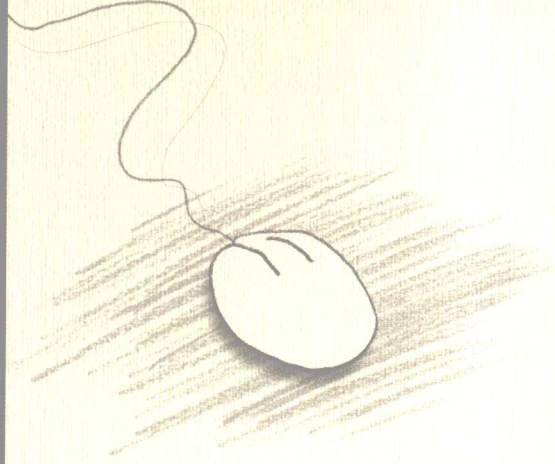

高等学校计算机实践教学系列教材

软件工程实践教程

谭庆平　毛新军　董　威　编著

高等教育出版社

高等学校计算机实践教学系列教材

软件工程实践教程

谭庆平　毛新军　董　威　编著

高等教育出版社

内容提要

本书以面向对象软件开发方法学、软件开发和管理的过程模型为主线，系统地介绍统一建模语言（UML）、基于 UML 的面向对象需求工程、基于 UML 的面向对象软件设计、面向对象的软件实现及软件测试、软件项目管理及软件工程过程模型。此外，本书还介绍了具有良好发展前景的先进的软件过程模型和软件开发技术，包括统一软件过程、敏捷软件开发和极限编程以及测试驱动、面向方面、面向 Agent 和模型驱动的软件开发方法。

本书是作者多年来在软件开发实践、软件工程科研和教学活动中积累沉淀的经验、体会和感悟的结晶。其结构设计和内容选材遵循实践性、可操作性、逻辑性、系统性、基础优先、技术和管理并重的原则，通过大量和系统的案例分析来诠释、演示软件工程技术和过程的综合使用方法，力求比较系统地解决长期困扰软件工程教育的“知易行难”问题。

本书可作为高等院校计算机专业、软件专业尤其是软件工程专业以及信息类相关专业高年级本科生和研究生“面向对象软件工程”课程的教材，还可作为软件开发人员、软件项目管理人员的技术参考书。

图书在版编目（CIP）数据

软件工程实践教程 / 谭庆平，毛新军，董威编著. —
北京：高等教育出版社，2009.4
ISBN 978-7-04-026437-1

Ⅰ. 软… Ⅱ. ①谭… ②毛… ③董… Ⅲ. 软件工程-教材 Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2009）第 032202 号

策划编辑 倪文慧　责任编辑 康兆华　封面设计 于文燕　责任绘图 尹　莉
版式设计 王　莹　责任校对 王　超　责任印制 宋克学

出版发行 高等教育出版社
社　　址 北京市西城区德外大街 4 号
邮政编码 100120
总　　机 010 - 58581000

经　　销 蓝色畅想图书发行有限公司
印　　刷 高等教育出版社印刷厂

开　　本 787 × 1092　1/16
印　　张 36.5
字　　数 820 000

购书热线 010 - 58581118
免费咨询 800 - 810 - 0598
网　　址 http://www.hep.edu.cn
　　http://www.hep.com.cn
网上订购 http://www.landraco.com
　　http://www.landraco.com.cn
畅想教育 http://www.widedu.com

版　　次 2009 年 4 月第 1 版
印　　次 2009 年 4 月第 1 次印刷
定　　价 35.00 元

物料号　26437 - 00

序　言

在现今人类社会所处的信息时代，软件已经发展成为信息技术的神经和灵魂。作为最典型的智力密集型产业之一，软件产业的竞争归根结底是人才的竞争。为了更好地履行造就高素质软件工程人才的使命，软件工程教育必须修正“重理论、轻实践，重技术、轻管理，重知识、轻体系”的错误倾向，通过系统梳理、见微知著、明辨本末，强化软件工程知识体系的系统性、逻辑性、实践性和可操作性，激发学习者在实践过程中学习的兴趣和热情，以图系统地解决长期困扰软件工程教育的“知易行难”问题。

