国外经典教材

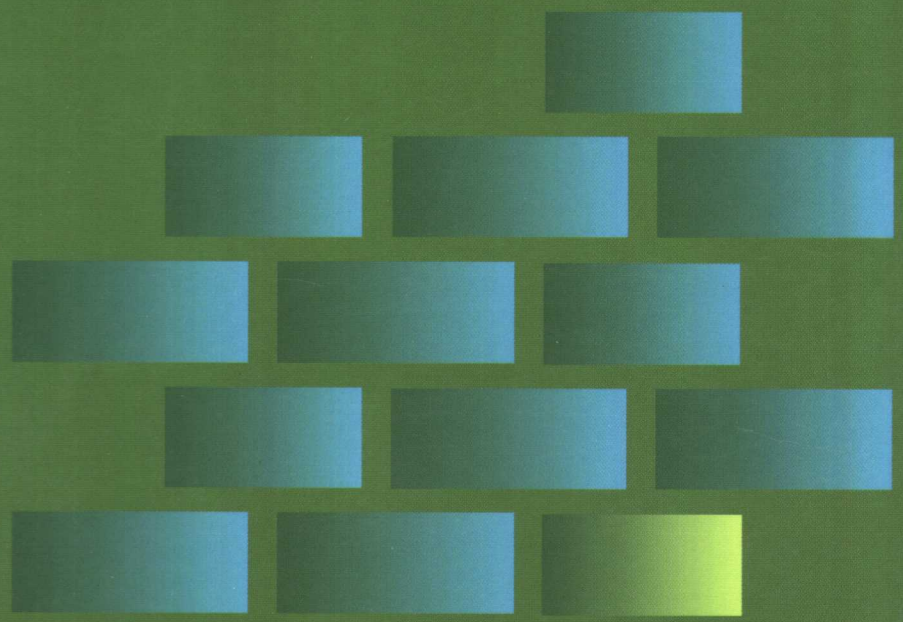
PEARSON Prentice Hall

软件工程基础（第2版）

Fundamentals of Software Engineering (Second Edition)

Carlo Ghezzi
(意) Mehdi Jazayeri 著
Dino Mandrioli

施平安 译



Pearson Education

清华大学出版社

国外经典教材

软件工程基础

（第2版）

Carlo Ghezzi
（意） Mehdi Jazayeri 著
Dino Mandrioli

施平安 译

清华大学出版社
北京

内 容 简 介

本书通过严格的形式化方法和非形式化方法阐述了软件工程原则和方法的重要性，有选择地介绍了软件工程基础；强调并确定了适用于整个软件生命期的基本原则，全面深入地介绍了这些基本原则在软件设计、规范、验证、软件生产过程和管理活动中的运用；书中提供了大量的练习和案例分析，既有助于理解书中介绍的理论知识，又可以让读者亲身体验如何应对复杂的现实问题。

本书可以作为大专院校计算机科学系和计算机工程系的本科生和研究生教材，也可以作为计算机软件人员和计算机用户的参考书。

图书在版编目（CIP）数据

软件工程基础（第2版）/(意) 盖兹，（意）扎查耶瑞，（意）曼德瑞利著；施平安译. --北京：清华大学出版社，2003

(国外经典教材)

书名原文：Fundamentals of Software Engineering,（ Second Edition ）

ISBN 7-302-07202-7

Ⅰ.软… Ⅱ.①盖… ②扎… ③曼… ④施… 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（2003）第 078801 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦

