免费提供
电子教案

高等院校规划教材
软件工程系列

软件工程实践教程

第2版

刘冰 刘锐 瞿中 吴渝 编著

机械工业出版社
CHINA MACHINE PRESS

高等院校规划教材·软件工程系列

软件工程实践教程

第2版

刘 冰　刘 锐　瞿 中　吴 渝　编著

机 械 工 业 出 版 社

本书详细介绍了软件工程、软件开发过程、软件计划、需求分析、总体设计、详细设计、编码、软件测试、软件维护、软件工程标准化和软件文档、软件工程质量、软件工程项目管理以及软件工程开发实例。各章均配有习题，部分章后附有经典例题讲解和实验内容。

本书可作为高等学校计算机专业课程的教材或教学参考书，也可作为通信、电子信息、自动化等相关专业的计算机课程教材，还可供软件工程师、软件项目管理者和应用软件开发人员阅读参考。

图书在版编目（CIP）数据

软件工程实践教程/刘冰等编著．—2版．—北京：机械工业出版社，2012.3
高等院校规划教材·软件工程系列
ISBN 978-7-111-37759-7

Ⅰ.① 软… Ⅱ.① 刘… Ⅲ.① 软件工程-高等学校-教材
Ⅳ.① TP311.5

中国版本图书馆CIP数据核字（2012）第047415号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
责任编辑：郝建伟 曹文胜
责任印制：乔 宇

三河市宏达印刷有限公司印刷

2012年5月第2版·第1次印刷
184mm×260mm·25.75印张·636千字
0001-3000册
标准书号：ISBN 978-7-111-37759-7
定价：49.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
社服务中心：（010）88361066
销售一部：（010）68326294
销售二部：（010）88379649
读者购书热线：（010）88379203

网络服务
门户网：http://www.cmpbook.com
教材网：http://www.cmpedu.com

封面无防伪标均为盗版

出 版 说 明

计算机技术的发展极大地促进了现代科学技术的发展，明显地加快了社会发展的进程。因此，各国都非常重视计算机教育。

近年来，随着我国信息化建设的全面推进和高等教育的蓬勃发展，高等院校的计算机教育模式也在不断改革，计算机学科的课程体系和教学内容趋于更加科学和合理，计算机教材建设逐渐成熟。在“十五”期间，机械工业出版社组织出版了大量计算机教材，包括“21世纪高等院校计算机教材系列”、“21世纪重点大学规划教材”、“高等院校计算机科学与技术‘十五’规划教材”、“21世纪高等院校应用型规划教材”等，均取得了可喜成果，其中多个品种的教材被评为国家级、省部级的精品教材。

为了进一步满足计算机教育的需求，机械工业出版社策划开发了“高等院校规划教材”。这套教材是在总结我社以往计算机教材出版经验的基础上策划的，同时借鉴了其他出版社同类教材的优点，对我社已有的计算机教材资源进行整合，旨在大幅提高教材质量。我们邀请多所高校的计算机专家、教师及教务部门针对此次计算机教材建设进行了充分的研讨，达成了许多共识，并由此形成了“高等院校规划教材”的体系架构与编写原则，以保证本套教材与各高等院校的办学层次、学科设置和人才培养模式等相匹配，满足其计算机教学的需要。

本套教材包括计算机科学与技术、软件工程、网络工程、信息管理与信息系统、计算机应用技术以及计算机基础教育等教材系列。其中，计算机科学与技术系列、软件工程系列、网络工程系列和信息管理与信息系统系列是针对高校相应专业方向的课程设置而组织编写的，体系完整，讲解透彻；计算机应用技术系列是针对计算机应用类课程而组织编写的，着重培养学生利用计算机技术解决实际问题的能力；计算机基础教育系列是为大学公共基础课层面的计算机基础教学而设计的，采用通俗易懂的方法讲解计算机的基础理论、常用技术及应用。