基于上述理念，本书以当前主流的面向对象软件开发方法学、软件开发和管理的过程模型为主线，全面、系统、深入地介绍统一建模语言（UML）、基于 UML 的面向对象需求工程、基于 UML 的面向对象软件设计、面向对象的软件实现及软件测试、软件项目管理及软件工程过程模型。此外，本书还介绍了具有良好发展前景的先进的软件过程模型和软件开发技术，包括统一软件过程、敏捷软件开发和极限编程、测试驱动开发、面向方面编程、面向 Agent 的开发和模型驱动的体系结构。

本书是作者多年来在软件开发实践、软件工程科研和教学活动中积累沉淀的经验、体会和感悟的结晶。全书在结构设计和内容选材方面力求体系完整、条理清晰、语言简练、深入浅出。为突出软件工程技术和项目管理方法的实践性和可操作性，本书将案例分析分为两层：第一层案例紧随技术和方法的描述而出现，其规模较小，简明易懂，用来诠释概念、演示基本使用方法；第二层案例出现在章（节）的末尾，其规模稍大，用来演示各章大部分重要技术的综合性使用方法和使用过程。希望这些案例分析能够有助于提高学习者的软件开发能力和软件项目管理能力。

本书可供高等院校计算机专业、软件专业尤其是软件工程专业以及信息类相关专业高年级本科生和研究生用做“面向对象软件工程”课程的教材，也可作为《软件工程》（第二版，齐治昌等，高等教育出版社）一书的配套实践教程，同时也适合于软件开发人员、软件项目管理人员用做技术参考书。在教学计划中，如果讲授 40 学时，建议采用第一章至第八章的自然顺序讲授，其中带星号“*”的内容可酌情处理。如果以 24～32 学时讲授本书，对于高年级本科生，笔者建议的教学内容及讲授次序为：第一、二章→第四章至第七章，其中有关软件项目管理的内容可以跳过；对于已在本科阶段学习过软件工程的硕士研究生，建议的教学内容及讲授次序为：第一章（第 1.4 节除外）→第三章→第四章至第七章→第八章。实习部分以 20～28 学时为宜。

谭庆平教授负责组织本书的编著工作，并撰写了第一、二、四、五章。第三章和第八章由毛新军教授撰写，第六章和第七章由董威副教授撰写。齐治昌教授在本书的创作过程中给予了多方面的指导。本书内容曾多次在国防科技大学计算机专业本科生和硕士研究生的教学中讲授过。

南京大学骆斌教授认真审阅了本书的初稿，并提出了许多中肯的修改意见。借此机会，作者谨向为本书付出辛勤劳动和智慧的骆斌教授以及所有曾经和即将使用本书进行教学的教师、学生和软件工程师表示诚挚的谢意。作者还要特别感谢齐治昌教授，没有齐教授长期以来的信任、宽容、鼓励和支持，本书是不可能面世的。

最后，诚恳欢迎读者和专家对本书提出批评意见和改进建议。

作 者

2008 年 10 月

目　录

第一章 绪 论

在人类社会当今所处的信息时代，信息技术带来的影响广泛而深远。半个多世纪的时间在人类漫长的发展历史中只不过是惊鸿一瞥，信息技术却经由这段时期从稚嫩的幼苗成长为参天大树，其根须深植人类社会的诸多领域，枝叶荫及人类生活的方方面面，异彩纷呈的果实带给人们惊喜连连。回想 1946 年，以世界上首台计算机 ENIAC（electronic numerical integrator and calculator，电子数字积分器和计算器）的诞生为标志，信息化时代的大幕初启，曙光乍现。自此以后的数十年间，计算机硬件的研发一直占据信息化舞台的核心位置，软件只是作为硬件的配角而存在。由于硬件技术的高速发展，计算机的存储容量和计算能力不断翻番，制造成本却直线下降，于是，计算机的应用领域持续拓宽，应用问题的规模迅速扩大，复杂度也随之提高。这种情形在客观上以需求牵引的方式直接带动了软件技术的发展壮大。到 20 世纪 90 年代初，信息化舞台依然好戏连台，软件已经发展成为信息化技术的中枢神经和灵魂，硬件成为软件得以运行的物质基础。何以见得？有案例为佐证。

