21世纪

21世纪高等院校创新精品规划教材

软件工程

主　编　景秀丽　吕洪林
副主编　高　红　栾　阳　郑学伟
主　审　杜文洁

中国水利水电出版社
www.waterpub.com.cn

21世纪高等院校创新精品规划教材

软件工程

主　编　景秀丽　吕洪林

副主编　高　红　栾　阳　郑学伟

主　审　杜文洁

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书详细地阐述了软件工程基础知识及其相关的实用技术。内容包括软件工程概述、软件可行性研究、软件需求分析、软件总体设计、软件详细设计、面向对象技术、统一建模语言 UML、程序编码、软件测试、软件实施与维护、软件重用技术、软件项目计划与管理、软件开发工具与环境以及软件开发实战（物流网络管理系统设计与实现）。

本书注重软件工程基本知识和基本概念的形象表述，注重内容系统性与实用性的恰当结合，力求做到通俗易懂，突出实践性。教材通过对软件工程常用方法的介绍，展现软件设计的实际运作过程，帮助读者掌握相关知识并在软件工程项目的开发上使用工程化标准。

本书适合高等院校本科层次和高职层次的计算机和软件技术相关专业的学生使用，也可作为软件开发技术培训的教材，同时还可作为从事软件开发人员的参阅资料。

本书配有免费电子教案，读者可以从中国水利水电出版社网站以及万水书苑下载，网址为：http://www.waterpub.com.cn/softdown/和 http://www.wsbookshow.com。

图书在版编目（CIP）数据

软件工程 / 景秀丽，吕洪林主编. -- 北京 : 中国水利水电出版社，2010.6
21世纪高等院校创新精品规划教材
ISBN 978-7-5084-7528-8

Ⅰ. ①软… Ⅱ. ①景… ②吕… Ⅲ. ①软件工程－高等学校－教材 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2010)第092851号

策划编辑：石永峰　责任编辑：李　炎　加工编辑：胡海家　封面设计：李　佳

书　名	21世纪高等院校创新精品规划教材 软件工程
作　者	主　编　景秀丽　吕洪林 副主编　高　红　栾　阳　郑学伟 主　审　杜文洁
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路1号D座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（营销中心）、82562819（万水）
经　售	全国各地新华书店和相关出版物销售网点
排　版	北京万水电子信息有限公司
印　刷	北京市天竺颖华印刷厂
规　格	184mm×260mm　16开本　17.5印张　426千字
版　次	2010年6月第1版　2010年6月第1次印刷
印　数	0001—3000册
定　价	30.00元

前　言

软件工程是一门运用工程学的原理和方法来组织和管理软件的开发、运行和维护，力求以高效率生产出高质量的软件产品的学科。软件工程是计算机科学技术领域的一个重要分支，在软件开发实战中起到基础指导的作用。

当今，国内软件工程教材较多，相关内容的深浅度、侧重点各有不同。我们依据高等院校本科生“软件工程”学科教学大纲所规定的教学要求编写了本教材，同时把多年来软件工程的教学经验和教学实践成果融入到书中，介绍了软件工程的基础理论、软件开发流程以及相关实践操作，从而指导学生进行软件开发活动。本教材也适用于高职高专计算机相关专业和软件技术相关专业的学生使用。

在教材编写上，本着深入浅出的原则，注重内容的先进性、系统性和实用性，力求反映软件开发技术的最新成果。在结构安排上，通俗易懂地阐述软件工程的基础理论知识，循序渐进，注重结合实践内容。在每章内容后面均附有小结和课后习题。

本书共 14 章，针对计算机专业和软件技术专业学生的实际特点设计了知识结构，全面系统地阐述了软件工程的知识体系、软件开发的相关流程，具体分布如下：

第 1 章，软件工程概述。主要介绍了软件工程的概念、软件生命周期和软件开发模型。

第 2 章，软件可行性研究。概括描述了可行性研究的任务、可行性研究的步骤和可行性研究的报告。

第 3 章，软件需求分析。系统介绍了需求分析概要、面向数据流的分析方法、需求分析方法与图形工具、实体一联系图和需求规格说明与评审。

第 4 章，软件总体设计。介绍了总体设计概要、总体设计的图形描述工具、模块化设计和面向数据流的设计方法。

第 5 章，软件详细设计。介绍了详细设计概述、详细设计的图形描述工具、Jackson 设计方法和 Warnier 设计方法。

第 6 章，面向对象技术。介绍了面向对象技术的概述、面向对象的开发模型、面向对象的分析、面向对象的系统设计、面向对象的实现、面向对象的程序设计语言以及类和应用程序的实现。

