第 1 章　建模基础知识

近年来，人们对环境工程、生物工程、软件工程等给予了极大的关注，许多高等院校都增设了这些专业，以培养研究这些新兴学科的专业人才。然而，由于人们对这些新兴工程学科认识不足，还处在一个艰难的探索阶段，如软件工程，因此国内许多软件企业对软件工程还不够重视，事实上，软件工程在计算机的发展和应用中的地位非常显著，它对软件产业的形成和发展起着定性的推动作用，是现代信息产业的巨大支柱。本章将对软件工程的目标、过程和原则，以及 UML 等方面的问题和基本概念进行简要介绍，以便读者在进入软件建模之前，对软件工程的一些基本知识有一定的理解。

1.1 软件工程概述

软件工程是在 20 世纪 60 年代末期提出的。这一概念的提出，其目的是倡导以工程的原理、原则和方法进行软件开发，以期解决当时出现的“软件危机”。

1.1.1 软件工程的产生

软件工程（Software Engineering）是一门指导计算机软件系统开发和维护的工程学科，涉及到计算机科学、工程管理科学、数学等多学科，其研究范围广泛，主要研究如何应用软件开发的科学理论和工程技术来指导大型软件系统的开发。例如，现代操作系统的开发，如果不采用软件工程的方法是不可能实现的。

在我国加入 WTO 后，大力推广、应用软件工程的开发技术及管理技术，提高软件工程的应用水平，对促进我国软件产业与国际接轨，推动我国软件产业的迅速发展起到了关键作用。

软件工程的产生和发展是与软件的发展紧密相关的。软件是计算机系统中与硬件相互依存的一部分，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。

自 20 世纪 40 年代中期出现了世界上第一台计算机以后，就有了程序的概念，开始了软件的开发，其后经历了几十年的发展，计算机软件经历了三个发展阶段。

- 程序设计时代（1946 年 ~ 1956 年）：采用“个体生产方式”，即软件开发完全依赖于程序员个人的能力水平。

- 程序系统时代（1956 年～1968 年）：由于软件应用范围及规模的不断扩大，个体生产已经不能够满足软件生产的需要，一个软件需要由几个人协同完成。在该阶段的后期，随着软件需求量、规模及复杂度的增大，生产作坊的方式已经不能够适应软件生产的需要，出现了所谓的软件危机。
- 软件工程时代（1968 年至今）：该阶段的主要任务是为了克服软件危机、适应软件发展的需要，而采用工程化的生产方式。

20 世纪 60 年代中期以后，需要开发一些大型软件系统。然而软件技术的进步一直未能满足形势发展的需要，在大型软件的开发过程中出现了复杂程度高、研制周期长、正确性难以保证的三大难题。遇到的问题找不到解决办法，致使问题堆积起来，形成了人们难以控制的局面，出现了所谓的“软件危机”。

软件危机（Software Crisis）是指在软件开发和维护中所产生的一系列严重的问题：一是如何开发新的软件，满足用户对软件的需求；二是如何维护数量众多的已有软件。其主要表现如下：

- 用户需求不明确、变更过多。
- 软件成本日益增长。
- 开发进度难以控制。
- 软件质量差。
- 软件维护困难。

软件危机的出现是由于软件的规模越来越大，复杂度不断增加，软件需求量增大。而软件开发过程是一种高密集度的脑力劳动，软件开发的模式及技术不能适应软件发展的需要，致使大量质量低劣的软件涌向市场，有的软件花费了大量的人力物力，却在开发过程中就夭折。例如：IBM 公司的 OS/360 操作系统由 5000 人开发，经费达数亿美元，但软件产品的每个版本均有 1000 多个大大小小的错误，系统根本无法正常运行。

到了 20 世纪 60 年代末期，软件危机已相当严重，这促使计算机领域的科学家们开始探索缓解软件危机的方法。他们提出了“软件工程”的概念，即利用现代工程的原理、技术和方法进行软件的开发、管理、维护和更新。于是，开创了计算机科学技术的一个新的研究领域。1968 年，北大西洋公约组织召开计算机科学会议，由 Fritz Bauer 首次提出了“软件工程”的概念。

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。它应用计算机科学、数学及管理科学等原理，借鉴传统工程的原则和方法创建软件以达到提高质量、降低成本的目的。其中，计算机科学和数学用于构造模型与算法；管理科学用于计划、资源、质量和成本等的管理。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。