① 20 世纪 80 年代中期，曾经辉煌一时的美国王安电脑公司错判行业发展趋势，聪明一世的技术天才、磁芯存储器之父王安博士居然对软件技术带来的灵活性和开放性置若罔闻，坚持走封闭的纯硬件式电子办公产品制造路线，导致该公司在 20 世纪 90 年代初以惊人的速度从巅峰轰然坠落。待公司决策者们恍然醒悟，却为时已晚，徒留一部令人扼腕的高科技公司兴衰史。无独有偶，以 VAX 小型机和 VMS 操作系统而名扬天下的 DEC 公司的决策层重硬轻软，缺乏明智地售出数据库及其他软件业务，自废武功，却将独步天下的梦想寄托于 Alpha 硬件平台，结果惨遭市场无情淘汰。

② 除却沉重和唏嘘，信息技术的发展史中也有轻松和欢快的一面。微软公司依仗桌面操作系统和办公软件傲视群雄；Google 和百度公司凭借互联网信息搜索服务再度演绎令人叹为观止的崛起神话；硬件制造领域的全球标志性企业 IBM 公司顺势而为，不断削减泥足深陷的低端硬件制造业务，将更多的资源投入软件研发及其增值服务，遂有收购 Lotus 和 Rational 公司等震惊业界的大手笔。

在此从信息产业的结构布局、就业形势和人才需求的角度来看软件重要性的上升趋势。1993 年，软件和信息服务业在全球信息产业的产值构成中已占有 56%的比例，在美国，这一比例更是高达 60%，而我国目前这一比例尚在 20%左右徘徊，因此软件产业在我国的发展前景颇为广阔[1-1] [1-2]。在就业形势方面，据美国政府部门的权威统计数据显示，2002 年美国软、硬件工程师的从业人数之比高达 9:1，就连被外界认为归属制造业的波音公司，竟有一半以上的员工在从事软件及相关工作[1-3]。在人才需求方面，近年来无论国内还是国外，信息化人才持续紧俏，软件工程师，尤其是系统分析师和设计师更是长期高居稀缺人才榜之首[1-4]。

笔者决无意贬损硬件的重要性，正如灵魂必须依附于肉体，作为知识和思想载体的软件若离开硬件就会变成无本之木、无源之水。但是，软件确已摆脱先前的从属地位，已发展成为信息产业技术竞争的制高点，它不仅作为效应倍增器带动着传统产业的升级和改造，而且作为基础性、战略性产业攸关国家的经济安全和军事安全，成为综合国力中的关键要素之一。

信息产业，特别是软件产业的竞争，主要是人才的竞争。软件系统分析师、高级软件设计师、软件测试工程师，尤其是软件技术和管理方面的复合型人才在全国乃至全世界范围内一直都是稀缺资源，已经成为软件产业发展的主要瓶颈[1-4]。在这些人才的培养方面，软件工程教育义不容辞。长期以来，重理论、轻实践，重技术、轻管理、重知识、轻体系的错误倾向严重损害了软件工程教育的成效和声誉。全面地分析造成此类现象的诸多原因，甚或提出完整的解决方案已超出了本书的论述范围，但笔者愿意将自己对相关问题的零星思索在这里和盘托出，仅供读者参考。

① 软件工程的内涵丰富，其中的知识单元众多，如果不对它们之间的逻辑关系善加梳理，不去深入发掘知识点中蕴藏的思想和原理，就很容易导致初学者视软件工程为空洞的说教、毫无灵性的清规戒律，这样无疑将扼杀初学者的学习兴趣和热情。

② 软件工程具有知易行难的特点。从概念、原则到方法、过程，对它们的理解并非难事，但许多初学者在面对实际应用问题时要么一筹莫展，要么将软件工程的清规戒律抛诸脑后。

③ 近年来软件工程的发展令人目不暇接。除了 Java[1-5]、UML[1-6]（unified modeling language，统一建模语言），还有构件技术[1-7]、设计模式[1-8]、软件体系结构[1-9]；除了 Agent 技术[1-10]，还有面向方面编程[1-11]（aspect-oriented programming，AOP）、模型驱动的体系结构[1-12]（model driven architecture，MDA）；除了软件能力成熟度模型[1-13]（capability maturity model，CMM），还有统一软件过程[1-14]（rational unified process，RUP）、极限编程[1-15]（extreme programming，XP），等等。如果不能明辨本末，不能合理地决定经典内容与先进技术之间的权衡与取舍，软件工程教育势必事倍功半。

