普通高等教育
软件工程 “十二五”规划教材

12th Five-Year Plan Textbooks
of Software Engineering

软件工程
——软件建模与文档写作

龙浩 王文乐 刘金 戴莉萍 ◎ 编著

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

普通高等教育
软件工程“十二五”规划教材

12th Five-Year Plan Textbooks
of Software Engineering

软件工程
——软件建模与文档写作

龙浩 王文乐 刘金 戴莉萍 ◎ 编著

人民邮电出版社
北京

图书在版编目（CIP）数据

软件工程 ：软件建模与文档写作 / 龙浩等编著. —
北京 ：人民邮电出版社，2016.8
普通高等教育软件工程“十二五”规划教材
ISBN 978-7-115-43024-3

Ⅰ. ①软… Ⅱ. ①龙… 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(2016)第185739号

内 容 提 要

本书根据现有软件工程教学和项目开发中存在的问题，结合软件工程的最新发展，以及目前软件工程教学的需要，围绕软件工程的三大要素——过程、方法和工具，以软件过程为引领，介绍软件开发工具和方法在不同软件开发阶段的建模和文档撰写。通过案例，以对比的方式，介绍结构化思想和面向对象思想在各个开发阶段中模型的体现，并在其中贯穿介绍了最新的软件工程应用技术。本书内容包括软件开发过程、软件建模工具、项目前期、需求分析、总体设计、详细设计与实现、软件测试、结构化开发案例、面向对象开发案例、综合实验等。在本书最后，介绍了安全设计、设计模式和 UML 语言等内容。

本书强调软件工程的理论与实践相结合，以软件开发过程为引导，介绍软件开发工具的使用和开发方法的应用。全书语言简练、通俗易懂，采用案例教学方法，注重培养软件项目实际建模能力和文档的写作能力，具有很强的实用性和可操作性。书中例题与习题丰富，便于教学和自学。

本书可作为高等院校计算机专业或信息类相关专业本科生软件工程相关课程的教材，也可作为高等职业技术学校信息类专业软件工程教材，也可供软件项目开发人员阅读参考。

◆ 编　　著　龙　浩　王文乐　刘　金　戴莉萍
责任编辑　刘　博
责任印制　沈　蓉　彭志环

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京隆昌伟业印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：21.25　　2016 年 8 月第 1 版
字数：589 千字　　2016 年 8 月北京第 1 次印刷

定价：49.80 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前 言

“软件工程”是高等院校计算机和软件工程专业教学计划中的一门主干课程，具有知识点众多、内容更新快、课程实践性强等特点。在教学中我们发现，如何将软件工程以及其他先导课程中涉及的众多知识点串接起来，方便广大学生理解和在实际软件项目开发中应用，是老师和学生都感到头痛的问题；而在实际软件项目，尤其是大中型软件项目的开发中，发现如何有效地把有关的软件工程思想、技术、方法和工具应用到具体项目中，是困扰多数软件开发企业或开发团队的问题。

高质量的软件产品，必须从管理和技术两方面着手，既要有良好的组织管理措施，也要有合理的技术措施（方法、工具和过程）。这要求开发团队必须强调设计合理恰当的软件开发过程，并在开发过程不同阶段使用合理的方法和工具。这不仅有利于管理者有效地组织和安排各项任务，也能够帮助开发人员理清各个阶段的目的和任务。合理的开发过程设计，一旦应用到软件开发企业的后续项目中，有利于提升软件开发企业的能力，最终提高软件项目的开发质量。

软件项目开发的各阶段必须获得规范合理的技术文档，可视化的模型则是技术文档的最重要内容。软件项目的可视化模型和规范化的软件文档，既有利于开发团队成员加深理解软件开发过程设计的必要性，也让他们更好地理解不同的开发工具、开发过程中各阶段的开发技术所体现出来的开发思想，保证开发团队始终用同一种开发思想来组织整个开发过程，最终达到在规定的时限和成本范围内，为客户提供高质量的软件系统。

为更好地配合“卓越工程师计划”，本书在内容组织上做了精心的安排。采用理论+案例的方法，以对比的方式介绍软件项目开发过程各个阶段的建模和文档编写。一方面，理论知识的介绍，方便初学者了解和掌握软件工程的有关基本概念、原理、方法，把有关的工具、技术、模型和具体的开发阶段活动结合起来，不同开发阶段中结构化思想和面向对象思想的对比应用，加深读者对软件工程的认识；另一方面，理论与实践（实际项目）的紧密结合，贯穿整个开发过程的案例，使得开发人员能够以本书为模板，直接进行实际开发项目的建模和文档撰写。

本教材基于作者多年来从事软件工程课程教学和实际软件项目开发实践的经验，特别是收集了学生的反馈和软件开发团队的需求，借鉴目前软件工程教材的优缺点并考虑到学生的学习特点而编著。本书的主要特色包括以下 3 个方面。

