国外经典教材·计算机科学与技术

PEARSON
Addison Wesley

Object-Oriented Software Development Using Java™(2nd Edition)

Java面向对象程序设计教程（第2版）

(美) Xiaoping Jia 著
杨茂江 译

Pearson Education
清华大学出版社

国外经典教材·计算机科学与技术

Java 面向对象程序设计教程
（第 2 版）

（美） Xiaoping Jia 著
杨茂江 译

清华大学出版社
北 京

内 容 简 介

本书使用 Java 语言作为范例语言，由浅入深、循序渐进地讲解了面向对象的开发方法和分析方法。与其他讲解面向对象的书籍相比，本书的独特之处在于紧扣面向对象开发和分析方法的前沿发展，结合大量实例，讲解了设计模式、框架、UML 语言等多种面向对象开发和分析的主流技术。

本书读者对象包括计算机专业高年级本科生、研究生、广大开发人员以及系统分析人员。

图书在版编目(CIP)数据

Java 面向对象程序设计教程（第 2 版）/（美）贾小平（Xiaoping Jia）著；杨茂江译. －北京：清华大学出版社，2004.11
书名原文：Object-Oriented Software Development Using Java™（2nd Edition）
国外经典教材・计算机科学与技术
ISBN：7-302-09809-3
I. J…　II. ①贾…　②杨…　III. Java 语言－程序设计－教材　IV. TP312
中国版本图书馆 CIP 数据核字（2004）第 110143 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：车立红
封面设计：久久度文化
印 装 者：北京鑫霸印务有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：33　**字数**：790 千字
版　　次：2004 年 11 月第 1 版　2004 年 11 月第1 次印刷
书　　号：ISBN 7-302-09809-3/TP・6769
印　　数：1～3000
定　　价：56.00 元

前　言

面向对象软件开发已经发展了近 20 年，最近几年在很大意义上已趋于成熟。下列领域的进展在该技术的成熟过程中扮演了重要角色：

- 诸多的面向对象建模技术和表示法的演变及归一，导致产生了统一建模语言（Unified Modeling Language , UML），并成为事实上的标准。
- Gamma 等人编著的《设计模式目录》一书的出版成为设计模式发展的里程碑，从而促使面向对象框架和设计模式得到了广泛采用。
- 软件界接受面向对象的开发思想在很大程度上应归功于 Java 语言的出现和普及。

总的来讲，面向对象技术在今天得到了前所未有的欢迎，特别是 Java 语言。然而，该技术的快速发展对计算机科学和软件工程专业的学生以及软件开发专业人员提出了学习上的挑战。本书旨在涵盖面向对象技术的各个方面及其内在关联性，包括使用 UML 的面向对象建摸，使用设计模式的面向对象设计，以及使用 Java 语言的面向对象编程。本书可作为计算机科学和软件工程专业的研究生入门教材以及高年级的本科生教材，同时也可作为专业开发人员的参考书。本书并不是 Java 语言编程的入门教材；本书的读者应当首先具备一定的编程经验，最好是 C 语言或 C++语言。

在面向对象领域中，编程和设计是两个不同的任务；但是，它们之间的内在联系要比在传统的编程模式中紧密得多。学习使用 Java 语言的面向对象软件开发不仅仅局限于学习 Java 语言的语法和类库。面向对象开发与传统开发在思维方式上存在巨大差异，掌握它需要全新的思想方式。在本书中，我试图循序渐进地将面向对象的思想方法传授给那些通过使用设计模式使用面向对象思想的人，剖析 Java 类库的设计，以及演示迭代软件开发方法。我本人既是软件工程和面向对象开发的教授者，也是实践者。我的意图是从不同的视角——学术界、工业界、理论和实践上，提出对面向对象软件开发的一个平衡的看法。

涵盖完整的 Java 语言和 Java 类库已经超出了本书的范围。因此，本书的重点是：

- 最重要的和最常使用的语言特性和类库。
- 使用 Java 类库来阐述面向对象设计原则和设计模式的应用。

本书可满足不同背景的学生对面向对象软件开发课程的不同目标要求。

- 第 1 章概述了面向对象软件开发所面临的挑战和解决方案。
- 第 2 章介绍了面向对象模式和统一建模语言（UML）。对于熟悉 UML 的学生来说，本章可作为一个复习。对于不熟悉 UML 的学生来说，本章可以作为入门读物，它包含了最重要的内容，这些内容在本书余下的章节中很有用。
- 第 3～5 章是为那些不熟悉 Java 语言，但有不同语言（如 C 或者 C++）编程经验的学生准备的。对于那些熟悉 Java 语言的学生来说，这一部分内容可作为复习。
- 第 6 章和第 7 章阐述了面向对象编程和设计中的一些关键问题。第 7 章还介绍了几个最常用的设计模式。

