高等学校计算机专业规划教材

软件工程

项目案例与实践指导

马小军 张玉祥
廖礼萍 张冰峰 编著

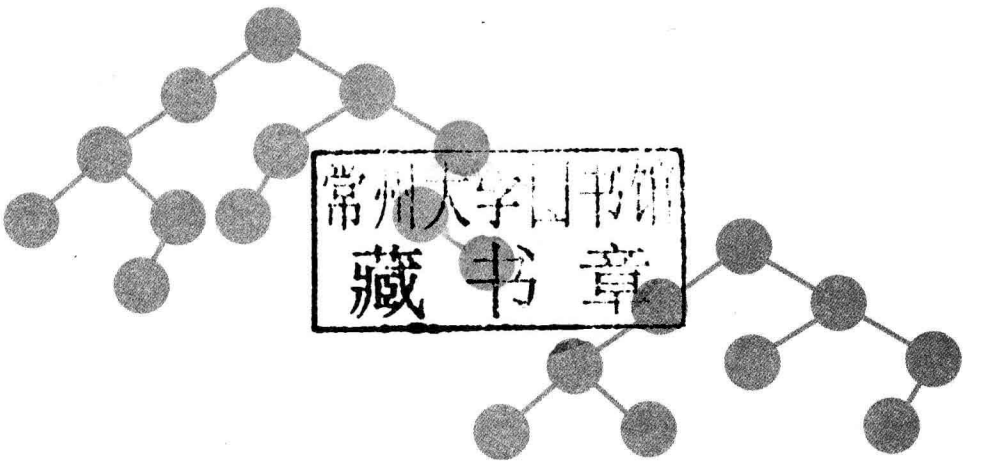
清华大学出版社

高等学校计算机专业规划教材

软件工程

项目案例与实践指导

马小军 张玉祥
廖礼萍 张冰峰 编著



清华大学出版社
北京

内 容 简 介

本书以软件工程的基本概念、当前主流技术与方法的简单介绍为导引，以项目开发过程中必须提供的且适合学生进行软件工程实践需要提交的主要开发文档的规范模板、编写案例与评分标准为核心内容，同时还提供了多个实际项目的需求描述，为学生选择有意义的模拟实践题目、了解不同领域业务工作的特点及用户对软件的基本要求与期望达到的目标提供参考，为今后实际参与项目开发、编写规范的文档打下一定的基础。全书共5章，第1章是软件工程与开发流程概述，主要包括软件工程的概念、软件生存周期的基本原理与组成、主流软件开发方法和基本建模工具；第2章是软件开发中的主要文档模板及各项内容编写说明，主要包括需求规格说明书模板、系统设计说明书模板(包括概要设计和详细设计)、软件测试计划模板和软件测试用例设计书模板；第3章是软件开发文档评分标准，主要为教师对文档各部分书写结果进行成绩评定提供参考；第4章是文档实例，围绕图书管理和宿舍分配这两个项目的文档编写进行介绍；第5章是实践项目，共包含8个小规模的软件项目的基本需求说明。

本书语言精练、论述清晰、内容实用、图形规范，所选案例是学生熟悉且有切身感受的，便于学习和理解。本书既可作为高等院校相关专业本科生软件工程实践课程的教材或参考书，也可作为项目管理人员、应用软件开发人员和专业技术人员编写技术文档的参考资料。

图书在版编目(CIP)数据

软件工程项目案例与实践指导/马小军等编著. --北京：清华大学出版社，2013
高等学校计算机专业规划教材
ISBN 978-7-302-32003-6

Ⅰ. ①软…　Ⅱ. ①马…　Ⅲ. ①软件工程－案例　Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2013)第078229号

责任编辑：龙启铭　战晓雷
封面设计：傅瑞学
责任校对：李建庄
责任印制：杨　艳

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn,010-62795954
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：6.5　　**字　　数**：164千字
版　　次：2013年6月第1版　　**印　　次**：2013年6月第1次印刷
印　　数：1～2000
定　　价：16.00元

产品编号：049316-01

前言

人类已经进入信息时代，在这个信息化环境中，软件产业受到了高度重视，也得到了空前发展，软件已经成为信息技术的核心和灵魂。软件开发不再只是软件专业学生必须掌握的知识和专有技能，而变得更加大众化和普遍化。同时，随着软件应用的广泛化与内部功能更新的频繁化，也要求软件开发必须遵循工程化和规范化的思想，唯此方能适应软件开发的要求。

