普通高等教育“十一五”国家级规划教材

UML系统建模与分析设计

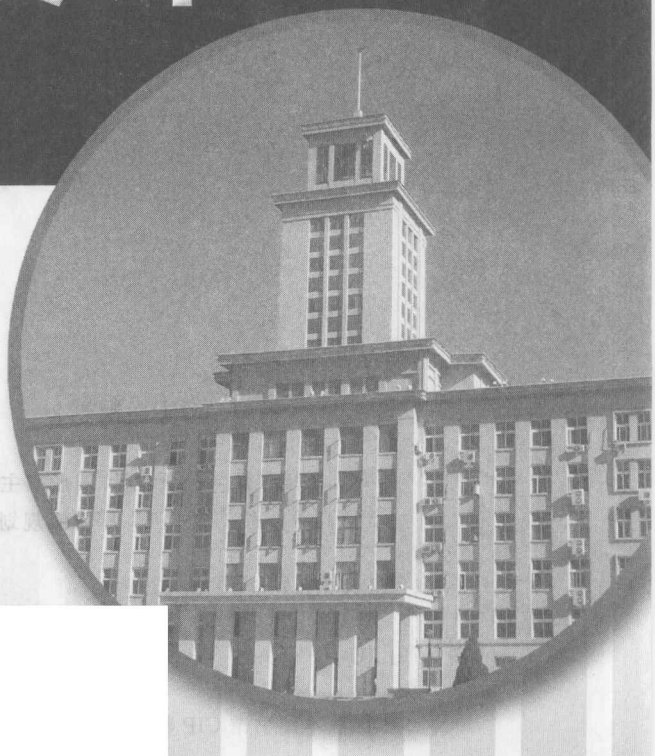
刁成嘉 主编
南开大学

为教师配有电子课件

机械工业出版社
China Machine Press

普通高等教育“十一五”国家级规划教材

UML系统建模与分析设计



刁成嘉 主编　　刁奕 等参编

机械工业出版社
China Machine Press

本书系统、全面地阐述基于UML的面向对象分析与设计的基本概念，详细介绍统一建模语言UML及其开发过程，以一个集成案例贯穿可行性研究、需求分析、系统分析与设计的全过程，并给出各阶段的基础模型范例和文档书写格式。本书还重点介绍面向对象的软件开发CASE集成环境、设计模式、软件复用技术、分布式对象技术、C/S模型、B/S模型、持久对象、往返工程、逆向工程和CORBA构件接口技术等内容。本书深入浅出、循序渐进，可使读者快速掌握面向对象的系统分析、设计方法。

本书可作为高等院校计算机专业本科生或研究生相关课程教材，同时也适合作为广大软件开发人员学习面向对象技术的自学指导书和技术参考书。

图书在版编目（CIP）数据

UML系统建模与分析设计/刁成嘉主编．-北京：机械工业出版社，2007.6
（普通高等教育“十一五”国家级规划教材）

ISBN 978-7-111-21384-0

Ⅰ.U…　Ⅱ.刁…　Ⅲ.面向对象语言，UML-程序设计-高等学校-教材　Ⅳ.TP312

中国版本图书馆CIP数据核字（2007）第060048号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：杨庆燕
三河市明辉印装有限公司印刷·新华书店北京发行所发行
2007年7月第1版第1次印刷
184mm×260mm·20.75印张
定价：33.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

Preface 前言

现代软件工程领域中，面向对象的系统分析和设计方法已逐渐取代了传统的方法，成为当前国内外计算机软件工程学的主流方法。其核心思想是利用面向对象的概念和方法为软件建立客户需求模型、系统分析模型和系统设计模型，采用面向对象程序设计语言完成系统实现，并对建成的系统进行面向对象的系统测试和系统维护。在今天，特别是随着 Internet/Intranet 的发展，网络分布计算的应用需求日益增长，面向对象技术为网络分布计算提供了基础性的核心技术支持。

