普通高等教育“十二五”规划教材

# UML系统建模及系统分析与设计


王欣　张毅　编著

中国水利水电出版社
www.waterpub.com.cn

普通高等教育“十二五”规划教材

# UML 系统建模及系统分析与设计

王欣　张毅　编著



中国水利水电出版社
www.waterpub.com.cn

## 内 容 提 要

本书结合统一建模语言 UML2.0 和软件开发项目案例，重点阐述以面向对象系统分析和设计方法为主线的软件工程领域关键技术，并介绍了模型驱动开发、基于复用的开发等最新技术。

全书共分 9 章，第 1 章为软件开发方法，第 2 章为 UML 建模语言，第 3 章至第 6 章为面向对象分析、面向对象设计与实现，第 7 章为软件复用与软件架构技术，第 8 章为 Rose 使用，第 9 章为案例。通过一个贯穿全书的案例对面向对象的软件开发过程和用例图、类图、交互图、活动图、状态机图以及构件图与部署图的绘制方法与步骤进行了具体的讲解，最后结合应用实例对软件开发与 UML 建模进行详细阐述，使学生掌握软件开发方法和 UML 建模技术及其应用。

本书理论与实际相结合、实用性与可读性相结合。可作为高等院校工科和管理类相关专业的教材或教学参考书，也可供有一定实际经验的软件工作人员和需要开发应用软件的广大计算机用户阅读使用。

**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/或 http://www.wsbookshow.com。**

**图书在版编目（CIP）数据**

UML系统建模及系统分析与设计 / 王欣，张毅编著
. -- 北京 : 中国水利水电出版社，2013.9
普通高等教育“十二五”规划教材
ISBN 978-7-5170-1097-5

Ⅰ. ①U… Ⅱ. ①王… ②张… Ⅲ. ①面向对象语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2013)第172943号

**策划编辑：石永峰　　责任编辑：李　炎　　封面设计：李　佳**

| | |
|---|---|
| 书　　名 | 普通高等教育“十二五”规划教材<br>**UML 系统建模及系统分析与设计** |
| 作　　者 | 王欣　张毅　编著 |
| 出版发行 | 中国水利水电出版社<br>（北京市海淀区玉渊潭南路 1 号 D 座　100038）<br>网址：www.waterpub.com.cn<br>E-mail：mchannel@263.net（万水）<br>sales@waterpub.com.cn<br>电话：（010）68367658（发行部）、82562819（万水） |
| 经　　售 | 北京科水图书销售中心（零售）<br>电话：（010）88383994、63202643、68545874<br>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 | 三河市铭浩彩色印装有限公司 |
| 规　　格 | 184mm×260mm　16 开本　20 印张　504 千字 |
| 版　　次 | 2013 年 9 月第 1 版　2013 年 9 月第 1 次印刷 |
| 印　　数 | 0001—3000 册 |
| 定　　价 | 36.00 元 |

# 前　言

笔者多年来一直为本科生和研究生讲授管理信息系统和软件工程课程，在讲授的过程中，对软件工程的基本理论进行了较为深入的研究。在实际开发过程中，深感明确一些基本概念，树立系统工程的开发思想是很重要的。

随着 UML 的广泛使用，发现使用 UML2.0 进行讲述的教材不多，而且目前出版的教材欠缺系统性，缺少案例，因此，笔者萌生了编著《UML 系统建模及系统分析与设计》的念头。从目前的系统开发方法发展来看，比较著名的有结构化方法、原型化方法和面向对象方法。笔者在本教材中对结构化开发方法进行了详细的阐述。本书以软件工程和面向对象技术的基本理论框架为基础，全面系统地讲述了软件工程的概念、原理，典型的软件开发方法学以及系统体系架构和软件复用理论，重点讲述基于 UML 的面向对象开发。对于创建对象系统来说，面向对象语言和 UML 是必要的，但重要的是理解“对象的思想”，对象的思想是本书的重点和难点。本书重点介绍在国内外广泛流行的面向对象方法及 UML 语言。笔者总结多年的教学与实践经验，认为：只讲 UML 语言不行，重要的是要清楚面向对象的思想。

本书的特色是：围绕案例逐步展开教学，在一些主要的章节介绍理论后，通过案例将知识点串联起来，使读者能够做到融会贯通；通过一个大型的综合案例，将本书所讲的主要内容应用于实践，提高读者的应用能力；每章开头都列出了本章的目的，便于学生掌握本章的重点、难点。本书附有制作精良的配套教学课件，读者可以免费到中国水利水电出版社和万水书苑网站上下载。

本书共分 9 章。第 1 章至第 5 章由王欣编写，第 6 章至第 9 章由张毅编写，全书由王欣统稿。本书在校稿过程中得到了吴言杰、刘泓利、刘宇航和李萍萍的大力帮助，并获得东北电力大学“优质教材”编写资助，同时本书的完成也离不开石永峰先生的热情鼓励，在此致以最诚挚的谢意！最后向中国水利水电出版社的广大员工致以深深地感谢！感谢他们对本书的大力支持！