对于初学者而言，什么叫软件生产的工程化？在开发过程中应该遵循怎样的步骤？系统分析、设计及测试等的具体内容是什么？必须编写什么文档？文档的规范格式是什么？评议文档的基本标准是什么？等等，都是很空泛、很抽象的概念。虽然学生在理论课上学习了，但是对于这些理论在实际工作中该怎样运用，很多学生依然是一头雾水，不知从何处下手、该怎样做。相信担任软件工程课程教学工作的很多教师在教学中都有与上述类似的经历和感受。目前已出版的软件工程方面的教材非常丰富，但课程实践指导类教材并不多，且基本上单纯以案例的介绍为主，案例所描述的项目过于复杂，学生不熟悉，缺乏感性认识，理解起来有一定困难，特别是对于一些非计算机专业的学生而言，欠缺很多软件方面的专业基础知识，学习使用这些教材难度更大。所以，编写一本集主要理论和实践练习指导于一体的实践指导教材是十分需要的。

有鉴于此，我们结合对软件工程的了解与感悟以及多年的教学体会与经验，将软件工程中最基本的概念、生存周期理念与开发过程、常用技术、主要模型和成果描述工具进行了回顾，用简明易懂的语言加以集中的介绍，读者即便未系统、完整地学习过软件工程理论知识，通过阅读学习第1章，也可以对软件工程有一个初步认知。软件的组成要素之一是文档，编写系统分析、系统设计和测试文档是软件开发最基本的工作，通过阅读第2章，读者能够了解这几个重要文档的规范模板，减少自己搜集的麻烦，同时避免出现因认识上的缺漏而导致书写内容不完整的情况。第3章主要是为教师提供文档评判的基本准则，同时也可以为文档编写者更好地编写文档提供参考。通过阅读第4章的文档实例，读者能够对实际文档的编写方法获得完整的感受和认识，并以此作为今后项目开发实践的参考资料。第5章列出了8个小规模软件项目的基本需求说明，主要是为学生参加软件工程实践选择项目时扩大视野并确定基本任务要求提供依据和参考。

本书由马小军主编，负责本书的结构组织和统稿，张玉祥、廖礼萍、张冰

峰参与编写。其中第1章由马小军编写，第2章由廖礼萍、马小军编写，第3章由张玉祥编写，第4章由马小军、廖礼萍编写，第5章由张冰峰编写。另外，本书的编写也得到了钟素芬老师和郑永荣同学的大力支持和协助，梁晔和付百文二位老师也对本书案例的叙述提供了资料和参考意见，在此一并表示感谢。

在本书的编写过程中，参阅了大量的文献和资料，在此向这些文献资料的作者致以衷心的感谢。

本书的编写以北京联合大学信息学院软件工程实践平台课教学改革为依托，所述内容具有如下特点：

(1) 突出实用性与实践性；

(2) 力求结构编排新颖，内容规范简捷，语言精练易懂；

(3) 所选案例是学生熟悉且有切身感受的；

(4) 文档系统化，且分别采用了结构化和面向对象技术，便于对这两种主流技术及应用的学习和理解。

(5) 书中所有图形(除界面截图以外)均是用 Microsoft Visio 2003 或 Rational Rose 2003 绘制的。

本书既可作为高等院校相关专业本科生软件工程课程实践的教材或参考书，也可作为项目管理人员、应用软件开发人员和专业技术人员编写技术文档的参考资料。

我们希望读者通过阅读本书，初步了解软件工程的理念，体会技术和工具的使用，特别是希望本书能够给学习者独立从事一个小型软件项目的开发提供参考和帮助。但由于时间关系且作者水平有限，在书中难免会存在问题和不妥之处，真诚地希望读者能够提出宝贵的意见和建议，以帮助我们逐步完善和修正。作者联系方式：xxtxiaojun@buu.edu.cn。

编　者

2013年3月

目录

第1章 软件工程与开发流程概述

一个国家的信息化建设和技术应用水平直接影响并决定了其国际上的地位。软件作为信息化的核心，其技术的先进性、实用性与可靠性关系到信息化的发展和经济发展、文化与系统安全，是综合国力的集中体现。众所周知，计算机系统由硬件和软件两部分组成，硬件的发展和应用离不开软件的支持，软件成果的实施又完全以硬件作为基础。硬件的发展给软件开发者的开发理念带来了无穷的刺激，推动了软件开发方法和技术的更新与完善。具备正确的开发思想，了解开发过程与基本原理，熟悉开发技术与工具的使用，才能够开发出用户满意的、先进的、实用的软件产品。