http://www.tup.com.cn　　**邮　　编**：100084

社 总 机：010-62770175　　**客户服务**：010-62776969

文稿编辑：赵欣奎

封面设计：立日新设计公司

印 刷 者：北京彩艺印刷有限公司

发 行 者：新华书店总店北京发行所

开　　本：185×260　**印张**：28.75　**字数**：714 千字

版　　次：2003 年 9 月第 1 版　2003 年 9 月第 1 次印刷

书　　号：ISBN 7-302-07202-7/TP・5244

印　　数：1～4000

定　　价：55.00 元

译 者 序

随着计算机网络技术的飞速发展，社会的发展和人们的生活越来越离不开软件及其提供的关键功能，人们对于更大、更精密的软件的需求也在不断地上升。为了满足这些期望，必须能够用合理的成本在合理的时间内生产可靠的软件，这正是软件工程的目标。软件工程是一门非常年青的工程学科，它运用工程学科的原则和方法指导软件工程师团队开发软件系统。软件工程原则具有持久性、通用性，只要掌握了它们，理解和掌握最新推出的技术和工具就容易了。基于上述认识，本书重点介绍了软件工程的原则和方法。

本书原著作者 Carlo Ghezzi 是 Politecnico di Milano 的计算机科学教授，主讲软件工程，为计算机协会的特别会员；Mehdi Jazayeri 是 Technische Universitat Wien 的计算机科学教授，主讲分布式系统，他有多年在硅谷从事软件开发的经验；Dino Mandrioli 是 Politecnico di Milano 的计算机科学教授，主讲理论计算机科学，主要研究方向是形式方法在软件工程实践中的应用。

本书在组织上不同于一般的基于软件开发生命期模型的软件工程教材，而是先介绍软件质量和软件工程原则，然后先后介绍这些原则在软件设计、规范、验证、软件生产过程和管理中的运用。全书通过严格的形式方法和非形式方法强调了原则和方法的重要性，强调并确定了适用于整个软件生命期的基本原则，使学生能够迅速适应当今社会瞬息万变的技术。全书提供 350 多个有助于理解书中介绍的理论知识的练习；还介绍了 20 多个全面深入的案例分析，使没有任何经验的学生能够感受到软件工程师每天都要面对的问题。

本书介绍了许多最新技术，包括面向对象、软件构架和组件、Z 和 UML 建模语言、需求工程描述和案例分析、最新的确认技术（包括模型检验）、质量改进模式（包括 GQM 和 CMM）、更多有关软件过程的资料（包括统一过程），以及一些关于过程、需求、设计、组织和系统工程的最新案例分析。

本书既可以作为大专院校计算机科学系和计算机工程系的本科生和研究生教材，也可以作为计算机软件人员和计算机用户的参考书。专业工程师和管理者可以从本书找到相关的材料，了解到现代软件工程实践是有用的，并了解到采用这些实践的必要性。本书适用于愿意花时间认真研读的专家，但不适于粗读。

全书由我一人翻译，施惠琼帮助进行了全书的录入和审校工作，在此对她表示感谢。我还要感谢我的家人们在我翻译本书过程中表现出来的无比耐心，感谢他们为我所提供的帮助和支持。尤其是我的女儿，她那天真烂漫的笑声给予了我无比的力量。没有他们的谅解和支持，很难想像我能够完成如此巨大的翻译工作。

责任编辑赵欣奎和文开琪在本书翻译过程中提供了很大的帮助，与他们通过互联网对本书中出现的难词、难句进行了广泛深入的交流。没有他们的支持，也很难想像本书能翻译得如此专业。书中出现的术语都经过仔细的推敲和研究，然而疏漏和争议之处在所难免，望广大读者提供宝贵的意见。

施平安
2003/4/20 于海军广州舰艇学院

第2版　前　言

本书第1版于1991年出版。自那以来，计算技术以及软件工程都取得了重大的进步。毫无疑问，Internet的普及对教育、科研、社会进步、商业、贸易产生了深远的影响。为使本书能反映软件工程最近10年的最新进展，我们决定推出第2版。

我们很高兴地发现，随着时间的推移，本书第1版所做的基本假设——即原则的持久性和重要性——已经得到证实：尽管技术有了进步，但是软件工程的原则没变。因此，我们能够在不改变本书原有结构的前提下对每章进行更新。本书的结构如下：

