教育部高等教育司推荐
国外优秀信息科学与技术系列教学用书

软件工程
——理论与实践
（第三版 影印版）

SOFTWARE ENGINEERING
Theory and Practice
(Third Edition)

■ Shari Lawrence Pfleeger
Joanne M. Atlee

高等教育出版社
Higher Education Press

教育部高等教育司推荐
国外优秀信息科学与技术系列教学用书

软件工程

——理论与实践

（第三版 影印版）

SOFTWARE ENGINEERING

Theory and Practice

(Third Edition)

Shari Lawrence Pfleeger

Joanne M. Atlee

高等教育出版社
Higher Education Press

图字：01-2006-0510 号

Software Engineering: Theory and Practice, Third Edition

Shari Lawrence Pfleeger, Joanne M. Atlee

原版 ISBN: 0-13-146913-4

图书在版编目（CIP）数据

软件工程：理论与实践：第 3 版＝Software Engineering：Theory and Practice，Third Edition /（美）弗莱格（Pfleeger，S. L.），（美）阿特利（Atlee，J. M.）—影印本. —北京：高等教育出版社，2006.9

ISBN 7-04-019876-2

Ⅰ. 软… Ⅱ. ①弗…②阿… Ⅲ. 软件工程—教材—英文 Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2006）第 108464 号

出版发行	高等教育出版社	**购书热线**	010—58581118
社　　址	北京市西城区德外大街 4 号	**免费咨询**	800—810—0598
邮政编码	100011	**网　　址**	http://www.hep.edu.cn http://www.hep.com.cn
总　　机	010—58581000	**网上订购**	http://www.landraco.com http://www.landraco.com.cn
经　　销	蓝色畅想图书发行有限公司	**畅想教育**	http://www.widedu.com
印　　刷	北京汇林印务有限公司		
开　　本	787×1092　1/16	**版　　次**	2006 年 9 月第 1 版
印　　张	47	**印　　次**	2006 年 9 月第 1 次印刷
字　　数	700 000	**定　　价**	48.00 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

物料号　19876—00

序

20世纪末，以计算机和通信技术为代表的信息科学和技术对世界经济、科技、军事、教育和文化等产生了深刻影响。信息科学技术的迅速普及和应用，带动了世界范围信息产业的蓬勃发展，为许多国家带来了丰厚的回报。

进入21世纪，尤其随着我国加入WTO，信息产业的国际竞争将更加激烈。我国信息产业虽然在20世纪末取得了迅猛发展，但与发达国家相比，甚至与印度、爱尔兰等国家相比，还有很大差距。国家信息化的发展速度和信息产业的国际竞争能力，最终都将取决于信息科学技术人才的质量和数量。引进国外信息科学和技术优秀教材，在有条件的学校推动开展英语授课或双语教学，是教育部为加快培养大批高质量的信息技术人才采取的一项重要举措。

为此，教育部要求由高等教育出版社首先开展信息科学和技术教材的引进试点工作。同时提出了两点要求，一是要高水平，二是要低价格。在高等教育出版社和信息科学技术引进教材专家组的努力下，经过比较短的时间，第一批引进的20多种教材已经陆续出版。这套教材出版后受到了广泛的好评，其中有不少是世界信息科学技术领域著名专家、教授的经典之作和反映信息科学技术最新进展的优秀作品，代表了目前世界信息科学技术教育的一流水平，而且价格也是最优惠的，与国内同类自编教材相当。

这项教材引进工作是在教育部高等教育司和高教社的共同组织下，由国内信息科学技术领域的专家、教授广泛参与，在对大量国外教材进行多次遴选的基础上，参考了国内和国外著名大学相关专业的课程设置进行系统引进的。其中，John Wiley公司出版的贝尔实验室信息科学研究中心副总裁Silberschatz教授的经典著作《操作系统概念》，是我们经过反复谈判，做了很多努力才得以引进的。William Stallings先生曾编写了在美国深受欢迎的信息科学技术系列教材，其中有多种教材获得过美国教材和学术著作者协会颁发的计算机科学与工程教材奖，这批引进教材中就有他的两本著作。留美中国学者Jiawei Han先生的《数据挖掘》是该领域中具有里程碑意义的著作。由