- 第 8 章从使用框架以及运用设计模式设计框架的角度讨论了 3 个重要的 Java 框架——集合、图形用户界面以及输入/输出。
- 第 9 章给出了一个完整的案例分析，用以演示迭代开发过程，该过程包含了多个设计模式和多次重构。
- 第 10 章给出了更多的经常使用的设计模式的例子，同时给出了具体的用 Java 语言实现的示例。本章可用于重点讲述面向对象设计和设计模式的课程。
- 第 11 章和第 12 章讨论了 Java 语言的并发和分布计算。

本书在多个例子中使用了 Java applet 来阐述面向对象编程和设计的概念。随着 Java 技术的最新进展，Java applet 在实践中的重要性和实用性都减弱了。但是，我认为 Java applet 仍然可以作为教授和学习面向对象编程和设计的有用的教学工具。因此，第 2 版中仍然保留了使用 Java applet 的大多数例子。

本版的改动

本版的主要改动包括：

- 1.4 节讨论了迭代软件开发过程，包括 Rational 统一过程（Rational Unified Process，RUP）和极限编程（Extreme Programming，XP）。
- 第 2 章对统一建模语言进行了更深入的探讨，包括用例建模和用例图。
- 6.2 节讨论了断言（assertion）、契约（contract）以及类的不变式（invariants of classes）。
- 6.4 节和 6.5 节增加了单元测试和项目建立的内容。
- 第 10 章对设计模式进行了深入的讨论。
- 第 9 章修改并扩充了本书第 1 版中的设计案例，以阐述迭代开发和重构。

标注和约定

下列字体用来代表程序代码：

1. 可以逐字拷贝的代码片段是等宽字体。粗体等宽的代码片段是重要的或有特殊含义的。
2. 直体 Roman（罗马）字体的代码片段是伪代码，这是程序逻辑的非正式描述。
3. 尖括号内的内容，比如：

   <108 页中的 doSomething()方法>

   代表占位符。应该把在其他地方定义的代码片段插到这里。页码代表被插入的代码片段在哪一页被定义。
4. 斜体名字，比如 *var*，可以被任何其他特定且不同的名字所代替。

在线补充资料

本书中的所有程序都是使用 Java 2 软件开发工具包（J2SDK）1.4 开发的。[1]可以从网站 *http://java.sun.com/*下载 J2SDK 和完整的 API 文档。

可以访问 *www.aw.com/cssupport* 得到本书所有例子的源代码。所有练习题的答案可提供给采用本书作为教材的教师。请与您当地的 Addison-Wesley 经销代理人联系以获得更多信息。

致　谢

本书的写作像是一次长途旅行。Java 语言的快速发展使得这个旅行既富有挑战性也充满激情。在编写过程中，我有幸得到了许多人的帮助。他们使我受益匪浅。感谢 Addison-Wesley 出版社的编辑——Maite Suarez-Rivas。如果没有她不断的鼓励、支持与指导，本书是不可能完成的。

感谢 Addison-Wesley 的员工：Katherine Harutunian（项目编辑）、Juliet Silveri（高级生产主管）、Jean Wilcox（封面设计人员）、Gina Hagen Kolenda（封面设计监督人员）、Michael Hirsch（执行市场经理）以及 Lesly Hershman（市场助理）。他们使本书的编写变得很愉快。我还要感谢 Peter Reinhart（技术编辑），他进行了仔细的编辑工作。感谢审阅者们所付出的辛勤工作：Charles Crowley（New Mexico 大学）、Anhtuan Q. Dinh（Mitre 公司及 George Mason 大学）、Shyamal Mitra（Austin 的 Texas 大学）、Gleb Naumovich（Polytechnic 大学，Brooklyn）、Juergen Rilling（Concordia 大学）、Ken Slonneger（Iowa 大学）、Joe Wong（Worcester Polytechnic 研究所）、A.Yanushka（Christian Brothers 大学），以及 Huiming Yu（North Carolina A&T 州立大学）。他们的评论和建议对本书有很大的改善。另外，我还要感谢 CTI 学院的两个很棒的同事，James Riely 和 Chris Jones，他们在授课中使用了第 2 版的各种不同版本的草稿。他们提出了非常宝贵的建议和反馈。同时还要感谢 Hongming Liu 和 Lizhang Qin 在开发本书的辅助材料方面的贡献。