本书参考了许多同行的著作，书后只是列出了部分参考文献。在此一并表示感谢！

由于本人水平有限，再加上编写时间仓促，书中一定有不妥之处，敬请读者批评斧正。

作　者

2013 年 6 月

# 目　　录

# 第 1 章　面向对象软件开发方法

20 世纪 60 年代中期爆发了众所周知的软件危机。为了克服这一危机，在 1968 年召开的著名的 NATO 会议上提出了软件工程这一术语，并在以后不断发展、完善。与此同时，软件研究人员也在不断探索新的软件开发方法，至今已形成了众多的软件开发方法。本章主要介绍软件、软件工程、软件过程和软件开发方法，重点介绍面向对象开发方法。

**本章目的：**

- 了解软件的发展和软件工程的概念。
- 了解软件开发的常用方法。
- 重点掌握面向对象技术的基本概念和开发过程。
- 了解几种典型的面向对象开发方法。
- 了解可行性研究方法。
- 掌握可行性分析报告的书写格式。

## 1.1　软件发展与软件工程

软件是一种特别的产品，随着其规模及应用领域的扩大和复杂性的进步，逐渐形成了工程。现在，被普遍接受的软件的定义是：软件（software）是计算机系统中与硬件（hardware）相互依存的另一部分，它包括程序（program）、相关数据（data）及其说明文档（document）。其中程序是按照事先设计的功能和性能要求执行的指令序列；数据是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各种图文资料。软件工程（Software Engineering，SE）是针对软件这一具有特殊性质的产品的工程化方法。软件工程涵盖了软件生存周期的所有阶段，并提供了一整套工程化的方法来指导软件人员的开发工作。

### 1.1.1　软件的发展与特征

1. 软件的发展阶段

第一个写软件的人是 Ada（Augusta Ada Lovelace），在 19 世纪 60 年代他尝试为 Babbage（Charles Babbage）的机械式计算机写软件。尽管他们的努力失败了，但他们的名字永远载入了计算机发展的史册。软件发展的历史可以大致分为如下的四个阶段：

第一个阶段（20 世纪 50 年代到 60 年代）是程序设计阶段，基本是个体手工劳动的生产方式。20 世纪 50 年代，软件伴随着第一台电子计算机的问世诞生了。以写软件为职业的人也开始出现，他们多是经过训练的数学家和电子工程师。20 世纪 60 年代美国大学开始有计算机专业，专门教人们写软件。在计算机系统发展的初期，硬件通常用来执行一个单一的程序，而这个程序，又是为一个特定的目的而编制的。大多数软件是由使用该软件的个人或机构研制的，软件往往带有强烈的个人色彩。早期的软件开发也没有什么系统的方法可以遵循，软件设计是在某个人的头脑中完成的一个隐藏的过程。而且，除了源代码往往没有软件说明书等文档。因

此，严格来说这个时期尚无软件的概念，基本上只有程序和程序设计概念，不重视程序设计方法。软件主要是用于科学计算，规模很小，采用简单的工具（基本上采用低级语言），硬件的存储容量小，运行可靠性差。20 世纪中期，计算机从军用领域转向民用领域应用，那时编写程序的工作被视为艺术家的创作。当时的计算机硬件非常昂贵，编程人员寻求的是如何在有限的处理器能力和存储器空间束缚下，编写出运行速度快、体积小的程序。在程序中充满了各种各样让人困惑的技巧。这时的软件生产非常依附于开发人员的聪慧才智。该阶段的主要特征是：

（1）程序设计只是一个隐含在开发者头脑中的过程，程序设计的结果，除了程序流程图和源程序清单可以留下来之外没有任何其他形式的文档资料保留下来。

（2）此时只有程序的概念，没有软件的概念。因此，这个时期软件开发就是指程序设计，其生产方式为个体手工方式，而且程序设计很少考虑通用性。

（3）主要采用汇编语言，甚至是机器语言，以解决计算机内存容量不够和运算速度太低的矛盾。由于过分追求编程技巧，程序设计被视为某个人的神秘技巧，程序除作者本人外，其他人很难读懂。