第 7 章，统一建模语言 UML。介绍了 UML 基本概念、UML 的概念模型、UML 的静态建模机制、UML 的动态建模机制和 UML 的物理架构建模。

第 8 章，程序编码。介绍了程序设计语言、结构化程序设计、程序设计风格、程序设计效率和程序复杂性度量。

第 9 章，软件测试。介绍了软件测试的基本概念、软件测试方法、软件测试流程、测试用例的设计、面向对象软件测试和软件测试的相关文档。

第 10 章，软件实施与维护。介绍了软件产品的实施、软件产品的维护活动、软件维护过程、软件维护文档、软件可维护性和软件维护的深化（软件再工程）。

第 11 章，软件重用技术。介绍了软件重用技术概要、基于构件的软件开发和面向对象的软

件重用技术。

第 12 章，软件项目计划与管理。介绍了软件项目的计划与组织、软件成本估算及控制、软件工程标准与软件文档等。

第 13 章，软件开发工具与环境。介绍了软件开发工具、软件工程环境和 CASE 技术。

第 14 章，软件开发实战。物流网络管理系统设计与实现。

本书由景秀丽、吕洪林任主编，高红、栾阳、郑学伟任副主编，陈悦、关雪梅参编。第 1、2、12 章由关雪梅编写，第 3、4 章由陈悦编写，第 5、6、11 章由栾阳编写，第 9、10 章由景秀丽编写，第 7、8、13、14 章由吕洪林编写。杜文洁老师作为整本书的主审，提出了许多宝贵的意见，郑学伟老师担任部分编写和校对工作。

由于时间仓促及作者水平有限，书中难免存在错误和不妥之处，恳请广大读者批评指正。

编 者

2010 年 4 月

目　录

第 1 章　软件工程概述

【本章引言】

自从 1946 年第一台电子计算机诞生以来，计算机的研究、生产和应用都得到迅猛的发展，计算机系统已经经历了四个不同的发展阶段，计算机科学也成为当今世界上发展最快和应用最广的学科之一。然而，我们仍然没有彻底摆脱“软件危机”的困扰，软件已经成为限制计算机系统发展的关键因素。为了克服这种困扰，软件工作者在不断地研究消除软件危机的方法，从而逐步形成了计算机科学技术领域中的一门新兴的工程学科——软件工程。

本章将对软件概念、软件分类、软件发展、软件危机、软件工程概念、软件生命周期及软件开发模型等方面做简要的介绍，通过本章学习，可为后几章软件工程的深入学习打下基础。

【本章重点】

- 软件工程概念
- 软件生命周期
- 软件开发模型

【学习目标】

- 理解软件工程的基本概念和软件生命周期
- 了解软件开发的几种模型

1.1　软件概述

1.1.1　软件的概念

随着计算机技术的快速发展，硬件性能有了极大的提高，计算机体系结构发生了深刻的变化，内存和存储容量的快速增加以及各种各样输入和输出的选择等，所有这些都促进了更高级和更为复杂的基于计算机系统的开发。

软件是计算机系统中与硬件相互依存的另一部分，它是包括程序、数据及其相关文档的完整集合。其中，程序是按事先设计的功能和性能要求执行的指令序列；数据是使程序能正常操纵信息的数据结构；文档是与程序开发、维护和使用有关的图文材料。

1.1.2　软件的分类

软件的类别多种多样，随着软件复杂程度的增加，软件之间的界限越来越不明显。按软件的作用，一般可以分为以下几类。

1. 系统软件

系统软件（System Software）是指能与计算机硬件紧密配合在一起，使计算机系统各个部件、相关的软件和数据协调高效地工作的软件。系统软件是计算机系统的重要组成部分，它支持应用软件的开发和运行。系统软件包括：操作系统、网络软件、编译程序、数据库管理程序、文件编辑系统、系统检查与诊断软件等。