简介：第1~3章；

产品：第4~6章；

过程和管理：第7~8章；

工具和环境：第9章。

与产品相关的各章内容按设计（第4章）、规范（第5章）和验证（第6章）的顺序进行编排。这与其他软件工程类书籍采用的方法不同，其他书一般先介绍规范，然后再介绍设计。我们之所以选择这种方法，是因为打算遵循本书所用的基于原则的方法。所有这些活动——设计、规范和验证——都是整个软件生命周期中必须了解和应用的基本活动。例如，对于设计，我们不仅要处理软件构架，而且还要处理软件规范。模块化设计方法有助于我们构造软件以及编写规范文档。其他软件工程类书籍往往先介绍规范后介绍设计，其理由是根据传统的软件过程，首先应指定一个软件，然后再设计它。相反，我们认为首先应学习设计活动和方法，激发学习规范所需的积极性，并提供形成那些规范所需的技巧和技术。

自本书第1版面世以来，虽然软件工程的所有领域都在发展，但是工具和环境领域的变化最明显。因此，我们对第9章进行了大量的修改；我们采用的方法是介绍基本原则，而不是具体的工具。近几年来，我们亲眼目睹工具随技术的发展而变化，并且选哪种具体工具进行研究取决于学生所处环境和所关注的重点。因此，我们提供了一个用于分析和评估软件工具的框架，而不具体地讨论任何特定的工具。

除了许多较小的改进和变化外，我们还进行了如下主要补充：

第3章增加了两个新的案例分析，一个与简单的编译器有关，另一个与电梯系统有关，这两个案例出现在书中大部分地方。它们是互补的，因为它们针对不同的应用领域并且提出了不同的设计问题。第3章以简单直观的方式介绍它们，以便面向学生，使其专注思考系统问题。它们旨在说明基本原理在具体实例中的应用。

第4章对面向对象、软件构架、组件和分布式系统方面的内容进行了扩充。

第5章新增了Z和UML方面的内容。新增的一节更系统地介绍了需求工程。

第6章新增了作为评估和验证技术的模型检查和GQM。

第7章新增了统一过程（unified process）、开放源过程（open-source process）及同步与稳定过程（synchronize-and-stabilize process）这三方面的内容。此外还新增了一个新的需求工程案例分析。

第 8 章新增了能力成熟度模型（capability maturity model），并描述了诺基亚（Nokia）软件工厂。

第 9 章新增了并发版本控制系统（concurrent versioning system，简称 CVS）相关介绍。

第 10 章介绍了软件工程道德准则（Software Engineering Code of Ethics）。

附录部分新增了一个关于行业内形式方法用法的案例分析。

面向对象的作用

本书对面向对象原则的介绍与其他方法并重，而不是将它作为实现软件工程的惟一方法。面向对象的分析、设计和编程确实得到了很大的发展，并成为软件工程的主要方法。但我们认为，软件工程的基本原则比对象更深奥。学生应当掌握可以不同方式来使用的原则和方法。学生应当知道如何从不同方法中选出所需的方法，并能在合适的情况下使用面向对象方法。例如，学生在学习对象和继承之前学习什么叫信息隐藏。

案例分析的目的

本书正文和附录中介绍的案例分析有两个目的：其一是提供在一个更大的场景中讨论的问题，目的是让学生更深刻地认识到原则或者技术的重要性；其二是为那些还没有接触过真实项目的学生提供一个实际项目的概况。为强调重要问题，我们对案例分析进行了必要的简化，但我们发现它们尤其适用于几乎没有经验的学生。软件工程的学习给大学的课程设置出了个难题，因为普通学生都没有接触过软件工程师每天都要面对的问题。书中提供的案例分析则试图克服该难题。

教师资料

本书配套光盘中，包含供教师参考的习题答案以及课程教学大纲示例。使用本书的学生和教师还可以通过出版社的网站联系作者。我们欢迎广大读者提出宝贵的意见和建议。

Carlo Ghezzi
意大利・米兰
Mehdi Jazayeri
美国・加利福尼亚州帕洛阿尔托
Dino Mandrioli
瑞士・卢加诺

第1版　前　言

这是一本**软件工程**教材。本书的基本主题是严格执行软件工程的重要性。涉及该主题的传统教材都基于软件开发的生存期模型——即需求、规范、设计、编码、维护——依次研究每个阶段。相比之下，我们的介绍则基于能独立于生存期模型进行应用的重要原则。我们的重点在于识别和应用适用于整个软件生存期的基本原则。