第二阶段（20 世纪 60 年代到 70 年代）是软件设计阶段，采取小组合作生产方式。这一时期计算机的应用领域得到进一步扩大，对软件系统的需求和软件自身的复杂度急剧上升，传统的开发方法无法适应用户在质量、效率等方面对软件的需求。此时，人们为摆脱汇编语言和机器语言编程的困难，相继研制出了一批高级程序设计语言（如 ALGOL、FORTRAN、BASIC、PASCAL、COBOL 等），这些高级程序设计语言的出现，大大加速了计算机应用普及的步伐，各种类型的应用程序相继出现。在这一时期软件开始作为一种产品被广泛使用，出现了“软件作坊”。“软件作坊”专门为有需求的人写软件。这时软件开发的方法基本上仍然沿用早期的个体化软件开发方式，但软件的数量急剧膨胀，软件需求日趋复杂，维护的难度越来越大。一些商业计算机公司为了扩大系统的功能，方便用户使用，合理调度计算机资源，提高系统运行效率，也投入了大量人力、物力从事系统软件和支撑软件的开发研究。但是，这个阶段的开发成本令人吃惊得高，而失败的软件开发项目却屡见不鲜。最为突出的例子是美国 IBM 公司于 1963 年～1966 年开发的 IBM360 系列机的操作系统。该项目的负责人 Fred Brooks 在总结该项目时无比沉痛地说：“……正像一只逃亡的野兽落到泥潭中做垂死挣扎，越是挣扎，陷得越深，最后无法逃脱灭顶的灾难，……程序设计工作正像这样一个泥潭……一批批程序员被迫在泥潭中拼命挣扎，……，谁也没有料到问题竟会陷入这样的困境……。”IBM360 操作系统的历史教训已成为软件开发项目中的典型事例被记入历史史册。“软件危机”就这样开始了。“软件危机”使得人们开始对软件及其特性进行更深一步的研究，人们改变了早期对软件的不正确看法。早期那些被认为是优秀的程序常常很难被别人看懂，通篇充满了程序技巧。现在人们普遍认为优秀的程序除了功能正确、性能优良之外，还应该容易看懂、容易使用、容易修改和扩充。为解决这个问题，1968 年秋季 NATO（北大西洋公约组织）的科技委员会召集了近 50 名一流的编程人员、计算机科学家和工业界巨头讨论并制定摆脱“软件危机”的对策。在联邦德国召开的这次国际学术会议上第一次提出了“软件危机”（software crisis）的概念。软件危机指的是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。概括来说，软件危机包含两方面问题：一是如何开发软件，以满足日益增长、日趋复杂的需求；二是如何维护数量不断膨胀的软件产品。落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重的问题。在这次会议上提出把软件开发从“艺术”和“个体行动”向“工程”和“群体协同工作”转化。其基础思想是利用计算机科学理论和技巧以及工程管理原

则和方法，按照预算和进度，实现满足用户请求的软件产品的定义、开发、发布和维护的工程。这次会议第一次提出了软件工程（software engineering）这个概念，从此出现了一门新的学科——软件工程。该阶段的主要特征是：

（1）由于程序的规模增大，程序设计已不可能由个人独立完成，而需要多人分工协作。软件的开发方式由“个体生产”发展到“小组软件作坊”。

（2）程序的运行、维护也不再由一个人来承担，而是由开发小组承担。

（3）程序已不再是计算机硬件的附属成分，而是计算机系统中与硬件相互依存、共同发挥作用的不可缺少的部分。在计算机系统的开发过程中，起主导作用的已经不仅仅是硬件工程师，同时也包括软件工程师。

这个时期的软件已经达到中小型规模，逻辑关系复杂，软件开发与维护难度很大。当软件投入运行时，需要纠正开发时期潜入的错误；补充开发用户新的需求；需要根据运行环境的改变对软件做出适应性调整。由于“小组软件作坊”本身的个体化开发特征，缺乏良好的小组管理水平，使得许多软件产品不可维护，最终导致“软件危机”。高级程序设计语言的使用，使得该时期结构化程序设计成为主要的开发技术和手段。

第三个阶段（20 世纪 70 年代到 90 年代）采用工程化的生产方式，是传统软件工程阶段。微处理器的出现与应用使个人计算机发展迅速，这个阶段的硬件向超高速、大容量、微型化以及网络化方向发展。软件系统的规模、复杂性增强，促进了软件开发过程管理及工程化。这个时期，软件开发除了编制程序外，还包括编制开发、使用和维护过程所需的文档。软件开发范围从需求定义、分析、设计、编码、测试、使用到维护等整个软件生命周期。该阶段的主要特征是：

（1）软件产业已经兴起，“软件作坊”已经发展为软件公司，甚至是跨国公司。软件的开发不再是“个体化”或“手工作坊”式的开发方式，而是以工程化的思想作指导，采用工程化的原则、方法和标准来开发和维护软件。

（2）软件开发的成功率大大提高，软件的质量也有了很大的保证。软件已经产品化、系列化、标准化、工程化。

（3）软件工程并发环境（Computer Aided Software Engineering，CASE）及其相应的集成工具大量涌现，软件开发技术中的度量问题受到重视，出现了 COCOMO 软件工作量估计模型、软件过程改进模型 CMM 等。20 世纪 80 年代后期，以 smalltalk、C++等为代表的面向对象技术出现，使得传统的结构化技术受到了严峻的挑战。