感谢 DePaul 大学计算机科学、电信、信息系统学院在 Java 技术发展的早期就将其引入到了课堂上。同时感谢学习 SE450 课程的所有学生：他们对 Java 语言和我讲课的热情使我对踏上这趟旅程充满了信心。他们对我的讲课以及本书早期草稿的反馈起了很大的帮助作用。我一直把他们当作这趟旅行的同伴，享受了有他们陪伴的快乐时光。

敬请指正

对任何能够帮助改进书中问题的建议和反馈我将不胜感激。请将您的意见发送到：*xjia@cti.depaul.edu*

---

① 大多数程序仅要求 JDK1.1 或更高版本。

出 版 说 明

近年来，我国的高等教育特别是计算机学科教育，进行了一系列大的调整和改革，急需一批门类齐全、具有国际先进水平的计算机经典教材，以适应当前我国计算机科学的教学需要。通过使用国外先进的经典教材，可以了解并吸收国际先进的教学思想和教学方法，使我国的计算机科学教育能够跟上国际计算机教育发展的步伐，从而培育出更多具有国际水准的计算机专业人才，增强我国计算机产业的核心竞争力。为此，我们从国外知名的出版集团 Pearson 引进这套“国外经典教材·计算机科学与技术”教材。

作为全球最大的图书出版机构，Pearson 在高等教育领域有着不凡的表现，其下属的 Prentice Hall 和 Addison Wesley 出版社是全球计算机高等教育的龙头出版机构。清华大学出版社与 Pearson 出版集团长期保持着紧密友好的合作关系，这次引进的“国外经典教材·计算机科学与技术”教材大部分出自 Prentice Hall 和 Addison Wesley 两家出版社。为了组织该套教材的出版，我们在国内聘请了一批知名的专家和教授，成立了一个专门的教材编审委员会。

教材编审委员会的运作从教材的选题阶段即开始启动，各位委员根据国内外高等院校计算机科学及相关专业的现有课程体系，并结合各个专业的培养方向，从 Pearson 出版的计算机系列教材中精心挑选针对性强的题材，以保证该套教材的优秀性和领先性，避免出现“低质重复引进”或“高质消化不良”的现象。

为了保证出版质量，我们为该套教材配备了一批经验丰富的编辑、排版、校对人员，制定了更加严格的出版流程。本套教材的译者，全部来自于对应专业的高校教师或拥有相关经验的 IT 专家。每本教材的责编在翻译伊始，就定期不间断地与该书的译者进行交流与反馈。为了尽可能地保留与发扬教材原著的精华，在经过翻译、排版和传统的三审三校之后，我们还请编审委员或相关的专家教授对文稿进行审读，以最大程度地弥补和修正在前面一系列加工过程中对教材造成的误差和瑕疵。

由于时间紧迫和受全体制作人员自身能力所限，该套教材在出版过程中很可能还存在一些遗憾，欢迎广大师生来电来信批评指正。同时，也欢迎读者朋友积极向我们推荐各类优秀的国外计算机教材，共同为我国高等院校计算机教育事业贡献力量。

清华大学出版社

国外经典教材·计算机科学与技术

编 审 委 员 会

译 者 序

由于对软件工程，特别是对面向对象的软件开发技术很有兴趣，我有幸得到翻译本书的机会。这对我本人来讲也是个难得的学习机会。在我的教学和工作过程中，我深深地体会到，软件工程这门课程从本科生到研究生一直到工作岗位，对教师来说，一直被认为是计算机专业里很难讲的一门课；对学生来说，他们也大多觉得课上完了，可没学到什么东西。究其原因，我认为一是因为教材更新较慢，目前大部分教材的内容没有反映软件工程技术的最新发展，学生学的东西和实际工作中使用的技术不能很好的衔接；二是因为软件工程是一门实践环节很重要的课程，光靠书本上的讲授很难让学生深刻地体会到软件工程的精髓。