1.1 软件与软件工程简介

软件工程的主体是软件，而人们对于软件这一概念的理解也在不断地深化和拓展。最初那种认为程序就是软件的认识是十分狭隘和错误的。程序只是为解决某个问题而利用计算机程序设计语言编写的语句序列。没有数据的支持，完全无从验证问题实现结果的正确性，也不能为用户提供任何有效的服务。软件是一个逻辑部件，不是制作出的，而是人类智慧劳动通过开发形成的结果，是高度依赖于逻辑思维的。因此，IEEE(美国电气和电子工程师协会)对软件给出了比较客观而准确的定义，即：软件是计算机程序、方法、规则、相关的文档资料以及在计算机上运行时需要的数据的集合。缺少上述任何一项内容，都不能称为软件。软件可以描述为下列公式：

$$\text{软件} = \text{程序} + \text{规则} + \text{文档} + \text{数据}$$

20世纪60年代中期以来，随着软件规模不断扩大，在开发软件的过程中出现了许多严重问题，以至于个人或开发小团队根本无法解决，使软件开发陷入了十分危机的境地，被后人称为软件危机时代。虽然一些新工具、新方法不断涌现，但依然未能摆脱危机的困扰，软件经常不能按期交付，交给用户的软件质量差，无法适应用户需求的变化和环境的变化，等等。全面分析后发现，导致这种情况发生的根本原因是对软件的认识和开发观念上的错误，只有从根本上改变观念，才能够彻底走出软件危机的局面。

1968年，北大西洋公约组织(North Atlantic Treaty Organization，NATO)科学委员会在联邦德国召开了一次会议，会议主题是研讨软件可靠性和软件危机的问题。在会上首次提出了“软件工程”一词。到目前为止，对于软件工程的定义并不唯一，其中有以下几个：

- 为了积极地获得可靠的且能在机器上有效运行的软件而建立和使用的完善的工程原理(1968年NATO给出的定义)。

- 软件工程是开发、运行、维护和修复软件的系统方法(1983 年 IEEE 给出的定义)。
- 软件工程是把系统的、规范的、可度量的途径应用于软件开发、运行和维护过程(1993 年 IEEE 给出的定义)。
- 软件工程是采用工程化的概念、原理、技术和方法来计划、开发、维护和管理软件,把经过实践检验注明是正确的管理技术和最佳的技术方法相结合,以经济地获得在计算机上运行的可靠软件的一系列方法。

上述定义虽然有一定的差异,但根本内容是一致的。软件工程的提出,为软件业引发了一场革命,在软件发展史上具有里程碑的重要意义。软件工程是工程技术、工程管理与工程经济等诸多内容的融合,已经成为指导人们进行计算机软件开发和维护的一门工程学科。

1.2 软件工程的基本原理

自从“软件工程”这一术语被正式提出以来,研究软件工程的专家学者们陆续提出了许多关于软件工程的准则或信条。美国著名的软件工程专家巴利·玻姆(Barry Boehm)综合这些专家的意见,并总结了美国天合公司(TRW)多年的开发软件的经验,于 1983 年提出了软件工程的 7 条基本原理:

- 用分阶段的生存周期计划严格管理。
- 坚持阶段评审。
- 实行严格的产品控制。
- 采纳现代程序设计技术。
- 结果必须接受严格、规范的审查。
- 开发小组的人员应少而精。
- 承认不断改进软件工程实践的必要性。

相比于早期的软件开发,运用工程化的思想开发软件更突出对任务实现过程的监控、技术的选择和成果的审查。有专业知识做基础,以业务过程为导向,遵循上述原理,将能够开发出高质量的软件。

1.3 软件生存周期

1.3.1 软件生存周期的提出和作用

提出软件工程的主要目的是明确软件制作要遵循工程化的思想,即之前要做好翔实具体的准备,过程中要严格按照预定的要求执行并予以监控管理,保证软件质量,以便交付后能够为用户提供全面的服务,同时尽可能延长软件的使用期。

由于人的一生具有孕育、生长、成熟、衰老和死亡的过程——生命期,大型建筑也具有设计、施工、使用、废弃的完整过程,一个软件产品随着用户工作环境、条件和要求的变化,不可能永远适用,必然也将被逐渐更新换代,特别是软件规模的不断扩大和复杂度的提

升，短时间内是很难完成的，也非单独一人能够胜任。因此，为了使软件质量最大程度地得到保障，对软件产品的生产过程进行全面监控，软件工程中提出了软件生存周期（也称为软件生命周期）的概念，具体定义为：软件从提出需求进行策划并着手实现开始，直到软件被停止使用或废弃为止的全过程。其核心是将软件开发过程划分为不同的阶段，每个阶段具有不同的任务，由不同的角色分别完成，每个阶段将产生各自的成果，并且成果必须被严格审核。

