能提出“项目需要在下个月之前完成”的要求，你回应说：“如果你需要在两个月内完成的话，需要两倍的成本”或“那不能做到。这通常需要3个月。我们可以尽力把它压缩到两个月，但不能更短了。”客户可能同意这一点，也可能说“如果下个月不能完成，那就没有用了”，并取消了项目。

1.1.5 设计决策

与排序问题的设计决策相关的步骤和想法可以归纳如下。

- *编程语言*：这通常是一个技术设计决策，有时候也被当作设计约束。所需的编程语言类型、性能和可移植性需求以及开发人员的技术专长往往在很大程度上影响编程语言的选择。
- *算法*：在实现系统时，通常可以通过算法的选择来影响多个部分。在我们的示例中，可以选择多种算法来进行排序。所使用的语言和可用的库也会影响算法的选择。例如，要进行排序，最简单的解决方案是使用编程语言提供的标准函数，而不是自己实现。因此，可使用实现选择的任何算法。性能通常是选择算法最重要的影响因素，但需要与实现算法所需的工作量和开发人员的熟悉程度相平衡。算法通常是设计决策，但也可以作为设计约束或功能需求。在许多业务环境中，可能会要求使用特定的算法或数学公式，在许多科学应用中，目标是测试几种算法，这意味着你必须使用某些算法。

1.2 测试

在定义、开发和完成程序后，测试程序总是一个好主意。这听起来像是显而易见的建