1.1.2 软件工程的内容和基本原理

软件工程研究的主要内容是软件开发技术和软件开发管理两个方面。在软件开发技术中，主要研究软件工程方法、软件工程过程、软件开发工具和环境。

- 软件工程方法为软件开发提供了“如何做”的技术。它包括了多方面的任务，如项目计划与估算、软件系统需求分析、数据结构、系统总体结构的设计、算法的设计、编码、测试以及维护等。软件工程方法常采用某种特殊的语言或图形的表达方法，以及一套质量保证标准。
- 软件工程过程是将软件工程的方法和工具综合起来以达到合理、及时地进行计算机软件开发的目的。软件工程过程定义了软件工程方法的使用顺序、要求交付的文档资料，为保证质量和协调变化所需要的管理及软件开发各个阶段完成的里程碑。
- 软件开发工具和环境为软件工程方法提供了自动的或半自动的软件支撑环境。目前，人们已经开发出了许多软件工具来支持上述的软件工程方法，而且已经有人把诸多软件工具集成起来，使得一种工具产生的信息可以被其他的工具所使用，从而建立起一种称为计算机辅助软件工程（CASE）的软件开发支撑系统。CASE 将各种软件工具、开发机器和一个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。

对于软件工程基本原理，早在 1983 年，B.W.Boehm 归纳提出了 7 条基本原则。

1．利用分阶段的生命周期计划严格管理

有人统计过，在不成功的软件项目中有 50%左右是由于计划不周造成的。应该把软件生命周期划分为若干阶段，并制定出相应的切实可行的计划，严格按照计划对开发和维护进行管理。B.W.Boehm 认为，应制定和严格执行 6 类计划：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。

2．坚持进行阶段评审

设计的错误占软件错误的 63%，编码错误只占 37%，而且在后期纠正错误的代价非常高，因此，必须严格坚持阶段评审，及早发现和纠正错误。

3．实行严格的产品质量控制

在现实中由于外部原因或要求对需求等进行修改是难免的，但必须有严格的管理制度和措施。

4．采用现代程序设计技术和软件工程技术

多年来，人们一直致力于研究新的程序设计技术，例如结构化程序分析（Structured Analysis）和结构化设计（Structured Design）等。

5．结果应能清楚地审查

由于软件是一种看不见摸不着的逻辑产品，对它的检验和审查很困难。因此，应提供可视化的检验标准和方法。

6．开发人员应少而精

软件开发小组的人员素质应该较高，人员不宜过多。人员素质低和人员过多，都会导致软件的错误率高，且开发效率下降，成本增加。

7．承认不断改进软件工程的必要性

软件工程是一门不断迅速发展的学科，必须学习和跟踪先进的技术和方法，也要不断总结经验、改进方法，要不断进行技术创新。

1.1.3 现代软件工程

软件不是纯物化的东西，其中包含着人的因素，于是就有很多变动的东西，不可能像理想的物质生产过程一般，基于物理学的原理来做。早期的软件开发仅考虑人的因素，传统的软件工程强调物性的规律，现代软件工程强调的是人和物的关系，即人和机器（工具、自动化）在不同层次的不断循环发展中的关系。

面向对象的分析、设计方法（OOA 和 OOD）的出现使传统的开发方法发生了翻天覆地的变化。随之而来的是面向对象建模语言（以 UML 为代表）、软件复用、基于组件的软件开发等新的方法和领域。

与之相应的是从企业管理的角度提出的软件过程管理，即关注于软件生存周期中所实施的一系列活动，并通过过程度量、过程评价和过程改进等涉及对所建立的软件过程及其实例进行不断优化的活动，使得软件过程循环往复，螺旋上升式地发展。其中最著名的软件过程成熟度模型是美国卡内基梅隆大学软件工程研究所（SEI）建立的 CMM（Capability Maturity Model），即能力成熟度模型。此模型在建立和发展之初，主要目的是为大型软件项目的招投标活动提供一种全面而客观的评审依据，而发展到后来，又同时被应用于许多软件机构内部的过程改进活动中。

1.2 建模概述

建模是所有建造优质软件活动中的中心环节，也是软件成功的一个基本因素，它在软件开发中是不可或缺的。

1.2.1 什么是模型

模型就是对现实客观世界的形状或状态的抽象模拟和简化。模型提供了系统的骨架（Sketch）和蓝图（Blueprint）。模型给人们展示系统的各个部分是如何组织起来的，模型既

可以包括详细的计划，也可以包括从很高的层次考虑系统的总体计划。一个好的模型包括那些有广泛影响的主要元素，而忽略那些与给定的抽象水平不相关的次要元素。每个系统都可以从不同的方面用不同的模型来描述，因而每个模型都是一个在语义上闭合的系统抽象。模型可以是结构性的，强调系统的组织，也可以是行为性的，强调系统的动态方面。对象建模的目标就是要为正在开发的系统制定一个精确、简明和易理解的面向对象模型。

通常，开发一个计算机系统是为了解决某个领域的特定问题。问题的求解过程就是从领域问题到计算机系统的映射。软件系统的模型用建模语言来表达，如 UML。模型包含语义信息和表示法，可以采用图形和文字等多种不同形式。建立模型是因为在某些用途中模型使用起来比操作实物更容易和方便。