（1）理论与实践相结合

以案例贯穿全教材。考虑到信息管理系统涉及的软件工程知识点最多最全面，开发过程最容易标准化，本书以图书馆管理系统为教学案例。案例源于已实际开发的项目，通过案例介绍软件项目开发过程不同阶段的文档撰写和建模，并将涉及的软件工程概念、方法、技术等有关知识点串接起来。

（2）突出组织逻辑和增加趣味性

目前的国内教材和国外经典教材，普遍侧重于概念原理的介绍，介绍的内容过于理论性，内容安排上过于求全，而对具体项目中相关的软件工程思想如何应用关注过少。本书重点不在于面面俱到地介绍软件工程有关的方法、原理、技术、工具，

而是有针对性地以瀑布过程模型和 RUP 模型为基础，设计了一个新型的开发过程模型，介绍了贯穿于该软件开发过程模型不同阶段的标准化的文档撰写和规范的可视模型构建。本书以直观、易于理解的可视化模型对看起来繁琐、细碎的软件工程知识点进行组织，方便读者的理解和这些知识点在软件开发不同阶段中的实际应用，让读者真正理解软件工程的“过程、方法和工具”三要素。

（3）增加软件产业热门和急需的技术知识

增加“项目前期”一章内容，强调从现实系统出发，进行软件项目的现状分析，并以此作为需求收集和后续需求分析的基础和信息来源，从而彻底解决困扰学生和开发人员“如何进行需求分析”这个大问题；将 MVC 设计思想、三层架构设计、设计模式、组件技术等新知识融合到软件开发过程中，有利于开阔学生的视野并为他们就业做好准备，也有利于其他读者将相关实践与软件工程理论结合。

本书将解决以下问题。

- 真正可行的软件项目开发过程应包含哪些主要环节？各个环节要解决的主要问题是什么？
- 软件开发过程模型、软件能力成熟度模型对于软件项目、软件企业的开发活动标准化有什么意义？
- 结构化方法、目前流行的面向对象方法和组件方法，具有什么特征？
- 软件开发过程的每个阶段应该撰写什么样的文档，建立什么模型？结构化思想和面向对象思想分别是如何在这些模型中体现的？
- 结构化方法下软件开发过程的前后阶段之间模型是如何衔接的？面向对象方法下软件开发过程的前后阶段之间模型是如何衔接的？
- 不同开发阶段的软件文档规范和建模标准是什么？常用的软件开发工具和编码规范有哪些？

本书主要内容包括软件开发过程（第 1 章）、软件开发工具（第 2 章）、项目前期（第 3 章）、软件需求（第 4 章）、概要设计（第 5 章）、详细设计和编码（第 6 章）、软件测试（第 7 章）、毕业论文系统——面向过程方法（第 8 章）、在线毕业论文系统——面向对象方法（第 9 章）。其中：第 8 章（面向过程方法的毕业论文系统）从现实系统的现状分析开始，介绍了需求分析、总体设计、详细设计各个阶段的结构化建模如何应用到系统开发中；第 9 章（面向对象方法的在线毕业论文系统）从现状分析出发，介绍了需求分析、总体设计、详细设计各个阶段的面向对象建模如何应用到实际系统开发中。本书在第 10 章给出了某电器生产公司中有关物料/半成品/成品管理的所有表单，希望学习者按照本书介绍的软件开发过程，采用结构化方法或面向对象方法，进行该生产企业仓库管理系统的建模和文档撰写。本书的三个附录，包括安全设计、设计模式、UML 语言，也能给读者提供很好的项目设计和知识点参考。

本书适合普通本科院校和职业院校软件工程专业学生作为软件工程课程的辅助教材使用，同时也可作为开发人员的参考指南。教师在进行教学内容组织时，“软件工程”课程可以先安排软件工程基本概念和结构化开发这部分内容的教学，在教学时间富余的情况下再进行面向对象开发的教学；“面向对象分析与设计”课程可以直接安排面向对象这部分内容的教学。

本书由龙浩博士主编，参与编写的成员包括王文乐、刘金、戴莉萍。本书是编写组成员对软件工程理论知识和大量软件项目开发实践经验的积累结果。研究生赵瑛承担了部分文字录入校对工作，郝志新、莫耀林、孙胜男、孙鸿杰等同学承担了部分文字录入工作，在此一并表示感谢。因时间仓促，可能存在不妥之处，欢迎指正。联系邮箱：hhlong2010@hotmail.com。