本套教材的内容源自致力于教学与科研一线的骨干教师与资深专家的实践经验和研究成果，融合了先进的教学理念，涵盖了计算机领域的核心理论和最新的应用技术，真正在教材体系、内容和方法上做到了创新。同时本套教材根据实际需要配有电子教案、实验指导或多媒体光盘等教学资源，实现了教材的“立体化”建设。本套教材将随着计算机技术的进步和计算机应用领域的扩展而及时改版，并及时吸纳新兴课程和特色课程的教材。我们将努力把这套教材打造成为国家级或省部级精品教材，为高等院校的计算机教育提供更好的服务。

对于本套教材的组织出版工作，希望计算机教育界的专家和老师能提出宝贵的意见和建议。衷心感谢计算机教育工作者和广大读者的支持与帮助！

机械工业出版社

前　言

《软件工程实践教程》第2版与2009年出版的《软件工程实践教程》第1版相比，增加了软件开发的实际案例分析和一些实用工具的使用等内容，并对基础知识部分的内容重新进行了编写。各章的习题也进行了重新组织和编写。

编写中，我们不仅重视将实践与理论相结合，在教学内容上也做了较大调整。为使学生掌握规范化的软件开发方法，本书强调结构化开发方法中软件生命周期各阶段的活动和产品，同时也加强了面向对象的系统分析与设计方法的介绍，并且将这部分的内容与“面向对象分析与设计”、“UML统一建模语言”、“软件测试”等课程的教学内容相结合，使学生能更深刻地理解这种先进的软件开发思想。

本书介绍了如下三方面的内容。

1）软件工程导论：包括软件与软件工程的概念、传统的开发方法、面向对象的开发方法、实现与测试、质量与质量保证、软件计划与管理、软件开发工具与环境等内容。通过对这部分内容的学习，让学生从方法、工具、过程三方面熟悉软件工程的基本概念、原理和技术。掌握需求分析和软件设计的一般方法、理解开发高质量软件的方法以及如何进行软件度量和管理。

2）软件建模技术：包括面向对象的软件建模与构造的基本概念、原理和方法。通过这部分内容的学习，学生应理解面向对象的软件设计与构造；掌握面向对象的基本概念、基本原理；掌握OOD的过程；深入理解OOD的每个阶段；掌握UML建模语言的一般方法及过程。

3）软件项目组织与管理：包括软件项目管理领域中所含的计划方法、活动排序与跟踪、质量保证、配置管理、风险防御、质量度量方法、软件人员的管理方法等内容。系统介绍了如何在CMM的开发模式框架下，利用这些技术完成软件项目的管理活动。

在软件工程课题组教师参与下，制作了本课程的多媒体课件。随着学科发展、知识更新和教材的更改，已经过了多次大的调整、补充和完善。

本书是在《软件工程实践教程》第1版的基础上改进完成的，由刘冰、刘锐、瞿中、吴渝编著。本书的顺利出版，得到了相关同事的大力支持和帮助，也得到了计算机教育界许多同仁的关心，在此一并致谢。

与本教材配套的电子教案和习题答案可以从机械工业出版社网站下载，网址为www. cmpedu. com，或与作者联系，邮箱为：cobly1837@ sina. com。

软件工程发展迅速，国内外有关软件工程技术与设计方面的资料很多，新理论、新技术层出不穷，由于编者水平有限，难免存在不足和错误之处，恳请读者批评指正。您的宝贵建议和意见请发至：jsjfw@ mail. machineinfo. gov. cn。