达特茅斯学院 Thomas Cormen 和麻省理工学院、哥伦比亚大学的几位学者共同编著的经典著作《算法导论》，在经历了 11 年的锤炼之后于 2001 年出版了第二版。目前任教于美国 Massachusetts 大学的 James Kurose 教授，曾在美国三所高校先后 10 次获得杰出教师或杰出教学奖，由他主编的《计算机网络》出版后，以其体系新颖、内容先进而倍受欢迎。在努力降低引进教材售价方面，高等教育出版社做了大量和细致的工作。这套引进的教材体现了权威性、系统性、先进性和经济性等特点。

教育部也希望国内和国外的出版商积极参与此项工作，共同促进中国信息技术教育和信息产业的发展。我们在与外商的谈判工作中，不仅要坚定不移地引进国外最优秀的教材，而且还要千方百计地将版权转让费降下来，要让引进教材的价格与国内自编教材相当，让广大教师和学生负担得起。中国的教育市场巨大，外国出版公司和国内出版社要通过扩大发行数量取得效益。

在引进教材的同时，我们还应做好消化吸收，注意学习国外先进的教学思想和教学方法，提高自编教材的水平，使我们的教学和教材在内容体系上，在理论与实践的结合上，在培养学生的动手能力上能有较大的突破和创新。

目前，教育部正在全国 35 所高校推动示范性软件学院的建设和实施，这也是加快培养信息科学技术人才的重要举措之一。示范性软件学院要立足于培养具有国际竞争力的实用性软件人才，与国外知名高校或著名企业合作办学，以国内外著名 IT 企业为实践教学基地，聘请国内外知名教授和软件专家授课，还要率先使用引进教材开展教学。

我们希望通过这些举措，能在较短的时间，为我国培养一大批高质量的信息技术人才，提高我国软件人才的国际竞争力，促进我国信息产业的快速发展，加快推动国家信息化进程，进而带动整个国民经济的跨越式发展。

教育部高等教育司

二〇〇二年三月

前　言

使用建模和设计在理论和实践之间架起桥梁

自 1968 年“软件工程”这个术语在一次北大西洋公约组织(North Atlantic Treaty Organization, NATO)的会议上首次使用以来，软件工程已经走过了很长的路。软件本身也以某种方式进入了我们的生活，甚至在十年前也很少有人能预见到这一点。因此，扎实的软件工程理论和实践基础对理解如何开发出优秀的软件和评价软件在生活中的风险和机遇是非常有必要的。本书体现了当前软件工程两个方面的结合：实践者的主要目标是构建能实现一定功能的优质产品，而研究者则努力找到改进产品质量以及提高产品生产率的方法。Edsgar Dykstra 常常提醒我们，理论和实践的困难检验了我们对软件工程的理解，帮助改进我们的思想、方法以及最终的产品。为问题和改进构建一个基本框架，这是我们在本书中所强调的精神。

在理论和实践之间构建框架和桥梁的关键在于建模和设计的思想。软件工程师比程序员编写更多的指令，就像厨师长比厨师烹饪更多的佳肴。构建一个优秀的软件是一门艺术，它体现在如何对问题的本质进行抽象和建模，并用这些抽象来设计解决方案的理解上。我们经常听到优秀的开发人员讨论“优雅的（elegant）”解决方法，这意味着这种解决方法体现了问题的核心，软件不仅解决了它当前形式下的问题，而且可以根据问题的发展变化而进行修改。正因如此，本书的第三版包含了大量如何对问题进行抽象和建模的材料，以及如何利用这些模型设计适当的解决方案。这样，学生学习如何将理论和实践相结合，将艺术和科学相结合，从而构建坚实的软件。

科学总是以事实为基础的。为了设计本科生的软件工程课程，本书绘制了一个软件工

程理论和实践的实用图，以使学生能将他们的所学直接应用到他们将要努力解决的真实问题中去。示例说明了学生经验有限，但清楚地表明了大型开发项目从需求到理想再到现实的进展。这些例子体现了读者可能遇到的很多情况：大的或小的项目，“灵活的”或高度结构化的方法，面向对象或面向过程的方法，实时方法或事务处理方法，开发和维护等。

本书也适用于介绍软件工程概念和实践入门的研究生课程，或那些希望拓展这些方面知识的专业人士。需要特别指出的是，第 12、13 和 14 章给出了引发读者思考的材料，它们专为对当前研究课题感兴趣的研究生而设计。