模型主要包含两个方面：语义方面的信息（语义）和可视化的表达方法（表示法）。

- 语义方面用一套逻辑组件表达应用系统的含义，如类、关联、状态、用例和消息。语义模型元素携带了模型的含义，即它们表达了语义，用于代码生成、有效性验证、复杂度的度量等，其可视化的外观与大多数处理模型的工具无关。语义信息通常称做模型。一个语义模型具有一个词法结构、一套高度形式化的规则和动态执行结构。这些方面通常分别加以描述，但它们紧密相关，并且是同一模型的一部分。
- 可视化的表达方式以可供人观察、浏览和编辑的形式展示语义信息。表达式元素携带了模型的可视化表达方式，即语义是用一种可被人直接理解的方式来表达的。它们并未增添新的语义，但用一种有用的方式对表达式加以组织，以强调模型的排列，因此它们对模型的理解起指导作用。表达式元素的语义来自于语义模型元素，但是，由于是由人来绘制模型图，所以表达式元素并不是完全来自于模型的逻辑元素。表达式元素的排列可以表达出语义关系的其他含义。

UML 作为一种可视化的建模语言，提供了丰富的基于面向对象概念的模型元素及其图形表示元素。

1.2.2 建模的原理

选择创建什么模型很重要，因为模型要反映难以处理的开发问题。建模的原理有以下 10 个：

- 选择建立什么样的模型对如何发现和解决问题具有重要的影响，正确的模型有助于提高开发者的洞察力。
- 模型要在不同的精度级别上来表示。
- 可以根据观察的角色和观察的原因来选择精度。
- 建造模型要和现实相连。
- 重要的系统需要用一组独立的模型去处理。在面向对象的软件体系中，为了理解系统的体系结构，需要几个互补和连锁的视图：用例图、设计视图、进程视图、实现视图和实施视图。
- 构造模型的基本技术是抽象，应突出与问题有关的特征，将与问题无关的性质略去。

- 不必追求绝对的真实和完整，只需从期望的角度看其是否充分。
- 应当刻画问题的关键方面，略去相对次要的因素。
- 建模语言应支持人们由模糊到清晰、由粗到细逐渐完善的认识过程。
- 应采用可视化的图形建模语言。

1.2.3 为什么要建模

使用模型建模不仅仅适用于建筑行业，例如，城市在进行规划的时候通常都有自己的规划模型等。如果不首先构造这些模型就进行城市的建设，那简直是难以想象的。一些电气设备，例如，ATM 机也需要一定程度的建模，以便更好地理解系统。在社会学、经济学和商业管理领域也需要建模，以证实理论的正确性。

建模是为了能够更好地理解正在开发的系统。通过建模，要达到以下 4 个目的：

- 模型有助于按照实际情况或按照所需要的样式对系统进行可视化。
- 模型能够规约系统的结构或行为。
- 模型给出了指导构造系统的模板。
- 模型可对做出的决策进行文档化。

那么具体到软件所涉及到的人员，包括系统用户、软件开发团队、软件的维护者和技术支持者，系统建模都有什么样的作用呢？

- 对于软件系统用户，软件的开发模型向他们描述了软件开发者对于软件系统需求的理解。让系统用户查看软件对象模型并且找到其中的问题，这样可以使用户开发者不至于从一开始就发生错误。需求分析阶段的错误将会导致大量的修复成本，让系统用户一开始就指出一些需求错误并修正它们，能够在很大程度上节约这些成本。
- 对于软件开发团队，软件的对象模型有助于帮助他们对软件的需求以及系统的架构和功能进行沟通。需求和架构的一致理解对于软件开发团队是非常重要的，可以减少不必要的麻烦。软件对象建模的受益者不仅仅包括代码的编写者，还包括软件的测试者和文档的编写者。
- 对于软件的维护者和技术支持者，在软件系统开始运行后的相当长的一段时间内，软件的对象模型能够帮助他们理解程序的架构和功能，迅速地对软件所出现的问题进行修复。

建模并不仅仅针对的是大型的软件系统，甚至一个小型的电话簿软件也能从建模过程中受益。事实上，系统越大、越复杂，建模的重要性就越大，一个很简单的原因就是：人对复杂问题的理解能力是有限的，人们往往不能完整地理解一个复杂的系统，所以要对它建模。

通过建模，可以缩小所研究问题的范围，一次只需要重点研究它的一个很小的方面，这就是“分而治之”的策略方法，即把一个困难问题划分成一系列能够解决的小问题，对这些小问题的解决也就构成了对复杂问题的解决。一个适当选择的模型可以使建模人员在较高的抽象层次上工作。