目 录

第1章 软件开发过程

人类社会已经跨入21世纪，计算机技术已经渗入社会生活的各个领域，作为计算机系统重要组成部分的软件已经成为社会经济的重要产业，也是影响社会发展的重要技术领域之一。为了快速有效地开发出用户需要的软件产品，必须以工程化的方法指导软件的开发。软件工程强调软件开发组织必须设计符合项目和开发团队本身实际的开发过程，并通过良好的开发过程对项目活动进行组织管理，并采用合适的方法、技术和工具来保证软件系统的质量。

1.1 软件工程概述

1.1.1 软件工程的发展历程

计算机系统由硬件、软件两部分组成。早期的计算机硬件购置成本高昂，整个系统主要以硬件为主，软件费用只占整个系统总费用很小的比例。随着集成电路制造技术的提升和计算机应用范围的扩大，硬件成本急剧降低，到20世纪90年代的时候，软件费用已经上升到了系统总费用的80%以上。

早期的计算机系统应用面狭窄，主要用于任务相对单一的科学或工程计算。由于要解决的问题相对简单，因此软件往往等同于程序，开发软件也就是编写程序。程序的编写者和使用者往往是同一个（一组）领域专家，主要使用低级语言（机器语言、汇编语言）直接面对机器编写程序代码。由于程序规模小，编写容易，既没有什么系统性的方法和指导思想，对软件开发工作也没有任何管理，是一种“个体化”的手工开发，软件成功与否完全依赖于开发人员的技能和经验。这一时期的软件开发成果除了程序清单之外，基本没有其他文档资料。

从20世纪60年代中期到70年代中期，是计算机系统发展的第二代。在这期间计算机技术有了很大发展。多道程序、多用户概念开始出现并变得普及，操作系统、数据库技术、高级编程语言也陆续出现了。软件应用范围与系统功能的增多促使软件产品数量急剧膨胀，软件也变得更加复杂。“软件作坊”是这段时期的主要软件开发组织方式，由于“软件作坊”基本还是采用早期的个体化开发方法，使得开发出来的软件依然广泛存在难以维护、不可复用的问题，出现了所谓的“软件危机”。

“软件工程”概念的提出（1968年），标志计算机系统发展到第三代。从此，人们考虑用工程化的概念、原理、技术和方法来开发和维护软件，避免软件危机带来的问题。因而“软件工厂”成为软件开发的主要组织方式。在这一时期，结构化的工程方法获得了广泛应用，并已成为了一种成熟的软件工程方法学；而自20世纪90年代起，面向对象的工程方法，也已被应用于软件开

发之中。此外，许多软件企业采用工程化的原理、技术和工具实施软件产品开发，以适应软件产业化发展的需要。软件也由单纯的程序发展成为了包括程序、数据、文档等诸多要素集合的软件产品。图 1-1 描述了软件构成。

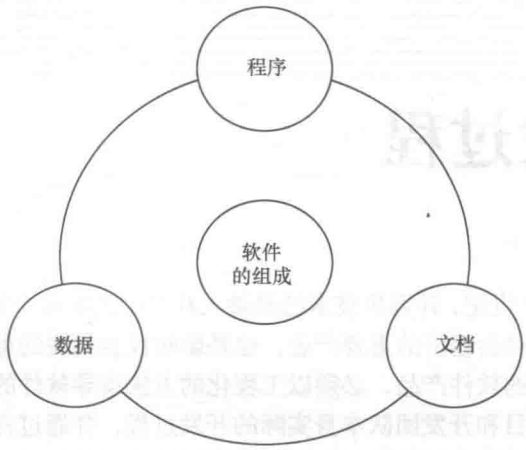

图 1-1　软件构成

1.1.2　软件的特征和分类

软件是人类思维的产物，它极大地扩充了计算机硬件的功能，是计算机系统中的重要逻辑成分。因此，软件具有与硬件完全不同的特征。

（1）软件的开发运行必须依赖于特定的计算机系统环境，比如硬件、网络配置和支撑软件等；

（2）软件是由开发或工程化而形成的，而不是传统意义上的制造产生的。具有复杂性、不可见性和易变性，难以处理；

（3）软件复制非常简单，软件不会“磨损”；

（4）大多数软件是定制的，绝大部分的软件都是新的，而且是不断变换的。

在计算机系统中往往需要许多不同的软件协同工作，可以从多个不同的角度来划分软件的类别。按功能可将软件划分为系统软件、支撑软件、应用软件；按工作方式将软件划分为实时处理软件、分时处理软件、交互式软件、批处理软件；按规模将软件划分为微型软件、小型软件、中型软件、大型软件；按服务对象将软件划分为通用软件、定制软件；按照软件是否分布式布置分为单机软件、网络软件。下面是一些软件类别的概要描述。

系统软件：系统软件是一种为其他程序服务的程序。其中的一些系统软件（如操作系统、驱动程序和通信程序等）实现对计算机系统资源的管理，是其他软件工作的基础，而其他的系统应用（如编译器、编辑器和文件管理程序）则用以支持开发人员、应用人员的基础性工作。它们均具有以下特点：与计算机硬件频繁交互，需要精细调度、资源共享及灵活的进程管理的并发操作，复杂的数据结构，多外部接口。