第四阶段（20 世纪 90 年代至今）是现代软件工程阶段。数据库、开发工具、开发环境、网络、分布式、面向对象技术等工具方法都得到应用。Internet 技术的迅速发展使得软件系统从封闭走向开放，Web 应用成为人们在 Internet 上最主要的应用模式，异构环境下分布式软件的开发成为一种主流需求，软件复用和构件技术成为技术热点，出现了以 Sun 公司的 EJB/J2EE、Microsoft 公司的 COM+/DNA（分布式网络架构）和 OMG（Object Management Group）的 CORBA/OMA 为代表的 3 个分支。与此同时，需求工程、软件过程、软件体系结构等方面的研究也取得了重要进展。进入 21 世纪，Internet 向智能网络时代发展，以网格技术和 Web 服务为代表的分布式计算日趋成熟，从而实现了信息充分共享和服务无处不在的应用环境。这个阶段的主要特征是：

（1）面向对象技术广泛使用。

（2）软件开发技术逐渐成熟。

这个时代的主流应用技术采用面向对象技术、软件复用技术（设计模式、软件框架、软

件体系结构等)、构件设计技术、分布式计算技术、软件过程管理技术等。

2. 软件的特征

软件同传统的工业产品相比，有其独特的特性。

(1) 软件是一种逻辑实体，具有抽象性。这个特点使它与其他工程对象有着明显的差异。人们可以把它记录在纸上、内存、磁盘和光盘上，但却无法看到软件本身的形态，必须通过观察、分析、思考、判断，才能了解它的功能、性能等特性。

(2) 软件没有明显的制造过程。一旦研制开发成功，就可以大量拷贝同一内容的副本。所以对软件的质量控制，必须着重在软件的开发方面下工夫。

(3) 软件在使用过程中，没有磨损、老化的问题。软件在生存周期后期不会因为磨损而老化，但会为了适应硬件、环境以及需求的变化而进行修改，而这些修改又不可避免地会引入错误，导致软件失效率升高，从而导致软件退化。当修改的成本变得难以接受时，软件就被抛弃。

(4) 软件对硬件和环境有着不同程度的依赖性，这就导致了软件移植的问题。

(5) 软件的开发至今尚未完全摆脱手工作坊式的开发方式，生产效率低。

(6) 软件是复杂的，而且以后会更加复杂。软件是人类有史以来生产的复杂度最高的工业产品。软件涉及人类社会的各行各业、方方面面，软件开发常常涉及其他领域的专门知识，这对软件工程师提出了很高的要求。

(7) 软件的成本相当昂贵。软件开发需要投入大量的、高强度的脑力劳动，成本非常高，风险也大。现在软件的开销已大大超过了硬件的开销。

(8) 软件工作牵涉到很多社会因素。许多软件的开发和运行涉及机构、体制和管理方式等问题，还会涉及到人们的观念和心理。这些人的因素，常常成为软件开发的困难所在，直接影响到项目的成败。

### 1.1.2 软件工程

虽然软件开发的工具越来越先进，人们的经验也越来越丰富，但是，需要解决的问题却越来越复杂。人们发现开发出的软件系统常常出现开发周期长、费用超过预算、最终产品不能满足用户需求、系统的可维护性差等问题，从而导致了软件危机。软件工程的方法就是基于软件危机的问题提出来的。大型的、复杂的软件系统开发是一项工程，必须按工程学的方法组织软件的生产和管理，必须经过系统的分析、设计、实现、测试和维护等一系列的软件生命周期阶段。这一认识促使了软件工程学的诞生。

1. 软件工程的概念与知识体系

软件开发是一项需要良好组织，严密管理且各方人员配合协作的复杂工作。软件工程正是指导这项工作的一门科学。软件工程是一门研究如何用系统化、规范化、数量化等工程原则和方法去进行软件的开发和维护的学科。软件工程包括两方面内容：软件开发技术和软件项目管理。软件开发技术包括软件开发方法学、软件工具和软件工程环境。软件项目管理包括软件度量、项目估算、进度控制、人员组织、配置管理、项目计划等。软件工程的三要素是方法、工具和过程。

统计数据表明，大多数软件开发项目的失败，并不是由于软件开发技术方面的原因，而是由于不适当的管理造成的。遗憾的是，尽管人们对软件项目管理的重要性认识有所提高，但在软件管理方面的进步远比在设计方法学和实现方法学上的进步小。迄今为止，为了达到最初设定的目标，软件工程界已经提出了一系列的理论、方法、语言和工具，解决了软件开发过程

中的若干问题。但是，由于软件固有的复杂性、易变性和不可见性，软件开发周期长、代价高和质量低的问题依然存在。IEEE 在 2002 年发表的报告中指出，即使是 IT 产业最发达的美国，在 2001 年美国本土公司开发的软件产品中平均每 1000 行代码中就有 0.37 个 bug，对于美国之外的其他国家，这个比例更高。