本书分 9 章来系统、全面地阐述基于 UML 的面向对象分析与设计方法。

第 1 章概要介绍系统建模与分析设计技术的演变历程和面向对象方法学的发展过程，以及面向对象分析与设计的基本概念。

第 2 章简要介绍统一建模语言 UML 及其开发过程，介绍面向对象分析与设计各个阶段产生的简单模型及其基本图符表示方法。

第 3～7 章详细介绍采用基于 UML 的面向对象分析与设计技术开发建立一个软件项目模型的全过程。以一个集成设计案例为范本，详细说明了从可行性分析、客户需求分析、系统分析到系统设计的各个阶段，建立了系统的业务用例模型、系统用例模型，以及对象的静态模型、动态模型和功能模型，并介绍了几个简单的设计模式，还介绍了可行性分析报告、客户需求规格说明、系统分析报告和系统设计报告的格式和基本内容。

第 8 章专门介绍设计模式，注重介绍设计模式的产生及特点、描述方法和使用规则，并对 11 种经典的设计模式进行了详细分析。

第 9 章重点介绍软件复用和构件接口技术。主要讨论了软件复用的方法和组织实施，COM+、EJB、CORBA 等构件接口技术模型，持久对象与关系数据库、面向对象数据库，客户/服务器模型、浏览器/服务器模型及分布式对象模型，计算机辅助软件工程（CASE）工具及面向对象的 CASE 集成环境的功能与结构等。

各章后面附有各种类型的习题供读者练习使用。附录部分对支持 UML 统一开发过程的 Rose 集成 CASE 开发环境及其使用做了简单介绍。

为了配合本教材的教学与学习，还配套出版了《UML 系统建模与分析设计课程设计》一书，提供一个案例的完整开发过程，供读者研习。

教学建议

建议在本课程开始时，为每个同学选择一个拟开发的课题作为开发案例。在教学过程

中，随着课程的深入，按照课程设计的要求，逐步开发、完善这个开发案例的系统模型设计。系统的实现可以采用任何一种面向对象程序设计语言（如 Java、C++等）。在学期末，要求学生能有一个完整的系统实现。

本教材由刁成嘉主编，其中第 9 章和附录由刁奕编写，金士英、陈艳秋、杨鹏飞、高建国、旷昊、蓝炳伟、郑伟、唐木玲、赵泳、杨志真等参加了部分习题的编写。由于编者水平所限，加之时间仓促，疏漏、欠妥、谬误之处在所难免，敬请读者批评指正。

刁成嘉

2007 年 6 月于南开园

Contents 目录

Chapter 1 第1章 系统建模与分析设计技术的演变

面向对象技术是软件工程学的一个重要分支，也是当今软件开发的主流方法。系统建模与分析设计是研究和应用如何以系统化、规范化、可度量的方法开发、运行和维护软件的一种层次化技术，包括过程、方法和工具三个要素。了解系统建模与分析设计技术的演变过程，采用最科学、最适用的系统建模与分析设计技术，学习和掌握其基本要点、基础理论和方法，是软件开发者充分发挥自身优势，运用团队合作能力，开发出高质量软件系统所必备的基本素质 。

本章目的：

- 理解软件的基本概念和特点。
- 了解软件的发展过程及软件的开发过程。
- 了解软件开发的方法。
- 掌握面向对象技术的基本概念及开发过程。
- 了解几种典型的面向对象方法。

自 20 世纪 40 年代计算机问世以来，计算机在各个领域得到了广泛的应用，使得计算机技术蓬勃发展。然而，长期以来计算机软件开发的低效率制约着计算机软件行业的发展。计算机业界努力探索和研究解决软件危机的途径，并提出了软件工程的思想和方法，这极大地提高了软件开发的效率和软件的质量。当代软件工程的发展正面临着从传统的结构化范型到面向对象范型的转移，这需要有新的语言、新的系统和新的方法学的支持，面向对象技术就是这种新范型的核心技术。面向对象的系统分析和设计方法已逐渐取代了传统的方法，成为当今世界计算机软件工程学的主流方法。

1.1 软件的概念、特点和分类

计算机技术发展的每一步几乎都能够在软件设计和程序设计语言中得到体现。软件是不断发展的，从 20 世纪 40 年代在 ENIAC 计算机上编制程序开始到提出软件工程术语的 20 多年时间里，人们对软件开发的理解就是编程序，且编程是在一种无序的、崇尚个人技巧的状态中完成的。在高级语言出现以前，汇编语言（或机器语言）是编程的主要工具，其表达软件模型的基本方式（或语言构成）是指令，表达模型处理逻辑的主要概念（机制）是顺序和转移。显然，这一抽象层次是比较低的。随着诸如 FORTRAN、PASCAL、COBOL、C 等高级语言的出现，软件开发人员使用变量、标识符、表达式、子例程、函数等概念作为语言