有鉴于此，本书的结构设计和内容选材遵循以下原则。

（1）可操作性

本书在阐述软件工程的每项技术或每种方法之后，即以案例进行解释并演示相应的使用过程。案例分为两层：第一层案例紧随技术和方法的描述而出现，其规模较小，简明易懂，用于诠释概念和基本使用方法；第二层案例出现在各章（第二章、第八章除外）的末尾，其规模稍大，用来演示该章中的重要技术的综合使用方法和过程。如果时间允许，笔者还计划在因特网上建立本教程的教学网站，在其中建设第三层软件项目案例库，其中所含案例的规模更大，应用问题的类型也更加丰富多彩，并且可以随时增加一些有趣的、更富有挑战性的项目案例，不断丰富其内涵。也诚恳地欢迎诸君贡献自己的经验和体会，为后学者指破迷津。

（2）逻辑性和系统性

本书以两条逻辑线索贯穿始终：面向对象的软件开发方法学，软件开发和管理的过程模型。这里的软件过程是指开发、维护软件及其相关产品（例如项目计划、需求分析文档、设计文档、

用户手册等）的一系列有序的技术开发活动和项目管理活动。从面向对象的基本概念，到基于UML 的需求分析、软件设计、软件实现、面向对象的软件测试，及至构件化开发方法、面向Agent 的软件工程，构成比较完整的软件开发技术体系；从软件过程的基本概念，到需求管理、项目计划、项目追踪、质量保证、配量管理，及至极限编程、统一软件过程、测试驱动的软件开发，构成比较完整的软件项目管理方法体系。在每条逻辑线索的内部，本书致力于揭示各环节之间的逻辑关系。在案例分析中，我们将两条线索有机地融于一体，构成**技术和管理并重**的软件工程实施方法学。

（3）基础优先

笔者以为，在庞大纷杂的软件工程知识体系中，堪称基础的知识体无非两条：面向对象开发方法学和软件工程过程。前者支撑起现代软件工程技术大厦，构件化方法、设计模式、软件体系结构、面向方面编程均可视为面向对象方法学的延伸或扩展；后者构成软件项目管理的基石，极限编程、统一软件过程、测试驱动的软件开发均建基于某种软件过程模型，只不过因为过程管理模块的组装、裁剪方式的不同而各有侧重，因此其适用范围也有所不同。由于篇幅有限，本书着重强调基础性内容，对于衍生型、扩展型的新技术、新方法则未详述。如果将来有机会，笔者希望创作一本《软件工程高级实践教程》之类的读物来弥补这一缺憾。

1.1 本书的结构及导读

全书共分 8 章、4 个逻辑部分。第一章和第二章构成全书的基础部分，其中第一章为绪论，依次介绍本书的创作思想、软件工程的观念、贯穿于全书的案例的初始问题描述，以及面向对象的基本概念与思想；第二章介绍作为面向对象开发方法的主流支持语言的 UML，概述其语法设计及基本使用方法。前两章的内容将为本书其余章节奠定概念和技术基础。第三章为有关软件项目管理的基础部分，其中包括软件过程的基本概念、软件过程模型、项目管理的主要活动及其目标和实施方法。将软件项目管理的基础内容置于全书的靠前位置，是因为在第三部分的软件开发过程中需要阐述各开发阶段必需的项目管理活动。第四章至第七章构成本书的主体部分，以面向对象的分析、设计、实现、测试为逻辑线索，环环相扣地串联起软件工程的技术和管理方法。第八章构成本书的软件工程过程（高级）模型及先进软件开发技术部分，其内容涵盖现今广为使用且具有良好发展前景的软件过程模型（敏捷软件开发和极限编程、测试驱动的软件开发）及软件开发技术（面向方面软件开发、面向 Agent 软件开发和模型驱动软件开发）。

对于软件工程的初学者，笔者建议依自然次序精读第一章至第七章，对第八章可以酌情处理（不读或粗读）。对于已经具备一定的软件工程知识基础、希望进一步提高软件工程能力的读者，笔者建议依自然次序读完全书内容，但对前两章可以酌情处理。

如果读者的兴趣聚焦于软件开发技术，对软件项目管理暂无暇顾及，那么第三章、第八章及第四章至第七章中有关项目管理的内容可以跳过或粗读；反之，第四章至第七章中有关软件开发技术的内容可以跳过或粗读。但是，在您做出此类取舍的决定之前，笔者愿再次强调：第