软件工程应该包括哪些知识？IEEE 的软件工程实施体系指南（Guide to the Software Engineering Body of Knowledge 2004Version，SEWBOK）界定了软件工程的 10 个知识领域（Knowledge Areas），即软件需求、软件设计、软件构建、软件测试、软件维护、软件配置管理、软件工程管理、软件工程过程、软件工程工具和方法及软件质量，每个知识领域还包括很多子领域，如表 1.1 所示。

**表 1.1　软件工程知识体系指南（2004）**

| 知识领域 | 内容 |
| --- | --- |
| 软件需求<br>software requirements | 软件需求基础、需求过程、需求获取、需求分析、需求规格说明、需求确认和实践考虑 |
| 软件设计<br>software design | 软件设计基础、软件设计关键问题、软件结构与体系结构、软件设计质量的分析与评价、软件设计符号、软件设计的策略与方法 |
| 软件构造<br>software construction | 软件构造基础、管理构造、实际考虑 |
| 软件测试<br>software testing | 软件测试基础和测试级别、测试技术、与测试相关的度量、测试过程 |
| 软件维护<br>software maintenance | 软件维护基础、软件维护的关键问题、维护过程、维护技术 |
| 软件配置管理<br>software configuration management | 软件配置管理、过程管理、软件配置标志、软件配置控制、软件配置状态统计、软件配置审核、软件发行管理和交付 |
| 软件工程管理<br>software engineering management | 启动和范围定义、软件项目计划、软件项目实施、评审与评价、关闭、软件工程度量 |
| 软件工程过程<br>software engineering process | 过程实施与改变、过程定义、过程评定、过程和产品度量 |
| 软件工程工具和方法<br>software engineering tools and methods | 软件工程工具、软件工程方法 |
| 软件质量<br>software quality | 软件质量基础、软件质量过程、实践考虑 |
| 相关学科<br>related disciplines | 计算机工程、计算机科学、管理、数学、项目管理、质量管理、软件人类工程学和系统工程 |

2. 软件工程的框架

软件工程的框架由软件工程目标、软件工程活动和软件工程原则三个方面的内容构成，如图 1.1 所示。

软件工程的目标是开发与生产出具有良好的软件质量和费用合算的产品，即生产具有正

确性、可用性以及合算的软件产品。正确性是指软件产品达到预期功能的程度。可用性是指软件基本结构、实现及文档为用户可用的程度。费用合算是指软件开发运行的整个开销能满足用户要求的程度。软件质量是指该软件能满足明确的和隐含的需求能力有关特征和特性的总和。软件质量可用六个特性来做评价，即功能性、可靠性、易使用性、效率、维护性、易移植性。软件的目标决定了软件过程、过程模型和工程方法的选择。

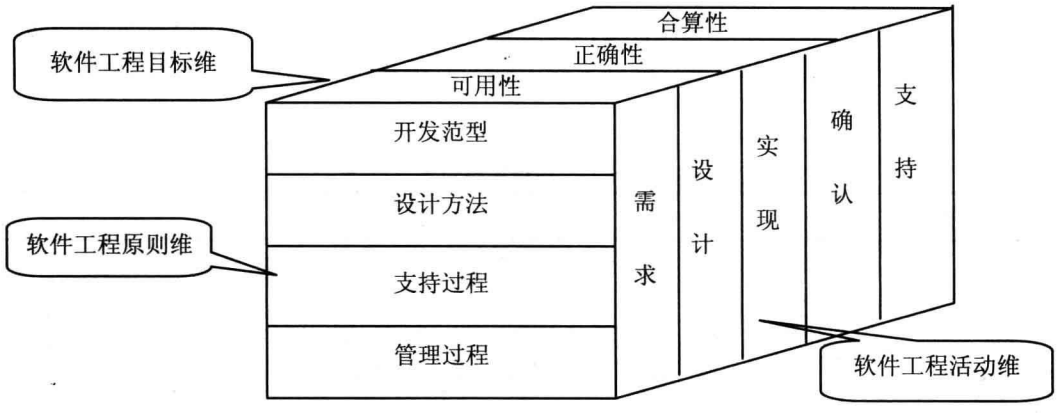


图 1.1　软件工程框架

软件工程活动包括需求、设计、实现、确认和支持。需求包括问题分析和需求分析。问题分析包括需求获取和定义，又称软件需求规约。需求分析包括生成软件功能规约。设计包括概要设计和详细设计。概要设计就是建立整个软件的体系结构，包括子系统、模块以及相关层次的说明、每一模块的接口定义等。详细设计是产生程序员可用的模块说明，包括每一模块中数据结构说明及加工描述。实现就是把设计结果转换为可执行的程序代码。确认贯穿整个开发过程，对完成的结果进行确认，保证产品满足用户的要求。支持是修改和完善活动。