软件生存周期是对软件生命过程的工程标准说明，也是软件开发、运行和维护过程的框架描述。软件生存周期的提出，不仅使软件的开发得到严格的控制和管理，同时也促进了软件开发工作的规范化。

1.3.2　软件生存周期的划分与组成

按照软件工程的思想和基本原理，软件生存周期总体划分为3个时期：软件定义时期、软件开发时期和软件运行时期，每个时期均由若干工作阶段组成，具体关系如图1.1所示。

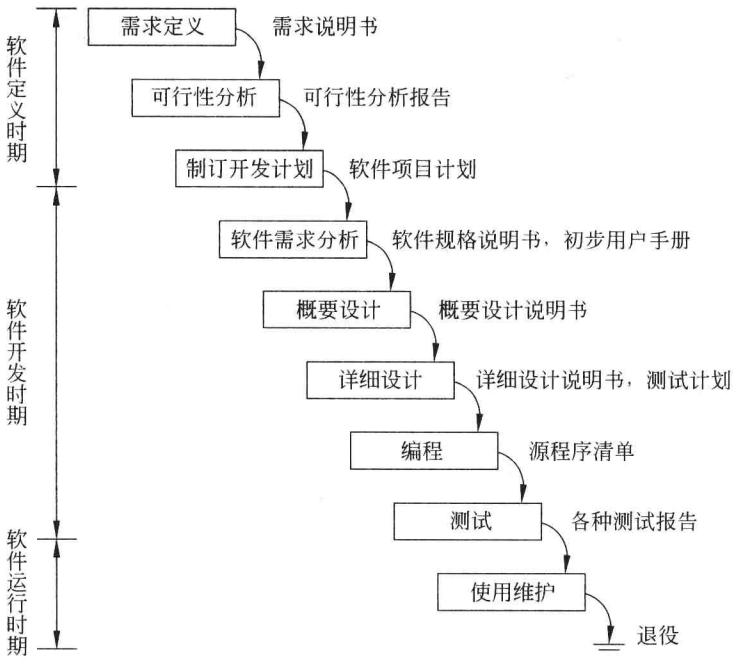


图1.1　软件生存周期组成模型

1. 软件定义时期

该时期也称为软件计划时期，主要完成软件开发前期的筹备与策划。包含需求定义、可行性分析与开发项计划制订3个阶段。

（1）需求定义也称为问题定义，主要进行市场调研，获取即将开发的项目需求，明确相关的用户角色，了解各类用户的工作范围与工作流程以及需要计算机帮助解决的问题。概括地说，即确定用户对未来软件提出的要求和寄予的期望。

(2) 可行性分析也称为可行性研究,目的是用最小的代价在尽可能短的时间内确定项目是否值得开发和能否成功开发。主要针对市场调研获得的基本需求和用户特点,从技术、经济、操作和社会与法律的可行性等方面进行全面分析,以确保未来项目开发有可行的技术方案,在经济上能够获得较高的效益,同时既适应用户的环境要求,也符合法律法规的要求。

(3) 开发计划的制订是在明确了项目有可行的方案且值得开发以后进行的一项工作,目的是保证后续开发工作能够有序进行,并且能够按期完成。

2. 软件开发时期

软件开发时期主要完成软件项目的具体研发,产生可以交付用户使用的成果。该时期由需求分析、系统设计、编码和测试等阶段组成。

(1) 需求分析也称为系统分析,核心任务是回答"软件做什么",主要工作是确定软件系统具备的基本功能,确定将来通过运行各功能可以为用户解决哪些方面的问题。它是一个对用户的需求进行去粗取精、去伪存真、正确理解,并用特定的文档形式进行表达的过程。本阶段的基本任务是对需求定义阶段获得的需求进行深入、全面的分析,建立软件的逻辑模型,编写需求规格说明书文档并最终得到用户的认可。采用结构化分析方法进行需求分析主要构建逻辑模型(数据流程图)和编写数据字典;若采用面向对象技术,则主要构建对象模型、功能模型和动态模型。

(2) 软件设计的核心任务是确定"软件怎么做",主要工作就是将软件分解成独立但彼此间又存在一定联系的功能模块并明确模块的内部结构与流程。模块是可以组合、分解和变更的功能单元,可以是一个函数、过程、子程序、一段带有程序说明的独立的程序和数据。