作　者

目　录

第1章　软件工程概述

本章要点

- 软件
- 软件工程
- 软件过程
- 软件过程模型

1.1　软件

一个完整的计算机系统是由软件和硬件组成的，它们相互依存、缺一不可。IEEE 给软件下的定义为：软件是计算机程序、规程以及运行程序所需要的相关文档和数据。软件是计算机程序和对该程序的功能、结构、设计思想以及使用方法等整套文字资料的说明。计算机程序是指为了得到某种结果而可以由计算机等具有信息处理能力的装置执行的代码化指令序列，或者可被自动转换成代码化的符号化指令序列。计算机程序包括源程序和目标程序。文档是指用自然语言或者形式语言所编写的文字资料和图表，用来描述程序的内容、组成、设计、功能规格、开发情况、测试结果及使用方法，如程序设计说明书、流程图、用户手册等。

根据软件服务对象的范围不同，软件可以划分为系统软件和应用软件两种类型。

（1）系统软件

系统软件负责管理、控制、维护、开发计算机的软硬件资源，提供给用户一个便利的操作界面，也提供编制应用软件的资源环境。系统软件主要包括操作系统，另外还有程序设计语言及其处理程序和数据库管理系统等。

操作系统在软件系统中居于核心地位，负责对所有软件、硬件资源进行统一管理、调度及分配，是用户和计算机的一个接口。

程序设计语言是供程序员编制软件，实现数据处理的特殊语言，语言处理程序提供对程序进行编辑、解释、编译和链接的功能。

数据库管理系统（DBMS）也是一类重要的系统软件，因为大量的应用软件都需要数据库的支持，如信息管理系统、电子商务系统等。目前比较流行的数据库管理系统有 Microsoft SQL Server、Oracle、Sybase 和 Informix 等。

（2）应用软件

应用软件是指为了解决某一领域的具体问题而编写的软件产品。可以是一个特定的程序，也可以是一组功能联系紧密，可以相互协作工作的程序的集合，也可以是一个由众多独立程序组成的软件系统。常见的应用软件有企业 ERP 系统、图形、图像处理软件、办公自动化软件、各类信息管理软件、卫星控制系统和空中交通指挥系统等。应用软件因其应用领

域的不同而丰富多彩。

1.1.1 软件的特点

（1）工具性特征

计算机软件包括程序和文档两个部分。计算机程序指“为了得到某种结果而可以由计算机等具有信息处理能力的装置执行的代码化指令序列，或者可被自动转换成代码化指令序列的符号化指令序列或者符号化语句序列”。计算机程序包括源程序和目标程序，源程序是用计算机高级语言（如 BASIC、Algol、Cobol、Fortran 等）编写的程序，表现为数字、文字和符号的组合，构成符号化指令序列或符号化语句序列；目标程序是用机器语言编写的，体现为电脉冲序列的一串二进制数（0 和 1）指令编码，直接用于驱动计算机硬件工作，保证计算机系统发挥各项功能，获得一定结果，因而又具有工具性特征。软件在调入计算机运行之前，首先表现为作品性，人们无法通过“阅读”或“欣赏”计算机程序与文档而制造任何有形产品和实现任何操作。但是，软件调入计算机运行时，则更主要地表现为工具性，即通过控制计算机硬件动作过程，获得某种结果。

（2）软件开发工作量大、成本高，但复制容易、成本低

软件开发必须经过功能限定、逻辑设计和编码三个步骤，要求软件开发人员必须具有丰富而超前的专业和相关知识，极强的逻辑和形象思维能力，了解计算机硬件的最新发展状况与发展前景，熟练掌握和使用编程语言。开发具有实用商业价值的计算机软件，通常需要按照专业化分工、流水线作业的方式由一批人共同完成，少数人几乎不可能开发计算机软件。可见，开发计算机软件必须具备相应的物质和技术条件，有充足的开发资金和良好的开发环境。计算机软件的复制是指把软件转载于有形物体的行为，如把软件打印在纸上或穿孔在卡片上，把软件转存于磁盘、磁带或 ROM 芯片等载体中。复制是对计算机软件的客观再现，不改变软件内容和本身的价值，复制后的软件以一定的客观物质形式体现，具有可感知性。计算机软件的可复制性决定其可以广泛传播和有效利用，创造经济和社会效益。