实时软件：管理、分析、控制现实世界中发生的事件的程序称为实时软件。实时软件的组成通常包括：一个数据收集部件，负责从外部环境获取和格式化信息；一个分析部件，负责将信息转换成应用时所需要的形式；一个控制/输出部件，负责响应外部环境；一个负责协调其他各部件的管理部件。实时系统有较严格的时间响应要求（一般从 1 毫秒到 1 分钟）否则可能带来灾难性的后果。

商业软件：商业信息处理是最大的软件应用领域。具体的“系统”（如工资表、帐目支付和接

收、存货清单等）均可归为管理信息系统（MIS）软件，它们可以访问一个或多个包含商业信息的大型数据库。该领域的应用将已有的现实手工数据重新构造，变换成一种能够辅助商业操作或管理决策的形式。辅助商业操作的通常称为交互式事务性操作软件（也称 OLTP，Online Transaction Processing，在线事务处理系统），管理决策的通常称为支持系统（也称为 OLAP，Online Analyze Processing，在线分析处理系统）。

工程和科学计算软件：工程和科学计算软件的特征是能够方便地进行“数值计算或分析”。此类应用涵盖面很广，但目前工程和科学计算软件已不仅限于传统的数值算法。计算机辅助设计、系统仿真和其他交互应用已经开始具有实时软件和系统软件的特征。

嵌入式软件：目前的智能化产品非常常见，嵌入式软件驻留在只读内存中，用于控制这些智能产品。嵌入式软件往往执行很有限但专职的功能（如微波炉的按钮控制），或是提供比较强大的功能及控制能力（如汽车中的数字控制，包括燃料控制、仪表板显示，刹车系统等）。

个人计算机软件：安装在计算机上以支持个人应用的软件，包括字处理、电子表格、计算机图形、多媒体、娱乐、数据库管理、个人及商业金融应用、外部网络或数据库访问等。

人工智能软件：人工智能（AI）软件利用非数值算法去解决复杂的问题，这些问题不能通过计算或直接分析得到答案。如基于知识的专家系统、模式识别（图像或声音）软件、定理证明程序和游戏软件。

1.1.3　软件危机

软件危机就是人们在开发和维护软件时遇到一系列的问题，如不能按时完成开发任务、开发经费超支、软件质量无法保证、开发工作效率低下等。具体体现在以下方面。

（1）软件开发进度难以预测，软件开发成本难以控制

软件产品不能在预算范围内、按照计划完成，拖延工期几个月甚至几年的现象并不罕见；投资一再追加，最终实际成本往往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，降低了软件开发组织的信誉，从而不可避免地会引起用户的不满。

（2）用户对产品功能难以满足

开发人员和用户之间很难充分有效地沟通，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。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。

（3）软件产品质量无法保证

开发团队缺少完善的软件质量评价体系和科学的软件测试规程，最终的软件产品存在很多缺陷和错误，而它们往往是造成重大事故的隐患。

（4）软件产品难以维护

软件产品本质上是开发人员的代码化的逻辑思维活动，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。

（5）软件缺少适当的文档资料

文档资料是软件必不可少的重要组成部分。实际上，软件的文档资料是开发组织和用户之间权利和义务的合同书，是系统管理者、总体设计者向开发人员下达的任务书，是系统维护人员的技术指导手册，是用户的操作说明书。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。

软件危机的出现和日益严重的趋势暴露了软件产业在早期发展过程中存在的各种问题，本质上这是由于人们对软件产品的认识不足以及对软件开发的内在规律理解的偏差造成的。概括起来说，软件危机的原因有以下 8 点。

① 从事软件开发的人员对这个产业认识不充分、缺乏经验；
② 缺乏统一的、标准化的开发过程设计，缺乏规范化的方法论进行指导；
③ 忽视软件开发前期的需求分析；
④ 文档资料不齐全、不准确；
⑤ 忽视测试的重要性；
⑥ 没有完善的质量保证体系；
⑦ 开发团队内部交流不顺畅、不充分；
⑧ 不重视维护，或由于以上原因造成维护工作的困难。

1.1.4 软件工程概念和基本原则

软件工程是人们为了应对软件危机，以借鉴传统工程的原则、方法，以提高质量、降低成本、控制工期为目的地指导计算机软件的开发和维护。软件工程有两方面的含义：一方面，软件工程是指导计算机软件开发和维护工程的学科；另一方面，它是人们把经过时间考验而证明正确有效的管理技术和当前能够得到的最好的技术方法结合起来，经济地开发出高质量的软件并有效地维护它。