2. 应用软件

应用软件（Application Software）则是在系统软件基础上，为解决特定领域的应用而开发的软件。按其性质不同可以分为以下几类：

（1）事务软件。事务信息处理是一个最大的软件应用领域，如工资单、收/支计算、存货盘点报表等。这些独立的系统可以组成管理信息系统（MIS）软件，它从一个或多个装有事务信息的数据库中存取数据。在这个领域中的应用是重新建立已有的数据，便于事务操作或做出管理决策。另外，除了传统的数据处理应用，事务软件还可以实现交互计算（如营业点的交易处理）。

（2）实时软件。监视、分析和控制现实世界中发生的事件，能以足够快的速度对输入信息进行处理并在规定的时间内做出反应的软件，称之为实时软件。实时软件包括四个组成部分：数据采集器（负责从外部环境中获取和格式化信息）、分析器（负责将信息转换成应用所需要的形式）、输出/控制器（负责响应外部环境）、管理器（负责协调系统各个部件工作，使系统能保持在一个可接受的响应时间内做出实时响应）。实时系统必须在严格的时间范围内响应，因此实时软件和计算机系统必须有很高的可靠性和安全性。

（3）工程和科学软件。工程和科学软件具有数值算法的特点。其应用范围从天文学到火山学；从自动应力分析到空间航天飞机轨道动力学；从分子生物学到自动化制造。但是，在工程和科学领域中的新的应用已经远离传统的数值算法。计算机辅助设计（CAD）、系统模拟和其他交互应用系统已经做到具有实时和系统软件的特点。

（4）嵌入式软件。嵌入式计算机系统将计算机嵌入在某一系统之中，使之成为该系统的重要组成部分、控制该系统的运行，进而实现一个特定的物理过程。用于嵌入式计算机系统的软件称为嵌入式软件（Embedded Software）。大型的嵌入式计算机系统软件可用于航空航天系统、指挥控制系统、武器系统等；小型的嵌入式计算机系统软件可用于工业的智能化产品之中，这时嵌入式软件驻留在只读存储内，为该产品提供各种控制功能和仪表的数字或图形显示功能等，如汽车的刹车控制、空调和洗衣机的自动控制等。

（5）个人计算机软件。个人计算机软件市场在过去的十几年逐渐兴起。发展了字处理、电子报表、计算机图形、家庭游戏、数据库管理、个人和事务财务应用、外部网络或数据库存取等数百种应用。事实上，个人计算机软件体现了一些最具有创新意识的软件——人机界面设计。

（6）基于 Web 的软件。浏览器检索的 Web 界面是软件，它结合了可执行的指令（CGI、HTML、Perl、Java）和数据（超文本、视频及音频等多种数据）。本质上，网络变成了一个巨大的计算机，提供了一个几乎无限的可通过任何调制解调器访问的软件资源。

（7）人工智能软件。人工智能（AI）软件采用非数值算法来解决不适于直接计算和分析的复杂问题。一般说来，这些问题都不能通过计算或直接分析而得到答案。迄今为止，在专家系统、模式识别、自然语言理解、人工神经网络、程序验证、自动程序设计、机器人学等领域开发了许多人工智能应用软件，用于疾病诊断、产品检测、自动定理证明、图像和语言自动识别、语言翻译等。

3. 工具软件

工具软件是 20 世纪 80 年代发展起来的，它是系统软件和应用软件之间的支持软件。一般用来辅助和支持开发人员开发和维护应用软件，以提高软件的开发质量和生产率。它包括需求分析工具、设计工具、编码工具、测试工具、维护工具和管理工具等。工具软件又可分为垂直工具软件和水平工具软件。垂直工具软件是指生命周期的某一阶段特定活动所使用的工具软件，如分析、设计、测试等活动；水平工具软件是指整个生命周期活动所使用的工具软件，如项目管理、配置管理等活动。

