普通高等教育“十二五”规划教材

软件工程

RUAN JIAN GONG CHENG

主编◎胡亚慧

辽宁大学出版社

普通高等教育“十二五”规划教材

软件工程

主　编　胡亚慧
副主编　程敏　涂文婕　郭乐江
参　编　肖蕾　胡亚敏

辽宁大学出版社

图书在版编目(CIP)数据

软件工程 / 胡亚慧主编. — 沈阳 : 辽宁大学出版社，2013.6
普通高等教育“十二五”规划教材
ISBN 978－7－5610－7310－0

Ⅰ. ①软… Ⅱ. ①胡… 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(2013)第 123487 号

出 版 者：辽宁大学出版社有限责任公司
（地址：沈阳市皇姑区崇山中路 66 号　邮政编码：110036）
印 刷 者：北京明兴印务有限公司
发 行 者：辽宁大学出版社有限责任公司
幅面尺寸：185mm×260mm
印　　张：16
字　　数：340 千字
出版时间：2013 年 6 月第 1 版
印刷时间：2013 年 6 月第 1 次印刷
责任编辑：张琢石　黄 铮
封面设计：可可工作室
责任校对：齐 悦

书　　号：ISBN 978－7－5610－7310－0
定　　价：32.00 元
联系电话：86864613
邮购热线：86830665
网　　址：http://WWW.lnupshop.com
电子邮件：lnupress@vip.163.com

前　言

软件工程是为计算机相关专业开设的一门重要课程。本书编写目的是让学生针对软件工程项目，能够全面掌握可行性分析、软件需求分析、软件总体设计、软件详细设计、软件测试、软件维护、面向对象的软件工程、软件项目管理等阶段的方法和技术。希望通过该课程的学习，提高学生按照软件工程的原理、方法、技术、标准和规范进行软件开发的能力，从而培养学生的合作意识和团队精神，培养学生编写技术文档的能力，进一步提高学生做软件工程的综合能力和项目管理能力。

软件工程不仅仅是一门课程，更是一项工程。它除了本身拥有的技术之外，还包括了适应其自身需要的管理方法和管理体制。本书共分为十章，围绕软件工程技术，涉及软件工程生命周期各阶段采用的软件开发技术、软件开发工具和环境。为了能够更好地理解相关知识，书中通过大量的实例加以解释说明，有助于对所学内容进行实际运用。

本书总结了我们多年软件工程教学与实践的经验。在本书编写的过程中，程敏，涂文婕花费了大量的时间和精力编写教材。同时，我们得到了鲁汉榕、李石君、郭乐江、肖蕾、胡亚敏的支持，在此对他们表示感谢。

由于软件工程是一门新兴学科，软件工程的教学方法本身还在探索之中，加之我们的水平和能力有限，错误及不当之处在所难免，敬请广大读者和同行们批评指正，以便今后的改进和提高。

胡亚慧

2012 年 12 月

目 录

第一章 软件工程概述

本章主要介绍软件及软件工程的概念、软件危机、软件工程基本原则、软件生命周期和常用的软件开发模型。其中，重难点部分是软件工程的概念、软件生命周期和常用的软件开发模型。

1.1 软件工程的概念

1.1.1 软件

把软件(Software)简单地视为程序(Program)，这样的理解是非常狭隘的。目前，计算机软件被公认的解释是程序、数据及其相关文档的完整集合。可以简单地说，软件由两部分组成，即程序与文档(Document)：一是机器可以执行的程序及有关的数据；二是机器不可以执行的文档。有以下两种较为普遍的定义：

1. 软件是与计算机系统操作有关的程序、规程、规则及任何与之相关的文档和数据。

2. 软件是程序以及开发、使用和维护程序所需要的文档，包括机器运行所需要的各种程序及有关资料。

程序是为了解决某一个问题而按事先设计的功能和性能要求执行的指令系列；或者说，用程序设计语言描述的适合于计算机处理的语句序列。

数据是使程序能正常操纵信息的数据结构。

文档是描述程序、数据和系统开发以及使用的各种图文资料。它具有永久性并能供人或机器阅读。文档的作用是：①记录；②通信和交流；③控制软件生产过程；④管理软件；⑤维护软件；⑥软件产品介绍等。

1.1.2 软件的特点

1. 计算机软件产品是一种逻辑产品部件而不是物理产品部件，它具有抽象性。软件可以记录在纸面上，保存在磁盘、磁带中，写在光盘上，可以在计算机上运行却无法看到它的形态。因为软件可以复制出大量同一内容的副本，由此带出一个软件产品的知识产权问题，有必要在技术上和法律上进行研究，并且采取有力的防范措施。

2. 软件的生产与硬件不同，软件是通过人们的智力活动，把知识与技术转化成信息的一种产品。与硬件相比，软件的开发更依赖于开发者的业务素质和能力，人员的组织、合作和管理。大多数的软件开发几乎都是这样。

3. 软件在开发过程中，尽管经过了严格的测试和试用，但仍然不能保证软件是没有错误的。所以，软件投入使用后，仍需要进行维护。这就带来软件维护复杂性的问题。软件产品维护比硬件产品维护复杂得多。