1.2.4 建模的目标和原则

建立模型可以帮助开发者更好地了解正在开发的系统。通过建模，要实现以下 4 个目标：

- 便于开发人员展现系统。
- 允许开发人员制定系统的结构或行为。
- 提供指导开发人员构造系统的模板。
- 记录开发人员的决策。

在工程学科中，对模型的使用有着悠久的历史，人们从中总结出了 4 条基本的建模原则。

- 选择要创建什么模型，对如何动手解决问题和如何形成解决方案有着意义深远的影响。换句话说，就是要认真地选择模型。正确的模型将清楚地表明最棘手的开发问题，提供不能轻易地从别处获得的洞察力；错误的模型将使人误入歧途，把精力花在不相关的问题上。
- 可以在不同的精度级别上表示每一种模型。如果正在建造一座大厦，有时需要从宏观上让投资者看到大厦的样子，感觉到大厦的总体效果。而有时又需要认真考虑细节问题，例如，对复杂管道的铺设，或对少见的结构件的安装等。无论如何，使用者的身份和使用的原因是评判模型优劣的关键。分析者和最终用户关心的是“这是什么”，而开发者关心的是“如何实现”。所有的参与者都想在不同的时期、从不同的角度了解系统。
- 最好的模型是与现实相联系的。如果建筑的物理模型不能以与真实的建筑相同的方式做出反应，则它的价值是很有限的；飞机的数学模型，如果只是假定了理想条件和完美制造，则可能掩盖真实飞机的一些潜在的、致命的现实特征。最好是有能够清晰地联系实际的模型，而当联系很薄弱时能够精确地知道这些模型怎样与现实脱节。所有的模型都对现实进行了简化，但有一点要记住，简化不要掩盖任何重要的细节。
- 孤立的模型是不完整的。每个重要的系统都是由多个几乎独立的模型组合出来的。如果正在建造一所建筑物，会发现没有任何一套单项设计图能够描述该建筑的所有细节。至少需要楼层平面图、立面图、电气设计图、采暖设计图和管道设计图。并且，在任何种类的模型中都需要从多视角来把握系统的范围（例如不同楼层的蓝图）。既能够单独地研究电气设计图，也能看到它如何映射到楼层平面图中，以及它与管道设计图中管道排布的相互影响。

1.3 UML 概述

UML 的目标是以面向对象的图形方式来描述任何类型的系统，其应用领域非常广泛。其中最常用的是建立软件系统的模型，但它同样可以用于描述非软件领域的系统，如机械系统、企业机构或业务过程，以及处理复杂数据的信息系统、具有实时要求的工业系统或工业

过程等。总之，UML 是一个通用的标准建模语言，可以对任何具有静态结构和动态行为的系统进行建模。

1.3.1 UML 的起源和发展

在面向对象建模上，被公认的面向对象建模语言最早出现于 20 世纪 70 年代中期。在面向对象建模的竞争过程中，最繁盛的时期是 1989 年~1994 年，在这短短的几年时间内，面向对象建模语言的数量从不到十种增加到了五十多种。从 20 世纪 90 年代中期开始，一些比较成熟的方法受到了学术界与工业界的推崇和支持，其中最具代表性的是 Booch 1993、OOSE 和 OMT-2 等，它们是当时影响最大的几种面向对象方法论。

尽管这些面向对象方法都比较优秀，但是不同程度和不同领域的开发人员却无法鉴别这些面向对象开发方法的长处，为了能够让不同程度和不同开发领域的开发人员能够进行很好的沟通，并交流他们在开发各种系统的过程中积累的经验和成果，业内研究人员和众多的厂商都开始意识到有必要对这些已经存在的并且比较好的方法进行充分分析，汲取众长，创建一种统一的建模语言。

统一的建模语言的创建开始于 1994 年 10 月，Grady Booch 和 Jim Rumbaugh 首先致力于这一工作的研究，他们将 Booch 93 和 OMT-2 统一起来，并于 1995 年 10 月发布了第一个公开版本，称之为统一方法 UM 0.8（Unfitied Method）。

而后，面向对象软件工程（Object-Oriented Software Engineer，OOSE）方法的创始人 Ivar Jacobson 也加入到这个队伍中，并且带来了其在 OOSE 方法中的成果。经过 Grady Booch、Jim Rumbaugh 和 Ivar Jacobson 三人的共同努力，于 1996 年 6 月和 10 月分别发布了两个新的 UML 版本，即 UML 0.9 和 UML 0.91，并且正式将 UM 重新命名为 UML（Unified Modeling Language）。