软件工程的原则是指围绕工程设计、工程支持以及工程管理在软件开发过程中必须遵循的原则。软件工程有以下四项基本原则：

（1）选取适宜开发范型。该原则与系统设计有关，在系统设计中，软件需求、硬件需求以及其他因素之间是相互制约、相互影响的，经常需要权衡。因此，必须认识需求定义的易变性，采用适宜的开发范型予以控制，以保证软件产品满足用户的要求。

（2）采用合适的设计方法。在软件设计中，通常要考虑软件的模块化、抽象与信息隐蔽、局部化、一致性以及适应性等特征。合适的设计方法有助于这些特征的实现，以达到软件工程的目标。

（3）提供高质量的工程支持。“工欲善其事，必先利其器”。在软件工程中，软件工具与环境对软件过程的支持颇为重要。软件工程项目的质量与开销直接取决于对软件工程所提供的支撑质量和效用。

（4）重视开发过程的管理。软件工程的管理，直接影响可用资源的有效利用、生产满足目标的软件产品和提高软件组织的生产能力等问题。因此，仅当软件过程得以有效管理时，才能实现有效的软件工程。

软件工程框架告诉我们，软件工程的目标是可用性、正确性和合算性；软件工程活动主要包括需求、设计、实现、确认和支持等活动，每一活动可根据特定的软件工程，采用合适的

开发范型、设计方法、支持过程以及过程管理。根据软件工程这一框架，软件工程学科的研究内容主要包括：软件开发范型、软件开发方法、软件过程、软件工具、软件开发环境、计算机辅助软件工程（CASE）及软件经济学等。

3. 软件工程的基本原理

自从 1968 年提出“软件工程”这一术语以来，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。美国著名软件工程专家勃姆（Barry Boehm）在总结软件工程准则和信条的基础上，于 1983 年提出软件工程的七条基本原理，也是软件项目管理应该遵循的原则。勃姆认为这七条原理是确保软件产品质量和开发效率的原理的最小集合，而且，它们是相互独立、缺一不可的最小集合；同时，它们又是相当完备的。

（1）用分阶段的生命周期计划严格管理。统计表明，不成功的软件项目中约有一半左右源自计划不周。本原则意味着，应该把软件生命周期划分成若干阶段，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。勃姆认为，在软件的整个生命周期中应该制定并严格执行六类计划，即项目概要计划、里程碑计划、项目控制计划、产品控制计划、验证计划、运行维护计划。不同层次的管理人员必须严格按照计划各尽其职地管理软件开发与维护工作，绝不能受顾客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。软件的质量保证工作不能等到编码阶段结束之后再进行，其理由为：第一，大部分错误始于编码之前，大约占 63%；第二，错误的发现与修改时间越晚，需要付出的代价就越高。因此，本原则意味着，在软件开发的每个阶段应该进行严格的评审，以便尽早发现软件开发过程中的错误。

（3）实行严格的产品控制。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。但实践告诉我们，需求的改动往往是不可避免的。这就要求我们要采用科学的产品控制技术来顺应这种要求。当改变需求时，为了保持软件各个配置成分的一致性，必须实行严格的产品控制，其中主要是实行基准配置管理。所谓基准配置又称基线配置，它们是经过阶段评审后的软件配置成分（各个阶段产生的文档或程序代码）。基准配置管理也称为变更控制，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，获得批准以后才能实施修改。要避免开发人员对软件随意进行修改。

（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。从 20 世纪 60 年代末提出的结构程序设计技术到如今的面向对象技术，人们不断创造先进的程序设计技术。实践表明，采用先进的技术既可提高软件开发的效率，又可提高软件维护的效率。

（5）结果应能清楚地审查。与其他有形产品不同，软件是看不见摸不着的逻辑产品。软件开发人员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以评价和管理。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，从而使所得到的结果能够清楚地审查。而且每一个阶段都应该进行评审，如图 1.2 所示。

（6）开发小组的人员应该少而精。软件开发项目的组成人员素质应该好，而人数则不宜过多。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。素质高的人员开发效率比素质低的人员开发效率可能高几倍至几十倍，而且素质高的人员所开发软件中的错误明显少于素质低的人员所开发的软件。此外，随着开发小组人员数目的增加，交流情况和讨论问题而造成的通信开销也急剧增加。当开发小组人员数为 $N$ 时，可能的通信路径有 $N(N\text{-}1)/2$