本书是最近本人所看到的一些讲解面向对象开发技术书籍中比较成功的一本。它基本上克服了上述缺点。一方面它基本涵盖了面向对象技术的方方面面，从 UML 到设计模式、从软件开发过程到框架，包括了面向对象最近发展的最新进展；另一方面，该书不是简单地介绍概念，而是循序渐进地用实例讲解这些设计模式、框架的使用，让读者能直观地体会这些技术的使用场合和使用技巧。另外，本书使用的示例语言是 Java 语言，这是本书的重要特点之一。原则上讲，使用何种语言并不是关键，但 Java 语言能以其固有的易读性让读者把主要精力放在理解面向对象开发的技巧，而不是语言的细枝末节上。我认为这一点本书是很成功的。

当然，我们也不能寄希望于通过本书完全掌握面向对象开发技术。有人说，软件工程是一门艺术而不是科学，我们姑且不讨论这句话的对与错，至少它从一个侧面反映了软件工程技术是需要长期积累和体会才可能真正领会其精髓的。因此，在阅读本书时，最好能够结合阅读其他讲解软件工程，特别是面向对象软件开发的书籍。本书大部分章节最后都有推荐的补充读物，读者可根据自身情况有选择地阅读。

本书在翻译过程中，得到了编辑车立红小姐的大力支持，在此表示最诚挚的谢意。同时感谢清华大学出版社给了我这个机会，让我对面向对象开发技术有了更深刻的理解。另外，微软全球技术中心的藤剑锋先生、德恒证券的潘彦先生、复旦大学的孙未未先生以及复旦德门软件有限公司的朱建秋先生参与了本书的翻译工作。在此向他们表示感谢，没有他们的辛勤工作，本书的完稿是不可想像的。最后，感谢在本书的翻译过程不断给了我鼓励和支持的所有朋友和家人。

杨茂江

目　录

第 1 章　面向对象软件开发

本章概述

本章概述了面向对象软件开发。我们从软件开发过程和软件产品的期望质量的一般讨论开始。接着，我们将讨论是什么使软件开发变得很困难以及软件工程和其他成熟的工程实践的区别。然后我们将深入研究迭代软件开发过程，包括 Rational 统一过程（Rational Unified Process，RUP）和极限编程（Extreme Programming，XP）。

无论你是使用计算机的新手，还是经验丰富的计算机用户或程序员，你都得承认我们生活在一个激动人心的时代，这是一个计算机软件和硬件不断推陈出新的时代。在过去的 20 年中，软件业是最成功的产业之一。这不仅是因为它的市场价值的突飞猛涨，更因为它不断的技术开发和产品革新的步伐势不可挡。今天，计算机软件已经在我们生活的方方面面变得很普遍了。人们越来越依赖于软件系统，从喷气客机的自动导航系统到股票市场的计算机交易系统乃至掌上电脑的个人记事簿。但是，软件是昂贵的。对于许多企业来说，购买、开发、维护以及升级软件系统的成本已经成为最大的一项开销了，而且这项开销还在不断增长。不断增长的软件成本与硬件成本的大幅下降以及硬件性能的大幅提高形成强烈反差。面向对象开发方法学旨在大幅改善现有软件开发实践。它已在最近几年得到了软件业的广泛采用。它已成为当今软件开发业在软件开发方法中的最佳选择。

1.1　软件开发面临的挑战

在过去的 20 年中，软件业开发了许多技术进步、创新和成功的商业产品。但是，开发这些成功产品的过程（也就是软件开发）是一个艰难、耗时并且耗资巨大的工程。比如，微软 Windows NT 操作系统的最初版本由 600 万行代码构成，耗资 1.5 亿美元，由 200 多名开发人员、测试人员和文档编写人员花了 5 年时间才完成。*Show-Stopper* [Zachary, 1994] 一书生动地展现了开发 Windows NT 的艰辛。此外，软件系统容易变得“故障百出（buggy）”，也就是说，它们会有一些故障，会妨碍甚至完全破坏其正常的功能或性能。小的故障可能仅仅是让人讨厌，但严重的故障可能会造成一场灾难。

- 1990 年 1 月 15 日，AT&T 的长途电话网络瘫痪，造成了美国全国范围内的长途电话服务中断超过 8 个小时。造成瘫痪的原因是因为用 C 语言写的程序中一个 break 语句的位置错误。
- 1996 年 6 月 4 日，由欧洲航天署开发的新一代的增强型阿里亚娜五号通信卫星运

载火箭执行她的处女航，在火箭升空后 37 秒发生了爆炸。一个由 64 位浮点数转换为 16 位的有符号整数致使软件异常的错误处理造成了这次灾难。

- 2001 年 6 月 8 日，前一天晚上刚刚安装在纽约证券交易所的新的交易系统中的一个软件问题造成了交易所一半席位无法进行交易，迫使纽约证券交易所关闭了所有席位的交易，时间超过一个小时。