4. 可重用软件

可重用技术是最近几年提出来的。实际上过去就有这种技术，如各种标准程序库，通常它是计算机厂家提供的系统软件中的一部分，对这些标准程序库里的标准子程序稍加改造，甚至不经改造就可以把它们编入新开发的程序。但过去的这种标准程序应用面比较窄，大多只限于一些数学子程序。今天，世界已把可重用范围扩展到算法以外，数据结构也可以重用。20 世纪 90 年代的可重用构件则是把数据和相应的操作两者封装在一起（通常叫作类或对象），使软件工程师能够用可重用构件来建立新的应用程序。例如，现在的交互式界面一般就是用这种可重用构件组成的。这些可重用构件能够建立图形窗口、下拉菜单以及各种交互机制。建立这样的界面所需要的数据结构和处理细节都包含在一个由界面构件所组成的可重用库里。

1.1.3　软件的发展

自 20 世纪 40 年代世界上第一台计算机问世以来，软件经历了程序设计、程序系统及软件工程三个阶段的发展。

（1）程序设计阶段（20 世纪 50 年代至 60 年代）的软件指的是程序，程序的开发采用个体工作的方式，开发工作主要依赖于开发人员的个人技能和程序设计的技巧，所以软件的质量得不到保证，缺乏与程序有关的文档。

（2）程序系统阶段（20 世纪 60 年代至 70 年代）的软件是指程序和说明书，软件开发采用开发小组工作的方式，开发工作主要依赖于开发小组的水平，所以有时文档资料的不齐全会给软件维护带来很大的难度，软件技术的发展不能满足需要。

（3）软件工程阶段（20 世纪 70 年代以后）的软件则是指程序、文档和数据，由开发小组及大中型软件开发机构承担开发软件的任务，开发工作主要依赖于整个机构的管理水平，采用多种开发技术。

目前在很多应用领域，人们开始采用面向对象的软件开发技术。专家系统、人工智能软件开始走向实际应用。软件技术呈现国际化、网络化、服务化等多种发展趋势。互联网作为 20 世纪最重要的科技成果之一，给人类生活和经济发展都带来了深远的影响，它所展现出的勃勃商机，吸引了众多厂商围绕互联网开发软件，与分布计算、网络和互联网相关的软件技术成为软件领域的主要技术热点。此外，自由软件潮流、智能化、简易化、多样化等趋势正极大地拓展软件产业的发展空间，派生出许多具有成长潜力的新兴领域。

1.1.4　软件危机

1. 软件危机的含义

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。这些问题绝

不仅仅是不能正常运行的软件才具有的。实际上，几乎所有软件都不同程度地存在这些问题。

概括地说，软件危机包含以下两方面的问题：第一，如何开发软件，以满足对软件日益增长的需求；第二，如何维护数量不断膨胀的已有软件。鉴于软件危机的长期性和症状不明显的特征，近年来有人建议把软件危机更名为“软件萧条(Depression)”或“软件困扰(Affliction)”。不过“软件危机”这个词强调了问题的严重性，而且也已被绝大多数软件工作者所熟悉，所以本书仍将沿用它。

2. 产生软件危机的原因

产生软件危机的原因很多，总结起来有以下几点：

(1) 由于缺乏软件开发经验和有关软件开发数据的积累，使得开发工作的计划很难制定，以致经常出现超出经费预算、无法遵循进度计划、完成开发的期限一再拖延等情况。

(2) 软件需求在开发的初期阶段不够明确或是未能得到确切的表达。开发工作开始后，软件人员和用户又未能及时交换意见，造成的矛盾在开发后期集中暴露。

(3) 开发过程没有公认、统一的方法论和规范进行指导，参加开发的人员各行其事。另外，设计和实现过程的资料很难维护。

(4) 未能在测试阶段做好充分的检测工作，提交至用户的软件质量差，在运行过程中暴露出大量的问题。

3. 软件危机的解决方法

为了消除软件危机，第一，应该对计算机软件有一个正确的认识。彻底清除在计算机系统早期发展阶段形成的“软件就是程序”的错误观念。一个软件必须由一个完整的配置组成。事实上，软件是程序、数据及相关文档的完整集合。其中，程序是能够完成预定功能和性能的可执行的指令序列；数据是使程序能够适当地处理信息的数据结构；文档是开发、使用和维护程序所需要的图文资料。1983 年 IEEE 为软件下的定义是：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程序时所必须的数据。虽然表面上看来在这个定义中列出了软件的 5 个配置成分，但是方法和规则通常是在文档中说明并在程序中实现的。