主要特点

本书具有许多区别于其他书的主要特征：

- 与其他书将度量和建模作为单独的问题来考虑不同，本书将度量和建模与软件工程的其他问题综合考虑。即将度量和建模作为软件工程学中不可分割的部分来考虑，而不是作为一个单独的问题。因此，学生学习如何抽象和建模，以及如何对日常活动进行评估和改进。他们可以利用模型来理解他们所要解决的问题或解决方案的重要部分，还可以利用度量来评价以个人、小组或项目为基础的进展。
- 同样地，复用、风险管理和质量工程等概念都结合到软件工程活动中去，软件工程受它们的影响，而不是将它们作为独立的问题来对待。
- 本版提出了敏捷的方法，包括极限编程方法（extreme programming）。介绍了给予开发者更多自治权的益处和风险，并将这种敏捷方法与更多的传统软件开发方法相比较。
- 每一章都将所介绍的概念应用到两个通用实例中：一个例子代表典型的信息系统，另一个例子代表实时系统。这两个例子都基于实际的项目。信息系统实例介绍了为一家大型英国电视公司确定广告时间与价格的软件，实时系统的实例则是 Ariane-5 的控制软件。我们考虑提出的问题，探讨软件工程是如何帮助确定并避

免这些问题的出现。学生可以跟随这两个项目的进展，看到本书介绍的不同实践方法是如何融入构建系统的技术中去的。

- 每章的最后用三种方式表达结果：本章内容对开发团队的意义，对开发人员个人的意义，以及对研究人员的意义。学生可以很容易地回顾每章的要点，并能看到每章与研究和实践的相关性。
- 本书有一个相关的 Web 网页，可以从 Prentice Hall 公司的 Web 网站 http://www.prenhall.com 获取。它包括了书中的例子和实际项目的真实结果的实例，还包括了与相关工具及方法的销售方的 Web 页面的链接。在此，学生可以找到实际的需求文档、设计、代码、测试计划以及更多内容，并为那些想查找具有一定深度信息的学生指导著名的、能够获取的出版物和 Web 网站。这些 Web 网页会定期更新，以保证书中的材料是最新的，网页中还包括了向作者和出版商提供反馈意见的途径。
- 本书还包括一张书附光盘，可以从 Prentice Hall 的代理商或该书的 Web 站点获取。光盘中包括了学生学习指南、解答手册，以及教师上课用的准备材料（如书中插图和表格的 PPT 幻灯片）。
- 本书采用了大量研究实例和文献中的例子。许多附注中的单页实例都在 Web 网页中得到详细扩充。学生可以从中学到如何将书中的理论概念应用到真实生活中。
- 本书的每一章都以引人深思的有关软件工程的法律和道德问题作为该章的结束。学生可以根据自己的社会和政治背景来理解软件工程。与其他科学一样，软件工程的决策必须根据它们对人们带来的影响做出评论。
- 每章都讨论了过程式的和面向对象的开发。此外，第 6 章中有关面向对象的内容解释了面向对象发展过程中的每一步。运用 UML 作为统一标识，从需求说明到程序设计，每一步都应用到一个通用例子中。
- 本书还包括一个带有注释的参考目录，它列出了许多有关软件工程的创造性论

文。另外，Web 网页中还包括了带注释的参考目录和专业讨论组，如软件的可靠性、容错性、计算机安全，等等。

- 每一章都包括一个学期课题的描述，它是一个借贷处理系统软件开发任务，教师可以将该项目或稍做变化后作为课外作业布置给学生。
- 每章的结尾都包括一个本章参考文献列表，它可以帮助学生找到本章所讨论的工具或方法的深度信息。

内容及组织

本书分为三部分。第一部分意在激励读者，并向从业人员和研究人员解释了软件工程知识的重要性。还讨论了理解过程的必要性，以决定开发人员的“灵活度”和完成项目计划。第二部分讨论了开发和维护的主要步骤，如需求的引出、建模和检查，设计问题的解决方案，代码的编写和测试，以及软件的交付，但未考虑用来构建软件的过程模型。第三部分讨论了软件的评估和改进，分析如何评价过程和产品的质量，并如何加以改进。

第 1 章：为什么需要软件工程？

本章描绘出全书的经脉，并激发读者的学习兴趣，突出将在后面章节中哪些地方解决有关的关键问题。特别地，介绍了 Wasserman 的帮助定义软件工程的关键要素：抽象、方法和符号的设计与分析、模块化和结构、软件生命周期和过程、复用、度量、工具和集成环境、用户接口和原形。本章讨论了计算机科学和软件工程的区别，阐明了可能遇到的主要问题类型，为本书的其余部分奠定基础。本章还探讨了采用一种系统方法来构建软件的必要性，介绍了每章中将要使用到的两个常用实例，以及学期课题的内容。