4. 软件在运行期间，没有机械磨损，物理上不会老化，但在使用过程中可能发生故障而要进行多次的修改，每一次的修改又可能引入新的错误，从而导致软件失效率升高，使得软件退化。

5. 一个软件系统的开发，简单地增加人力并不能成比例地提高软件开发的数量和质量，它要求软件开发群体要有团体协作精神。一个系统开发的全过程中，对人员的组织、协调、管理等问题必须很好地解决，处理不周就会出现混乱状态，导致系统开发的失败。

1.1.3 软件危机

20 世纪 60 年代末 70 年代初，西方工业发达国家曾出现过软件危机。所谓软件危机是指在软件开发和维护过程中所遇到的一系列严重问题。具体地说，就是要解决：①如何开发新的软件，以满足对软件日益增长的需求；②如何维护数量不断膨胀的软件。

造成软件危机的原因很多，其中，由于软件本身的复杂性，人类无法抗衡或解决。软件产品的特殊性和人类智力的局限性，导致存在人们无力处理软件本身的复杂问题。首先，开发者不能有效地独立自主地处理大型软件的全部关系和各个分支问题以及错漏；其次，与软件本身的特点有关，软件是一种逻辑部件而不是物理部件，规模越来越庞大，而且越来越复杂，软件不像硬件一样容易维护；再其次，没有采用正确的开发方法和技术，缺乏有力的方法和技术以及工具方面的支持，过分地依靠技巧性和个人创造性。此外，软件开发人员对用户需求的理解与用户的本来愿望有差异，对一些软件需求描述不精确，导致软件开发时间长、质量差、维护难等问题。在大型软件项目的开发中，人员组织、协调得不好，缺乏科学的管理也是引起软件危机的重要因素。

1.1.4 软件工程

软件开发由于已不是过去的简单编程序，不能是个体劳动，程序量的增大又不能简单地多投入人力来解决，开发人员必须组成一个良好的群体，他们应该有各自的能力，素质好，能互相配合。完成一个软件产品就是完成一个工程项目。所以，人们设想、构思、沿用工业化比较成熟的经验和工程上的概念、原理、技术与方法进行软件开发。

1968 年北大西洋公约组织在联邦德国的一次学术会议上，首次提出软件工程的概念，从而形成一门新兴的学科——软件工程学。从此软件生产开始了一个新的飞跃。

1. 软件工程的定义

软件工程是一门工程学科，它涉及到软件生产的各个方面，从初期软件系统的构思和描述到系统维护，都属于该学科的研究范畴。尽管有许多对软件工程的定义，但主要的思想都是强调在软件开发中应用工程化原则的重要性。简单地说，软件工程是指导计算机软件开发和维护的工程学科，采用工程上成熟的概念、原理、技术、方法来开发和维护以及管理软件。

Boehm 曾为软件工程下了定义："运用现代科学技术知识来设计并构造计算机程序及为开发、运行和维护这些程序所必需的相关文件资料。"这里的"设计"应包括软件的需求分析和对软件修改时所进行的再设计活动。

Fairley 认为："软件工程学是为在成本限定以内按时完成开发和修改软件产品所需的系统生产以及维护技术和管理科学。"

FritzBauer 则给出一个软件工程学的定义："建立并使用完善的工程化原则，以较经济的手段获得能在实际机器上有效运行的可靠软件的一系列方法。"

IEEE1983 年的定义："软件工程是开发、运行、维护和修复软件的系统方法。"

1993 年 IEEE 又有进一步的定义："把系统化的、规范化的、可度量的途径应用于软件开发、运行和维护的过程，也就是把工程化应用于软件中……"

2. 软件工程方法学的内容

软件工程主要研究软件设计方法、软件工具、软件工程标准和规范、软件工程管理以及有关理论等。软件工程方法与技术、软件工具与环境以及软件过程三部分称为软件工程三要素。

软件工程方法是完成软件工程项目的技术手段。它应该包括多方面的任务，例如，项目计划与估算、软件需求分析、系统总体结构设计、数据结构设计、算法设计、编码、软件测试以及维护等。软件工程方法可以采用某种特殊的语言或图形的表达方法及一套质量保证标准。

软件工程使用的工具与环境是人类在开发软件的活动中智力和体力的扩展和延伸，它提供了自动或半自动的软件支撑环境。事实上，人们已经开发了许多软件工具，已经能够支持软件工程方法，支持各种软件文档的生成等等。而且有人试图把许多的软件工具按照一定的方法或模型组织起来，使得它成为软件工具集。在工具集中，一种工具产生的信息可以为其他的工具所使用，从而建立起一种称为计算机辅助软件工程(CASE)的软件开发支持系统。CASE 将各种软件工具、开发机器和存放开发过程信息的工程数据库组合起来形成一个软件工程环境。

3. 软件工程的基本目标

近几十年来，软件工程的理论和方法得到广泛的研究和发展，软件工具和技术也越来越成熟。人们一直希望在不同的应用领域中，有相应的一套系统软件方法。软件工程的基本目标如下：