第二，更重要的是，必须充分认识到软件开发不是某种个体劳动的神秘技巧，而应该是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目。必须充分吸取和借鉴人类长期以来从事各种工程项目所积累的行之有效的原理、概念、技术和方法，特别要吸取几十年来人类从事计算机硬件研究和开发的经验教训。

第三，应该推广使用在实践中总结出来的开发软件的成功的技术和方法，并且研究探索更好更有效的技术和方法，尽快消除在计算机系统早期发展阶段形成的一些错误的概念和做法。

第四，应该开发和使用更好的软件工具。正如机械工具可以“放大”人类的体力一样，软件工具可以“放大”人类的智力。在软件开发的每个阶段都有许多烦琐重复的工作需要做，在适当软件工具的辅助下，开发人员可以把这类工作做得既快又好。如果把各个阶段使用的软件工具有机地集合成一个整体，支持软件开发的主过程，则称为软件工程支撑环境。

总之，为了消除软件危机，既要有技术措施（方法和工具），又要有必要的组织管理措施。软件工程正是从管理和技术两方面研究如何更好地开发和维护计算机软件的一门新兴学科。

1.2　软件工程的概念

1.2.1　软件工程的定义和原理

1. 软件工程的定义

软件工程（Software Engineering）这个名词是北大西洋公约组织（NATO）科学技术委员会于 1968 年秋在当时的联邦德国召集了近 50 名第一流的编程人员、计算机科学家和工业界巨头，制定摆脱软件危机的办法时提出来的。尽管当时专家们无法设计出一张指导软件业走向更牢固阵地的详细路线图，但他们借鉴硬件工程的办法，为切实解决软件这一难题，不仅创造了一个新名词——软件工程，还使软件工程有了方向。从 1968 年到现在已经过去 40 多年，应该说，在今天，软件工程已发展成为一门独立的学科。

关于什么是软件工程的定义，每一位软件工程人员都给出了自己的不同理解。他们把它定义为：运用工程学的原理和方法来组织和管理软件的生产和维护，以保证软件产品开发、运行和维护的高质量和高生产率。

1993 年，IEEE 在《IEEE Standard Collection：Software Engineering》一文中给出了以下全面的定义：

（1）应用系统的、规范的和可量化的方法去开发、运行和维护软件，即软件的工程化应用。

（2）对（1）中所述方法的研究。

2. 软件工程的基本原理

自从 1968 年在联邦德国召开的国际会议上正式提出并使用了“软件工程”这个术语以来，研究软件工程的专家学者们陆续提出了 100 多条关于软件工程的准则或“信条”。著名的软件工程专家 B.W.Boehm 综合这些学者们的意见并总结了 TRW 公司多年开发软件的经验，于 1983 年在一篇论文中提出了软件工程的 7 条基本原理。他认为这 7 条原理是确保软件产品质量和开发效率原理的最小集合。这 7 条原理是互相独立的，其中任意 6 条原理的组合都不能代替另一条原理。因此，它们是缺一不可的最小集合。然而这 7 条原理又是相当完备的，人们虽然不能用数学方法严格证明它们是一个完备的集合，但是可以证明在此之前已经提出的 100 多条软件工程原理都可以由这 7 条原理的任意组合蕴含或派生。

下面简要介绍软件工程的 7 条基本原理。

（1）用分阶段的生命周期计划严格管理。经统计发现，在不成功的软件项目中有一半左右是由于计划不周造成的，可见把建立完善的计划作为第一条基本原理是吸取了前人的教训而提出来的。

在软件开发与维护的漫长生命周期中，需要完成许多性质各异的工作。这条基本原理意味着，应该把软件生命周期划分成若干个阶段，并相应地制定出切实可行的计划，然后严格按照计划对软件的开发与维护工作进行管理。Boehm 认为，在软件的整个生命周期中应该制定并严格执行六类计划，它们是项目概要计划、里程碑计划、项目控制计划、产品控制计划、验证计划和运行维护计划。