1993 年 IEEE 给出了软件工程的全面定义。即软件工程是：①把系统化的、规范的、可度量的途径应用于软件开发、运行和维护的过程，也就是把工程化方法应用于软件中；②研究系统化的、规模化的、可度量的途径。

以工程化的形式组织软件的开发和维护，应能够保证达到以下目标。

① 开发成本控制在预计的合理范围之内；
② 开发周期能够控制在预计的合理时间范围之内；
③ 软件的功能和性能能够满足用户需求；
④ 软件具有较高的质量；
⑤ 软件具有较高的可靠性；
⑥ 软件产品易于移植、维护、升级和使用。

简而言之，就是按时、按质开发用户需要的软件产品。那么到底如何进行软件的开发，才可视为真正的工程化软件开发呢？自从软件工程提出以来，很多该领域的专家学者提出很多关于软件工程的准则或“信条”。著名软件工程专家 Barry W.Beohm 综合这些专家学者的意见并总结 TRW 公司多年开发软件的经验，提出软件工程的七条基本原则。

（1）用分阶段的生命周期计划严格管理。这条基本原理要求把软件生命周期划分为若干个阶段，并制订相应的切实可行的计划，然后严格按照计划对软件的开发和维护工作进行管理。

（2）坚持进行阶段评审。在每个阶段都进行严格的评审，以尽早发现软件开发过程中所犯的错误。

（3）实行严格的产品控制。不能随意对软件进行修改，必要的修改必须按照严格的规程进行评审，获得批准后才能实施修改。

（4）采用现代程序设计技术。采用先进的开发技术来提高开发和维护效率、降低开发中可能出现的错误，提高软件产品质量。

（5）结果应能够清楚地审查。根据软件开发项目的总目标和完成期限，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）开发小组的人员应小而精。小的开发小组，可以降低交流成本，精练的开发人员可以极大地提高开发效率，并显著地降低错误。

（7）承认不断改进软件工程实践的必要性。积极主动地采用新的软件技术，注意不断总结经验，对于促进软件产品的质量也有莫大的效果。

这七条软件工程的基本原理是互相独立的，彼此不能替代，它们共同确保软件产品质量和开发效率，是缺一不可的、完备的最小集合。

1.2　软件生命周期

同任何事物一样，一个软件产品或软件系统也要经历一个包含孕育、诞生、成长、成熟、衰亡等阶段的生存过程，称为软件生命周期。通常把整个软件生存周期划分为若干阶段，使得每个阶段有明确的任务，使规模大、结构复杂和管理复杂的软件开发变得容易控制和管理。概括地说，软件生命周期包含软件定义、软件开发、软件运行维护三个时期，并可以进一步细分为可行性研究、项目计划、需求分析、概要设计、详细设计、编码实现与单元测试、系统集成测试、系统确认验证、系统运行与维护等几个阶段。这是软件生命周期的基本构架，在实际软件项目开发中，应该根据所开发软件的规模、种类，软件开发机构的习惯做法，以及采用的技术方法等，对各阶段进行必要的合并、分解或补充。

1.2.1　软件定义期

软件定义是软件项目的早期阶段，主要由软件系统分析人员和用户合作，针对有待开发的软件系统进行分析、规划和规格描述，确定软件是什么，为今后的软件开发做准备。这个时期往往需要分阶段地进行以下 4 项工作。

（1）软件任务立项

软件项目往往开始于任务立项，并需要针对项目的名称、性质、目标、意义和规模等做出回答，以此获得对准备着手开发的软件系统的最高层描述。

（2）项目可行性分析

在软件任务立项报告被批准以后，接着需要进行项目可行性分析。可行性分析是针对准备进行的软件项目进行的可行性风险评估。因此，需要对准备开发的软件系统提出高层模型，并根据高层模型的特征，从技术可行性、经济可行性和操作可行性这三个方面，对项目做出是否值得往下进行的回答，由此决定项目是否继续进行下去。

（3）制定项目计划

在确定项目可以进行以后，接着需要针对项目的开展，从人员、组织、进度、资金、设备等多个方面进行合理的规划，并制定项目开发计划。

（4）软件需求分析

软件需求分析是软件规格描述的具体化与细节化，是软件定义时期需要达到的目标。需求分析要求以用户需求为基本依据，从功能、性能、数据、操作等多个方面，对软件系统给出完整、准确、具体的描述，用于确定软件规格。

