1 软件工程概述

软件工程是研究软件开发技术和软件项目管理的一门工程学科，从工程化的角度来指导软件开发、测试和项目管理等活动。

本章主要讲授软件工程的基本概念和知识，内容包括软件的概念和特点、软件危机、软件工程的概念及原理、软件生存周期及其模型等，本章的内容是学习以后各章的基础。

1.1 软件的概念和特点

1.1.1 计算机系统的构成及实现

计算机系统由硬件系统及软件系统构成，硬件系统的实现靠硬件工程，软件系统的实现靠软件工程，如图 1-1 所示。

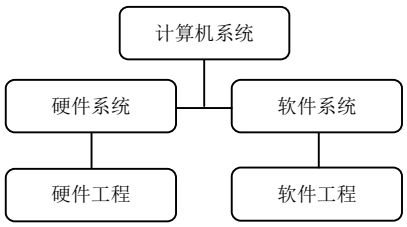

图 1-1　计算机系统的构成及实现

1.1.2 软件的概念

软件（Software）是计算机系统中与硬件相互依存的另一部分，它包括程序、数据及其相

关文档的完整集合。

软件＝程序+数据+文档

程序是为实现软件的功能和性能要求而编写的指令序列。

数据是指使程序能够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资料。

1.1.3 软件的特点

与计算机硬件相比，计算机软件具有如下特点：

（1）软件是一种逻辑实体而非物理实体，因而软件具有抽象性。

（2）软件的开发是人智力的高度发挥，而不是传统意义上的硬件制造。

（3）软件可能被废弃，但不会用坏，不存在磨损、消耗问题。

（4）软件的开发和运行常常受到计算机系统的限制，对计算机系统有着不同程度的依赖性。

（5）软件的开发至今尚未完全摆脱手工艺的开发方式，使软件的开发效率受到很大限制。

（6）软件开发是一个复杂的过程。主要表现在实际问题的复杂性和程序逻辑结构的复杂性两个方面。

（7）软件成本非常高昂。

1.1.4 计算机软件的分类

计算机软件一般分为系统软件和应用软件两大类。

1. 系统软件

系统软件为计算机使用提供最基本的功能，可分为操作系统和支撑软件，其中操作系统是最基本的软件。系统软件是负责管理计算机系统中各种独立的硬件，使得它们可以协调工作。系统软件使得计算机使用者和其他软件将计算机当作一个整体而不需要顾及到底层每个硬件是如何工作的。

操作系统是用来管理计算机硬件与软件资源的程序，同时也是计算机系统的内核与基石。操作系统身负诸如管理与配置内存、决定系统资源供需的优先次序、控制输入与输出设备、操作网络与管理文件系统等基本事务。操作系统也提供一个让使用者与系统交互的操作接口。

支撑软件是支撑各种软件的开发与维护的软件，又称为软件开发环境（SDE）。它主要包括环境数据库、各种接口软件和工具组；包括一系列基本的工具，比如编译器、数据库管理、存储器格式化、文件系统管理、用户身份验证、驱动管理、网络连接等方面的工具。著名的软件开发环境有 IBM 公司的 Web Sphere、微软公司的 Studio.NET 等。

2. 应用软件

系统软件并不针对某一特定应用领域，而应用软件则相反，不同的应用软件根据用户和

所服务的领域提供不同的功能。

应用软件是为了某种特定的用途而被开发的软件。它可以是一个特定的程序，比如一个图像浏览器；也可以是一组功能联系紧密，可以互相协作的程序的集合，比如微软的 Office 软件；也可以是一个由众多独立程序组成的庞大的软件系统，比如数据库管理系统。

1.2 软件的发展和软件危机

1.2.1 计算机软件的发展过程

软件是由计算机程序和程序设计的概念发展演化而来的，是在程序和程序设计发展到一定规模并且逐步商品化的过程中形成的。软件开发经历了程序设计阶段、软件设计阶段和软件工程阶段的演变过程。

1. 程序设计阶段

程序设计阶段出现在 1946～1955 年。此阶段的特点是：尚无软件的概念，程序设计主要围绕硬件进行开发，规模很小，工具简单，无明确分工（开发者和用户），程序设计追求节省空间和编程技巧，无文档资料（除程序清单外），主要用于科学计算。