一，软件项目的成功取决于技术和管理活动的良好协调，二者不可偏废；第二，兼具软件工程技术和软件项目管理能力的复合型人才对于软件产业而言具有更重要的价值。

此外，本书目录中带“*”标记的章节不属于基础内容，其难度略高于其他部分，读者可根据自己的需要不读、粗读或精读，跳过它们并不会影响全书内容体系的连贯性。

鉴于软件工程的实践性特点，笔者强烈建议读者在阅读本书的过程中，选择各章习题中的部分应用问题，与阅读进度**同步**地开展软件项目的开发与管理活动，以达学以致用、事半功倍之效。

1.2 软件工程的观念

错误的观念和思维定式是妨碍软件工程学习，尤其是软件工程实践的隐形杀手。正确的观念不仅有助于学习者更深刻地理解软件工程的知识体系，而且有益于实践者将软件工程的原则、技术和方法转化为更富成效的**自觉**行为。本节针对初学者容易陷入的误区，阐述软件工程中一些重要的、基本的观念。

（1）中、大型软件项目的成功有赖于系统化的软件开发方法学和有纪律的软件工程过程

在软件技术发展历史的早期，曾经有人认为精巧的算法设计、巧妙的编程技巧就能确保软件项目取得成功；也有人认为增加软件项目组的人数就足以应付规模更大的应用问题。但是，为数众多的由软件引发的灾难性事故，以及更多的软件项目失败案例无情地宣告了作坊式、工艺式软件生产方式的幻灭。究其原因，人类个体智能在面对大型、复杂应用问题时表现出来的“百密必有一疏”的特征首当其冲，而任由个体自由发挥才智的多人软件开发又会因为个体的差异而导致误会丛生、效率低下、质量失控。痛定思痛，在克服软件危机的漫漫征途中，人们终于摸索到软件项目取得成功的不二法门：科学的、系统化的软件开发方法学可以帮助开发人员从技术上化解应用问题的复杂度；软件工程过程模型规定由合适的人、在合适的时间做合适的事情，以提高软件的质量并改善多人协同的有效性。

（2）软件质量要素不仅包含正确性、可靠性、有效性和可用性，也包含可理解性、可扩展性、可维护性

前 4 项要素构成软件质量的基础，也是软件工程追求的最主要目标。但是，对于一名优秀的软件工程师而言，后 3 项要素同样不容忽视，这是因为，对软件系统的修改几乎是必然的。这种修改既可源于用户需求的变更、运行环境的改变，也可能来自软件开发过程中软件设计、实现方案的修订，或者触发自测试阶段对于错误的发现。软件系统修改发生的时间段既可处于开发过程中，也可能位于软件维护阶段。如果软件在可理解性、可扩展性、可维护性方面存在较大的缺陷，那么软件系统修改的成本必然居高不下。更为严重的是，软件系统的修改较易引入质量隐患，导致屡经修改的软件在正确性、可靠性、有效性和可用性方面的质量每况愈下。此外，后 3 个质量要素的缺失也折损了软件作为知识载体的使用价值。

（3）用户优先

软件的价值来源于用户的使用和认可，而非软件内部的先进技术或精巧结构。因此，确立以用户为中心的观念，追求软件的易用性理所应当成为软件工程师的自觉行动。具体来说，在需求分析阶段，必须透彻地理解应用问题的相关业务背景和应用需求，设身处地为用户构思优化的软件应用场景；在后续的设计、实现及测试阶段要尽力确保软件的界面表现、功能、行为及性能符合或优于用户的需求；在软件文档的创作过程中，要特别重视安装手册、用户手册之类的文档，必须简明扼要、准确直观，避免使用与特定专业无关的术语，更要杜绝晦涩和含混的现象；在软件发布阶段，应当为用户提供简捷的安装（部署）工具，提升初始使用体验的愉悦度；在软件使用阶段之初，如果培训是必需的，则要系统策划、精心组织，不仅在培训过程中传授软件的使用方法，还可以考虑通过培训宣传软件产品的特色和优势，甚至收集用户的反馈意见，发掘潜在的软件易用性的改进点；在软件使用的过程中，不仅要做到及时响应、提供优质的人性化服务，还要从软件错误，甚至是用户的错误操作中反思用户文档的写作或软件的开发过程，谋求改进之道，避免重蹈覆辙。