本书具有如下基本特征：

- **本书讨论的是软件工程而不是编程**。因此，我们不讨论任何编程问题。例如，我们不讨论任何程序设计语言的结构（诸如 goto）、循环等等。我们认为软件工程专业的学生应当已经熟悉这些问题了，这些问题在程序设计语言教材中介绍比较合适。另一方面，我们确实讨论了将软件设计结构映射为特定的程序设计语言的问题。我们重点讨论模块间的问题，并且认为编写单个模块的能力是必要条件。
- **本书强调原则和技术而不是特定的工具**（但例子中可能会用到它们）。现在，许多公司都在积极开发软件工程工具和环境，我们希望随着软件工程知识不断增加，开发出更好更精致的工具。一旦学生理解了工具所基于的原则和技术，就很容易掌握工具了。原则和技术的使用在不同工具间是共通的，相比之下，掌握了如何使用任何一种特定工具则无助于学生更好地掌握其他工具的用法。再说，在不理解工具基本原理的情况下使用工具是危险的。
- **本书介绍了工程原则，但不是一本工程手册**。原则是通用的，而且一般在多年内经久不变，特定的技术方法则会由于技术、不断增长的知识等而发生变化。至于**如何**运用某种特定的技术，可以参考工程手册：此类书籍包含该技术的一系列使用说明。另一方面，本书旨在使读者理解**为何**该用、为何**不该**用某种特定的技术。即使我们确实介绍了如何用一种特定的技术来实施一个给定的原则，但我们主要强调的仍然是如何理解“为何”这类问题。

本书体现了我们使用基本原理的信条以及理论在工程实践中的重要性。我们同时在大学和各方面软件工程专业课中使用过书中的材料。

本书的读者对象

本书既可以作为在校生软件工程的教材，也可以作为软件工程的自学教材。专业工程师和经理可以从本书找到相关的材料，了解到现代软件工程实践是有用的，并了解到采用这些实践的必要性。本书适用于愿意花时间认真研究的专家，但却不适用于粗读。特别是在需要的时候，我们都会牺牲广度以求深度。对于专业人士，关于需要进一步阅读的参考资料的说明尤其有用。还出版了与本书配套的教师手册，主要讲述课程组织并提供了某些练习的答案。

前提条件

本书适用于计算机科学专业的初级、高级学生或者低年级研究生使用。读者必须学过数据结构课程，并能熟练掌握一种或者多种编程语言。我们假定读者已精通编程。虽说分析推理并非绝对需要，但是它能提高读者理解本书中更深层概念的能力。可以通过诸如微积分、离散数学或更好的计算机科学理论等数学课程的学习来发展这种分析推理能力。“数学成熟度（Mathematical maturity）”对于任何工程学科的学生来说，都是必要的。

本书的内容编排

软件工程是一门大型的、多学科交叉的学科。组织一本涉及该主题的教材确实不是一件容易的事情，因为教材应当按顺序介绍内容，但是软件工程的很多方面是互相交织的，各主题间不存在最优的顺序。我们对本书的组织基于软件工程的观点：

我们准备构造一个**产品**：软件；

我们用一个**过程**来构造这个产品；并且

我们使用**工具**来支持这个过程。

因此，本书包含3个技术部分，它们分别涉及到软件产品（第4章到第6章）、软件工程过程和管理（第7章和第8章）以及软件工程环境（第9章）。第1章到第3章简要介绍了软件工程领域以及本书后面技术性更强的各节内容。