尽管像这样的灾难性错误是很少见的，但在几乎所有软件系统中都存在一些小缺陷。换句话说，故障百出的软件是很正常的。

但是，软件开发的现状已远远不同于此前人们曾说的“软件危机”（software crisis）。软件开发方法学和软件工程过程等多方面的发展使得人们能够开发出很多在大多数时间内可以按期望运行的大规模软件。我们无法提交一个 100%可靠的软件，而且也没要求我们这样做。问题是：多好才是足够好？

软件开发是人工密集型的。大多数软件开发项目都会超出预算并且延期。软件开发的现实是：软件是很昂贵的，并且往往不可靠。尽管软件工业在技术发展和革新上取得了长足进步，但要按时并在预算内交付高质量的软件绝不是一件轻而易举的事情。面向对象的软件开发方法学是软件工业当前广泛采用的解决方案，其目的是改善软件系统的可靠性和软件开发的成本效益。

为了改善软件开发实践，让我们首先看看一些造成软件开发高难度的根本原因：复杂性、长期性和演变性以及高用户期望。

复杂性 现在开发的软件系统一般都很庞大而且复杂。复杂性是由系统所要解决的问题和所提供的服务决定的。从工程的观点看，二者均超出了软件开发人员的控制范围。大型软件系统非常复杂，没有任何人能够了解系统的每个细节。为了开发一个复杂的软件系统，必须将系统分解为可管理的模块，并需要开发团队的共同协作，而不仅仅是个人的努力。个人开发小系统时很有效的方法学、技术和工具对由团队开发的大系统通常没有什么效果。

长期性和演变性 由于经济、政治和其他的一些限制因素，软件系统往往需要使用很长的时间。今天，一些遗留系统的使用时间已超过了 20 年。在它们的生命期内，软件系统必须不断地更新以满足用户需要和环境的变化。但是，修改软件系统（即维护）是一项困难的任务。另外，维护不仅昂贵耗时，而且往往还会降低被维护系统的质量。一般来说，一个软件系统在其生命期内的维护开销要远远大于其最初的开发费用。

高用户期望 过去，当计算机主要在大学、研究机构和大公司里使用的时候，软件系统的用户大部分都是科学家和工程师，在使用系统的过程中遇到故障时，他们具有处理这些故障的技术能力。今天，计算机已经在家庭、学校、不同规模的企业中广泛用于娱乐和工作。今天的软件用户大多数都是普通的非技术人员。人们认为计算机软件产品越来越像消费品，并期望它能像家电产品那样可靠。从前认为可以接受的偶然故障在今天已变得无法忍受。人们期望软件系统是“无缺陷（bug-free）”的，但这样尽善尽美的软件是可望不可得的。

软件开发面临的挑战就是寻求有效的解决方案，以控制软件系统的复杂性，管理软件系统的长效性和演变性，并交付具有更高可靠性和可用性的软件系统。

1.2　工程的观点

“软件工程”一词是在由 NATO（北大西洋公约组织）于 1968 年主办的研讨会上提出来的。它表达了人们的一种愿望，即把软件开发实践建立在一个可靠的科学基础上，同时达到像土木工程和机械工程这种成熟工程学科所拥有的可靠性和生产力水平。

软件工程是一个工程学科，它关注的是以经济高效的方式开发并交付高质量的有用软件的方方面面。软件工程定义了软件开发中的各种“活动”以及与这些活动相关的产品或可交付产品。软件工程还定义了“软件开发过程”，它定义了实施开发活动的顺序和各个活动的可交付产品的判别标准。

1.2.1　软件开发活动

很显然，软件开发最重要的产品就是软件。大多数人将软件等同于计算机程序或源代码。然而，源代码只是软件开发所产生的产品的一部分。在软件工程里，“软件”这个词定义得更宽泛。它包括源代码和软件开发过程的各个活动中所产生的相关文档。软件文档可能包括需求规范、构架和设计文档、配置数据、安装和用户手册等等。软件开发一般包括下列活动：