（3）软件具有无形性，可以多次使用

计算机软件是智力劳动产生的精神产品，如计算机程序、说明程序的文档等都是智力劳动的直接产物，不具有任何形状，人们只有借助于一定的物质载体和工具才能感知其存在。计算机软件在同一时间可以为若干人分别使用，软件只要不受操作失误、计算机病毒等影响，就可以无限制反复使用。但是，计算机软件又具有工具性，主要通过“使用”而发挥其功用，因而应该具有使用寿命，使用寿命在流通领域表现为商业寿命。在科学技术飞速发展、新软件层出不穷的今天，计算机软件的商业寿命正日益缩短。一般而言，10 年以上的软件效率差，实用价值不大，已很难有效占领市场。

1.1.2 软件的发展

软件的发展大致经历了三个阶段：

第一阶段（20 世纪 40 年代到 50 年代中期）是软件发展的初期。在这个阶段，软件开发采用低级语言，开发效率比较低，应用领域基本局限于科学和工程的数值计算，人们不重视软件文档的编制，只注重代码的编写。

第二阶段（20 世纪 50 年代中期到 60 年代后期）。相继诞生了大量的高级程序设计语

言，程序开发的效率明显提高，并产生了成熟的操作系统和数据库管理系统。在后期，由于软件规模不断扩大，复杂程度大幅度提高，产生了“软件危机”，同时也出现了有针对性地进行软件开发方法的理论研究和实践。

第三阶段（20 世纪 70 年代至今）。软件应用领域和规模持续扩大，大型软件的开发成为一项工程性的任务，由此产生了“软件工程”并得到长足发展。同时软件开发技术继续发展，并逐步转向智能化、自动化、集成化、并行化和工程化。

1.1.3 软件危机

20 世纪 60 年代以前，计算机刚刚投入实际使用，软件设计往往只是为了一个特定的应用而在指定的计算机上设计和编程，采用密切依赖于计算机的机器代码或汇编语言，软件的规模比较小，文档资料通常也不存在，很少使用系统化的开发方法，设计软件往往等同于编制程序，基本上是个人设计、个人使用、个人操作、自给自足的私人化、作坊式的生产方式。60 年代中期，大容量、高速度计算机的出现，使计算机的应用范围迅速扩大，软件开发急剧增长。高级语言的出现和操作系统的发展引起了计算机应用方式的变化，大量数据处理导致第一代数据库管理系统的诞生。软件系统的规模越来越大，复杂程度越来越高，软件可靠性问题也越来越突出。原来的个人设计、个人使用的方式不再能满足要求，迫切需要改变软件生产方式，提高软件生产率，软件危机开始爆发。1968 年北大西洋公约组织的计算机科学家在原联邦德国召开了国际会议，第一次讨论软件危机问题，并正式提出“软件工程”一词，从此一门新兴的工程学科“软件工程学”为研究和克服软件危机应运而生。

软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。20 世纪 60 年代末至 20 世纪 70 年代初，“软件危机”一词在计算机界广为流传。事实上，几乎从计算机诞生的那一天起，就出现了软件危机，只不过到了 1968 年在原联帮德国加密施（Garmish）召开的国际软件工程会议上才被人们普遍认识到。概括地说，软件危机主要包含两方面的问题：如何开发软件，怎样满足对软件日益增长的需求；如何维护数量不断膨胀的已有软件。

“软件危机”使得人们开始对软件及其特性进行更深入的研究，人们改变了早期对软件的不正确看法。早期那些被认为是优秀的程序常常很难被别人看懂，通篇充满了程序技巧。现在人们普遍认为优秀的程序除了功能正确，性能优良之外，还应该容易看懂、容易使用、容易修改和扩充。

程序设计语言虽然为计算机的应用开拓了无比广阔的前景，但“软件危机”依然存在。因为软件的开发不仅受到程序设计的方法、结构的制约，而且受到开发周期以及软件开发成本的限制，更重要的是软件质量的保障与其程序设计的正确性关系极大。如果所开发的软件其可靠性得不到保障，在运行中就会产生不堪设想的严重后果。