不同层次的管理人员都必须严格按照计划各尽其职地管理软件开发与维护工作，绝不能受客户或上级人员的影响而擅自背离预定计划。

（2）坚持进行阶段评审。人们已经认识到，软件的质量保证工作不能等到编码阶段结束之后再进行。这样说至少有两个理由：第一，大部分错误是在编码之前造成的，例如，根据Boehm 等人的统计，设计错误占软件错误的 63%，编码错误仅占 37%；第二，错误发现与改正得越晚，所需付出的代价也越高。因此，在每个阶段都进行严格评审，以便尽早发现在软件开发过程中所犯的错误，是一条必须遵循的重要原则。

（3）实行严格的产品控制。在软件开发过程中不应随意改变需求，因为改变一项需求往往需要付出较高的代价。但是，在软件开发过程中改变需求又是难免的。由于外部环境的变化，相应地改变用户需求是一种客观需要，显然不能硬性禁止客户提出改变需求的要求，而只能依靠科学的产品控制技术来顺应这种要求。也就是说，当改变需求时，为了保持软件各个配置成份的一致性，必须实行严格的产品控制，其中主要是实行基准配置管理。所谓基准配置又称为基线配置，它们是经过阶段评审后的软件配置成份（各个阶段产生的文档或程序代码）。基准配置管理也称为变动控制：一切有关修改软件的建议，特别是涉及到对基准配置的修改建议，都必须按照严格的规程进行评审，获得批准后才能实施修改。绝不能谁想修改软件（包括尚在开发过程中的软件）就随意进行修改。

（4）采用现代程序设计技术。从提出软件工程的概念开始，人们一直把主要精力用于研究各种新的程序设计技术。20 世纪 60 年代末提出的结构化程序设计技术，已经成为绝大多数人公认的程序设计技术。以后又进一步发展出各种结构分析（SA）与结构设计（SD）技术。近年来，面向对象技术已经在许多领域中迅速地取代了传统的结构化开发方法。实践证明，采用先进的技术不仅可以提高软件开发和维护的效率，而且可以提高软件产品的质量。

（5）结果应能清楚地审查。软件产品不同于一般的物理产品，它是看不见摸不着的逻辑产品。软件开发人员（或开发小组）的工作进展情况可见性差，难以准确度量，从而使得软件产品的开发过程比一般产品的开发过程更难于评价和管理。为了提高软件开发过程的可见性，更好地进行管理，应该根据软件开发项目的总目标及完成期限，规定开发组织的责任和产品标准，从而使得所得到的结果能够清楚地审查。

（6）开发小组的人员应该少而精。这条基本原理的含义是，软件开发小组的组成人员要有较高的素质，而且人数不宜过多。开发小组人员的素质和数量，是影响软件产品质量和开发效率的重要因素。素质高的人员的开发效率比素质低的人员的开发效率可能高几倍甚至几十倍，而且素质高的人员开发的软件中的错误明显少于素质低的人员开发的软件中的错误。此外，随着开发小组人员数量的增加，因为交流情况、讨论问题而造成通信开销也急剧增加。当开发小组人数为 N 时，可能的通信路径有 N(N-1)/2 条，可见随着人数 N 的增大，通信开销将急剧增加。因此，组成少而精的开发小组是软件工程的一条基本原理。

（7）承认不断改进软件工程实践的必要性。虽然遵循上述六条基本原理，就能够按照当代软件工程基本原理实现软件的工程化生产。但是，仅依靠上述六条原理并不能保证软件开发与维护的过程能赶上时代前进的步伐和技术的不断进步。所以，Boehm 提出应把承认不断改进软件工程实践的必要性作为软件工程的第七条基本原理。按照这条原理，不仅要积极主动地采纳新的软件技术，而且要注意不断总结经验。如收集进度和资源耗费数据、收集出错类型和问题报告数据等。这些数据不但可以用来评价新的软件技术的效果，而且可以用来指明必须着重开发的软件工具和应该优先研究的技术。

1.2.2　软件工程的目标

组织实施软件工程项目，要从技术上和管理上采取多项措施来最终得到项目的成功。成功指的是达到以下几个主要目标：

- 付出较低的开发成本。
- 达到要求的软件功能。
- 取得较好的软件性能。
- 开发的软件易于移植。
- 需要较低的维护费用。
- 能按时完成开发工作并及时交付使用。