需求分析　需求分析的目的是为将要开发的软件建立功能、服务和约束条件。对于定制软件，即为特定客户开发的软件，需求分析一般通过与系统用户沟通来完成。对于商业（“薄膜包装”，shrink-wrapped）软件，也就是即将进入市场，销售给任何想购买它的客户的软件，需求分析往往通过对潜在客户的需要进行市场调查和/或通过现有客户的反馈来完成。有如下两类需求：“功能需求”，主要是与软件所执行的功能和服务相关；以及“非功能需求”，主要是与软件运行所必须受到的限制有关，比如响应时间、内存消耗以及用户友好性。需求分析的主要任务是定义要解决的问题。需求应以需求规范或者系统规范的形式形成文档。

设计　设计的目的是为问题建立一个解决方案，通过建立一个整体软件构架、将软件拆分为组件或子系统、确定组件或子系统间的关系和依赖性来达到这个目的。这些设计活动常常被进一步分为系统设计和详细设计。系统设计主要关心的是如何把复杂的问题分解为可管理的模块；详细设计主要关心的是每个模块的解决方案。软件设计往往使用各种图来形成文档。

实现和单元测试　实现是将设计实现为程序，即源代码。单独实现每个模块。单元测试是对每个单独的模块或单元进行独立测试。单元测试的目的是保证每个单元在集成之前，它的功能按照其规范正常运行。

集成和系统测试　单独的模块或单元被集成在一起，并进行测试，以保证整个软件系统的功能按照其规范正常运行。

维护　维护涉及到软件系统交付后的各种活动。这些活动包括修改错误、提高性能、增强功能或服务、修改以适应新的环境等。只要软件还在使用，就要对其进行维护。它往往是软件生命期内最长和成本最高的活动。

1.2.2 软件开发过程

最著名的软件开发过程是如图 1.1 所示的“瀑布”模型，它已成为软件开发过程事实上的标准。在瀑布模型中，开发活动的各个阶段按照线性连续的方式展开：需求分析、设计、实现和单元测试、集成和系统测试以及维护。“阶段”是两个主要里程碑之间的一个时间跨度，在此期间，要满足预先定义的目标集合、完成制品，并做出是否进入下一阶段的决定。原则上，必须批准（签字确认）每个阶段的交付产品后才能开始下一阶段的工作。其基本根据是在需求分析阶段实现对需求规范的修改的耗资要远远小于后面几个阶段实现对需求规范的修改。需求修改得越晚，其耗资越高。因此，我们的目标是尽可能减少和避免在文档提交后的更改变化。这要求每个阶段的任务要完成得彻底，同时每个阶段的提交物在提交和确认后应该被封存而不再更改。

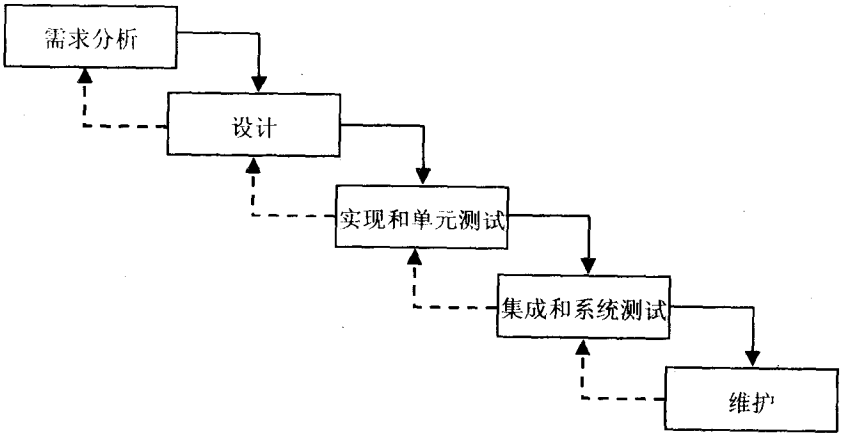


图 1.1 软件开发的瀑布模型

但是，瀑布模型并不现实。各种不同形式的更改变化在开发过程里的每个阶段都是很普遍的。变化来自于多个方面：规范和设计的错误可能在实现阶段才发现；设计阶段所做的假设可能在系统测试时发现是错误的；客户要求的特性可能在系统测试时被证明太慢、资源消耗过多或者根本不可行；用户需求可能在需求分析结束后发生了变化。因此，在实践中，瀑布模型中的各个阶段常常必须有多次迭代。然而，瀑布模型的一个主要缺点是它无法实现这样的迭代。

有几个设计用来以迭代方式执行软件开发活动的可选软件开发过程。迭代软件开发过程已变得很流行，并在实践中被人们所接受，部分原因是因为面向对象开发方法学被广泛接受，而面向对象软件开发方法学特别适合迭代开发。我们将在 1.4 节中讨论两个常用的迭代开发过程。