1996 年，一些机构将 UML 作为其商业策略已日趋明显。UML 的开发者得到了来自公众的正面反应，并倡议成立了 UML 成员协会，以完善、加强和促进 UML 的定义工作。当时的成员有 DEC、HP、I-Logix、Itellicorp、IBM、ICON Computing、MCI Systemhouse、Microsoft、Oracle、Rational Software、TI 以及 Unisys 等 700 多家公司。这些公司表示支持采用 UML 作为其标准建模语言。这一机构对 UML 1.0（发布于 1997 年 1 月）及 UML 1.1（发布于 1997 年 11 月 17 日）的定义和发布起到了重要的促进作用。1997 年 11 月 17 日，对象管理组织（OMG）开始采纳 UML 为其标准建模语言，成为业界的标准。从此，UML 的相关发布、推广等工作交由 OMG 负责。

至此，UML 作为一种定义良好、易于表达、功能强大且普遍适用的建模语言，溶入了软件工程领域的新思想、新方法和新技术，成为面向对象技术学习中不可缺少的一部分。UML 的作用不仅在于支持面向对象的分析与设计，还支持从需求分析开始的软件开发的全过程。

从 UML 纳入到 OMG 开始，OMG 对于 UML 的修订工作就从来没有停止过，产生了 UML 1.2、UML 1.3、UML 1.4 和 UML 2.0 版本，目前，该组织正在为 UML 2.x 版本努力。

目前，许多的软件工具开发厂商在自己的产品中支持或计划支持 UML 标准。许多的软件工程方法学家也正在使用 UML 的表示法进行以后的研究工作。UML 的出现深受计算机界

的欢迎，因为它集中了许多专家的经验，减少了各种软件开发工具之间无谓的分歧。

1.3.2 UML 的主要特点

标准建模语言 UML 的主要特点可以归纳为以下三点：

- UML 统一了 Booch、OMT 和 OOSE 等方法中的基本概念。
- UML 吸取了面向对象技术领域中其他流派的长处，其中也包括非 OO 方法的影响。UML 符号考虑到了各种方法的图形表示，删掉了大量易引起混乱的、多余的和极少使用的符号，也添加了一些新符号。因此，在 UML 中汇入了面向对象领域中很多人的思想。这些思想并不是 UML 的开发者们发明的，而是开发者们依据最优秀的 OO 方法和丰富的计算机科学实践经验综合提炼而成的。
- UML 在演变过程中还提出了一些新的概念。在 UML 标准中添加了模板（Stereotypes）、职责（Responsibilities）、扩展机制（Extensibility mechanisms）、线程（Threads）、过程（Processes）、分布式（Distribution）、并发（Concurrency）、模式（Patterns）、合作（Collaborations）、活动图（Activity Diagram）等新概念，并清晰地区分类型（Type）、类（Class）和实例（Instance）、细化（Refinement）、接口（Interfaces）和组件（Components）等概念。

因此，可以认为，UML 是一种先进实用的标准建模语言，但其中某些概念尚待实践来验证，UML 也必然存在一个进化过程。

1.3.3 UML 的应用领域

UML 的目标是以面向对象图的方式来描述任何类型的系统，具有很宽的应用领域。其中最常用的是建立软件系统的模型，但它同样可以用于描述非软件领域的系统。总之，UML 是一个通用的标准建模语言，可以对任何具有静态结构和动态行为的系统进行建模。

此外，UML 适用于系统开发过程中从需求规格描述到系统完成后测试的不同阶段。在需求分析阶段，可以用用例来捕获用户需求。通过用例建模，描述对系统感兴趣的外部角色及其对系统（用例）的功能要求。分析阶段主要关心问题域中的主要概念（如抽象、类和对象等）和机制，需要识别这些类以及它们相互间的关系，并用 UML 类图来描述。为实现用例，类之间需要协作，这可以用 UML 动态模型来描述。在分析阶段，只对问题域的对象（现实世界的概念）建模，而不考虑定义软件系统中技术细节的类（如处理用户接口、数据库、通信和并行性等问题的类）。这些技术细节将在设计阶段引入，因此设计阶段为构造阶段提供更详细的规格说明。

UML 模型还可作为测试阶段的依据。系统通常需要经过单元测试、集成测试、系统测试和验收测试。不同的测试小组使用不同的 UML 图作为测试依据：单元测试使用类图和类规格说明；集成测试使用部件图和合作图说明；系统测试使用用例图来验证系统的行为；验收测试由用户进行，以验证系统测试的结果是否满足在分析阶段确定的需求。

总之，标准建模语言 UML 适用于以面向对象技术来描述类型的系统，而且适用于系统开发的不同阶段，包括从需求规格描述直至系统完成后的测试和维护。

1.3.4 UML 的模型种类

软件有孕育、诞生、成长、成熟和衰亡的生存过程，我们称为“软件生命周期”。软件生命周期可以分为多个阶段，即制定计划、需求分析、设计、编码、测试、运行和维护。软件开发可以采用多种途径进行。软件开发模式是跨越整个软件生存周期的系统开发，并运行和维护所实施的全部内容的结构框架，它给出了软件开发活动各个阶段之间的关系。软件项目可以遵循不同类型的开发过程，目前，可以将常见的软件开发模式大致分为如下几种类型。