2. 软件设计阶段

软件设计阶段出现在 1956～1970 年。此阶段的特点是：硬件环境相对稳定，出现了“软件作坊”的开发组织形式。开始广泛使用产品软件（可购买），从而建立了软件的概念。随着计算机技术的发展和计算机应用的日益普及，软件系统的规模越来越庞大，高级编程语言层出不穷，应用领域不断拓宽，开发者和用户有了明确的分工，社会对软件的需求量剧增。但软件开发技术没有重大突破，软件产品的质量不高，生产效率低下，从而导致了“软件危机”的产生。

3. 软件工程阶段

自 1970 年起，软件开发进入了软件工程阶段。由于“软件危机”的产生，迫使人们不得不研究、改变软件开发的技术手段和管理方法。从此软件发展进入了软件工程时代。此阶段的特点是：硬件已向巨型化、微型化、网络化和智能化四个方向发展，数据库技术已成熟并广泛应用，第三代、第四代语言出现；第一代软件技术——结构化程序设计在数值计算领域取得优异成绩；第二代软件技术——软件测试技术、方法、原理用于软件生产过程；第三代软件技术——处理需求定义技术用于软件需求分析和描述；第四代软件技术——使软件工程公认的模块化、信息隐蔽、抽象、局部化、软件重用等原则在面向对象机制下得到了充分的体现。

每个发展阶段都具有不同的特点，见表 1-1 所示。

1.2.2 软件危机

20 世纪 60 年代末 70 年代初，西方工业发达国家经历了一场“软件危机”。这场软件危机表现在：一方面软件十分复杂，价格昂贵，供需差日益增大，另一方面软件开发时又常常受挫，

质量差，指定的进度和完成日期很少能按时实现，研制过程很难管理，即软件的研制往往失去控制。

表 1-1　计算机软件发展的三个阶段及其特点

阶段 / 特点	程序设计	软件设计	软件工程
软件所指	程序	程序及说明书	程序+数据+文档
主要程序设计语言	汇编及机器语言	高级语言	软件语言
软件工作范围	程序编写	设计和测试	整个软件生命周期
需求者	程序设计者本人	少数用户	市场用户
开发软件的组织	个人	开发小组	开发小组及大、中型开发机构
软件规模	小型	中、小型	大、中、小型
决定质量的因素	个人技术	小组技术水平	技术与管理水平
开发技术和手段	子程序、程序库	结构化程序设计	数据库、开发工具、集成开发环境、工程化开发方法、标准和规范、网络及分布式开发、面向对象技术、计算机辅助软件工程
维护责任者	程序设计者	开发小组	专职维护人员
硬件的特征	高价、存储量小、可靠性差	降价，速度、容量和可靠性明显提高	向超高速、大容量、网络化、微型化方向发展
软件的特征	完全不受重视	软件的技术发展不能满足需求，出现软件危机	开发技术有进步，但仍未完全摆脱软件危机

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问题的现象称为软件危机。

1. 软件危机的表现

（1）经费预算经常突破，完成时间一再拖延。

（2）开发的软件不能满足用户要求。

（3）开发的软件可靠性差。

（4）开发的软件可维护性差。

2. 软件危机产生的原因

（1）软件规模越来越大，结构越来越复杂。

（2）软件开发管理困难。

（3）软件开发费用不断增加。

（4）软件开发技术落后。

（5）生产方式落后。

（6）开发工具落后，生产率提高缓慢。

3. 消除软件危机的途径

消除软件危机的途径既要有技术措施（方法、工具），又要有组织管理措施，需将两者结合起来以现代工程方法来开发软件。

（1）消除错误的观点和做法。

（2）推广使用成功的开发技术和方法。

（3）开发使用软件工具和软件工程支持环境。

（4）加强软件工程管理。

1.3 软件工程及其原理

1.3.1 软件工程的概念

软件工程是一门研究用工程化方法构建和维护有效的、实用的和高质量的软件的学科。它涉及程序设计语言、数据库、软件开发工具、系统平台、标准、设计模式等方面。在现代社会中，软件应用于多个方面。典型的软件有电子邮件、嵌入式系统、人机界面、办公套件、操作系统、编译器、数据库、游戏等。同时，各个行业几乎都有计算机软件的应用，如工业、农业、银行、航空、政府部门等。这些应用促进了经济和社会的发展，也提高了工作和生活效率。