第 2 章　过程和生命周期的建模

本章总结了各种过程和生命周期模型的类型，包括流水线模型、V 模型、螺旋模型和各种原型化模型。说明了敏捷方法的必要性，与传统的软件开发方法相比，敏捷方法的开发人员被赋予更大的自主权。本章还介绍了几种建模技术和工具，包括系统动态模型、

SADT，以及几种其他常用的方法，并采用其中几种技术对两个通用实例的相关部分均进行了建模。

第3章 项目的计划和管理

本章研究项目的计划和管理。介绍了活动、里程碑、工作分解结构、活动图、风险管理、成本及成本估计等概念。采用了评估模型来评估本书两个通用实例的成本和进度。本章主要研究实例，包括F-16战斗机和Digital's alpha AXP程序的软件开发管理。

第4章 需求获取

本版中这一章得到了很大程度的加强。强调了在优秀的软件工程中抽象和建模的关键原则。特别地，本章使用模型来从已有的需求中梳理出误解和遗漏的细节，并与他人沟通需求。研究了多种不同的建模范例，为每种范例研究样本符号，讨论何时采用何种范例，并为如何做出特定的建模和抽象的决定提供建议。还讨论了需求的不同资源和不同类型（功能需求vs质量需求vs设计约束），解释如何编写易测试的需求，并描述如何解决冲突。另外，本章还讨论了需求引出、需求文档、需求评审、需求质量及度量，以及如何选择一个规格说明方法的示例。最后还将一些方法应用到本书的两个通用实例中去。

第5章 系统设计

本章主要研究软件体系结构问题，首先讨论了Shaw和Garlan两种软件体系结构的框架，然后介绍概念设计和技术设计的区别。讨论了完成设计人员的角色，介绍了两个基本的设计方法：组合法和分解法。接下来，确定了优秀设计的特征，介绍了几种设计策略，并给出了几个系统设计技术和工具的示例。本章中，读者还将学习到客户机-服务器的体系结构、可重用的设计组件、人机接口设计、安全可靠的系统设计（包括错误处理和容错能力）、设计模式、形式化设计方法，以及如何评定设计的折衷方案。在解释如何评估和验证一个设计的质量以及如何对结果进行归档后，接着又讨论了程序设计问题。

本章解释了程序设计的规则，包括自顶向下和自底向上的方法、模块性和独立性，以

及逻辑设计和物理设计之间的区别。讨论了并发系统和具有严格安全要求系统的设计，分析了导致 Therac-25 故障的设计缺陷。还讨论了几种设计工具，并全面地讨论了设计质量以及如何度量设计质量。本章介绍了设计复用、评定、审查，并解释了文档设计基本原理。最后，本章给出了信息系统和实时系统的设计实例。

第 6 章 对象

这一章回过头来考虑面向对象开发的特性。首先介绍用例场景，讨论如何从自然语言中获取对象和它们的特征。采用第 4 章中介绍的建模工具，将需求表达为各种不同的模型，一并考虑，从多种角度描述手边的问题。接着，分析系统设计，以了解如何采用模型来生成解决问题所需的高级信息。然后，强化系统设计，增加了非功能性需求和程序设计中所需的细节。采用 UML 及其模式为皇家服务站生成了一个面向对象的规格说明和设计。

本章详细分析了面向对象的度量问题，将一些常用的面向对象的度量方法应用到皇家服务站的例子中去。指出如何采用度量的变化来帮助我们决定如何分配资源和查找错误。最后，将面向对象的概念应用于信息系统实例和实时系统实例。

第 7 章 程序编写

本章介绍了代码级设计决策问题，以及有关高质量代码设计的实现问题。讨论了标准和过程，给出了一些简单的编程指南。提供了用几种语言编写的示例，包括面向对象语言和面向过程语言。本章讨论了程序文档和错误处理策略的必要性，最后，将其中一些概念应用于本书的两个通用实例。