- 在第一代软件开发过程模式中，软件需求是要求完全确定的，如瀑布模型。这类开发模式的特点是软件需求在开发阶段已经基本确定，软件生命周期的各项活动依顺序固定，软件开发阶段性进行。其缺点是如果在开发后期要改正早期已经存在的问题需要付出昂贵的代价，从用户角度来讲，开发需要等待较长时间才能够看到软件产品，这样大大增加了软件开发的风险系数。并且，如果在开发过程中存在开发瓶颈，则将严重影响开发效率。根据开发的瓶颈制定开发计划，成为第一代软件开发过程不可缺少的一部分。
- 对于第一代软件开发过程模式的改进诞生了在开始阶段只提供基本需求的渐进式开发模型，如喷泉模型和演化模型等。这类开发模型的特点是在软件开发的开始阶段只需要提供基本的需求，软件开发过程的各个活动是迭代的，所以也被称为迭代式开发。通过迭代过程实现软件的逐步演化，最终得到软件产品。在此引入了风险管理，采取早期预防措施，增加项目成功几率，提高软件质量。其缺点是由于在开始阶段需求的不完全性，对于软件的总体设计带来了困难，从而也削弱了产品设计的完整性，这样就对风险技能管理水平提出了很大的挑战。
- 以体系结构为基础或基于构件的开发模型，如基于构件的开发模型和基于体系结构的开发模型等。这类模型的特点是首先利用获取的需求分析结果设计出软件的总体结构，然后通过基于构件的组装方法来构造软件系统。这样软件体系结构的出现使得软件的结构框架更清晰，有利于系统的设计、开发和维护。
- 轻量级的开发模型，这种开发模型强调适应性而非预测性，强调以人为中心，而不以流程为中心，以及对变化的适应和对人性的关注，其特点是轻载、基于时间、紧凑、并行并基于构件的软件过程。在所有的敏捷方法中，XP（eXtreme Programming）是一种最引人注目的轻型开发方法。

下面简单地对每种类型的代表，即瀑布模型、螺旋模型、基于构件的开发模型、XP（eXtreme Programming）方法等软件开发模型进行分析。

1. 瀑布模型

瀑布模型也被称为“生存周期模型”，其核心思想是按照相应的工序将问题进行简化，

将系统功能的实现与系统的设计工作分开，便于项目之间的分工与协作，即采用结构化的分析与设计方法将逻辑实现与物理实现分开。瀑布模型将软件生命周期划分为软件计划、需求分析、软件设计、软件实现、软件测试、软件运行和维护这 6 个阶段，并且规定了它们自上而下的次序，如同瀑布一样下落。每一个阶段都是依次衔接的。采用瀑布模型的软件开发过程如图 1-1 所示。

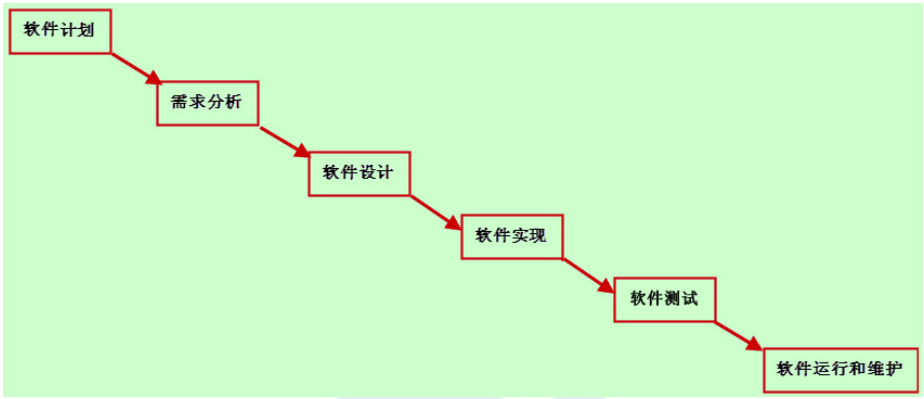

图 1-1 采用瀑布模型的软件过程

瀑布模型是最早出现的软件开发模型，在软件工程中占有重要的地位，它提供了软件开发的基本框架。其过程是：从上一项活动接收该项活动的工作对象作为输入，利用这一输入实施该项活动应完成的内容，给出该项活动的工作成果，并作为输出传给下一项活动。同时评审该项活动的实施，若确认，则继续下一项活动；否则返回前面，甚至更前面的活动。

瀑布模型为项目提供了按阶段划分的检查点，这样有利于软件开发过程中人员的组织及管理。瀑布模型让你在完成当前阶段后，才去关注后续阶段，这样有利于开发大型项目，然而软件开发的实践表明，瀑布模型也存在一定的缺陷。