20 世纪 60 年代中期以后，计算机硬件技术日益进步，计算机的存储容量、运算速度和可靠性明显提高，生产硬件的成本不断降低。计算机价格的下跌为它的广泛应用创造了极好的条件。在这种形势下，迫切要求计算机软件也能与之相适应。因而，一些开发大型软件系统的要求提了出来。然而软件技术的进步一直未能满足形势发展的需要，在大型软件的开发过程中出现了复杂程度高、研制周期长、正确性难以保证的难题。遇到的问题找不到解决办法，致使问题堆积起来，形成了人们难以控制的局面，出现了所谓的“软件危机”。

“软件危机”最为突出的例子是美国 IBM 公司于 1963 年～1966 年开发的 IBM360 系列机的操作系统。该软件系统花了大约 5000 人/年的工作量，最多时，有 1000 人投入开发工作，编写了近 100 万行的源程序。尽管投入了这么多的人力和物力，得到的结果却极其糟糕。据统计，这个操作系统每次发行的新版本都是从前一版本中找出 1000 个程序错误而修正的结果。该项目的负责人 F·D·希罗克斯在总结该项目时无比沉痛地说：“……正像一只逃亡的野兽落到泥潭中作垂死挣扎，越是挣扎，陷得越深，最后无法逃脱灭顶的灾难，……程序设计工作正像这样一个泥潭……一批批程序员被迫在泥潭中拼命挣扎，……，谁也没有料到问题竟会陷入这样的困境……”。IBM360 操作系统的历史教训已成为软件开发项目中的典型事例被记入史册。

如果开发的软件隐含错误，可靠性得不到保证，那么在运行过程中很可能对整个系统造成十分严重的后果。轻则影响到系统的正常工作，重则导致整个系统的瘫痪，乃至造成无可挽回的恶性事故。如银行的存款可能被化为乌有，甚至成为赤字；工厂的产品全部报废，导致工厂破产。1963 年，美国用于控制火星探测器的计算机软件中的一个“,”号被误写为“.”，而致使飞往火星的探测器发生爆炸，造成高达数亿美元的损失。这些均是由软件危机所产生的严重后果。

1. 软件危机的表现

早期出现的软件危机主要表现在：

（1）软件开发进度难以预测

由于软件是逻辑、智力产品，软件的开发需建立庞大的逻辑体系，这是与其他产品的生产不一样的。如工厂里要生产某种机器，在时间紧的情况下可以要工人加班或者实行“三班倒”，而这些方法都不能用在软件开发上。在软件开发过程中，用户需求变化等各种意想不到的情况层出不穷，令软件开发过程很难保证按预定的时间完成，给项目计划和论证工作带来了很大的困难。

Brook 曾经提出“在已拖延的软件项目上，增加人力只会使其更难按期完成”。事实上，软件系统的结构很复杂，各部分之间的联系极大，盲目增加软件开发人员并不能成比例地提高软件开发能力。相反，随着人员数量的增加，人员的组织、协调、通信、培训和管理等方面的问题将更为严重。许多重要的大型软件开发项目，如 IBM OS/360 和世界范围的军事命令控制系统，在耗费了大量的人力和财力之后，由于离预定目标相差甚远而不得不宣布失败。

拖延工期几个月甚至几年的现象并不罕见，这种现象降低了软件开发组织的信誉。以丹佛新国际机场为例：该机场规模是曼哈顿机场的两倍，宽为希思机场的 10 倍，可以全天候同时起降三架喷气式客机。投资 1.93 亿美元建立了一个地下行李传送系统，总长 21 英里，有 4 000 台遥控车，可按不同线路在 20 家不同的航空公司柜台、登机门和行李领取处之间发送和传递行李。支持该系统的是 5 000 个电子眼、400 台无线电接收机、56 台条形码扫描仪和 100 台计算机。按原定计划在 1993 年万圣节前启用，但一直到 1994 年 6 月，机场的计划者还无法预测行李系统何时能达到可使机场开放的稳定程度。