在软件项目进行过程中，需求分析是从软件定义到软件开发的最关键步骤，其结论不仅是今后软件开发的基本依据，同时也是今后用户对软件产品进行验收的基本依据。

1.2.2　软件开发期

在对软件规格完成定义以后，接着可以在此基础上对软件实施开发，并由此制作出软件产品。

这个时期需要分阶段地完成以下 5 项工作。

（1）软件概要设计

概要设计（也称总体设计）是针对软件系统的结构设计，用于从总体上对软件给出设计说明。软件开发团队有开发人员、高层管理者、安装配置人员、运行维护人员、软件系统实际操作者（用户），不同的人员对于软件构成有不同的观察角度，所关心的软件系统构成元素有所不同。开发人员关心软件系统的构造、接口、全局数据结构和数据环境等，高层管理者关心系统的构造，安装配置人员、运行维护人员关心硬件系统和相关软件配置，软件实际操作者关心功能模块结构。概要设计的结果将成为详细设计与系统集成的基本依据。

对开发人员而言，结构化方法下，模块是概要设计时构造软件的基本元素，因此，概要设计中软件也就主要体现在模块的构成与模块接口这两个方面上。面向对象方法下，对象是概要设计时构建软件的基本元素，因此概要设计时软件主要体现在对象的原型——类设计上。概要设计时并不需要说明模块/类方法的内部细节，但是需要进行全部的有关它们构造的定义，包括功能特征、数据特征和接口等。对于模块，需要给出模块名、输入输出参数和要实现的功能描述；对于类，需要给出类名、属性名和数据类型、方法名称、方法的输入输出参数和方法的功能描述。

在进行概要设计时，模块/类的独立性是一个有关质量的重要技术性指标，可以使用模块的内聚、耦合这两个定性参数对模块独立性进行度量。

（2）软件详细设计

设计工作的第二步是详细设计，它以概要设计为依据，用于确定软件结构中每个模块的内部细节，为编写程序提供最直接的依据。详细设计需要从实现每个模块功能的程序算法和模块内部的局部数据结构等细节内容上给出设计说明。

（3）编码和单元测试

编码是对软件的实现，一般由程序员完成，并以获得源程序基本模块为目标。编码必须按照“详细设计说明书”的要求逐个模块地实现。在基于软件工程的软件开发过程中，编码往往只是一项语言转译工作，即把详细设计中的算法描述语言转译成某种适当的高级程序设计语言或汇编语言。

为了方便程序调试，针对基本模块的单元测试也往往和编码结合在一起进行。单元测试也以详细设计结果为依据，用于检验每个基本模块在功能、算法与数据结构上是否符合设计要求。

（4）系统集成测试

所谓系统集成也就是根据概要设计中的软件结构，把经过测试的模块，按照某种选定的集成策略，例如渐增集成策略，将系统组装起来。在组装过程中，需要对整个系统进行集成测试，以确保系统在技术上符合设计要求，在应用上满足需求规格要求。

（5）系统确认验证

在完成对系统的集成之后，接着还要对系统进行确认验证。系统确认验证需要以用户为主体，以需求规格说明书中对软件的定义为依据，由此对软件的各项规格进行逐项的确认，以确保已经完成的软件系统与需求规格的一致性。为了方便用户在系统确认期间能够积极参与，也为了系统在以后的运行过程中能够被用户正确使用，这个时期往往还需要以一定的方式对用户进行必要的培训。

在完成对软件的验收之后，软件系统可以交付用户使用，并对项目进行总结。

1.2.3 软件运行与维护期

软件系统的运行是一个比较长久的过程，跟软件开发机构有关的主要任务是对系统进行经常性的有效维护。软件的维护过程，也就是修正软件错误，完善软件功能，由此使软件不断进化升级的过程，以使系统更加持久地满足用户的需要。因此，对软件的维护也可以看成为对软件的再一次开发。在这

个时期，对软件的维护主要涉及三个方面的任务，即改正性维护、适应性维护和完善性维护。

1.3　软件开发过程模型

随着软件的规模和复杂性不断增大，以开发人员的经验和技术来保证软件产品质量，单纯对结果进行检验以评估软件系统质量已经成为不可能的任务。更多情况下，必须将质量保证的观点贯穿于整个软件开发过程。这要求软件开发必须从管理和技术两方面着手，既要有良好的技术措施（方法、工具和过程），又要有必要的组织管理措施。从技术角度来说，过程设计是影响软件产品质量的决定性因素，方法和工具只有在合理设计的开发过程中，才能发挥最大功效。软件过程模型是人们在软件开发实践中总结出来的，适用于具有某一类特征项目的标准开发过程。软件开发模型提供了一个框架并把必要活动映射在这个框架中，包括主要的开发阶段、各个阶段要完成的主要任务和活动、各个阶段的输入输出。

常见的软件开发过程模型很多，包括瀑布模型、演化模型（包括原型模型、增量模型和螺旋模型）、喷泉模型、RUP 过程等。在实践中，软件项目开发团队必须依据拟开发项目的特点以及对用户需求的把握程度，选择某一开发过程模型做一定的剪裁，设计出适合具体项目的软件开发过程。