第2章讨论了软件的很多方面和共同期望的软件特征。这些特征为软件构造人员和所用的过程施加了一些限制。第3章介绍了构造高质量软件的原则。通过研究原则而不是特定的工具，学生可以学到不依赖于某种特定的技术和应用环境的知识。因为技术不断在进步，环境不断在变化，所以学生应当掌握可以在不同的应用领域中使用的原则和技术。第4章到第8章分别讨论了如何将第3章中讨论的原则应用于设计、规范、验证、工程过程和工程管理。第9章讨论了使用计算机本身来帮助构造软件。我们将任何特定工具的讨论推迟到这一章进行。

虽然前面两部分材料应当经得起时间的考验，但第3部分中的材料很可能会过时（我们希望这样），因为更新更好的工具会开发出来。鉴于程序设计语言是软件工程师的基本工具，所以我们将第9章当做设计问题（见第4章）和特定程序设计语言结构之间的桥梁。

练习

本书包含大量练习，主要分为以下3类：

- 简短的书面练习，旨在扩展本书所学知识，或者更深入地应用所学知识；这些练习散布在各章之中。
- 较长的书面练习位于每章最后，要求综合应用当前章所学内容进行解答。
- 学期项目，要求由一个小团队来开发一个实际的软件系统。

部分练习的答案在各章最后给出，更多的练习答案请参阅教师手册。

案例分析

文中采用了几个案例分析，以说明不同概念的综合和在实际环境中比较不同的方法。此外，本书最后给出了三个实际的软件工程项目及其分析的案例。可以在不同时间出于不同目的阅读和研究这些案例分析。通过这些案例分析，没有从业经验的学生能够很快地体验到工业实践中面临的各种问题。稍有经验的学生也许可以识别出这些案例分析中的某些特征，并从其他案例中吸取知识。案例分析可以与正文同步阅读。书中的几个练习引用了这些案例分析。

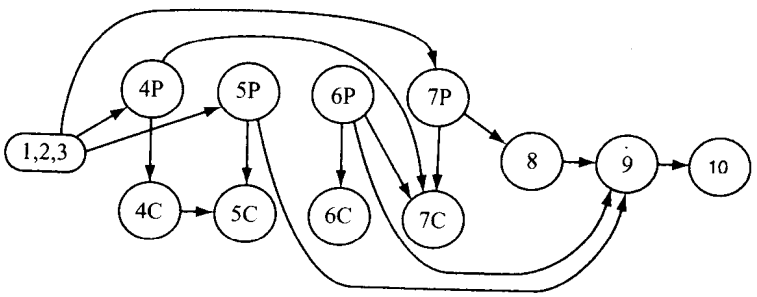
实验课程

许多软件工程课程把讲授同实验结合起来讲。在一个学期内同时进行讲授和实验是非常困难的。教师会发现自己在讨论组织问题，学生却在专心解决他们的日常调试问题。我们认为，必须像所有其他工程学科一样来讲授软件工程，首先为学生提供一个坚实的“理论”基础。只有打下了良好的理论基础，然后获取实验室经验才能增强学生的知识。这就意味着学生项目必须从接近中期或中期开始启动，而不是在学期一开始就启动。在我们看来，最好用一个学期进行理论教学，再用一个学期做实验。教师手册提供了几种基于本书组织实验课程的观点。

阅读顺序图

本书可以按不同的顺序和不同的级别进行阅读。第一次阅读本书或粗读本书时，第 4 章到第 7 章中的部分内容可以跳过不看。第 1 章到第 3 章是阅读以后各章的基础。阅读顺序图画出了各章之间的依赖关系以及阅读本书的各种途径。符号 *n*P 指阅读第 *n* 章中的部分内容，有的节可以跳过；*n*C 表示全部阅读。

教师手册基于本书讨论了不同的课程组织。传统的一学期项目的软件工程课可以遵循如下顺序：1,2,3,7P,5P,4P,6P,8,9,10。我们自己喜欢遵循如下顺序：1,2,3,4P,5P,6P,7P,8,9,10。无论采用那种上课顺序，学生必须在 5P 后开始项目。