系统设计分为概要设计和详细设计两个阶段。概要设计就是结构设计,也称为总体设计,其主要目标是给出软件的模块结构。详细设计的主要任务是设计模块的执行流程、确定功能实现中使用的算法和数据的组织结构,设计和搭建数据库。若采用结构化设计方法,则利用结构图描述系统功能模块的物理构成,利用程序流程图或问题分析图反映模块功能的具体实现思路。若采用面向对象技术,则需完善对象模型,细化动态模型。

(3) 编程是把软件设计结果转换成计算机可以接受的程序代码的过程,即写成以某一计算机程序设计语言表示的源程序清单。编写出的程序一定要与设计方案保持一致,体现模块的执行流程和算法,且结构简洁清晰,有良好的可读性。

(4) 测试是软件开发时期的最后一项工作,也是将软件交付用户使用前的最终检查与审核阶段。软件测试的最根本目的是以较小的代价发现尽可能多的错误。在早期的软件开发中,通常忽视这一阶段的工作,即便进行测试,也是很简单的走马观花、轻描淡写。随着软件规模的不断扩大,软件内部及外部接口的复杂度提高,未经过细致、严谨测试的软件是不能被验收通过的,更不能直接用于工作中。

鉴于软件测试过程是一个发现软件错误的过程,要真正实现软件测试的目标,关键在于选择适当的测试方法,设计出具有典型特征的测试用例。所谓测试用例,就是精心设计的一组测试数据及相应的预期结果。常用的测试方法包括黑盒测试和白盒测试。黑盒测试主要用于发现软件的功能及各模块接口的错误,其中接口错误包括内部和外部接口、资

源管理、集成化以及系统错误。白盒测试主要用于发现源程序内部的逻辑结构和数据定义与使用方面存在的错误。测试阶段需要编写详细的测试计划、测试用例设计书和测试报告。

为提高测试质量，实现测试目的，Grenford J. Myers 专门提出了两个重要观点：

(1) 一个好的测试用例在于能发现至今未发现的错误。

(2) 一个成功的测试是发现了至今未发现的错误的测试。

3. 软件运行时期

软件运行时期主要是将软件交付用户实际使用，并对在使用过程中发现的错误、功能上的漏洞以及用户提出的调整要求进行修改和补充，即软件维护。这是软件生存周期中最后的一个时期，也是最长的时期。软件维护分为修正性维护、适应性维护和完善性维护。如果之前的软件分析和设计工作做得细致、全面，则软件维护工作就相对简单、易行，软件的使用寿命也可以自然延长，由此降低用户资金的再投入，同时为用户带来明显的更多的经济效益。

通过以上所述充分反映出，软件生存周期各阶段的工作是不可或缺和不可替代的，而且大量的开发事实和经验证明，需求分析、系统设计和软件测试这 3 项工作对于系统是否能够满足用户的要求，操作过程是否合理，运行结果是否正确等，则具有更加至关重要的作用，是软件开发的关键与核心。可以说，需求分析和系统设计对系统开发成败具有决定性的作用，系统测试则是成果投入运行前的把关者。为此，在本书中主要围绕软件开发中的需求分析、系统设计(概要设计和详细设计)和测试阶段的主要文档的规范和编写要求进行详细阐述，同时配有采用结构化方法和面向对象技术进行开发的实例文档，以此为教师进行软件工程理论知识教学提供案例描述、为学生正确理解和掌握软件技术和工具的使用以及进行相应的软件开发实践提供参考。

1.4 主流开发方法与工具

到目前为止，遵循软件生存周期的基本理论，从事软件工程和技术研究的学者先后提出了多种开发技术和方法，如结构化技术、原型方法、面向对象技术和敏捷软件开发等。下面对结构化技术、面向对象技术和敏捷软件开发分别进行简单介绍。

1.4.1 结构化技术

结构化技术(structure technology)是在结构化程序设计语言被广泛应用的基础上形成和推出的，是软件生存周期所定义的开发过程的最完整体现，也是最早被广泛采用的、经典的开发技术。结构化技术以用户需求为出发点，映射出系统的功能，再以模块的形式描述系统的功能结构组成，确定各模块的内部实现算法，进而逐一实现这些功能模块。

结构化技术强调开发计划和文档的重要性，文档的审核通过是开发进程推进的唯一动力。该技术主要由结构化分析(Structure Analysis，SA)、结构化设计(Structure Design，SD)和结构化编程(Structure Programming，SP)3 个主要环节组成。