(1)定义良好的方法学，面向计划并开发维护整个软件生存周期的方法学。

(2)确定的软件成分，记录软件生存周期每一步的软件文件资料，按步骤显示轨迹。

(3)可预测的结果，在生存周期中，每隔一定时间进行复审。

软件工程学的最终目的，是以较少投资获得易维护、易理解、可靠、高效率的软件产品。软件工程学是研究软件结构、软件设计与维护方法、软件工具与环境、软件工程标准与规范、软件开发技术与管理技术的相关理论。

1.1.5　软件工程基本原则

为了开发出低成本高质量的软件产品，软件工程学应遵循以下基本原则：

1. 分解

分解是人类分析解决复杂问题的重要手段和基本原则，其基本思想是从时间上或是从规模上将一个复杂抽象的问题分成若干个较小的、相对独立的、容易求解的子问题，然后分别求解。软件瀑布模型，结构化分析方法，结构化设计方法，Jackson 方法，模块化设计都运用了分解的原则。

2. 抽象和信息隐蔽

尽量将可变因素隐藏在一个模块内，将怎样做的细节隐藏在下层，而将做什么抽象到上一层做简化，从而保证模块的独立性。这就是软件设计独立性要遵守的基本原则。模块化和局部性的设计过程使用了抽象和信息隐蔽的原则。

3. 一致性

研究软件工程方法的目的之一，就是要使开发过程标准化，使软件产品设计有共同遵循的原则。要求软件文件格式一致，工作流程一致，且软件开发过程要标准化、统一化。

4. 确定性

软件开发过程要用确定的形式表达需求，表达的软件功能应该是可预测的。用可测试性、易维护性、易理解性、高效率的指标来具体度量软件质量。

组织实施软件工程项目，从技术和管理上采取了多项措施后，项目的成功主要要达到的目标有：开发成本较低；软件的功能能够达到用户要求并具有较好的性能；软件具有良好的可移植性，易于维护且维护费用较低；软件的开发工作能按时完成并及时交付使用。

1.2 软件生命周期

就像一个人从出生开始经过幼儿期、少年期、中年期、老年期直到死亡的过程一样，一个软件从提出设想到完成使命、报废为止，也经历了一个漫长的时期。通常把软件经历的这个漫长时期称为软件生存周期(Software Life Cycle)。软件生存周期是借用工程中产品生存周期的概念而得来的。引入软件生存周期概念对于软件生产的管理、进度控制有着非常重要的意义，使得软件生产有相应的模式、相应的流程、相应的工序和步骤。在划分软件生存周期阶段时，应遵循的一条基本原则是各阶段的任务应尽可能相对独立，同一阶段各项任务的性质尽可能相同，从而降低每个阶段任务的复杂程度，简化不同阶段之间的联系，有利于软件项目开发的组织管理。一般软件生存周期有软件定义、软件开发和运行维护三个时期组成。软件定义时期通常进一步划分成三个阶段，即问题定义、可行性研究和需求分析。开发时期通常有下述三个阶段组成：总体设计、详细设计、编码和测试。

1.2.1 问题定义

这一阶段的工作是确定“要解决的问题是什么”，如果不知道要解决什么问题就盲目去做，只会白白浪费时间和金钱，最终得到的结果很可能是毫无意义的。尽管确切地定义问题的必要性是十分明显的，但是在实践中它却可能是最容易被忽略的一步。

1.2.2 可行性研究

这一阶段的工作是确定系统在经济上、技术上和操作上是否可行。具体地说，就是要确定软件系统的工作范围；预测开发的系统所需要的资源，包括硬件、软件和人员；对软件开发成本进行初步估计，并写出软件计划任务书。

可行性分析的任务首先需要进行概要的分析研究，初步确定项目的规模和目标，确定项目的约束和限制，把它们清楚地列举出来。然后，分析员进行简要的需求分析，抽象出该项目的逻辑结构，建立逻辑模型。从逻辑模型出发，经过压缩的设计，探索出若干种可供选择的主要解决办法，对每种解决方法都要研究它的可行性。可从以下三个方面分析研究每种解决方法的可行性。

1. 技术可行性

对要开发项目的功能、性能、限制条件进行分析，确定在现有的资源条件下，技术风险有多大，项目是否能实现。这里的资源包括已有的或可以搞到的硬件、软件资源，现有技术人员的技术水平和已有的工作基础。

技术可行性常常是最难解决的方面，因为项目的目标、功能、性能比较模糊。技术可行性一

般要考虑的情况包括：

开发的风险：在给出的限制范围内，能否设计出系统并实现必需的功能和性能？

资源的有效性：可用于开发的人员是否存在问题？可用于建立系统的其他资源是否具备？

相关技术：相关技术的发展是否支持这个系统？

开发人员在评估技术可行性时，一旦估计错误，将会出现灾难性后果。

2. 经济可行性

进行开发成本的估算以及取得效益的评估，确定要开发的项目是否值得投资开发。对于大多数系统，一般衡量经济上是否合算，应考虑一个“底线”，经济可行性研究范围较广，包括成本效益分析、公司经营长期策略、开发所需的成本和资源、潜在的市场前景等。