条，可见随着人数 $N$ 的增大，通信开销将明显增加。因此，在开发进程中，切忌中途加人。构建和维持少而精的开发团队甚至标杆团队是软件工程的一条基本原理。

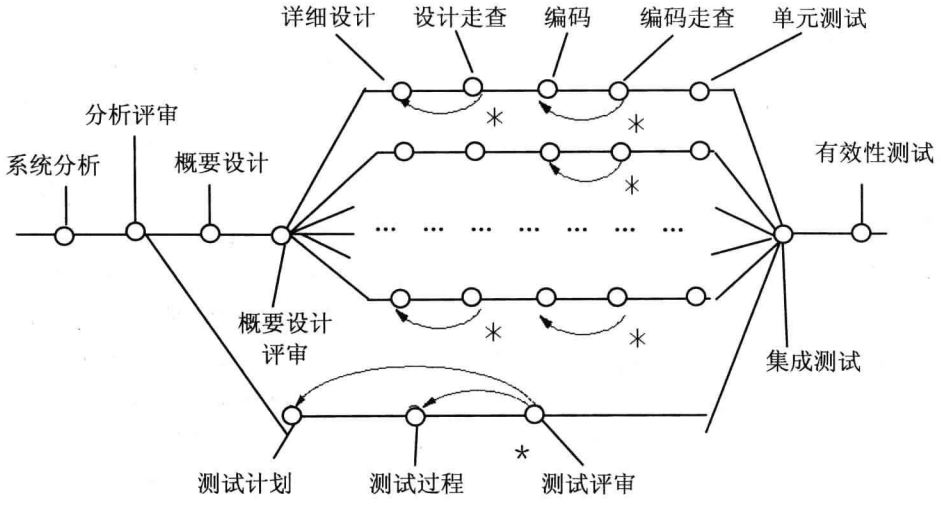


图 1.2 阶段审查图

（7）承认不断改进软件工程实践的必要性。遵循上述六条基本原则，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，仅遵循上述六条原则并不能保证软件开发与维护过程能赶上时代前进的步伐和技术的不断进步。因此，勃姆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则。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。

## 1.2 软件过程和开发方法

为迎接软件危机的挑战，人们进行了不懈的努力。主要包括两个方面工作：一是从管理的角度希望实现软件开发过程的工程化。这方面最为著名的成果就是提出了大家都很熟悉的“瀑布式”生命周期模型。后来针对该模型的不足，有人提出了快速原型法、螺旋模型、喷泉模型等，这些模型在软件开发的实践中被广泛采用。另外还提出了一些重要文档格式标准，包括变量、符号的命名规则以及源代码的规范式等。二是侧重对软件开发过程中分析、设计方法的研究。如 20 世纪 70 年代风靡一时的结构化开发方法、原型化方法以及 80 年代著名的面向对象开发方法等。

### 1.2.1 软件过程

软件过程是为了获得高质量的软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。软件过程描述为了开发出客户满意的软件，什么人（who）、在什么时候（when）、做什么事（what）以及怎么做（how）这些事以实现某一个特定的具体目标。通常用生命周期模型简洁地描述软件过程。生命周期模型规定了把生命周期划分成哪些阶段及各阶段的执行顺序，因此，称为软件过程模型，又称为软件开发模型。软件开发模型是指软件开发

中的所有过程、活动和任务的结构框架，它能清晰、明确地表达软件开发的全过程。对于不同的软件系统，可以采用不同的软件开发过程和方法、不同的软件工具和软件工程环境、不同的管理方法和手段来实现软件项目的跟踪和把控。软件开发通常包括需求、设计、实现、确认和支持等阶段。下面讲述的模型不是针对某个特定的项目，因此，使用“通用的”阶段划分方法。常见的软件过程模型有瀑布模型、快速原型模型、增量模型、螺旋模型、喷泉模型等，此外还有迭代模型、V 模型和智能模型等。本书只介绍常见的软件过程模型。

1. 瀑布模型

1970 年温斯顿 • 罗伊斯（Winston Royce）提出了著名的“瀑布模型（Waterfall Model）”，有时也称为传统生存周期模型或线性顺序过程模型。20 世纪 80 年代之前，它一直是唯一被广泛采用的软件开发模型，现在它仍然是软件工程中应用最广泛的过程模型之一。传统软件工程方法学的软件过程，基本上可以用瀑布模型来描述。瀑布模型将软件生命周期划分为软件计划、需求分析、系统设计、软件编写、软件测试和运行维护等六个基本活动，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。瀑布模型如图 1.3 所示。在瀑布模型中，软件开发的各项活动严格按照线性方式进行，只有前一项活动的任务完成才能进行下一项活动，前一项活动的工作结果是后一项工作的依据。前一项活动的工作结果需要进行验证，验证通过的结果作为下一项活动的输入，继续进行下一项活动，否则返回修改。