这里必须强调用户界面设计对于软件易用性的极端重要性。优秀的用户界面具备以下特征：简明——界面元素数量适度、分组合理，尽可能减少需要用户干预的次数，降低手工录入的信息量，减轻用户的记忆负担，界面中的提示性或解释型文字应通俗易懂、一目了然，最好能提供单键触发式的帮助信息；直观——界面布局、操作次序、界面元素之间的关联性等方面符合业务操作规律和用户使用习惯，在耗时较长的处理过程开始之前完整地收集所需的用户输入（避免占用用户时间），在处理过程中及时反馈进度情况；标准——遵循操作系统平台或者业界公认的界面元素定义标准（例如，在 Microsoft Windows 平台上的应用软件通常以 Ctrl + O 快捷键触发文件打开操作），这样可以缩短用户的学习时间并与其使用习惯相吻合；防护——在执行破坏性的或数据难以恢复的操作之前必须提示用户加以确认，在适当的时机校验用户输入数据的合法性，动态禁用或隐藏不合时宜的界面元素，尽力避免因用户的误操作而导致越权访问、软件失效等情形；灵活——界面的布局、美学风格等可以动态定制，适应更多用户的审美情趣或操作习惯，以解众口难调之困；典雅——界面的色彩、装饰、布局应该美观而不失端庄。由是观之，软件的界面设计并非单纯的软件技术知识所能胜任的，还需要对应用问题所处的业务领域有深刻的理解，对用户心理和使用习惯有所洞察，以及对美学设计原则能够心领神会。有志于成为专业的用户界面设计师的读者可参考文献[1-16]、[1-17]、[1-18]。

（4）合理权衡

在从事软件设计与开发工作的人群中，精益求精、追求完美的倾向并不鲜见。这种倾向并非坏事，有时甚至可以作为一种宝贵的职业精神而广为提倡。但是凡事均须适度，否则物极必反，正所谓“水至清则无鱼”。软件工程师必须深谙“工程”之理：从某种意义上讲，在软件开发过程中，必须以全局利益为依归，在必要时适当地牺牲局部利益，容忍无关紧要的疵瑕。软件项目的成功绝不仅仅取决于完善的软件需求模型、“以不变应万变”的软件结构、精巧的算法设计，甚至也不单纯地取决于广义的软件质量要素。如果因为追求阶段产品的完美无瑕而致项目延期，痛失市场推广机会；如果因为对软件质量的苛刻要求而致成本超支，项目中途夭折，

那么前述的完美追求其意义何在？更何况，普天之下，何来绝对完美、一尘不染？因此，“权衡”在软件开发过程中势难避免，是软件技术和管理决策的主要内容。当然，在进行权衡时，如何恰到好处地把握折中度，取决于实际情况以及决策者的知识、经验和判断力。这也是软件工程“知易行难”的症结所在。从这个意义上讲，在软件工程领域，任何书本、理论知识都无法取代实践者个性化的经验和体会。

（5）放眼全局

优秀的软件工程师必须具有时间和空间两方面的全局观。时间全局观是指，从事软件开发和管理活动时不仅要着眼于当前，更要考虑未来，正所谓“不谋万世者，不足以谋一时”。例如，程序和文档中无意义的命名方式虽然可以减少字符的数量，但将来无疑会在软件系统修改过程中付出惨痛的代价，得不偿失。空间全局观是指，不仅要考虑问题和解决方案的侧面或局部，更要宏观地考虑整体和全局；不仅要考虑自己，更要考虑协作者的便利。例如，子系统的设计者不仅要理解、分析子系统自身的功能和性能方面的需求，还要在更大的范围内了解它与其他子系统之间的交互和协作关系，如此才能确定子系统的接口设计。又如，软件开发人员如果草率地报告自己的工作进度，就会给项目管理人员的计划追踪和调整造成困扰，进而影响后续的资源调配、阶段衔接等工作。依笔者观察，全局观是软件项目团队将软件过程纪律从约束转变为向导的关键推动因素。