的基本构造，并使用顺序、分支和循环3种基本控制结构来表达软件模型的计算逻辑，因此软件开发人员可以在一个更高的抽象层次上进行程序设计。其间出现了一系列开发范型和结构化程序设计技术，实现了模块化的数据抽象和过程抽象，提高了人们表达客观世界的抽象层次，并使开发的软件具有一定的构造性和演化性。

面向对象方法自从在20世纪80年代中期诞生以来，即开始逐步流行，为人们提供了一种以对象为基本计算单元，以消息传递为基本交互手段来表达的软件模型。面向对象方法的实质是以拟人化的观点来看待客观世界，即客观世界是由一系列对象构成，这些对象之间的交互形成了客观世界中各式各样的系统。面向对象方法中的概念和处理逻辑更接近于人们解决问题的思维模式，使开发的软件具有更好的构造性和演化性。

目前，人们更加关注软件复用问题，构建比对象粒度更大、更易于复用的基本单元——可复用构件，并研究以构件复用为基础的软件构造方法，更好地凸显软件的构造性和演化特性，因为易于复用的软件具有很好构造性和演化性。

1. 软件的概念和特点

软件是人们对客观世界中问题空间与解空间的具体描述，是客观事物的一种反映，是知识的提炼和“固化”。形象一点儿说，软件就是程序以及开发、使用、维护程序所需要的所有文档，即：软件＝程序＋文档。由于客观世界是不断变化的，因此，构造性和演化性是软件的本质特征。如何使软件模型具有更强的表达能力、更符合人类的思维模式，也即如何提升计算环境的抽象层次，从一定意义上来讲，就是围绕软件的本质特征——构造性和演化性——不断提升和具体实施的问题。

进入20世纪60年代后，由于客观实际的需求，要制作一些大型的软件系统。在这种需求的驱动下，软件开发方法和管理受到重视，逐渐产生出各种不同类型的软件开发方法和模型。随着时间的推移和实际工作的需要，软件开发方法也在不断地更新变化。

与20世纪60年代以前的软件开发相比，今天的软件开发具有以下特点：

1）软件规模大。过去，由于硬件发展水平的限制，只有少数专业人员才去关心软件，而且这些专业人员必须懂得硬件，根据硬件的性能编写一些简单、单一的应用软件。这一点从根本上阻碍了软件的广泛应用，限制了软件的规模。现在，软件开发方法已逐渐走向成熟，成为了一种系统的工程技术，经常要完成一些大规模或者超大规模的项目工程。

2）软件开发规范化并趋于标准化。以往大部分软件开发人员不太关心别人的工作，他们往往醉心于自己的编程技巧，决定软件质量和开发时间的唯一因素就是编程人员的素质，即个人的经验、智慧和技巧。如今，软件的易读易懂和文档化是人们追求的目标，一个规范化、标准化的软件开发过程更加注重的是团队合作开发，而且各种软件制品可以比较容易地整合集成。

3）软件开发方法多，有大量的软件工具支持。计算机刚刚发展的时候，由于缺少有效的软件开发方法和软件工具支持，使人们错误地认为编程仅仅是一门艺术，只是少数人的专利。现如今，有众多科学的软件开发方法及其配套的开发工具供人们选择使用。

4）注重软件开发的管理。在传统的软件开发中，由于人们只重视个人的技能，软件开发过程的透明度低，致使管理人员几乎不清楚软件技术人员的工作进展情况，造成管理活动很少甚至不存在。而今天，“管理是软件开发的生命”这种观念已经深入人心，使人们更加重视软件设计、开发、运行和维护等各个环节的管理。

5）软件维护相对过去容易得多。

在某种程度上，软件产品在某些方面有些相似于其他一般工程中的有形产品，如桥梁、建筑物、机床、计算机，但它们之间也有一些重要的差别，从而不能简单地把一般工程方面的知识、方法和技术直接应用到软件工程上来。软件工程尽管无时无刻都在发展，但还不能说其已经成熟。软件是逻辑产品而不是实物产品，像磁盘、集成电路块都只是软件的载体。由于软件是逻辑产品，其功能只有依赖于硬件和软件的运行环境以及人们对它的操作才能得以体现。没有计算机相关硬件的支持，软件没有实用价值。同样，没有软件支持的计算机硬件，也会是毫无价值的机器，软件与硬件密切相关的程度是一般工程所没有的。一般情况下，软件产品要完成多种多样的功能，以使用户清晰、准确地描述他们要解决的问题，同时软件产品也要具备较强的可靠性、易移植性和易使用性。