第 8 章 程序测试

本章探讨了程序测试的几个方面，区分了传统测试方法与“净室”（cleanroom）测试方法的不同，并分析了如何测试不同的系统。提出了软件问题的定义和分类，并讨论如何使用正交错误分类使得数据收集和分析更有效。之后解释了单元测试和集成测试的区别。在介绍几种自动测试工具和技术之后，解释了测试生命周期的重要性，以及如何将工具集

成到其中。最后，本章将这些概念应用于书中的两个通用实例。

第 9 章　系统测试

本章首先介绍了系统测试的原则，包括测试集和数据的复用，以及仔细配置管理的必要性。介绍的概念包括功能测试、性能测试、验收测试和安装测试。分析了测试面向对象系统的特殊需要，介绍了几个测试工具，并讨论了小组成员的任务。接着，向读者介绍了软件可靠性建模，并讨论了软件的易用性、可维护性和可用性。读者从中可学习如何使用测试结果评估交付产品的性能，同时还介绍了几种测试文档类型，并以介绍本书两个通用实例的测试策略作为本章的结束。

第 10 章　系统交付

本章讨论了培训和文档的必要性，提供了几个信息系统和实时系统例子中的培训和文档的示例。

第 11 章　系统维护

本章分析了系统改变的后果。解释了在系统的生命周期中改变是如何发生的，系统设计、代码、测试过程和文档如何适应变化。讨论了典型的维护问题，以及仔细配置管理的必要性。本章全面地讨论了采用度量来预测可能发生的变化，并评估变化带来的影响。考虑了对遗留系统进行再生工程和重构造。最后，根据可能的变化对书中两个通用实例进行了评估。

第 12 章　产品、过程和资源的评价

由于许多软件工程的决策涉及到对现有组件的合并和集成，本章给出了评价过程和产品的方法。讨论了经验评估的必要性，并给出几个例子以体现如何用测量标准来建立质量和生产率的基准。分析了几个质量模型，如何评估系统的复用性，如何完成算后检查（post-mortems），以及如何理解信息技术投资的回报。所有这些概念都被应用于书中的两

个通用实例中。

第 13 章 预测、产品、过程和资源的改进

本章建立在第 11 章的基础之上，展示了如何实现预测、产品、过程和资源的改进。它包括几个深入的实例研究，以说明如何理解预测模型、检测技术以及软件工程其他方面的问题，并使用多种研究性技术来加以改进。本章的结尾给出了一组对评估当前状态和确定改进的机会的指导方针。

第 14 章 软件工程的前景

本书的最后一章研究几个有关软件工程的开放性问题。回顾了 Wasserman 的概念，将它作为一个原则来了解我们做得如何。本章还分析技术转移和制定决策的几个问题，以决定是否做好了从理论到实践的重要转变工作。在这一版本中，新增了对几个有争议问题的分析，如软件工程作为专业工程的许可，以及更多特殊领域解决方案和方法的趋势。

致谢：

本书的完成要归功于朋友和家人提供的技术和情感的支持。在此，我们不可能列出所有在本书的编写和校对过程中曾经支持过我们的人，所以首先要为可能的疏忽而致歉。非常感谢本书以前版本的读者，他们的仔细阅读对本书的修订提供了极好的建议。据我们所知，所有这些建议都收编到了这个版本中，我们将继续重视读者的反馈，不管是正面的还是反面的。

Carolyn Seaman (马里兰大学，巴尔的摩校区)是本书第一版的很棒的评论家，提出了既清晰又简单的方法，使本书更为紧凑，更易于理解。同时，她还为本书的大多数习题提供了解答，并帮助建立了本书早先的 Web 网站。我非常感激她的友谊和帮助。Yiqing Liang 和 Carla Valle 更新了这个网站，并为本书的第二版补充了新的资料。Maria Augusta Vieira Nelson (Minas Gerais 的 Catholic University, Brazil)为第三版修改了该网站，增加了建模符号和敏捷方法的内容。

我们将极大的感谢献给 Forrest Shull(马里兰大学，夫琅和费中心)和 Roseanne Tesoriero (Washington University)，他们规划了本书最早的学习指南，还要感谢 Maria Nelson，他修订了第三版的学习指南和解答手册。我们同样十分感激 Guilherme Travassos (Rio de Janeiro 的 Federal University)，我们采用了他与 Maryland-College Park 大学的 Pfleeger 开发所用的材料，后者丰富并扩展了子类的使用。