(1) 结构化分析的核心工作是确定用户需求，推导出系统的逻辑模型。阶段成果主

要是数据流程图和数据字典。数据流程图(Data Flow Diagram,DFD)用于体现软件的逻辑功能,需要采取自顶向下、逐层分解细化的方法。通常要构建顶层、0 层和 1 层,后续的图要视具体情况而定。数据字典是对数据流程图的辅助说明,是系统中数据的详细描述。数据流程图如图 1.2 所示。

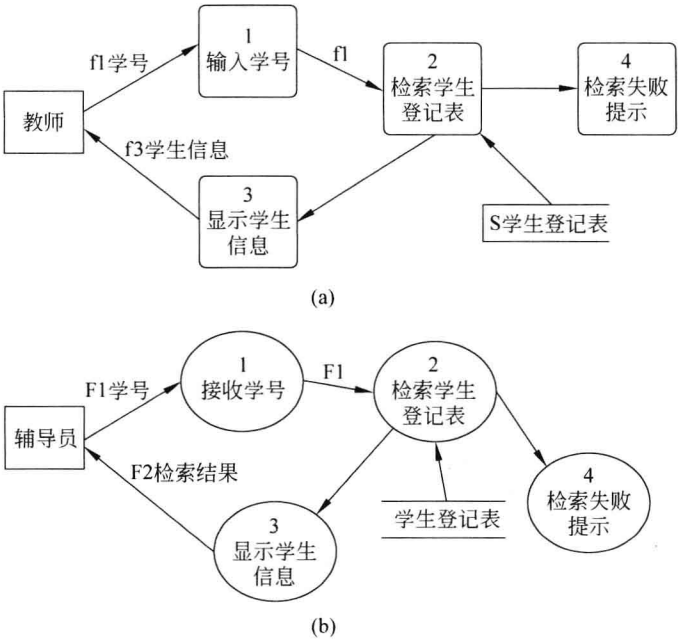


图 1.2 数据流程图样例

补充说明:图 1.2(a)和(b)中使用的符号来自两套不同的标准,可以任意选择使用,但不要混合使用。

(2) 结构化设计的核心工作是以模块的形式构建系统的物理结构,同时描述重要模块的内部实现过程和算法。阶段成果是模块结构图(Structure Chart,SC)、程序流程图(Program Flow Chart,PFC)、问题分析图(Problem Analysis Diagram,PAD)或过程设计语言(Process Design Language,PDL)等。其中,模块结构图主要体现软件系统的模块组成及相关的数据,其余几种工具主要描述模块内部执行过程与算法。模块结构图如图 1.3 所示,程序流程图如图 1.4 所示。

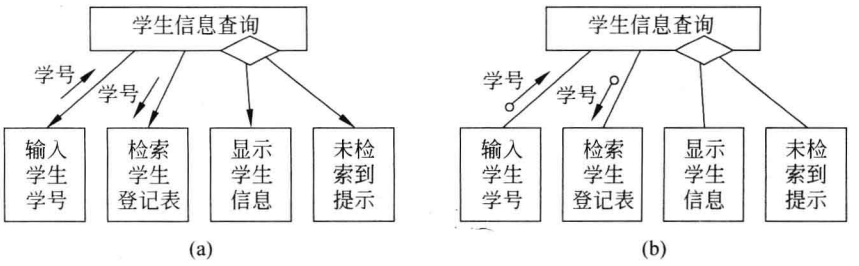


图 1.3 模块结构图样例

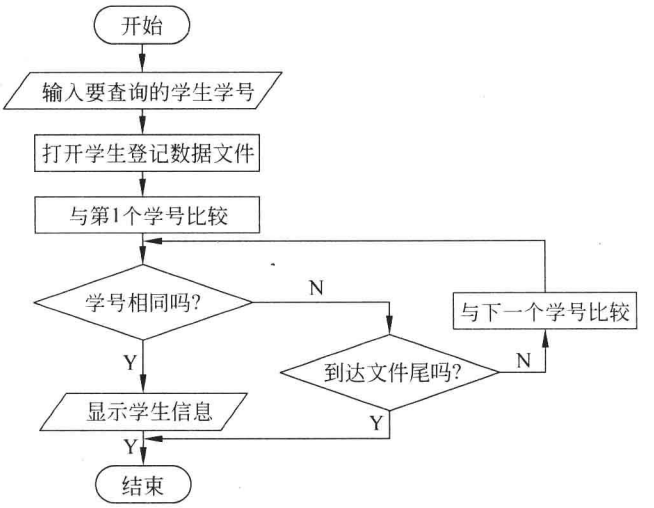


图 1.4　程序流程图样例

(3) 结构化编程的核心工作是用结构化的程序设计语言编写程序代码。

1.4.2　面向对象技术

面向对象技术(Object-Oriented Technology)是 20 世纪 80 年代末推出、20 世纪90 年代中期开始盛行的软件开发技术,在分析问题、描述系统组成、构建系统模型、编写程序实现用户需求等方面与其他技术都有极大的区别。该技术以类、对象、方法、消息和消息传递等概念为主导,强调系统由类组成,每个类内部既包含静态数据,也包含可执行的动态操作,具有对象唯一性、抽象性、封装性、继承性和多态性等特性。