软件工程的概念于 1968 年 NATO（North Atlantic Treaty Organization，北大西洋公约组织）在德国召开的一次会议上被首次提出。为了消除和缓解软件危机，1968 年德国软件大师鲍尔（Bauer）在北大西洋公约组织会议上提出软件工程的定义：“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”。

软件工程大师勃姆（Boehm）对软件工程的定义：“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”。

1983 年 IEEE 的软件工程定义：“软件工程是开发、运行 、维护和修复软件的系统方法”。1993 年 IEEE 的一个更加综合的定义：“将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程，即将工程化应用于软件中”。

1.3.2 软件工程的要素

软件工程有三个基本要素，包括方法、工具和过程。

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、算法过程的设计、编码、测试以及维护等。

软件工具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，已经推出了许

多软件工具，这些软件工具集成起来，形成一个计算机辅助软件工程（CASE）的软件开发支撑环境。

软件工程过程则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。过程定义了方法使用的顺序、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。

1.3.3 软件工程的目标

软件工程的目标是“以较少的投资获得高质量的软件”，具体包括：

（1）付出较低的开发成本。

（2）达到要求的软件功能。

（3）取得较好的软件性能。

（4）开发的软件易于移植。

（5）需要较低的维护费用。

（6）按时完成开发工作，及时交付使用。

软件工程的不同目标之间是互相影响和互相牵制的，有些是互补关系，有些是互斥关系，如图 1-2 所示。例如，提高软件生产率有利于降低软件开发成本，但过分追求高生产率和低成本便无法保证软件的质量，容易使人急功近利，留下隐患。但是，片面强调高质量使得开发周期过长或开发成本过高，由于错过了良好的市场时机，也会导致所开发的产品失败。因此，我们需要采用先进的软件工程方法，按照目标的重要性确定其优先级并进行适当地折衷，使质量、成本和生产率三者之间的关系达到最优的平衡状态。

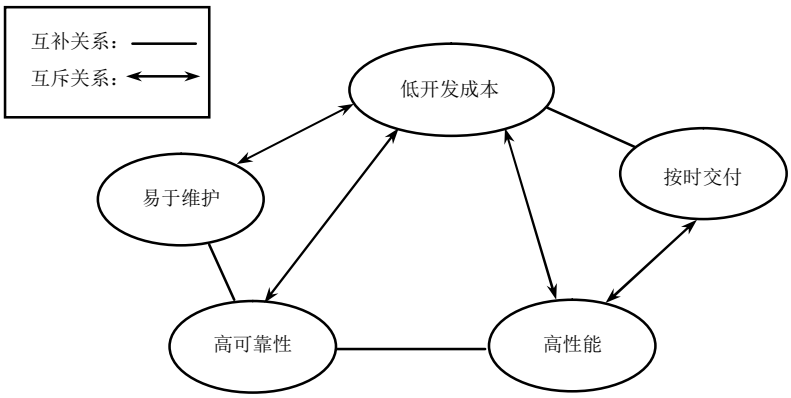

图 1-2　软件工程目标之间的关系

1.3.4 软件工程的原则

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。

1. 选取适宜开发模型

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，采用适宜的开发模型予以控制，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，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。

这一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；实施一个软件工程要选取适宜的开发模型，要采用合适的设计方法，要提供高质量的工程支撑，要实行开发过程的有效管理；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，采用合适的开发模型、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：软件开发模型、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。

1.3.5 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。美国著名的软件工程专家巴利·勃姆（Barry Boehm）综合这些专家的意见，总结了美国天合公司（TRW）多年的开发软件的经验，于 1983 年提出了软件工程的七条基本原理。

1. 用分阶段的生命周期计划严格管理

这一条是吸取前人的教训而提出来的。统计表明，50%以上的失败项目是由于计划不周而造成的。在软件开发与维护的漫长生命周期中，需要完成许多性质各异的工作。这条原理意味着，应该把软件生命周期分成若干阶段，并相应制定出切实可行的计划，然后严格按照计划对软件的开发和维护进行管理。