（6）结构制胜

相对于模块内部之算法设计和程序实现而言，软件系统的结构对产品质量的影响程度更大。评判软件结构优劣的标准难有定论，但自然性和可伸缩性（scalability）应是软件结构设计者孜孜以求的重要目标。自然性是指软件结构直接反映应用领域中问题自身结构的程度；可伸缩性包含可扩展性和可裁剪性，其实质是软件结构接纳、容忍、适应修改的柔韧程度。为了搭建良好的软件结构，设计者必须具有自顶向下（top-down）的思维能力，意即在软件设计逐步精化过程中的每个抽象级别上，将关注点聚焦于当前抽象级别上的模块划分和模块接口设计，而将模块内部的结构或算法设计留待抽象级别降低后再予以解决，切忌囿于细节而忽视全局结构。在同一抽象级别上进行模块布局时，必须遵循著名的“强内聚、松耦合”原则，不要试图将不同性质、不同类别的软件元素混装在一起，更不要在软件模块之间设置“剪不断、理还乱”的交互式关联。软件系统内部的和谐来源于各模块尽可能独立地各司其职，仅在必要时才进行尽可能简单的协作。例如，适当分离软件需求中稳定的和易变的需求的实现模块，将有益于软件的可维护性；将数据持久存储服务从业务逻辑处理中分离出来，将使软件系统的大部分模块独立于数据存储机制（数据库管理系统或文件系统）。

（7）化繁就简

技术人员往往对高深的理论和方法情有独钟。但是，软件工程倡导用简单的方法求解复杂的问题，而非反其道而行之。当然，由于应用问题的规模和难度而导致的解决方案的复杂性是可以理解的，但是切忌杀鸡用牛刀，为炫耀技术而舍弃简单方法。如果较复杂的方法在某些方面（例如时空效率、灵活性、可扩展性、可复用性等）具有一定的优势，就需要视实际情况进行合理权衡、科学决策。必须特别强调的是，不能仅仅用数量（例如文档的页数、源代码的行

数）来衡量简单性，软件产品及其开发方法的简单性主要源自概念的简明、一致，结构的自然、直观。例如，对两种类图的设计进行比较，不能仅以类的多寡来判断其简单与否，更应该考察类及类间关系的设置是否自然、贴切、易理解、易实现。

千万不要认为简单方法比复杂方法更易获得，事实与此正好相反。同其他学科一样，简单的解决方案往往建立在设计者对问题的透彻理解、对技术的深刻领会之上，爱因斯坦的质量和能量转化方程 $E=mc^2$ 简单至极，但其中凝聚了多少智慧和灵感，对人类科技发展的影响又何其深远！

以上并非软件工程观念的全部，其他观念性的论题将随本书内容的展开而出现。希望读者在从事软件工程实践的过程中有意识地提炼自己独到的经验和体会，将其升华为观念，并将正确的观念渗入自己的潜意识中，再自觉地表现为合理、高效的软件开发和项目管理行为。

1.3 案 例 说 明

贯穿全书的案例共有两个：课程注册管理系统和机票预订管理系统。前者相对简单，主要用于解释软件工程中的概念、原则、技术和过程；后者规模稍大，更接近实际应用问题，用于综合演示每章所述的软件工程技术和过程的应用方法。

案例 1：课程注册管理系统

为某所学校的教务管理部门开发一个课程注册管理系统。在每学期开学前，教务管理人员可利用该系统输入课程信息、设定课表（每门课程的任课教师、上课时间和地点）。开学后，学生可利用该系统查询课程和课表信息，在第一周内注册课程或撤销对课程的注册。软件系统负责将学生所选课程的列表通知计费系统以确定学生应缴纳的选课费用。任课教师在学期内可随时查询选修其所授课程的学生的信息，学生可随时查询课程信息、课表、本人所选课程列表，教务管理人员可随时查询所有信息。

案例 2：机票预订管理系统

为某家机票预订服务商开发一个机票查询和预订管理系统。该系统中的航班和机票信息由多家航空公司负责提供。客户可以通过电话或者上网方式查询航班时刻表、机票可用信息、折扣信息，可以远程订票、退订、通过银行信用卡支付票款。客户分为三类：金牌客户、银牌客户、普通客户。普通客户通过订票积分至一定的额度可以逐步升级为银牌客户、金牌客户。不同类别的客户所享受的折扣可以不同。该系统还必须按每日、每周、每月给出机票销售的统计分析报表。

以上只是对本书案例的初步描述，随着对问题和需求的理解的不断深入，后续章节在引用这些案例时会有所拓展、发掘或精化。