面向对象技术主要由面向对象分析(Object-Oriented Analysis,OOA)、面向对象设计(Object-Oriented Design,OOD)、面向对象程序设计(Object-Oriented Programming,OOP)和面向对象测试(Object-Oriented Test,OOT)4 个环节组成。其中面向对象分析和面向对象设计的工作内容基本一致,阶段成果的表达形式也基本相同,只是面向对象设计的描述更加详细、完整。通过面向对象分析,能够实现对现实世界的抽象,构建出对象模型、动态模型和功能模型;面向对象设计则是将面向对象分析的结果转化为符合成本要求和质量要求的实现方案,完成问题域子系统设计、任务管理子系统设计、数据子系统设计和人-机交互子系统设计。

UML(Unified Modeling Language,统一建模语言)是基于面向对象技术而提出的标准建模语言,它吸收了不同学者和流派对面向对象技术进行研究取得的多项成果的长处,并对这些成果进行了整合,统一并提出了一套新的建模理念、简洁的图形符号和完善的视图规范。在 UML 中,提出了以下 5 类视图。

(1) 用例视图:用于描述系统的基本功能,用用例图表示。

(2) 静态视图(也称为逻辑视图):用于描述系统的静态结构和组织关系,用类图表示。

(3) 动态视图:用于描述系统的动态特征和行为变化,用状态图、活动图和顺序图等表示。

(4) 组件视图：用于描述系统实现的结构和行为特征，用组件图表示。

(5) 部署视图：用于体现系统的实现环境，反映系统的物理结构，用配置图表示。

用例图、类图、顺序图和活动图的基本形式如图 1.5 至图 1.8 所示。

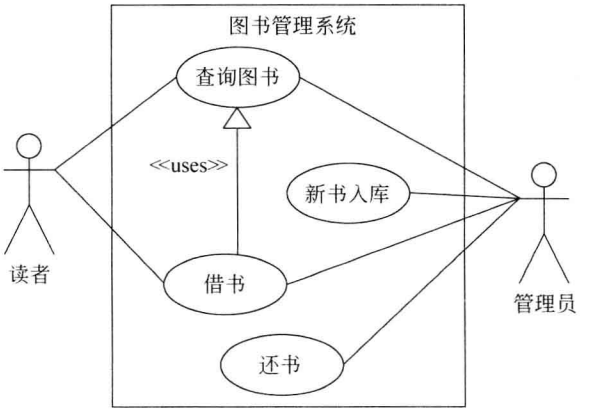


图 1.5　用例图样例

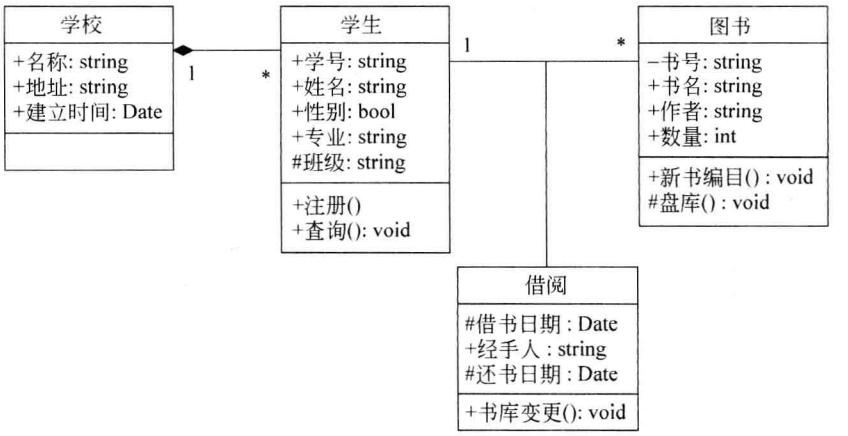


图 1.6　类图样例

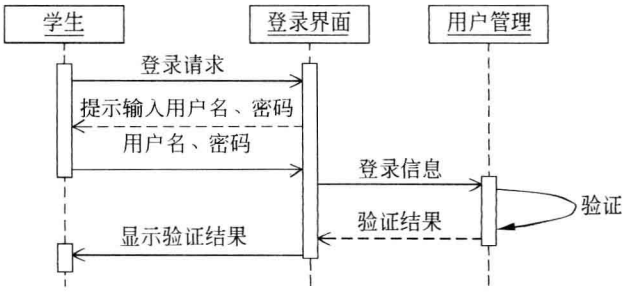


图 1.7　顺序图样例

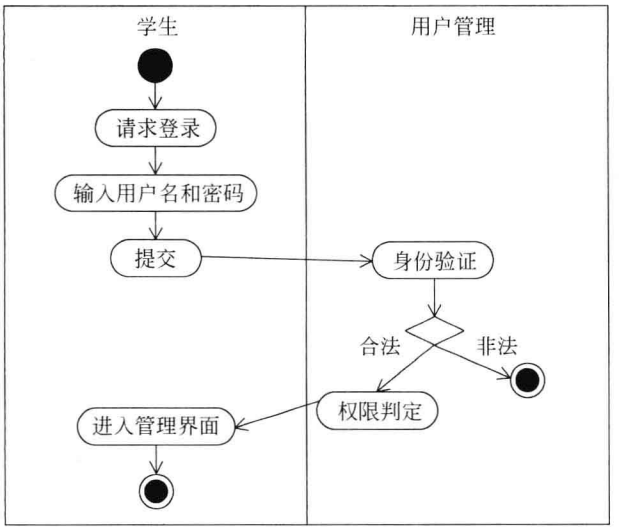


图 1.8　活动图样例(有泳道分隔的)

上述 UML 视图的使用将面向对象分析和面向对象设计的工作过程和成果表示进一步明确化,为后续实现策略的确定提供精确的描述,同时通过这些模型可以直接映射成不同面向对象语言的程序框架,保证了系统程序体系的严谨性、一致性和完整性。因此,UML 已经成为当今软件开发企业最流行的软件系统建模语言和工具。