3. 操作可行性

要开发的项目是否存在任何侵犯、妨碍等责任问题，要开发项目的运行方式在用户组织内是否行得通，现有管理制度、人员素质、操作方式是否可行。

操作可行性所涉及的范围也比较广，包括合同、责任、侵权、用户组织的管理模式及规范，其他一些技术人员常常不了解的陷阱等。

典型的可行性研究有下列几个步骤：

(1)确定项目规模和目标

分析员对有关人员进行调查访问，仔细阅读和分析有关的材料，对项目的规模和目标进行定义和确认，清晰地描述项目的限制和约束，确保分析员正在解决的问题确实是要解决的问题。

(2)研究正在运行的系统

正在运行的系统可能是一个人工操作的系统，也可能是旧的计算机系统，要开发一个新的计算机系统来代替现有系统。因此，现有的系统是信息的重要来源，要研究它的基本功能、存在什么问题、运行现有系统需要多少费用、对新系统有什么新的功能要求、新系统运行时能否减少使用费用等。

应该收集、研究、分析现有系统的文档资料，实地考察现有系统。在考察的基础上访问有关人员，然后描绘现有系统的高层系统流程图，与有关人员一起审查该系统流程图是否正确。这个系统流程图反映了现有系统的基本功能和处理流程。

(3)建立新系统的高层逻辑模型

根据对现有系统的分析研究，逐渐明确了新系统的功能、处理流程以及所受的约束，然后使用建立逻辑模型的工具——数据流图和数据字典来描述数据在系统中的流动和处理情况。注意，现在还不是软件需求分析阶段，不是完整、详细地描述，只是概括地描述高层的数据处理和流动。

(4)导出和评价各种方案

分析员建立了新系统的高层逻辑模型之后，要从技术角度出发，提出实现高层逻辑模型的不同方案，即导出若干较高层次的物理解法。根据技术可行性、经济可行性、社会可行性对各种方案进行评估，去掉行不通的解法，就得到了可行的解法。

(5)推荐可行的方案

根据上述可行性研究的结果，应该决定该项目是否值得去开发的问题。若值得开发，那么可行的解决方案是什么，并且说明该方案可行的原因和理由。该项目是否值得开发的主要因素是从经济上看是否合算。这就要求分析员对推荐的可行方案进行成本—效益分析。

(6)编写可行性研究报告

将上述可行性研究过程的结果写成相应的文档，即可行性研究报告，提请用户和使用部门仔细审查，从而决定该项目是否进行开发，是否接受可行的实现方案。

1.2.3 需求分析

需求分析阶段的任务不是具体地解决问题，而是准确地确定软件系统必须做什么，确定软件系统必须具备哪些功能。

用户了解他们所面对的问题，知道必须做什么，但是通常不能完整、准确地表达出来，也不知道怎样用计算机解决他们的问题。软件开发人员知道怎样用软件完成人们提出的各种功能要求，但是对用户的具体业务和需求不完全清楚。这是需求分析阶段的困难所在。系统分析员要和用户密切配合，充分交流各自的理解，充分理解用户的业务流程，完整地、全面地收集、分析用户业务中的信息和处理，从中分析出用户要求的功能和性能，并完整地、准确地表达出来。这一阶段要给出软件需求规格说明书。

1.2.4 概要设计

这一阶段的主要任务是解决系统“怎么做”的问题。概要设计决定软件系统的总体结构即模块结构。在这个阶段，开发人员要把确定的各项功能需求转换成需要的体系结构。在该体系结构中，每个成分都是意义明确的模块，即每个模块都和某些功能需求相对应，并给出模块的相互调用关系、模块间传递的数据及每个模块的功能说明。同时还要设计该项目的应用系统的总体数据结构和数据库结构，即应用系统要存储什么数据，这些数据是什么样的结构，它们之间有什么关系。这个阶段的文件资料是软件结构图和模块功能说明。

1.2.5 详细设计

详细设计对概要设计产生的功能模块逐步细化，把模块内部的细节转化为可编程的程序过程性描述。详细设计给出每一个模块内部过程的描述，包括算法与数据结构、数据分布、数据组织、模块间接口信息、用户界面等的设计。这一阶段完成软件详细设计说明书。

1.2.6 编码和测试

这一阶段是把软件设计方案加以具体实施，即根据软件详细设计说明书的要求，为软件系统中每一个功能模块编写程序并进行模块调试。当然，写出的程序应是结构好、清晰易读并且与设计相一致。

这一阶段的文件资料是程序说明书、源程序。

测试是保证软件质量的重要手段。软件测试阶段的主要任务是发现和排除错误，也就是对软件系统进行从上到下全面的测试和检验，看它是否符合软件总体设计方案规定的功能要求。这期间要提出测试标准，制订测试计划，确定测试方法。经过测试、纠错得到可运行的程序，同时写出软件测试报告。