1.2.3 软件系统的期望质量

现在让我们把注意力集中在软件开发的产品——软件系统上。下面是软件系统最期望达到的质量：

可用性（Usefulness）：软件系统应该充分满足用户在解决问题和提供服务方面对软件提出的要求。

及时性（Timeliness）：应该及时完成软件系统并将其发运。否则可能会因为用户需求或者运行环境发生变化，而使软件系统的用处变小或者没有任何用处。及时性对于软件厂商保持竞争力也同样重要。

可靠性（Reliability）：软件系统应依照用户的期望运行，保证执行功能的正确性和服务的可及性 (availability), 并将故障率控制在可接受的限度内。

可维护性（Maintainability）：软件系统应该能很容易维护，也就是说，对软件系统的排错、修改、扩展是可能的，并且不产生不当的费用。

可重用性（Reusability）：软件系统的组件不应该设计成在特定环境下针对特定问题的特别解决方案，而应该设计为在不同环境下针对某类问题的“通用”解决方案。这个通用模块可以被“修改”和“重用”多次。

用户友好性（User friendliness）：软件系统应提供针对目标用户的能力和背景量身定制的用户友好的界面，以方便用户的使用，并能够访问到系统的全部功能。

高效性(Efficiency)：软件系统不应该浪费系统资源，包括处理时间、内存和磁盘空间。

这些期望质量既不会同时满足，也并不是同等重要。软件系统开发的一个关键就是在这些不同的质量间寻求合理的平衡。显然，面向对象开发方法并不能直接改善所有上述质量。它主要着眼于改善软件系统的可维护性和可重用性。应把可维护性作为开发过程关注的焦点，原因如下：首先，软件系统的生命期很长，维护费用会远大于其初始开发费用。在可维护性上做妥协是不明智的，因为与长期的维护开销相比，最初的任何节省都会相形见绌。第二，系统在初始发布时，当时的开发技术很难达到高度的可靠性。可靠性是通过在开发阶段和软件系统整个生命期内反复修改错误达到的。软件系统的可靠性会因低劣的可维护性而无法增强。第三，高可维护性要求软件系统设计和实现上的灵活性。这种高度灵活性促进了增量开发方法，增量开发方法有利于增强软件的可靠性、可用性、用户友好性以及节省费用。

决定软件系统的可维护性的因素有以下几个：

灵活性：灵活性意味着能够很容易地修改软件的各个方面。修改所造成的影响能被限制在很小的区域内，而且能够在局部验证修改正确与否，也就是说只需检查受影响的小区域而不是检查整个软件。

简单性：人总是要犯错误的，这是不可避免的。但是，在处理很简单的事情时，人犯的错误就少得多，而且要确认事情处理得是否正确也要容易得多。即使有错误，错误也会很明显，纠正错误也会容易得多。复杂的软件系统可以通过使用分治法（divide-and-conquer）技术有效地被简化。

可读性：可维护性的一个前提是可读性，或者称为可理解性，因为在修改软件系统之前必须先理解它。可读性依赖于设计和程序代码的清晰和简单、相应文档的清晰和完整以及简单和一致的设计、实现和文档风格。

这些因素是我们在后面章节中对方法和技术进行讨论的重点。

1.2.4 软件开发是否是工程过程

尽管软件工程已有 30 多年的历史，但人们对“软件工程”的确切定义仍然没有取得一致意见，甚至对于把“软件工程师”作为一个专业头衔的合理性都存在争议。就像 Shaw 和 Garlan[1996]指出的那样，软件工程只是一个标签：它是“代表软件开发的管理过程、软件工具和设计活动的一个总称。但是，在实践上却与成熟的工程学科大相径庭。”

通过仔细研究传统的、成熟的工程学科，我们发现这些学科所拥有的几个本质特性在今天的软件开发中并不具备。

设计分析

几个世纪以来的历史清楚地证明，工匠的卓绝技艺可建造出宏伟的建筑，如埃及金字塔、罗马市政管道以及巴黎圣母院。但是，现代工程所拥有的确定性、可预测性及效率却是工匠技术所不能满足的。工程技术和工匠技术之间的一个主要区别在于工程项目的成功可以通过对工程设计的分析而得到保证，而工匠项目的成功则是通过在本项目或以前项目中的不断尝试、不断失败和不断总结而得来的。