- 只有在项目生命周期的后期才能看到结果：由于开发模型呈线性，所以当开发成果尚未经过测试时，用户是无法看到软件效果的，不能在开发过程中得到及时反馈，增加了项目开发过程的风险。在软件开发前期未发现的错误传到后面的开发活动中，进而可能会造成整个软件项目开发失败。
- 通过过多的强制完成日期和里程碑来跟踪各个项目阶段：在每个项目开发阶段，瀑布模型是通过强制固定的完成日期和里程碑进行项目跟踪的，在项目开发过程中缺乏足够的灵活性，特别是对于需求不稳定的项目更加麻烦。
- 在软件需求分析阶段，要完全确定系统用户的所有需求是一件比较困难的事情，甚至可以说是不太可能的。

尽管瀑布模型存在一定的缺陷，但是它对很多类型的项目而言依然是有效的，特别是一些大型的项目。如果使用正确，可以节省大量的时间和金钱。对于所开发的项目而言，是否使用这一模型主要取决于能否理解客户的需求以及在项目的进程中这些需求的变化程度，对于能够在前期进行确定的需求分析项目，瀑布模型还是具有一定价值的。

2. 喷泉模型

喷泉模型是一种以对象为驱动、以用户需求为动力的模型，主要用于描述面向对象的软件开发过程。该模型认为软件开发过程自下而上的各阶段是相互重叠和多次反复的，就像水喷上去又可以落下来，类似一个喷泉。各个开发阶段没有特定的次序要求，并且可以交互进行，可以在某个开发阶段中随时补充其他任何开发阶段中的遗漏。采用喷泉模型的软件开发过程如图 1-2 所示。

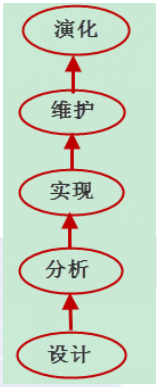

图 1-2　采用喷泉模型的软件过程

喷泉模型主要用于面向对象的软件项目，软件的某个部分通常被重复多次，相关对象在每次迭代中随之加入渐进的软件成分。各活动之间无明显边界，例如设计和实现之间没有明显的边界，这也称为“喷泉模型的无间隙性”。

喷泉模型不像瀑布模型那样，需要在分析活动结束后才开始设计活动，设计活动结束后才开始编码活动。该模型的各个阶段没有明显的界限，开发人员可以同步开发。其优点是可以提高软件项目的开发效率，节省开发时间，适应于面向对象的软件开发过程。由于喷泉模型在各个开发阶段是重叠的，因此在开发过程中需要大量的开发人员，不利于项目的管理。此外这种模型要求严格管理文档，使得审核的难度加大，尤其是在面对可能随时加入各种信息、需求与资料的情况下。

3. 基于构件的开发模型

基于构件的开发模型利用模块化方法将整个系统模块化，并在一定构件模型的支持下复用构件库中的一个或多个软件构件，通过组合手段高效率、高质量地构造应用软件系统。基于构件的开发模型融合了螺旋模型的许多特征，由软件的需求分析和定义、体系结构设计、构件库建立、应用软件构建、测试和发布 5 个阶段组成，采用这种开发模型的软件过程如图 1-3 所示。

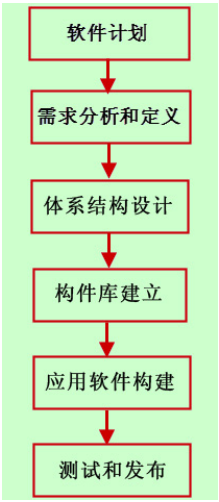

图 1-3　采用基于构件的开发模型的软件过程

构件作为重要的软件技术和工具得到了极大的发展，这些新技术和工具有 Microsoft 的 DCOM、Sun 的 EJB，以及 OMG 的 CORBA 等。基于构件的开发活动从标识候选构件开始，通过搜查已有构件库，确认所需要的构件是否已经存在。如果已经存在，则从构件库中提取出来复用；否则采用面向对象方法开发它。之后利用提取出来的构件通过语法和语义检查将这些构件通过胶合代码组装到一起实现系统，这个过程是迭代的。

基于构件的开发方法使得软件开发不再一切从头开发，开发的过程就是构件组装的过程，维护的过程就是构件升级、替换和扩充的过程。其优点是构件组装模型导致了软件的复用，提高了软件开发的效率。构件可由一方定义其规格说明，被另一方实现，然后供给第三方使用，构件组装模型允许多个项目同时开发，降低了费用，提高了可维护性，可分步提交软件产品。

但由于该模型采用自定义的组装结构标准，缺乏通用的组装结构标准，因而引入了较大的风险。可重用性和软件高效性不易协调，需要精干的有经验的分析和开发人员，并且由于过分依赖于构件，所以构件库的质量影响着产品质量。