致谢

我们忠心地感谢本书早期草稿的评审者：康涅狄格大学的 Reda A.Ammar、杨百翰大学的 Larry C.Christensen、爱荷华州大学的 William F.Decker、堪萨斯州立大学的 David A.Gustafson、位于圣巴巴拉的加利福尼亚大学的 Richard A.Kemmerer、维吉尼亚大学的 John C.Knight、华盛顿大学的 Seymour V.Pollack 以及美国北卡罗来纳州大学 K.C.Tai。

我们还要感谢为本书各稿提供宝贵意见的朋友：Vincenzo Ambriola, Paola Bertaina, David Jacobson 和 Milon Mackey。

Hewlett-Packard 实验室和 Politecnico di Milano 支持了 Mehdi Jazayeri 于 1988 年春季在 Politecnico di Milano 举办的课程，从而使本书的构想成为可能。Alfredo Scarfone 和 HP Italiana 为我们在意大利提供了支持。感谢 Hewlett-Packard 实验室管理人员的支持，尤其是帕洛阿尔托的 John Wilkes, Dick Lampman, Bob Ritchie 和 Frank Carrubba，以及意大利比萨的 Peter Porzer。感谢 Bart Sears 在各系统方面提供的帮助，感谢 John Wilkes 用他的数据库来管理参考资料。我们还得到了 Consiglio Nazionale delle Ricerche 的支持。

Carlo Ghezzi
意大利 • 米兰
Mehdi Jazayeri
美国 • 加利福尼亚帕洛阿尔托
Dino Mandrioli
瑞士 • 卢加诺

目　录

第 1 章　软件工程：预览

软件工程是研究软件系统构造的计算机科学领域，这些软件系统很大、很复杂，要一个或多个工程师团队才能完成。通常，这些系统以多个版本存在，并且要使用很多年。在它们的生存期内，要经历很多变化：修复故障、增强现有特性、增加新特性、消除旧特性或者对它们进行改写以适应在新环境中运行。

我们可以将软件工程定义为“工程在软件中的应用”。更精确地讲，IEEE 标准 610.12-1990 的软件工程术语的标准术语表（ANSI）将软件工程定义为：系统的、规范的、定量的方法在软件的开发、操作和维护中的应用。

Parnas[1978]将软件工程定义为“多人构造多版本软件”。该定义抓住了软件工程的本质，并强调了程序设计和软件工程之间的差别。一个程序员编写一个完整的程序，而一个软件工程师编写一个软件组件，它要与其他软件工程师编写的组件进行组合才能构建一个系统。一个软件工程师编写的组件可以被其他软件工程师修改；在组件的原始工程师离开项目很久以后，其他工程师可以用它建造不同版本的系统。程序设计主要是个人行为，而软件工程从本质上讲是团队行为。

实际上，术语“软件工程”是 20 世纪 60 年代末提出的，即在认识到所有从如何很好地进行编程中得到的经验无助于建造更好的系统之后。虽然通过系统地研究算法和数据结构以及“结构化程序设计”的提出，程序设计领域已经取得了巨大的进步，但是在建造大型软件系统方面仍然存在重大困难。物理学家编写一个程序以计算一次试验的微分方程的解的技术，对于建造一个操作系统甚或一个库存跟踪系统的团队中的程序员是不适用的。在这些复杂的情况下，真正需要的是一种经典的工程方法：明确地定义试图解决的问题，然后使用和开发标准的工具和技术来解决它。

固然，软件工程自 20 世纪 60 年代以来取得了很大的进步。该领域已经使用了标准技术。软件工程逐渐地被越来越多在传统上与工程有关的学科所实践，而不是作为一种技术在使用。然而，软件工程仍然与传统的工程学科存在差别。在设计一个电子系统时，诸如一个放大器，电子工程师可以精确地规定该系统。所有的参数和容许量都可以明确地规定，并且可以为客户和工程师所理解。而对于软件系统，这样的参数仍然是不可知的。我们不知道要规定哪些参数以及如何规定它们。