土木工程师在项目破土之前依据力学，就能够信心十足地预测出他们新设计的大桥或大楼能够按照所期待的那样矗立及运行。航空工程以空气动力学为理论基础，依赖于空气动力学和模拟技术，航空工程师就能够充满信心地在新设计的飞机开始制造前，预测出它一定能飞起来。

相反，软件开发者却在很大程度上依赖于测试和调试（即尝试和失败）以建立他们对自己产品的信心。软件开发很像是用工匠技术建造现代摩天大厦，因为软件开发项目的成功很少能提前保证。

不重复失败

失败，有时候甚至是灾难也会在成熟的工程领域里发生。1940 年坍塌的 Tacoma Narrows 大桥也许是工程历史上最为“壮观”的失败之一。它的设计是非传统、富有创新性的，大桥外观异常秀丽端庄。大桥设计师做了详尽的分析，以确保它在 45 英里/小时的大风中完全能够承受其自重及设计载重。然而，始料未及的是，纤薄的桥面在弱于 40 英里/小时的中强侧风中产生了如飞机机翼一般的效应，升力和横向拉力将大桥桥身扭断。造成坍塌的原因一旦被了解，人们就找到了在未来阻止类似失败的手段。因此，在成熟的工程领域，相同类型的错误很少会重复。

在软件开发中，相同类型的错误却总是重复出现。几乎没有有效的手段能够保证特定类型的错误不出现。软件开发中令人难过的事实是，没有一个人能保证阿里亚娜五号的那种失败永远不再发生。

知识的积累与记载

成熟的工程领域的成功很大程度上归因于知识的积累和编纂成文，以使成功的经验和知识得以反复运用。设计知识和解决方案往往被组织并编纂为指南或手册，以使一般和常

规的设计不仅更简单更快捷，而且更加可靠、可信赖和可管理。设计者往往在手册中找解决方法，并把这些解决方法采纳、组织到他们特定的设计问题中去。难得会需要创新或新颖的解决方法。通常，指南或手册中包括正反两方面的经验，即什么是可行的，什么是该避免的。

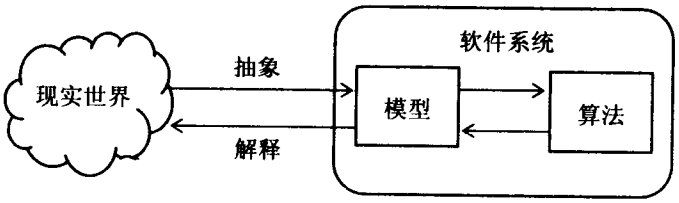
在软件开发中，尽管已经积累了许多设计知识和经验，但很少被系统地编纂成手册。无法受益于以往的经验和设计方案，每个软件系统的设计都被当成创新设计。因此，软件设计成为困难的、耗时的、不可靠的就不足为奇了。

因此，软件开发与传统的工程学科有很大的不同。充其量也就是一个不成熟的工程学科。为了使软件开发成为真正的工程学科，软件开发人员必须有一个机制，以完成设计分析、保证已知失败不重复出现并将设计知识编纂成手册。

1.3　面向对象的原则

1.3.1　对现实世界建模

软件开发的主要目的是建立软件系统，这些软件系统能为人们提供服务并增强人们在现实世界中解决问题的能力。一个软件系统一般由两个基本组成部分构成：一个是模型，它代表了现实世界中的相关部分；一个是算法，它包含操纵和处理模型所涉及到的计算。



现实世界是庞大复杂的。它的许多特性是模糊、未知或者是无法感知的。相反，软件系统中用到的编程模型却必须是精确的并且规模相对要小。模型必须是现实世界的“抽象”。它只从一个特定的视角获取现实世界里本质的和相关的特征，而忽略了其他特征。模型是用来被操纵或处理的，它们的行为应该模仿现实世界中的行为，以反映所选择的视角具有合理的正确性。操纵的结果能够通过“解释”（即将含义赋给模型中的实体）反馈到现实世界中，它往往代表了现实世界中问题的解决方案。

1.3.2　编程模型的演变

编程语言是软件开发人员用以描述计算机模型的主要工具。人们需要开发越来越复杂和越来越高效的模型，正是这个需要驱动了编程语言和编程方法学的演变。而这个需要又是被现代计算机持续增长的能力和人们迫切需要使用这些能力所驱动的。

软件开发中的一个基本问题是，“人们怎样对现实世界建立模型？”答案很大程度上