1.2.7 维护

由于经过测试的软件仍然可能有错，用户的需求和软件的操作环境也可能发生变化，因此交付运行的软件系统仍然需要维护。所以，软件维护的实质是对软件继续进行查错、纠错和修改。一般维护分为以下几方面：

1. 改正性维护：对软件性能、功能、处理和实现中出现的错误进行纠正。
2. 适应性维护：当软件处理对象或数据环境变化时，依某些适应性进行修改。
3. 完善性维护：为了提高软件的性能或者对可维护性方面所做的某些修改。
4. 预防性维护：为了改善将来的可靠性或可维护性而对软件进行的修改或补充。

软件维护是软件生存周期中时间最长的阶段。已交付的软件投入正式使用后，便进入软件维护阶段，它可以持续几年甚至几十年。软件运行过程中可能由于各方面的原因，需要对它进行修改。其原因可能是运行中发现了软件隐含的错误而需要修改，也可能是为了适应变化了的软件工作环境而需要做适当变更，也可能是因为用户业务发生变化而需要扩充和增强软件的功能等。

软件生存周期划分为上述七个阶段，这为工程化地开发软件系统提供了一个框架。但是，实际的开发工作不可能是直线的，常常存在着反复。例如，在软件设计阶段发现软件规格说明书有不完整或定义不确切之处，就要回到需求分析阶段进行“再分析”；测试阶段发现模块内部或接口中的错误，就要回到设计阶段对原来的设计进行修改。

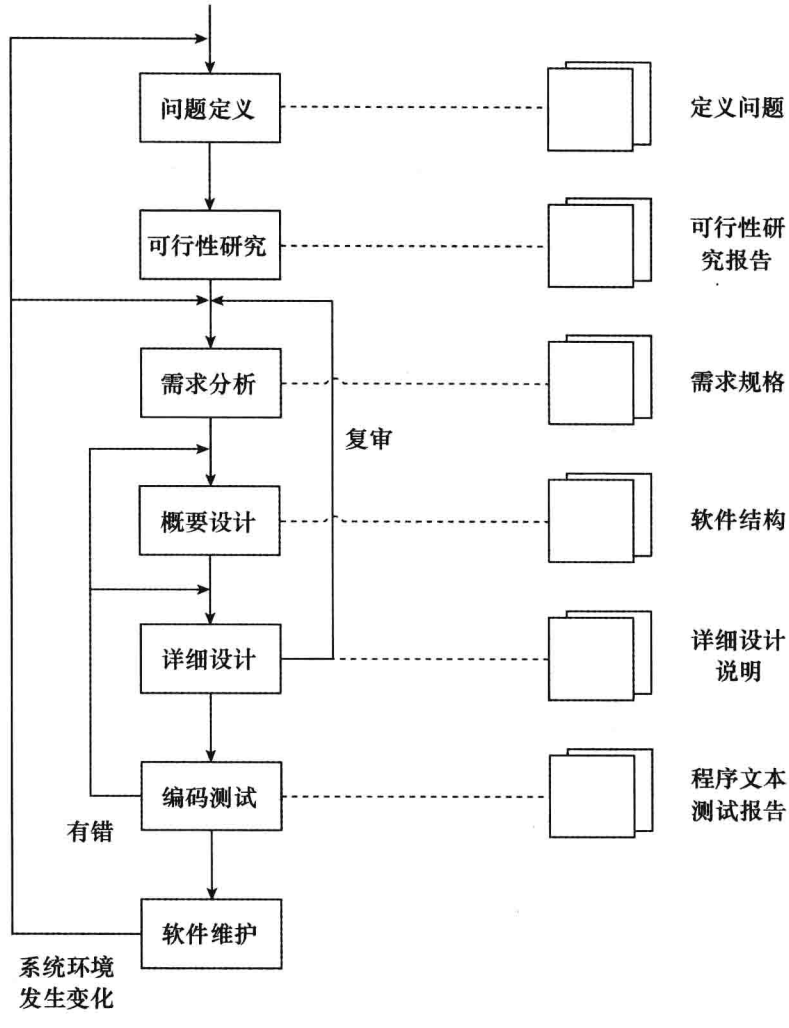

图 1.1　软件工程设计流程

实际的软件工程设计流程如图 1.1 所示。

在软件开发期中，测试阶段工作量约占整个开发期总工作量的 40%，而在软件的整个生存周期中软件维护的周期最长，工作量也最大。

1.3　软件开发模型

软件开发模型(Software Development Model)也称生命周期模型，是指反映整个软件生命

周期中系统开发、运行、维护等实施活动的一种结构框架。使用该软件开发模型能清晰、直观地表达软件开发全过程，明确规定软件生命周期划分的阶段，以及各个阶段要完成的主要活动和任务。软件开发模型可以作为指导软件项目开发的基础。到目前为止已经提出了多种软件开发过程模型，主要有瀑布模型、原型模型、增量模型、螺旋模型、喷泉模型、构件组装模型、统一过程模型 RUP 和第四代技术等。模型的选择是基于软件的特点和应用领域。下面介绍几种典型的模型。