在古典工程学科中，工程师可以借助工具和数学培训，独立于那些设计特性来规定产品的特性。例如，一个电子工程师根据数学方程来验证一个设计没有违反电源需求。在软件工程中，还没有开发出这么好的数学工具，并且它们的可用性还存在争议。典型的软件工程师更多地依赖于经验和判断而不是数学技术。虽然技术和判断是必不可少的，但是形式分析工具在工程实践中也是不可少的。

本书将软件工程作为一门工程学科进行介绍。提供了一些我们认为对于“多人构造多版本的软件”所必不可少的原则。这样的原则比任何一种特定的软件构造表示法或者方法论都更重要。原则使得软件工程师能够评价不同的方法，并在合适的地方应用它们。第 3 章介绍这些原则；本书的其余内容可以看成这些原则在软件工程的各种问题中的应用。

本章回顾软件工程的演进及其与其他学科的联系。本章的目标是正确地放置软件工程领域的位置。

1.1 软件工程在系统设计中的作用

软件系统通常是一个更大的系统的一个组件。因此，软件工程活动是一个更大的系统设计活动的一部分，大系统中软件的需求根据正在设计的系统的其他部分的需求综合确定。例如，一个电话交换系统包括计算机、电话线和电缆、电话、其他诸如卫星等硬件，以及控制各组件的软件。预期满足整个系统的需求的是所有这些组件的组合。

诸如“系统在 20 年内的故障时间不得超过 1 秒钟”或者“提起电话听筒时，拨号音在半秒钟内响起”等需求，可以通过硬件、软件和特殊设备的综合应用而得到满足。要做出决策来最佳地满足需求，需要综合考虑很多方面。其他一些关于系统的例子如发电厂或者交通监控系统、银行系统和医院管理系统，它们都表现出需要将软件作为一个更大的系统的一个组件。

软件逐渐嵌入到各种系统中，从电视机到飞机都配备了软件。开发这样的系统要求软件工程师更加广泛地考虑系统工程的基本问题。它要求软件工程师参与制定整个系统的需求；要求软件工程师在着手考虑软件必须满足什么抽象界面之前，必须尽量理解应用领域。例如，如果与用户接口的硬件设备有原始的数据输入设备，则该系统将不需要复杂的文字处理系统。

将软件工程看成系统工程的一部分使我们认识到折中考虑的重要性，这是任何工程学科的特点。经典的折中考虑牵涉到选择在软件方面应当干什么及在硬件方面应当干什么。软件实现保证了灵活度，而硬件实现则保证了性能。例如，在第 2 章中，我们将会看到一个投入硬币即发生作用的机器的例子，既可以将它建成有多个硬币投入槽，每种类型的硬币一个投入槽；也可以将它建成只有一个硬币投入槽，而让软件去识别不同的硬币。一种更加基本的折中考虑牵涉到决定什么应当自动完成和什么应当手动完成。

1.2 软件工程简史

软件工程作为计算机科学的一门学科，其产生和发展可以追溯到对编程行为的发展和成熟的看法。在计算技术的早期，程序设计问题基本上被看成如何将一个指令序列放在一起，以使计算机实现某些有用的功能。要通过编程进行解决的问题已经得到很好地理解，例如，如何解微分方程。程序由相应的人编写，假如由物理学家编写用来解其感兴趣的方程。问题仅仅存在于用户和计算机之间；不会牵涉到他人。

随着计算机越来越廉价和普及，越来越多的人开始使用它们。20 世纪 50 年代末发明了高级语言，使得人与计算机的交流更加容易了。但是，使计算机实现某种有用的功能的活动，基本上是由一个人为一个良好定义的任务编写一个程序。

就是在这个时候，“程序设计”确立了它的职业地位：你可以请一个程序员为你编写一个程序，而不用亲自去编程。这种安排引入了用户和计算机之间的一种区别：现在用户必须以一种不同于以前所用的精确的编程符号那样的形式规定好任务。然后程序员阅读该规范，并把它转变成精确的计算机指令集合。当然，这有时会使程序员误解用户的意图，甚