2. 软件的分类

世界上有多种多样的计算机软件做着各自不同的工作，要给软件做出科学的分类很困难。对于不同类型的软件工程对象，其开发和维护有着不同的要求和处理方法，可以按软件的功能、规模、工作方式、服务对象和使用频度等对其进行分类。

(1) 按软件的功能划分

- 系统软件。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。例如，操作系统、设备驱动程序以及通信处理程序等。
- 支撑软件。协助用户开发软件的工具性软件，其中包括帮助程序人员开发软件产品的工具，也包括帮助管理人员控制项目开发进程的工具。
- 应用软件。在特定领域内开发、为特定目的服务的一类软件。应用软件的种类繁多，涉及范围也很广。如商业数据处理软件、工程与科学计算软件、事务管理软件、办公自动化软件、中文信息处理软件和计算机辅助教学软件，还有系统仿真、人工智能型软件等。

(2) 按软件的规模划分

- 微型软件。一个人可用很短时间完成的软件，程序不超过 500 行语句，仅供个人专用。
- 小型软件。长度约 2000 行左右的程序，一般用于数值计算或数据处理问题。
- 中型软件。一般由一个开发小组花费一年多时间完成，长度在 5000～50 000 行之间的程序。这种软件需要开发人员之间、开发人员与用户之间频繁联系，密切协调，因而要有详细的开发计划、建立分析设计模型、进行必要的技术审查和完善的文档资料储备等。
- 大型甚至超大型软件。多个研发小组，甚至是成百上千人组成的开发团队，利用 3 到 5 年甚至更长时间完成的软件项目。这种软件往往需要进行二级开发，或者将项目划分为若干个子项目，每个子项目都是一个中型以上的软件，子项目之间具有复杂的接口，采用统一的标准，而且必须具备完备、严密的审查环节和详细的文档资料。例如编译程序、分时系统、实时控制系统以及实时处理系统、多任务系统、大型数据库管理系统等。

(3) 按软件工作方式划分

- 实时处理软件。指在事件或数据产生时，立即予以处理，并及时反馈信息，对过程需要进行监测和控制的软件。包括数据采集、分析、输出三部分，对时间有严格的

限定。

- 分时软件。这种软件允许多个联机用户同时使用一台计算机的处理器。系统把处理器时间轮流分配给各联机用户，使各用户都感到只是自己在使用计算机。
- 交互式软件。能实现人机通信的软件。具有友好的用户界面，及时应答用户的输入信息，发出反馈信息。主要用于终端设备，实现人机交互。
- 批处理软件。以批处理的方式把一组作业或一批数据输入到计算机，计算机按顺序依次处理每个数据的软件。

(4) 按软件服务对象的范围划分

- 项目软件。针对某个特定客户的委托，由开发机构在合同的约束下开发出来的软件。这种软件带有试验、研究性质，一般需要软件开发机构拥有良好的质量管理、技术实力、开发经验以及信誉等。
- 产品软件。由软件开发机构开发出来直接提供给市场的软件，或是为千百个用户服务的软件。例如，文字处理软件、文本处理软件、财务处理软件、人事管理软件等。

(5) 按使用的频度划分

- 一次性使用软件。如人口普查、工业普查软件，要若干年才使用一次，属于一次性使用软件。
- 使用频度较高的软件。如天气预报软件则属于使用频度较高的软件。

(6) 按软件失效的影响程度划分

- 一般性软件。如有的软件因发生故障而造成软件失效，但它对整个系统带来的影响不大，这属于一般性软件。
- 关键性软件。而有的软件，如金融财务、国防军工、航空航天等领域中使用的软件，可靠性要求高，一旦失效将会造成灾难性的后果，这类软件属于关键性软件。

1.2 软件的发展与软件工程

计算机软件伴随着计算机科学的发展进程而不断发展、完善，软件危机的出现，促使人们对软件的特性和设计方法不断进行更深入的研究，逐步形成了软件工程学科。同时，结构化程序设计方法、快速原型法、面向对象设计方法等软件设计方法相继产生和发展，加快了软件工程的建设，减少了软件危机所带来的影响。