1.3.1 瀑布模型

瀑布模型(Waterfall Model)是在 1970 年由 W. Royce 最早提出的软件开发模型。它将软件生命周期的各项活动规定为依固定顺序连接的若干阶段工作。这些工作之间的衔接关系是从上到下、不可逆转，如同瀑布一样，因此称为瀑布模型。传统的瀑布模型将软件开发过程划分成若干个互相区别而又彼此联系的阶段，这几个阶段分别为问题定义、需求分析、软件设计、编码、测试、运行和维护，每个阶段的工作都以上一个阶段工作的结果为依据，同时又为下一个阶段的工作提供前提。因此，这几个阶段的工作必须是前一个阶段完成后，才能开始下一个阶段的工作。阶段工作是顺序和依赖的关系，如图 1.2 中向下的箭头所示。

瀑布模型的顺序活动的特点，使得软件开发人员在进行开发活动时，必须按照阶段顺序安排工作，避免了软件开发人员接到任务后，急于开始编写程序，而忽略前期的各项准备工作。以往的经验告诉我们，当开发的软件项目较大时，编码开始的越早，项目完成的时间很可能越长。这是因为过早进入编码往往意味着大量的返工，而在编码过程中发现错误进行修改是一件十分麻烦的工作，如果有错误一直隐藏到编码结束，很可能出现无法弥补的问题，造成无法挽回的损失。因此，瀑布模型明确要求在项目前期，即需求分析和总体设计阶段只考虑系统的逻辑模型，不涉及系统具体实现，直到设计结束，软件开发人员主要考虑的是系统逻辑实现模型。将系统逻辑设计与物理实现严格分开，尽可能在设计结果通过审查确定无误时开始编码工作。推迟物理实现的观点是瀑布模型的一个重要指导思想。

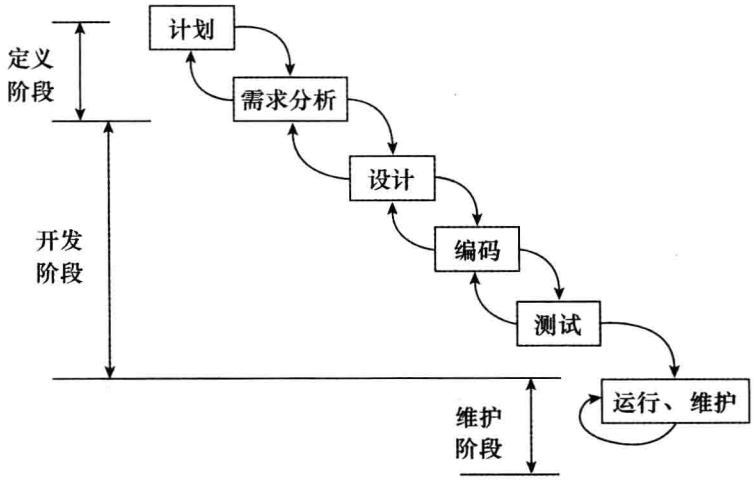

图 1.2 瀑布模型

为了提高软件质量，在瀑布模型中要求每个阶段的工作都要有完整、准确的文档资料，并且每个阶段结束前都要对文档进行审查，尽早发现问题，尽早解决，最大限度减少软件开发成本。若某阶段的评审未通过，则返回前一阶段，甚至更前的阶段，如图 1－2 中向上的箭头所示。

瀑布模型自提出以来，一直是一种被广泛采用的开发模型，它配合结构化方法和严格的软件开发管理手段，在软件工程化开发中起到了重要作用。但是，在经过长期的实践活动中，人们发现这种模型有如下一些缺点：

1. 在项目开始阶段，开发人员和用户对需求的描述常常是不全面的。用户由于对工作十分熟悉，因此常常忽略一些日常的行为活动，而开发人员通常对项目所涉及的领域不了解，也无法知道用户的描述是否全面，甚至可能存在对用户描述内容理解上的偏差。如果需求阶段未发现这些问题，就会影响到后面各阶段的工作。

2. 瀑布模型是由文档驱动的。也就是说，瀑布模型中各阶段所做的工作都是文档说明，那么这些描述文字对于不很熟悉计算机使用的用户来说，很难全面理解文字描述背后的软件产品。当用户在使用软件时往往会产生一些新的想法，或多或少地对软件的使用及功能方面提出一些意见，而这时对系统修改难度已经很大了。

3. 开发过程中，事先选择的技术或需求迅速发生变化，需要返回到前面某个阶段，对前面的一系列内容进行修改，这样势必影响整个软件开发角度。

总的来说，瀑布模型是一种应付需求变化能力较弱的开发模型，因而很多在该模型基础上开发出来的软件产品不能够真正满足用户需求。

1.3.2　原型模型

事实证明，很多用户在开始使用为他开发的系统时，会对系统的功能、界面或操作方式等提出许多新的建议，其中有一些建议是非常合理的，软件开发人员确实需要进行修改，而这样做势必增加开发成本。为了尽量减少这种状况发生，人们提出了原型模型技术。