1.3.1　瀑布模型

瀑布模型（也称线性顺序模型）诞生于 20 世纪 70 年代，是最早出现并获得最广泛应用的软件过程模型。瀑布模型中的“瀑布”意味着过程中的开发活动是严格线形的，就像山顶倾泻下来的水，逐级下落。瀑布模型的过程如图 1-2 所示。

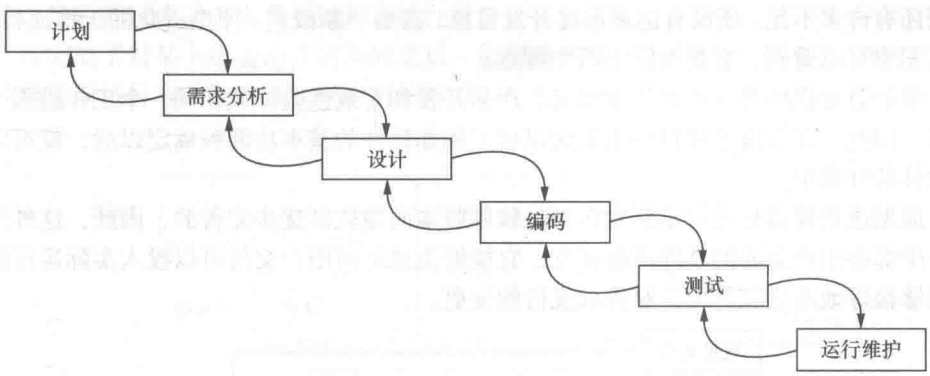

图 1-2　瀑布模型

瀑布模型是一种基于里程碑的、阶段性的过程模型，它所提供的是里程碑式的软件开发工作流程，文档是瀑布模型中每个阶段的成果体现，模型的回溯性很差。因此，瀑布模型要求项目严格按规程推进，瀑布模型从上至下按顺序进行的几个阶段有固定的衔接次序，瀑布模型中的阶段只能逐级到达，不能跨越。每个阶段都有明确的任务，都需要产生确定的成果。并且前一阶段输出的成果被作为后一阶段的输入条件，在某个阶段的工作任务已经完成，并准备进入到下一个阶段之前，需要针对这个阶段的文档进行严格的评审，直到确认以后才能启动下一阶段的工作。

瀑布模型必须等到所有开发工作全部做完以后才能获得可以交付的软件产品，它适用于具有以下特征的项目。

① 需求稳定、变化很小且开发人员能够一次性获取全部需求的项目；

② 软件开发人员具有丰富经验，对于应用领域非常熟悉；

③ 软件项目本身的风险很低。

例如，瀑布模型可以用于编译系统、操作系统、财务会计系统、企事业单位的事务管理系统等项目的开发。瀑布模型的优点在于阶段性强，易于对项目进行管理；缺点在于灵活性差，并不支持对软件系统进行快速创建，对于一些急于交付的软件系统的开发，瀑布模型有操作上的不便。

1.3.2 原型模型

1. 快速原型方法

快速原型方法是原型模型在软件分析、设计阶段的应用，用来解决用户对软件系统在需求上的模糊认识，或用来试探某种设计是否能够获得预期结果。快速原型方法具有以下特点。

（1）快速原型用来获取用户需求，或是用来试探设计是否有效。一旦需求或设计确定，原型就将被抛弃。因此，快速原型要求快速构建、容易修改，以节约原型创建成本、加快开发速度。快速原型往往采用一些快速生成工具创建，例如 4GL 语言。Visual Basic、Delphi 等基于组件的可视化开发工具是非常有效的快速原型创建工具，也被应用于原型创建和进化。

（2）快速原型是暂时使用的，因此并不要求完整。它往往针对某个局部问题建立专门原型，如界面原型、工作流原型、查询原型等。

（3）快速原型不能贯穿软件的整个生命周期，它需要和其他的过程模型相结合才能产生作用。例如，在瀑布模型中应用快速原型，以解决瀑布模型在需求分析时期存在的不足。

2. 原型进化模型

原型进化对开发过程的考虑是，针对有待开发的软件系统，先开发一个原型系统给用户使用，然后根据用户使用情况的意见反馈，对原型系统不断修改，使它逐步接近并最终到达开发目标。原型进化所要创建的原型则是一个今后将要投入应用的系统，只是所创建的原型系统在功能、性能等方面还有许多不足，还没有达到最终开发目标，需要不断改进。原型进化的工作流程如图 1-3 所示。从图中可以看到，它具有以下两个特点。

（1）原型进化模型将软件的需求定义、产品开发和有效性验证放在同一个工作进程中交替或并行运作。因此，在获得了软件需求框架以后，例如软件的基本功能被确定以后，就可以直接进入到对软件的开发中。

（2）原型进化模型是通过不断发布新的软件版本而使软件逐步完善的，因此，这种开发模式特别适合于那些用户急需的软件产品开发。它能够快速地向用户交付可以投入实际运行的软件成果，并能够很好地适应软件用户对需求规格的变更。