在实际开发的具体项目中，企图让以上几个目标都达到理想的程度往往是非常困难的。而且上述目标很可能是互相冲突的。一方面，若只顾降低开发成本，很可能同时也降低了软件的可靠性。另一方面，如果过于追求提高软件的性能，可能造成开发出的软件对硬件有较大的依靠，从而直接影响到软件的可移植性。

如图 1-1 所示表明了软件工程目标之间存在的相互联系。其中有些目标之间是互补关系，如易于维护和高可靠性之间、低开发成本与按时交付之间。还有一些目标是彼此互斥的，如上述指出的互相冲突的情况。

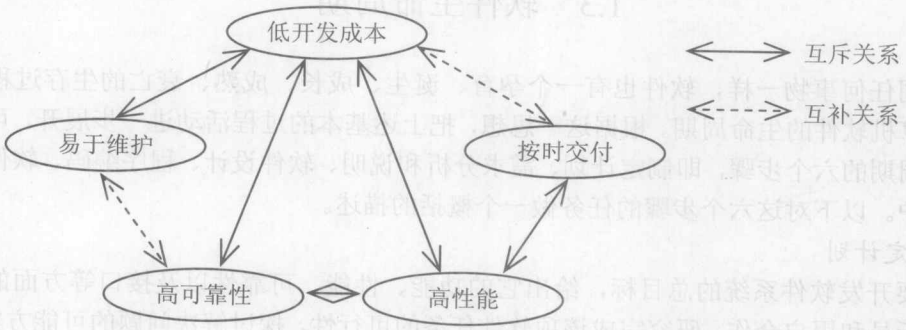

图 1-1　软件工程目标之间的关系

这里提到的几个目标很自然地成为判断软件开发方法或管理方法优劣的衡量尺度。如果提出一种新的开发方法，人们关心的是它满足哪些目标比现有的方法更有利。实际上，实施软件开发项目就是力图使以上目标的冲突取得一定程度的平衡。

1.2.3　软件工程的原则

软件工程的目的是提高软件生产率、提高软件质量、降低软件成本。为了达到这个目的，在软件的开发过程中必须遵循以下软件工程原则。

- 抽象。抽象事物最基本的特征和行为，忽略非基本细节。采用分层次抽象，自顶向下、逐层细化的办法控制软件开发过程的复杂性。
- 信息隐藏。将模块设计成“黑箱”，实现细节隐藏在模块内部，不让模块的使用者直接访问，这就是所谓信息封装（使用与实现分离）的原则。使用者只能通过模块接口

访问模块中封装的数据。

- 模块化。模块是程序中在逻辑上相对自主的成分，是独立的编程单位，应有良好的接口定义。如C语言程序中的函数过程、C++语言程序中的类。模块化有助于信息隐蔽和抽象，有助于表示复杂的系统。
- 局部化。在一个物理模块内集中逻辑上相互关联的计算机资源，保证模块之间有松散的耦合，模块内部有较强的内聚，这有助于控制软件的复杂性。
- 确定性。软件开发过程中所有概念的表达应是确定的、无歧义的、规范的。这样有助于人们在交流时不会产生误解、遗漏，保证整个开发工作的协调一致。
- 一致性。整个软件系统（包括程序、文档和数据）的各个模块应使用一致的概念、符号和术语；程序内、外部接口应保持一致；软件同硬件、操作系统的接口应保持一致；用于形式化规格说明的公理系统应保持一致。
- 完备性。软件系统不会丢失任何重要成分，可以完全实现系统所要求的功能。为了保证系统的完备性，在软件开发和运行过程中需要严格的技术评审。
- 可验证性。开发大型的软件时需要对系统自顶向下、逐层分解。系统分解应遵循使系统易于检查、测试、评审的原则，以确保系统的正确性。

使用一致性、完备性和可验证性可以帮助开发者设计一个正确的系统。

1.3 软件生命周期

正如同任何事物一样，软件也有一个孕育、诞生、成长、成熟、衰亡的生存过程。一般称其为计算机软件的生命周期。根据这一思想，把上述基本的过程活动进一步展开，可以得到软件生命周期的六个步骤，即制定计划、需求分析和说明、软件设计、程序编码、软件测试及运行与维护。以下对这六个步骤的任务做一个概括的描述。