4. XP 方法

敏捷方法是近几年兴起的一种轻量级的开发方法，它强调适应性而非预测性，强调以人为中心，而不以流程为中心，以及对变化的适应和对人性的关注，其特点是轻载、基于时间、紧凑、并行并基于构件的软件过程。在所有的敏捷方法中，XP（eXtreme Programming）是一种最引人注目的轻型开发方法。它规定了一组核心价值和方法，消除了大多数重量型开发过程中的不必要产物，建立了一个渐进型开发过程。该方法将开发阶段的 4 个活动（分析、设计、编码和测试）混合在一起，在全过程中采用迭代增量开发、反馈修正和反复测试的方法。它把软件生命周期划分为用户场景、系统架构、发布计划、迭代、验证测试和小型发布共 6 个阶段，采用这种开发模型的软件过程如图 1-4 所示。

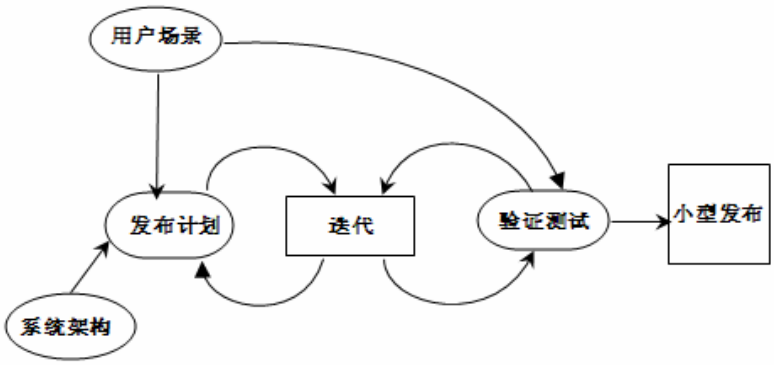

图 1-4　采用 XP 方法的软件过程

XP 模型通过对传统软件开发的标准方法进行重新审视，提出了由一组规则组成的简便易行的过程。由于这些规则是通过在实践中观察使软件高效或缓慢的因素而得出的，因此它既考虑了保持开发人员的活力和创造性，又考虑了开发过程的有组织、有重点和持续性。XP 模型是面向客户的开发模型，重点强调用户的满意程度。开发过程中对需求改变的适应能力较高，即使在开发的后期，也可较高程度地适应用户的改变。

XP 开发模型与传统模型相比具有很大的不同，其核心思想是交流（Communication）、简单（Simplicity）、反馈（Feedback）和进取（Aggressiveness）：XP 开发小组不仅包括开发人员，还包括管理人员和客户，该模型强调小组内成员之间要经常进行交流；在尽量保证质量的前提下力求过程和代码的简单化；来自客户、开发人员和最终用户的具体反馈意见可以提供更多的机会来调整设计，保证把握正确的开发方向；进取则包含于上述 3 个原则中。

XP 开发方法中有许多新思路，如采用“用户场景”代替传统模型中的需求分析。“用户场景”是由用户用自己领域中的词汇并且不考虑任何技术细节准确地表达自己的需求。XP 模型的优点如下：

- 采用简单计划策略，不需要长期计划和复杂模型，开发周期短。
- 全过程采用迭代增量开发、反馈修正和反复测试的方法，软件质量有保证。
- 能够适应用户经常变化的需求，提供用户满意的高质量软件。

上面的开发模型或方法或许不能一概而论地说是以面向对象的建模为基础的开发模式，但是在各种开发方法中，都包含了软件的需求分析、软件的设计、软件的开发、软件的测试和软件的部署。在每一个阶段，都可以借助于面向对象的建模和这些开发模型形成一套适合自己或企业的开发方式，主要体现在以下几个方面。

- 在软件的需求分析阶段，对系统将要面临的具体管理问题以及用户对系统开发的需求进行调查研究，即先弄清要干什么的问题，然后分析问题的性质和求解问题，在繁杂的问题域中抽象地识别出对象以及其行为、结构、属性、方法等。一般称之为面向对象的分析，即 OOA。
- 在软件的设计阶段整理问题，对分析的结果作进一步的抽象、归类、整理，并最终

以范式的形式将它们确定下来。一般称之为面向对象的设计，即 OOD。

- 在软件的开发阶段，即程序实现阶段，用面向对象的程序设计语言将上一步整理的范式直接映射（即直接用程序设计语言来取代）为应用软件。一般称之为面向对象的程序，即 OOP。

1.4 扩展练习

1. 简述建模的重要性、目标以及原则。
2. UML 是什么？它的主要特点和应用领域在哪里？
3. 简述软件开发的生命周期。
4. 常见的软件开发模式有哪几种？
5. 简述软件工程的内容和基本原理。