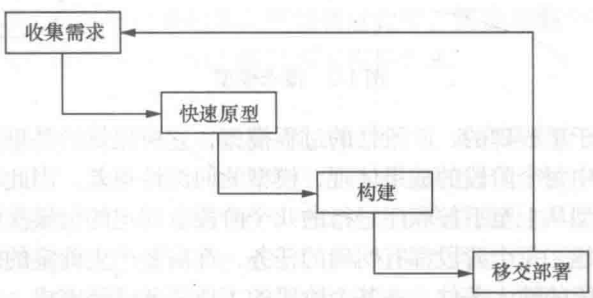

图 1-3 快速原型模型

原型模型适用于具有以下特征的软件项目开发。

① 对现有软件产品进行升级或功能完善；

② 开发人员和用户交流困难，需求获取困难；

③ 开发人员对技术熟悉或把握性不大；

④ 具有支持快速开发的工具。

原型进化模型的优点在于灵活性好，简单快速，能够适应软件需求的中途变更；缺点在于需要额外的花费来构造原型，且缺乏有效的管理规程，软件版本的快速变更还可能损伤软件的内部结构，使其缺乏整体性和稳定性。

1.3.3　增量模型

瀑布模型较难适应用户的需求变更，开发速度慢。但是，瀑布模型提供了一套工程化的里程碑管理模式，能够有效保证软件质量，并使得软件容易维护。相反地，原型进化模型则可以使对软件需求的详细定义延迟到软件实现时进行，并能够使软件开发进程加速。但是，原型进化模型不便于工业化流程管理，也不利于软件结构的优化，并可能使得软件难以理解和维护。基于以上因素的考虑，增量模型对这两种模型的优点进行了结合。

增量模型在整体上按照瀑布模型的流程实施项目开发，以方便对项目的管理；但在软件的实际创建中，则将软件系统按功能分解为许多增量构件，并以构件为单位逐个地创建与交付，直到全部增量构件创建完毕，并都被集成到系统之中交付用户使用。

如同原型进化模型一样，增量模型逐步地向用户交付软件产品，但不同于原型进化模型的是，增量模型在开发过程中所交付的不是完整的新版软件，而只是新增加的构件。

图 1-4 所示是增量模型的工作流程，它被分成以下 3 个阶段。

（1）在系统开发的前期阶段，为了确保系统具有优良的结构，仍需要针对整个系统进行需求分析和概要设计，需要确定系统的基于增量构件的需求框架，并以需求框架中构件的组成及关系为依据，完成对软件系统的体系结构设计。

（2）在完成软件体系结构设计之后，可以进行增量构件的开发。这个时候，需要对构件进行需求细化，然后进行设计、编码测试和有效性验证。

（3）在完成了对某个增量构件的开发之后，需要将该构件集成到系统中去，并对已经发生了改变的系统重新进行有效性验证，然后再继续下一个增量构件的开发。

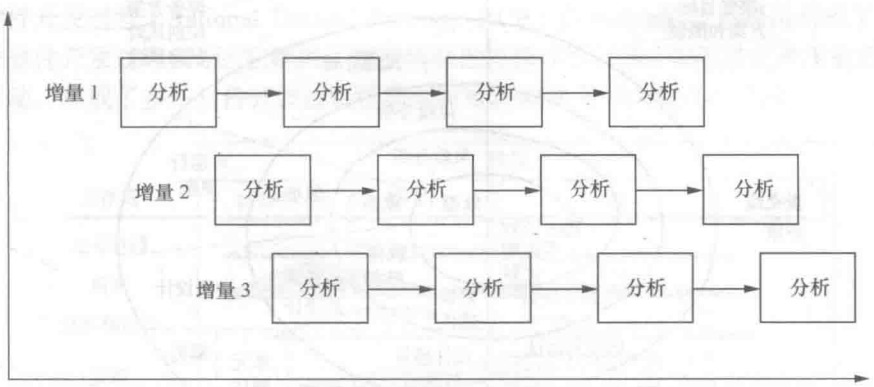

图 1-4　增量模型

增量模型具有以下特点。

（1）开发初期的需求定义可以是大概的描述，只是用来确定软件的基本结构，而对于需求的细节性描述，则可以延迟到增量构件开发时进行，以增量构件为单位逐个地进行需求补充。

（2）可以灵活安排增量构件的开发顺序，并逐个实现和交付使用。这不仅有利于用户尽早地用上系统，而且用户在以增量方式使用系统的过程中，还能够获得对软件系统后续构件的需求经验。

（3）软件系统是逐渐扩展的，因此，开发者可以通过对诸多构件的开发，逐步积累开发经验，从总体上降低软件项目的技术风险，还有利于技术复用。