至一些经常碰到的小任务也会这样。

在 20 世纪 60 年代早期，几乎没有完成什么大型的软件项目，并且它们都是由计算机专家承担的。例如，MIT 开发的 CTSS 操作系统确实是一个大型项目，但它是由知识渊博、积极向上的专家们完成的。

到了 20 世纪 60 年代中期，真正的大型软件系统的开发开始走上商业化。这些项目中记载的最好的是 IBM 360 计算机家族所用的 OS 360 操作系统。这些大型项目中的成员很快就认识到建造大型软件系统与建造小型系统非常不同。将小型程序的开发技术扩展到大型软件开发有着根本的困难。术语“软件工程”就是在这个时候引入的，并举行会议来讨论这些项目在提交承诺的产品时面临的困难。大型软件项目普遍都超出预算并且落后于预定进度。这个时期引入的另一个术语是“软件危机”（software crisis）。

看来建造大型软件系统时碰到的问题不仅仅是将计算机指令放在一起的问题。确切地讲，这个正在着手解决的问题还没有得到很好的理解，至少还没有被项目中的每一个人所理解，项目中的每一个人还要用大量的时间互相进行交流，而不仅仅是编写代码。某些成员有时甚至会离开项目，这不仅会影响他们正在从事的工作，而且会影响到依赖于他们的其他人的工作。替换一个人要求大量关于项目需求和系统设计方面的培训。最初的系统需求的任何变动似乎都会影响到项目的很多方面，进而延迟系统的提交。这些类型的问题在早期的“程序设计”时代是不存在的，看来需要一种新方法才能解决。

解决方案已经提了很多，并且都得到了尝试。一些人认为解决方案在于更好的管理技术；另一些人则建议不同的团队组织方式；还有一些人则赞成更好的程序设计语言和工具。很多人要求组织范围的标准，诸如统一的编码约定等。少数人要求使用数学方法和形式化方法。所有这些观点都没有错。最后一致认为，解决软件建造问题应当采用与工程师建造其他大型复杂系统（诸如桥梁、精炼厂、工厂、船舶和飞机等）一样的方法。这种观点将最终的软件系统看成一种复杂的产品，而将其建造过程看成一项工程性的工作。工程方法需要有管理、组织、工具、理论、方法学和技术。自此软件工程应运而生。

Brooks 在 1987 年发表的一篇经典论文中提出，软件开发存在两种问题：根本问题和次要问题。次要问题是那些与我们当前所用的工具和技术有关的问题，例如，我们在使用程序设计语言时出现的语法问题。我们可以用更好的工具和技术来克服这些难题。然而对于根本问题，新工具却不能很好地奏效。对于复杂的设计问题，诸如建立和表示一个有助于预报天气或者经济情况的模型，要求脑力劳动、创造力和时间。Brooks 认为解决软件工程师面临的根本问题没有妙药，即没有“万能的药方”。

Brooks 的观点揭示了术语“软件危机”背后隐藏的错误假设。之所以发明该术语是因为软件工程项目不断地出现延误和超过预算的问题。结论是，该问题是临时的，并且可以借助于更好的工具和管理技术得到解决。实际上，项目延误是因为应用复杂，客户和开发者对应用理解不够，他们都不知道如何估计任务存在的难题以及要用多长时间才能解决这些难题。尽管有时仍然使用术语“软件危机”，但是大家一致认为软件开发的根本问题不是一个短期问题。新的和复杂的应用领域本来就是很难处理的，都不是在短期内能够迅速解决的问题。

前文所述的历史强调了软件工程的成长过程，它起源于程序设计。其他技术发展对该领域的成长也起到了重要作用。最重要的影响应当算硬件和软件价格平衡上的改变。虽然计算机化系统的成本过去主要取决于硬件成本，软件仅仅是一个次要因素，而现在软件组