所有三个版本中，那些提供了很多帮助和很有见地的评论家包括：Barbara Kitchenham (Keele 大学，英国)、Bernard Woolfolk (Lucent 科技)、Ana Regina Cavalcanti da Rocha (Rio de Janeiro 的 Federal 大学)、Frances Uku（Berkeley 的 California 大学）、Lee Scott Ehrhart（MITRE）、Laurie Werth (Texas 大学)、Vickie Almstrum (Texas 大学)、Lionel Briand (Carleton 大学, Ottawa)、Steve Thibaut (Florida 大学)、 Lee Wittenberg (New Jersey 的 Kean 学院)、Philip Johnson (Hawaii 大学)、Daniel Berry (Waterloo 大学，加拿大)、Nancy Day (Waterloo 大学)、Jianwei Niu (Waterloo 大学)、Chris Gorringe (East Anglia 大学，英国)、Ivan Aaen (Aalborg 大学)，以及一些由 Prentice Hall 提供的匿名评论家。与 Greg Hislop（Drexel 大学）、John Favaro (Intecs Sistemi，意大利)、Filippo Lanubile (Università di Bari，意大利)、John d'Ambra (New South Wales 大学，澳大利亚)、Chuck Howell (MITRE)、Tim Vieregge (美国 Army Computer Emergency Response Team)、James 和 Suzanne Robertson (Atlantic System Guild，英国)的讨论使本书得到许多改进和增强。

再次感谢 Toni Holm 和 Alan Apt，是他们使此第三版本变得有趣而轻松，也感谢 James 和 Suzanne Robertson，他们使用了 Piccadilly 的例子；还有 Norman Fenton，他使用了书中的软件度量材料。

非常感谢那些允许某些图和示例复制到本书中的出版商们。从 *Complete Systems Analysis* (Robertson and Robertson 1994)、*Mastering the Requirements Process* (Robertson and Robertson 1999) 中选用的材料得到了 Dorset House Publishingr 的许可，在 www.dorsethouse.com，保留所有权利。练习 1.1 中的材料是经 Associated Press 许可后从

Washington Post 复制过来的。图 12.15 和 12.16 是经 John Wiley 和 Sons Limited 许可后从 Barghouti 等（1995）中复制而来的。图 12.14 和 12.15 是经 John Wiley 和 Sons Limited 许可后从 Rout (1995)中复制而来的。

第 2、3、4、5、9、11、12 和第 14 章中注明为 IEEE 版权的图和表是经电气电子工程师协会许可的再版。同样地，第 14 章中注明为 ACM 版权的三张表均是得到美国计算机协会许可的再版。表 2.1 和图 2.11（Lai,1991）是经软件生产力联盟许可后的复制。图 8.16 和 8.17 (Graham,1996a)是经 Dorothy R. Graham 许可后的复制。图 12.11 和表 12.2（Liebman, 1994）是经 Center for Science in the Public Interest, 1875 Connecticut Avenue NW, Washington DC 许可后所采用。表 8.2、8.3、8.5 和 8.6 是经 McGraw-Hill 公司许可后所复制。源于 Shaw 和 Garlan（1996），Card 和 Glass（1990），Grady（1997），以及 Lee 和 Tepfenhart (1997) 的图和表是经 Prentice Hall 公司许可后所复制。

表 9.3、9.4、9.6、9.7、13.1、13.2、13.3 和 13.4，以及图 1.15、9.7、9.8、9.9、9.14、13.1、13.2、13.3、13.4、13.5、13.6 和 13.7 是经 Norman Fenton 部分或完全的许可，从 Fenton 和 Pfleeger (1997)中复制或改编过来的。图 3.16、5.19 和 5.20 是从 Norman Fenton 的课程笔记中复制或改编过来的，感谢他友善的许可。

特别感激我们的职员、RAND 公司和 Waterloo 大学，尤其是他们在我们准备此版本时对我们的鼓励。同时，也感谢我们的朋友和家人，编写此书占用了他们的时间，感谢他们付出的善意、支持和耐心。特别地，Shari Lawrence Pfleeger 感谢 Manny Lawrence；感谢皇家服务站的经理和他的会计 Bea Lawrence，不仅感谢她的工作和她的学生对皇家系统的规格说明，而且也包括他们其他工作带来的影响和指导：像她的父母一样；Jo Atlee 特别感谢她的父母： Nancy 和 Gary Atlee，他们支持和鼓励她所做的任何事情（包括尝试），还感谢她的同事和学生，他们在她写作的主要时期友好地分担了很多工作。最为特别的是，我们要感谢 Charles Pfleeger 和 Ken Salem，谢谢他们的支持、鼓励和幽默感。