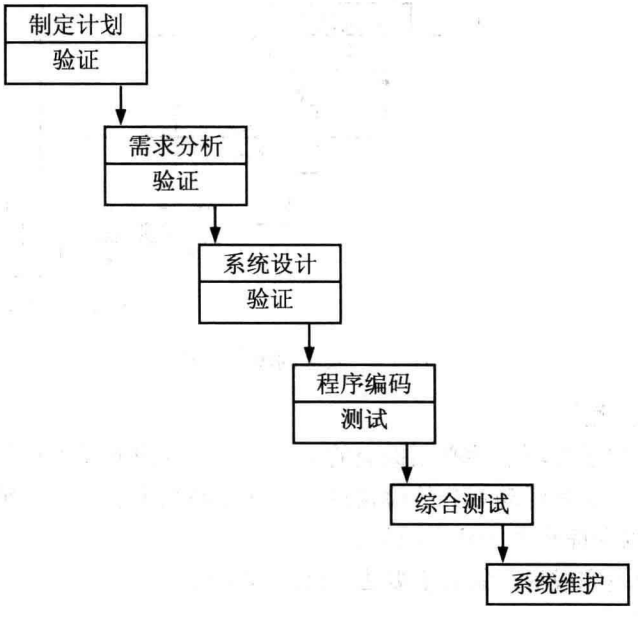


图 1.3　传统的瀑布模型

瀑布模型强调文档的作用，并要求每个阶段都要仔细验证。瀑布模型的主要特点是：

（1）阶段间具有顺序性和依赖性。使用瀑布模型时，只有前一项活动完成之后，才能进入后一项活动。前一项活动的输出文档是后一项活动的输入文档，只有前一项活动的输出文档正确，后一项活动才能正确。

（2）推迟实现的观点。实践表明，对于规模较大的项目开发，往往编码开始的越早，最终完成开发工作需要的时间反而越长。这是因为前面的分析和设计工作没有做好，过早地进行

了程序的实现，往往会导致大量的返工，有时甚至发生无法弥补的错误，导致项目开发失败。

（3）质量保证的观点。瀑布模型在实现之前无法了解项目的实际情况，只有实现了才知道。因此，为了保证所开发软件的质量，在使用瀑布模型开发时需要注意两点：一是每项活动结束前，必须完成规定的文档，没有交出合格的文档，就是没有完成该项活动的任务。二是每项活动结束前必须对所完成的文档进行评审，以便尽早地发现问题，并及时改正。因为最早潜入的错误，是最晚暴露出来的，暴露出来的时间越晚，改正错误所付出的代价越高。所以，及时验证是保证软件质量、降低软件成本的重要措施。

传统的瀑布模型过于理想化，实际上，人们在软件开发过程中不可能不犯错误，在设计活动中可能会发现需求分析中存在的错误，而设计中的缺陷和错误可能在实现的过程中发现，在综合测试中又可能会发现需求分析、系统设计或编码活动中的错误。因此，实际的瀑布模型是带反馈环的，如图 1.4 所示（图中实线箭头表示开发过程，虚线箭头表示维护过程）。

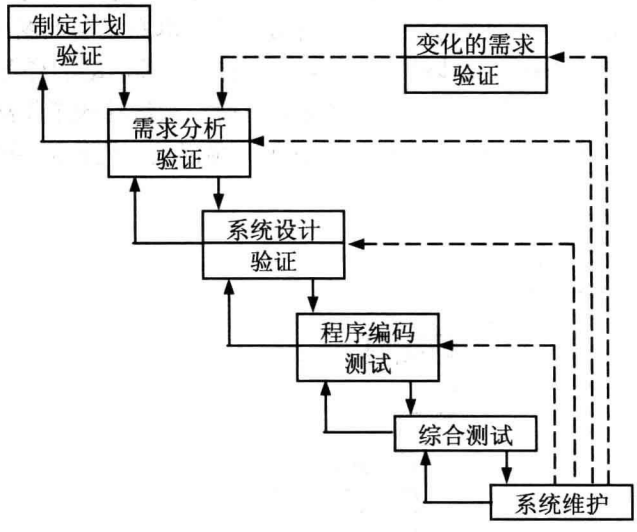


图 1.4　实际的瀑布模型

瀑布模型有以下优点：

（1）严格地规定了每项活动必须提交的文档。可以强迫开发人员采用规范的开发方法。

（2）为项目提供了按活动划分的检查点。每项活动交出的产品必须经过质量保证小组的仔细验证，这样可以保证软件的开发质量。

（3）当前一阶段完成后，您只需要去关注后续阶段。

瀑布模型的缺点：

（1）各个阶段的划分完全固定，阶段之间产生大量的文档，极大地增加了工作量。

（2）由于开发模型是线性的，用户只有等到整个过程的末期才能见到开发成果，从而增加了开发的风险。

（3）早期的错误可能要等到开发后期的测试阶段才能发现，进而带来严重的后果。

瀑布模型是线性的，“线性”是人们最容易掌握并能熟练应用的思想方法。当人们碰到一个复杂的“非线性”问题时，总是千方百计地将其分解或转化为一系列简单的线性问题，然后逐个解决。一个软件系统的整体可能是复杂的，而单个子程序总是简单的，可以用线性的方式