1.4.3　敏捷软件开发方法

受经济发展和社会大环境的影响,用户单位在发展规划、制定政策等方面也在不断变化,导致用户需求不断变更。为了高效地开发软件系统,提升软件开发应对需求变化的能力,自 20 世纪 90 年代末,一些软件领域的专家学者提出了一个快速开发软件的新方法——敏捷软件开发,它提供了一套与以往的开发方法(比如结构化技术)完全不同的开发思想和策略,总体概括如下:

(1) 相比于过程和工具,每一个体(人)及其间的交互更加重要。

(2) 相比于面面俱到的文档,能够正常运行的软件更加重要。

(3) 相比于严肃紧张的合同谈判,客户合作更加重要。

(4) 相比于严格遵循软件开发计划,响应用户需求的变化更加重要。

敏捷软件开发不仅追求过程简捷、高效,尤其重视与软件相关的所有人员的作用发挥及交流合作。为保证软件开发的质量和进程的顺利执行,敏捷软件开发方法对欲采用此方法进行开发的人们提出了下列 12 条指导原则:

(1) 应该尽早并持续地交付有价值的软件,由此获得用户的满意。

(2) 即使到了软件开发的后期,用户需求发生变化也要高兴地面对,并能够尽快应对处理。

(3) 要能够经常性地交付可运行的软件,交付的间隔可以从几周到几个月,时间间隔越短越好。

(4) 在整个软件开发期间,用户和开发人员最好能够每天一起工作。

(5) 选择一些工作积极主动的人员承担项目开发,为其提供需要的环境和支持,并充分信任他们。

(6) 在团队内部应该采取最有效的传递信息的方式——面对面交谈。

(7) 以可运行的软件作为衡量开发进度的首要标准。

(8) 为保证可持续开发,出资方、开发方和用户应保持长期、恒定的开发速度。

(9) 要关注优秀的技能和良好的设计,由此增强敏捷性。

(10) 考虑问题和解决问题的方法要简单化,切忌华而不实。

(11) 最好的架构、需求和设计通常出于自组的团队。

(12) 开发团队应该定期针对怎样提高工作效率进行反思,并做出相应的调整。

敏捷软件开发方法并没有提出很多新的概念和特有的技术,完全是建立在之前的那些技术之上,是将经过多年检验的软件工程准则加以有机地融合,从而达到软件开发的小巧、简单、快捷、灵活的目标。但是有一点还需提醒:采用敏捷方法进行软件开发,最好有一定的开发经验做基础,同时密切结合上述 12 条原则开展工作,这样才能使成果得到保证,真正体现和发挥敏捷开发的优势。