Preface

USING MODELING AND DESIGN TO BRIDGE THE GAP BETWEEN RESEARCH AND PRACTICE

Software engineering has come a long way since 1968, when the term was first used at a NATO conference. And software itself has entered our lives in ways that few had anticipated, even a decade ago. So a firm grounding in software engineering theory and practice is essential for understanding how to build good software and for evaluating the risks and opportunities that software presents in our everyday lives. This text represents the blending of the two current software engineering worlds: that of the practitioner, whose main focus is to build high-quality products that perform useful functions, and that of the researcher, who strives to find ways to improve the quality of products and the productivity of those who build them. Edsgar Dykstra continually reminded us that rigor in research and practice tests our understanding of software engineering and helps us to improve our thinking, our approaches, and ultimately our products. It is in this spirit that we have enhanced our book, building an underlying framework for this questioning and improvement.

Key to building the framework and building a bridge between research and practice are the notions of modeling and design. Software engineers are more than programmers following instructions, much as chefs are more than cooks following recipes. There is an art to building good software, and the art is embodied in understanding how to abstract and model the essential elements of a problem and then use those abstractions to design a solution. We often hear good developers talk about "elegant" solutions, meaning that the solution addresses the heart of the problem, such that not only does the software solve the problem in its current form but it can also be modified as the problem evolves over time. For this reason, this third edition contains extensive material about how to abstract and model a problem, and how to use the models to design appropriate solutions. In this way, students learn to blend research with practice and art with science, to build solid software.

The science is always grounded in reality. Designed for an undergraduate software engineering curriculum, this book paints a pragmatic picture of software engineering research and practices so that students can apply what they learn directly to the real-world problems they are trying to solve. Examples speak to a student's limited experience but illustrate clearly how large software development projects progress from need to idea to reality. The examples represent the many situations that readers are likely to experience: large projects and small, "agile" methods and highly structured ones, object-oriented and procedural approaches, real-time and transaction processing, development and maintenance situations.

The book is also suitable for a graduate course offering an introduction to software engineering concepts and practices, or for practitioners wishing to expand their

knowledge of the subject. In particular, Chapters 12, 13 and 14 present thought-provoking material designed to interest graduate students in current research topics.

KEY FEATURES

This text has many key features that distinguish it from other books.

- Unlike other software engineering books that consider measurement and modeling as separate issues, this book blends measurement and modeling with the more general discussion of software engineering. That is, measurement and modeling are considered as an integral part of software engineering strategies, rather than as separate disciplines. Thus, students learn how to abstract and model, and how to involve quantitative assessment and improvement in their daily activities. They can use their models to understand the important elements of the problems they are solving as well as the solution alternatives; they can use measurement to evaluate their progress on an individual, team and project basis.
- Similarly, concepts such as reuse, risk management, and quality engineering are embedded in the software engineering activities that are affected by them, instead of being treated as separate issues.
- The current edition addresses the use of agile methods, including extreme programming. It describes the benefits and risks of giving developers more autonomy and contrasts this agility with more traditional approaches to software development.
- Each chapter applies its concepts to two common examples: one that represents a typical information system, and another that represents a real-time system. Both examples are based on actual projects. The information system example describes the software needed to determine the price of advertising time for a large British television company. The real-time system is the control software for the Ariane-5 rocket; we look at the problems reported, and explore how software engineering techniques could have helped to locate and avoid some of them. Students can follow the progress of two typical projects, seeing how the various practices described in the book are merged into the technologies used to build systems.
- At the end of every chapter, the results are expressed in three ways: what the content of the chapter means for development teams, what it means for individual developers, and what it means for researchers. The student can easily review the highlights of each chapter, and can see the chapter's relevance to both research and practice.
- The book has an associated Web page, reachable from Prentice Hall's corporate Web site, http://www.prenhall.com. It contains current examples from the literature and examples of real artifacts from real projects. It also includes links to Web pages for relevant tool and method vendors. It is here that students can find real requirements documents, designs, code, test plans, and more. Students seeking additional, in-depth information are pointed to reputable, accessible publications and Web sites. The Web pages are updated regularly to keep the material in the textbook current, and include a facility for feedback to the author and the publisher.