1. 制定计划

确定要开发软件系统的总目标，给出它的功能、性能、可靠性以及接口等方面的要求；由系统分析员和用户合作，研究完成该项软件任务的可行性，探讨解决问题的可能方案，并对可利用的资源（计算机硬件、软件、人力等）、成本、可取得的效益、开发的进度做出估计，制定出完成开发任务的实施计划和可行性研究报告，提交管理部门审查。

2. 需求分析和说明

对待开发软件提出的需求进行分析并给出详细的定义。软件人员和用户共同讨论决定哪些需求是可以满足的，并对其加以确切的描述。然后编写出软件需求说明书或系统功能说明书及初步的系统用户手册，提交管理机构评审。

3. 软件设计

软件设计是软件工程技术的核心。在设计阶段中，设计人员把已确定了的各项需求转换成一个相应的体系结构。结构中的每一组成部分都是意义明确的模块，每个模块都和某些需求相对应，即总体设计。进而对每个模块要完成的工作进行具体的描述，为源程序编写打下基础，即详细设计。所有设计中的考虑都应以设计说明书的形式加以描述，以供后续工作使用并提交评审。

4. 程序编码

把软件设计转换成计算机可以接受的程序代码，即写成以某一种特定程序设计语言表示

的“源程序清单”。这一步工作也称为编码。自然，写出的程序应当是结构良好、清晰易读的，且与设计相一致。

5. 软件测试

软件测试是保证软件质量的重要手段，其主要方式是在设计测试用例的基础上检验软件的各个组成部分。首先是进行单元测试，查找各模块在功能和结构上存在的问题并加以纠正；其次是进行组装测试，将已测试过的模块按一定顺序组装起来；最后按规定的各项需求，逐项进行有效性的测试，决定已开发的软件是否合格，能否交付用户使用。

6. 运行与维护

已交付的软件投入正式使用，便进入运行阶段。这一阶段可能持续若干年甚至几十年。软件在运行中可能由于多方面的原因，需要对它进行修改。其原因可能有：运行中发现了软件中的错误需要修正；为了适应变化了的软件工作环境，需做适当的变更；为了增强软件的功能需做变更。

1.4　软件开发模型

为了指导软件的开发，用不同的方式将软件生存周期中的所有开发活动组织起来，即形成不同的软件开发模型。软件开发模型是从软件项目需求定义直至软件经使用后废弃为止，跨越整个生命周期的系统开发、运作和维护所实施的全部过程、活动和任务的结构框架。到现在为止，已经提出了多种软件开发模型。如瀑布模型、快速原型模型、增量模型、螺旋模型、喷泉模型等。

1.4.1　瀑布模型

瀑布模型（Waterfall Model）是 1970 年由 W.Royce 提出的。瀑布模型规定了各项软件工程活动，包括制定开发计划、进行需求分析和说明、软件设计、程序编码、软件测试及运行与维护。如图 1-2 所示。并且规定了它们自上而下、相互衔接的固定次序，如同瀑布流水，逐级下落。

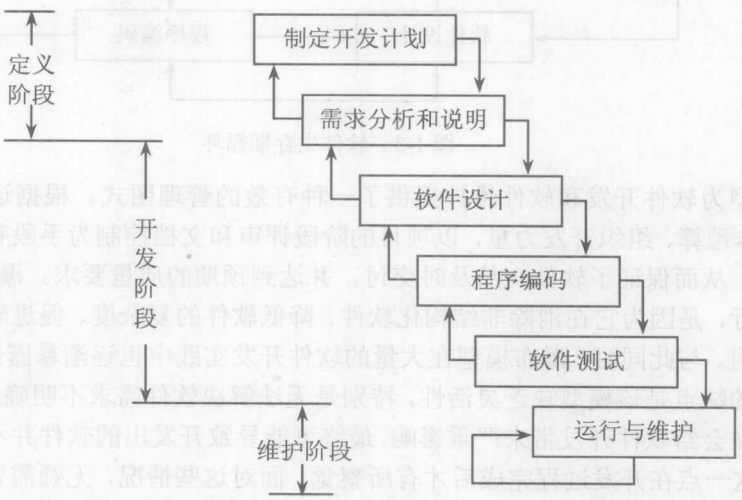

图 1-2　软件生命周期的瀑布模型