（2）软件开发成本难以控制

投资一再追加，令人难于置信。往往是实际成本比预算成本高出一个数量级。而为了赶进度和节约成本所采取的一些权宜之计又往往损害了软件产品的质量，从而不可避免地会引

起用户的不满。

（3）开发的软件不能满足用户要求

开发人员和用户之间很难沟通、矛盾很难统一。往往是软件开发人员不能真正了解用户的需求，而用户又不了解计算机求解问题的模式和能力，双方无法用共同熟悉的语言进行交流和描述。在双方互不充分了解的情况下，就仓促上阵设计系统、匆忙着手编写程序，这种“闭门造车”的开发方式必然导致最终的产品不符合用户的实际需要。

在“软件作坊”里，由于缺乏工程化思想的指导，程序员几乎总是习惯性地以自己的想法去代替用户对软件的需求，软件设计带有随意性，很多功能只是程序员的“一厢情愿”，这也是造成软件不能令人满意的重要因素。

（4）软件的可靠性差

尽管耗费了大量的人力物力，而系统的正确性却越来越难以保证，出错率大大增加。软件项目即使能按预定日期完成，结果却不尽如人意。软件是逻辑产品，质量问题很难以统一的标准度量，因而造成质量控制困难，软件可靠性差。软件产品并不是没有错误，而是盲目检测很难发现错误，而隐藏下来的错误往往是造成重大事故的隐患。

1965 年至 1970 年，美国范登堡基地发射火箭多次失败，绝大部分故障是由应用程序错误造成的，程序的任何一些微小错误都可以造成灾难性的后果。在美国肯尼迪发射一枚阿脱拉斯火箭，火箭飞离地面几十英里高空开始翻转，地面控制中心被迫下令炸毁。后经检查发现是飞行计划程序里漏掉了一个连字符。就是这样一个小小的疏漏造成了这支价值 1850 万美元的火箭试验失败。

（5）开发出来的软件难以维护

软件产品本质上是开发人员逻辑思维的代码化活动，他人难以替代。除非是开发者本人，否则很难及时检测、排除系统故障。为使系统适应新的硬件环境，或根据用户的需要在原系统中增加一些新的功能，又有可能增加系统中的错误。而且，由于在软件设计和开发过程中，没有严格遵循软件开发标准，随意性很大，没有完整地真实反映系统状况的记录文档，给软件维护造成了巨大的困难。特别是在软件使用过程中，原来的开发人员可能因各种原因已经离开原来的开发组织，使得软件几乎不可维护。

另外，软件修改是一项很“危险”的工作，对一个复杂的逻辑过程，哪怕进行一项微小的改动，都可能引入潜在的风险。有资料表明，工业界为维护软件支付的费用占全部硬件和软件费用的 40% ~75% 。

（6）软件缺少适当的文档资料

软件的文档资料包括开发组织和用户之间的权利和义务的合同书，系统管理者、总体设计者向开发人员下达的任务书，系统维护人员的技术指导手册和用户的操作说明书。缺乏必要的文档资料或者文档资料不合格，将给软件开发和维护带来许多严重的困难和问题。最典型的失败例子是：IBM 公司开发 OS/360 系统，共有 4000 多个模块，约 100 万条指令，投入 5000 人年，耗资数亿美元，结果还是延期交付，交付使用后的系统中仍发现大量（2000 个以上）的错误。

目前，软件危机不仅没有消失，还有加剧之势，主要表现在：

1）软件成本日益增长。在计算机发展的早期，大型计算机系统主要应用于军事领域，在这个时期，研制计算机的费用主要由国家财政提供，研制者很少考虑到研制代价问题。随