事实上，早期的软件发展之路是异常艰难的，所设计的软件产品经常发生错误。1962年6月，美国飞向金星的第一个空间探测器——水手1号，因飞行舱中计算机导航程序中的一条语句的语义出错，致使其偏离航线，导致整个计划失败。阿波罗8号太空飞船的一个计算机软件错误，造成存储器一部分信息丢失；而阿波罗14号在整个飞行过程中，出现了18个软件错误。可以说，软件系统的可靠性几乎得不到保证。

另外，当时计算机硬件成本每隔2～3年降低一半，内存和外存的成本每年降低40%左右，硬件的性能价格比每十年提高一个数量级，导致软件需求量大大增加。然而，所需的软件很少能在开发成本、时间进度、功能规模、维护能力等方面达到要求，特别是其可靠性难以符合人们的需要。那时，在开发一些大型软件系统上，遇到的困难相当多，有些系统甚至彻底失败。有些系统虽然完成了，但比原定计划推迟了好几年，而且费用大大超出了预算。

有些系统未能完全符合用户当初的期望，有些系统则无法进行修改维护。

究其原因，是因为大型软件系统大大地增加了软件的复杂度，即技术复杂性和管理复杂性成指数数量级上升。就拿软件维护工作来说，它比计算机硬件的维护要复杂得多。美国Dalg研究了一个16万条汇编语句与17万个逻辑门组成的大型实时系统的维护记录日志，结果发现：

- 软件维护时修改一条汇编语句的工作量是硬件维护时修理一个逻辑门的4倍。
- 软件维护费用是硬件维护费用的4倍。
- 编写一条汇编语句所需工作量是研制一个硬件逻辑门的两倍。

在那时，大型软件系统中存在潜在错误的情况已司空见惯，要开发一个没有错误的软件极其困难。

软件是计算机系统中与硬件相互依存的另一部分，它包括程序、相关数据及其说明文档。其中程序是按照事先设计的算法要求执行的指令序列，数据是程序能正常操作的信息，文档是与程序开发维护和使用有关的各种图文资料。人们对软件的认识经历了一个由浅到深的过程，从追求编程技巧到崇尚清晰好用、易于修改和扩充，其过程实际上就是软件设计方法的发展历史。

计算机科学的不断发展，使得软件的需求量不断增大，它的要求、复杂度、开发成本也越来越高，但软件的开发方法和技术却还停留在“小程序”、“个体化”的操作上面，致使软件设计犹如泥潭，大批的设计者深陷其中，甚至出现上述存在的种种缺点，人们称其为软件危机。虽然，“软件危机”的概念是在1968年NATO（北大西洋公约组织）的计算机科学家在联邦德国召开的国际学术会议上才第一次提出，其实它几乎从计算机产生的那一天起就出现了。它是计算机科学发展进程的必然产物，只不过到后来这种现象日渐严重，已经影响到计算机事业的发展，才引起各界的关注。

具体地说，软件危机就是软件开发和维护过程中所遇到的一系列严重问题，其主要有以下表现：

- 对软件开发成本和进度的估计常常不准确。
- 用户对“已完成”系统不满意的现象经常发生，产品无法符合用户的实际需求。
- 软件产品的质量往往靠不住。故障一大堆，补丁一个接一个。
- 软件的可维护程度非常低。
- 软件通常没有适当的文档资料。
- 软件的成本不断提高。
- 软件开发生产率的提高跟不上硬件的发展和人们需求的增长。

软件危机产生的原因主要有两个：一是与软件本身的特点有关，另一个是与软件开发和维护的方法不正确有关。通过分析软件危机的表现和原因，再加上不断地实践和总结，人们终于认识到：按照工程化的原则和方法组织软件开发工作，是摆脱软件危机的一个主要出路，这就是软件工程的概念。

软件工程把软件的生产过程分为需求分析、系统分析、系统设计、功能设计、实现、测试、运行和维护等几个主要阶段，并按各个阶段进行实施，使生产出来的产品可以较有效地抑制软件危机现象。软件作为一种新的事物，它的开发过程介于一般物质工程和社会工程之间，其产品又介于传统意义上的商品和人类思想成果之间，具有太多的二相性，用任何一种

已有的思维方式和工作方法都不能有效地处理它。同时，软件工程思想是用物质工程的思维方式来考察软件的物质工程的一面，其手段包括严密的需求分析，按部就班的开发过程模型和文档管理等，没有太多的自由空间，都不能到达各自的“极限”。