原型模型(Prototype Model)的基本思想是：软件开发人员在与用户进行需求分析时，以比较小的代价快速建立一个能够反映用户主要需求的原型系统，让用户在计算机上进行操作，在实践过程中提出改进意见。开发人员根据用户的意见，对原型进行补充和完善，然后再由用户试用、评价、提出建议，重复这一过程，直到用户对开发的原型系统满意为止，如图 1.3 所示。

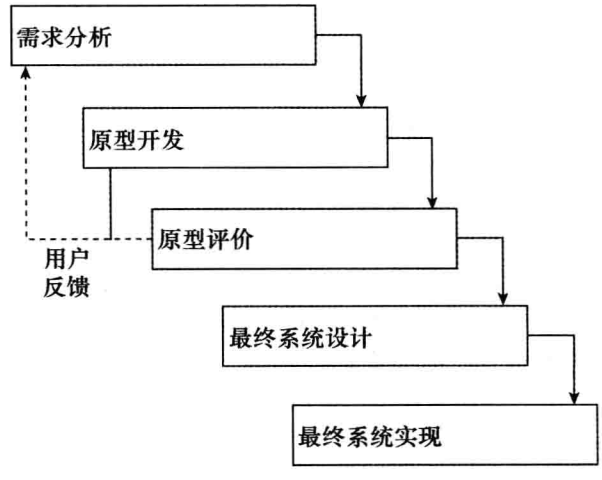

图 1.3　原型模型

开发人员根据该原型书写说明文档，作为后面开发工作的依据。采用原型模型有以下优点：

1. 原型模型让用户有机会实践系统的基本功能，因而可以对不尽合理的内容提出修改意见和建议。

2. 原型模型法可以使开发者和用户充分交流，对一些模糊需求也能够处理。

3. 开发人员通过建立原型模型对系统有了更深层次的理解，在设计和编码时可以尽量减少出错，有助于软件的开发工作顺利进行。

4. 当快速模型的某个部分是利用软件工具自动生成的时候，可以把这部分内容用到最终的软件系统中，比如一些界面或生成的报表等。

5. 原型模型使总的开发费用降低，时间缩短。

6. 原型模型可以使用户对系统更为满意，也有利于维护。

7. 用户在使用原型系统时已经对系统有了初步了解，因此，建立模型的过程也相当于是用

户的一个学习软件的过程。

需要提醒开发者注意的是，建立原型要避免进入以下误区：

1. 由于开发者对软件的应用领域不熟悉，将需求的次要部分当做主要内容建立模型。

2. 用户在原型模型基础上不断提出要求，开发者因为修改错误而忽略了主要部分的设计。

3. 修改原型模型同时忽略更新文档。

4. 不断修改原型模型以满足用户需求，但忘记了用户的实际应用环境是否适合。原型模型特别适合人机界面的设计，用户通过交互界面的内容，能够提出有关操作、功能上的建议；而对一些类似实时控制软件、嵌入式软件则不适合。

1.3.3 增量模型

增量模型也称为渐增模型，如图 1.4 所示。使用增量模型开发软件时，把软件产品作为一系列的增量构件来设计、编码、集成和测试。每个构件由多个相互作用的模块构成。使用增量模型时，第一个增量构件往往实现软件的基本需求，提供最核心的功能。例如使用增量模型开发字处理软件时，第一个增量构件提供基本的文件管理、编辑和文档生成功能；第二个增量构件提供更完善的编辑和文档生成功能；第三个增量构件实现拼写和语法检查功能；第四个增量构件完成高级的页面排版功能。把软件产品分解增量构件时，应该使构件的规模适中，规模过大或过小都不好，最佳分解方法因软件产品特点和开发人员的习惯而异。分解时必须遵守的约束条件是，当把新构件集成到现有软件中时，所形成的产品必须是可测试的。

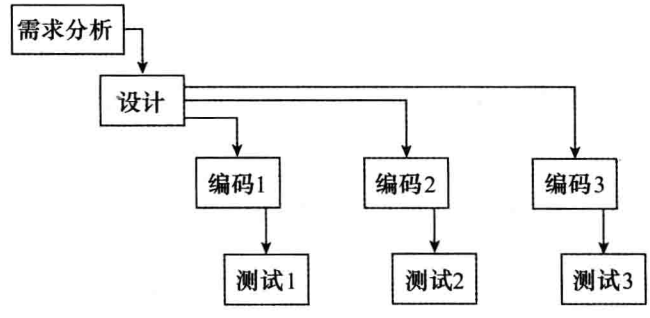

图 1.4 增量模型

增量模型的优点是能在较短时间内，向用户提交可完成部分工作的产品。另一个优点是，逐步增加产品功能，可以使用户有较充裕的时间学习和适应新产品，从而减少一个全新的软件可能给客户组织带来的冲击。

使用增量模型的困难是，在把每个新的增量模型构件集成到现有软件体系结构中时，必须不破坏原来已经开发出的产品。此外，必须把软件体系结构设计得便于按这种方式进行扩充，向现有产品中加入新构件必须简单、方便，也就是说软件体系结构必须是开放的。但是，从长远观点来看，具有开放结构的软件拥有真正的优势，这样的软件可维护性明显好于封闭结构的软件。

1.3.4 螺旋模型

螺旋模型(Spiral Model)因在其指导下的开发呈现为一个螺旋式上升的过程而得名。螺旋模型将瀑布模型和原型模型结合起来，并加入了两种模型忽略的风险分析，弥补了两者的不足。

螺旋模型提供了一种渐增式的开发方法，从建立原型开始，循序渐进地演进至目标系统。初始原型很简单，只体现人们的最初想法，之后不断地以容易实现的较小的增量扩展，在已有的原型上加入新的功能。所以该方法实际上是多次反复原型，随着系统开发人员对系统的认识不断加深，原型不断扩充和完善。

任何软件的开发都存在风险，只是对于不同规模、不同类型的项目，风险的大小不同而已。

通常情况下，项目越大，软件的复杂性越复杂，需要承担的风险也越大。风险是软件开发不可忽视的潜在的不利因素，它对于软件的开发过程和软件质量都有影响。因此，建立开发模型时如果能够对风险进行识别和分析，则有助于降低风险对软件开发的影响。

螺旋模型在构建的同时进行了风险分析，在多次反复原型时，附加进行相应层次上的多次风险分析，以降低某些类型的风险。螺旋模型如图 1.5 所示，图中以笛卡尔坐标为基准，在 4 个象限上分别表示四个方面的活动。这些活动分别为：

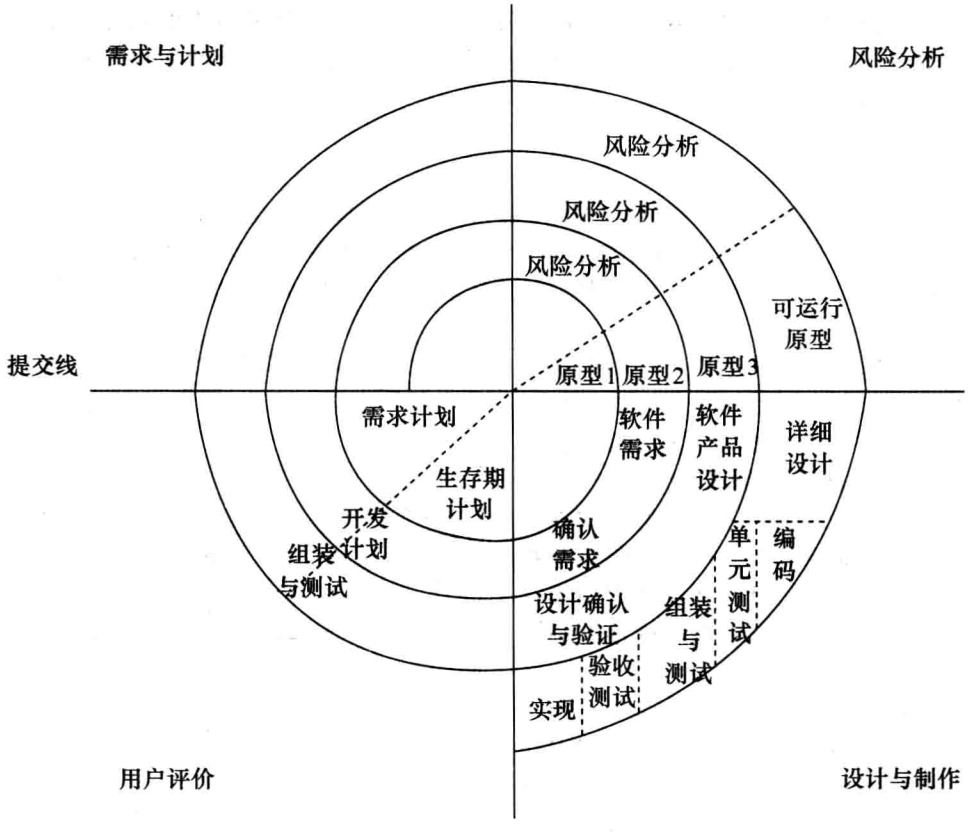

图 1.5　螺旋模型

1. 制订计划：确定软件目标，选择实施方案，设定约束条件；
2. 风险分析：评价方案、识别风险、消除风险；
3. 实施工程：实施软件开发；
4. 客户评估：评价开发工作，提出修正建议。

螺旋线由内向外代表开发进度，每个周期代表一个开发阶段，每转动一圈得到一个较为完整的版本。每个阶段开始时，确定该阶段的目标、选择方案和限制条件以后，转入右上象限，开始评估方案，识别和分析风险。如果风险存在，通常用建造原型的方法排除风险；如果风险过大，开发者和用户无法承受，则终止项目开发的各项活动。如果确定继续开发项目，且成功规避了风险，则进入右下象限，开始下一开发步骤。这一步骤的工作过程相当于纯粹的瀑布模型，最后对该阶段工作进行评价，并计划进入下一阶段工作。

螺旋模型的主要优势在于它是风险驱动的。它采用了软件工程的“演化”的概念，使得开发