在一定程度上，软件工程的出现，降低了软件危机发生的可能性，但是必须承认这样一个事实，那就是软件危机是不可能完全消失的。因为软件的实质是人们以计算机编程语言为桥梁，将客观感知世界映射到计算机世界中，以解决人们在客观感知世界中要解决的问题。这里牵涉三个主要的范畴：客观感知世界——计算机编程设计语言——计算机世界。人们对客观世界的认识一直都处在“深入中”状态，永远不可能到达极限，因此也就不能保证开发出的软件能完整地反映现实世界。如果把“解决软件危机”比作一个极限，那么只能逼近它，而永远不能到达。事物总是一分为二的，正因为软件危机的“不可解”性，使得人们必须用积极的态度去面对，想尽办法去解决。在经历了几十年的不懈努力后，软件工程的理论已得到极大的丰富和完善，各种软件设计方法层出不穷，软件行业一片繁荣，从而也促进了计算机科学的不断向前发展。

软件工程的基础是一些软件开发的指导性原则，概括如下：

- 变动的软件需求。必须充分认识软件需求的变动性，保证提供满足用户要求的产品。要根据模块化、抽象化、信息隐蔽、局部化和一致性原则进行软件设计。
- 稳妥的设计方法。精良的开发工具与开发环境可以大大方便软件开发，达到软件工程的目标。
- 高效的软件开发支持技术。软件工程项目的质量和效益取决于其支持技术的质量和效用。
- 有效的过程管理。只有对软件工程开发过程进行有效的管理，才能产生有效的软件工程。

随着软件工程的研究和实践取得了长足的进步，出现了一些具有里程碑意义的软件开发技术，它们包括：

- 20 世纪 60 年代末至 70 年代中期，在一系列高级语言应用的基础上，出现了结构化程序设计技术，并开发出了一些支持结构化软件开发方法的工具。
- 20 世纪 70 年代中期至 80 年代，计算机辅助软件工程（CASE）成为研究热点，并开发了一些对软件技术发展具有深远影响的软件工程开发环境。
- 20 世纪 80 年代中期至 90 年代，出现了面向对象语言和方法（这种方法逐渐成为主流的软件开发技术）。同时开展了软件过程及软件过程改善的研究，注重软件复用和软件构件技术的研究与实践。

1.3 软件开发模型的演变和生存周期

1.3.1 软件开发过程与模型的演变

由于认识到软件的设计、实现、维护和传统的工程规则有相同的基础，在提出“软件危机”的同时，“软件工程”应运而生，它也是于 1968 年由 NATO（北大西洋公约组织）在德国格密斯（Garmish）举行的学术会议上第一次正式提出的。在软件工程概念中，软件开发过程占主要部分。经过软件工程的实践，人们总结出一系列开发模型。通过对这些模型的运用和研究，又不断改进和提出新的模型。但各种模型都有其缺点，都只是适应某类具体情

况，没有统一的可适用的模型，因此，选择模型很重要。选择正确的适用于某个具体软件工程项目的软件开发模型，是保证软件开发成功的第一步，也是一条重要的指导性原则。

要掌握软件开发模型，就要了解软件开发模型的概念。软件开发模型是软件开发全部过程、活动和任务的结构框架。软件开发模型能清晰、直观地表达开发全过程，明确规定了要完成的主要活动和任务，可以作为软件项目开发工作的基础。对于一个具体的软件系统来说，应采用适合的开发方法，使用适宜的程序设计语言，组织具有相应技能的开发人员参与，并采用适合的管理方法和手段对整个开发活动的全过程进行有效的控制。还应采用最适宜的、高效的软件开发工具和相应的软件工程环境，其采用的软件开发模型应该是稳定有效和普遍适用的。

需要提醒注意的是：开发模型仅对软件系统的开发、运行、维护过程有意义。生存周期模型是一个框架，它规定了软件开发、运行和维护所需的过程、活动和任务。软件生存周期模型和软件开发模型是同一概念。下面介绍几个经典的软件开发模型，读者可以从中看到软件开发模型的演变过程。

1. 瀑布模型

1970 年，W. Royce 提出了著名的“瀑布模型（waterfall model）”，它是最早出现的软件开发模型，直到 20 世纪 80 年代早期，一直是唯一被广泛采用的软件开发模型。顾名思义，瀑布模型就是将软件开发过程中的各项活动规定为按固定顺序连接的若干阶段工作，换句话说，它将软件开发过程划分为若干个互相区别而又彼此联系的阶段，每个阶段中的工作都以上一个阶段工作的结果为依据，同时为下一个阶段的工作提供了前提。简单地说，瀑布模型分为以下几个阶段：

- 确定客户需求。
- 定义系统需求。
- 系统分析与设计。
- 系统测试。
- 系统实现与集成。
- 系统调整与改进。
- 系统交付或使用。

我国曾在 1988 年依据该开发模型制定并公布了“软件开发规范”国家标准，这对我国软件开发起到了较大的促进作用。瀑布模型之所以能广泛流行多年，有两个原因：首先，它在支持开发结构化软件、控制软件开发复杂度、促进软件开发工程化方面起到了显著作用；其次，它为软件开发和维护提供了一种当时较为有效的管理模式，根据这一模式制定开发计划、进行成本预算、组织开发人员，以阶段评审和文档控制为手段，有效地对软件开发过程进行指导，从而对稳定软件质量提供了一定程度的保证。

瀑布模型在广泛流行的同时，也暴露出一些不足。由于各阶段的顺序固定，使得前期阶段工作中造成的差错越到后期阶段所造成的损失和影响就越大，为了纠正它而花费的代价也就越高。

注意，在以下所列几点中，如果你的开发项目满足其中之一，你就要慎重考虑是否使用瀑布模型：

- 不能充分理解客户需求或客户需求有可能迅速发生变化。

- 系统太大、太复杂，不能一次做完所有的事。
- 事先拟采用的技术迅速发生变化。
- 提供的资源有限。
- 无法利用各开发阶段的某一个中间产品。

在以下情况下可以使用瀑布模型：

- 系统所有的功能、性能要求客户可以一次性准确交付。
- 是首次开发的新系统并且淘汰全部老系统。

2. 渐增模型

渐增模型（incremental model）的开发活动是由一组有计划的、循环渐增的、不断改进的过程版本组成。第一个中间版本先纳入一部分需求，下一个中间版本将增加更多的需求，依此类推，逐渐增加，直到系统完成。每个中间版本都要执行必要的过程、活动和任务。渐增模型的例子如图 1-1 所示。渐增模型也称为有计划的产品改进模型。

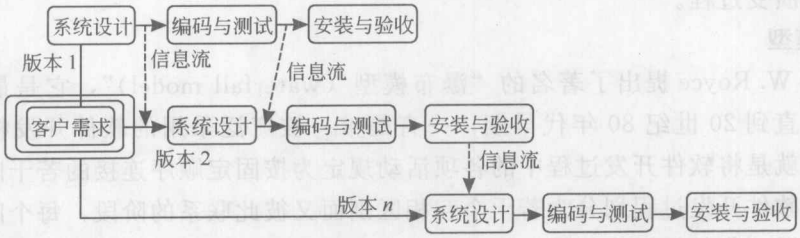


图 1-1 渐增模型

渐增模型在开发过程中的各个中间版本可以并发开发，其过程、活动和任务可以在各中间版本间平行使用。

注意，如果在你的开发项目中出现任何一种情况满足下列几点之一者，你就要慎重考虑是否使用渐增模型：

- 不能充分理解客户需求或客户需求有可能迅速发生变化。
- 事先拟采用的技术迅速发生变化。
- 客户突然提出一些新的功能需求。
- 长时期内仅有有限的资源保证（开发人员和资金）。

在以下情况下可以使用渐增模型：

- 需要在尽短的时间内得到系统基本功能的演示或使用。
- 各版本都有中间阶段产品可提供使用。
- 系统可以被自然地分割成渐增的模式。
- 开发人员与资金可以逐步增加。

3. 演化模型

演化模型（evolutionary model）也称为原型（prototype）法模型，对于事先不能完整定义需求的软件项目的开发可以使用演化模型。演化模型可以减少由于需求不明给开发工作带来的风险。为了避免开发项目返工，人们根据客户给出的基本需求，对待开发软件进行首次试验性开发，目的是探索其可行性，弄清需求，获得有用的反馈信息，以支持软件的最终设计和实现。通常把首次试验性开发出的软件称为原型。演化模型有以下几种形式：