勃姆认为，在整个软件生命周期中应指定并严格执行 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

统计结果显示：大部分错误是在编码之前造成的，大约占 63%。错误发现得越晚，改正

它要付出的代价就越大，要差 2 到 3 个数量级。因此，软件的质量保证工作不能等到编码结束之后再进行，应坚持进行严格的阶段评审，以便尽早发现错误。

3. 实行严格的产品控制

开发人员最痛恨的事情之一就是改动需求。但是实践告诉我们，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。也就是要采用变动控制，又叫基准配置管理。当需求变动时，其他各个阶段的文档或代码随之相应变动，以保证软件的一致性。

4. 采纳现代程序设计技术

从二十世纪六、七十年代的结构化软件开发技术，到最近的面向对象技术，从第一、第二代语言，到第四代语言，人们已经充分认识到：方法大于气力。采用先进的技术既可以提高软件开发的效率，又可以减少软件维护的成本。

5. 结果应能清楚地审查

软件是一种看不见、摸不着的逻辑产品。软件开发小组的工作进展情况可见性差，难于评价和管理。为更好地进行管理，应根据软件开发的总目标及完成期限，尽量明确地规定开发小组的责任和产品标准，从而使所得到的标准能清楚地审查。

6. 开发小组的人员应少而精

开发人员的素质和数量是影响软件质量和开发效率的重要因素，应该少而精。这一条基于两点原因：高素质开发人员的效率比低素质开发人员的效率要高几倍到几十倍，开发工作中犯的错误也要少得多；当开发小组为 N 人时，可能的通讯信道为 N(N-1)/2，可见随着人数 N 的增大，通讯开销将急剧增大。

7. 承认不断改进软件工程实践的必要性

遵从上述六条基本原理，就能够较好地实现软件的工程化生产。但是，它们只是对现有的经验的总结和归纳，并不能保证赶上技术不断前进发展的步伐。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原理。根据这条原理，不仅要积极采纳新的软件开发技术，还要注意不断总结经验，收集进度和消耗等数据，进行出错类型和问题报告统计。这些数据既可以用来评估新的软件技术的效果，也可以用来指明必须着重注意的问题和应该优先进行研究的工具和技术。

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。它们是相互独立的，是缺一不可的最小集合；同时，它们又是相当完备的。

人们当然不能用数学方法严格证明它们是一个完备的集合，但是可以证明，在此之前已经提出的 100 多条软件工程准则都可以有这七条原理的任意组合蕴含或派生。

1.3.6 软件开发方法

1. 结构化方法

结构化方法（Structure Method）的基本思想可以概括为：自顶向下、逐步求精，采用模

块化技术、分而治之的方法，将系统按功能分解成若干模块；模块内部由顺序、分支、循环三种基本控制结构组成；子程序实现模块化。

2. 面向对象方法

面向对象方法（Objected-Oriented）认为：客观世界由各种对象组成，每个对象都有各自内部状态和运动规律，不同对象之间相互作用和联系构成了各种各样的系统，构成了客观世界。

面向对象方法吸取了结构化方法的基本思想和主要优点，将数据与操作放在一起，作为一个相互依存、不可分割的整体进行处理。它综合了功能抽象和数据抽象，采用数据抽象和信息隐蔽技术，将问题求解看作是一个分类演绎过程。与结构化方法相比，面向对象更接近人们认识事物和解决问题的过程和思维方式。

1.4 软件生存周期及其模型

软件生存周期（Software Life Cycle）是指软件产品开发的一系列相关活动的整个生命期，即从软件定义开始，经过软件开发、交付使用到运行与维护，直到最终被废弃的整个时期。它由三个时期构成，每个时期进一步划分成若干阶段。

1.4.1 软件定义时期

软件定义时期的任务：确定软件开发工程必须完成的总目标；确定工程的可行性；导出实现工程目标应该采取的措施与系统必须完成的功能；估算完成该工程需要的资源和成本；制定工程进度表。

该阶段的任务又称为系统分析，由系统分析员负责完成。软件定义部分又可划分为问题定义、可行性研究和需求分析三个阶段。

1. 问题定义

问题定义的任务是：搞清楚“要解决的问题是什么”。如果不知道要解决的问题是什么，就没有办法解决问题，或者只能盲目的解决，最终的结果不但解决不了问题，还浪费人力、物力、时间和金钱。

系统分析员通过对问题的调研，写出关于问题性质、工程目标和工程规模的书面报告，并且要得到用户的确认。

2. 可行性研究

可行性研究的任务是：了解用户的要求及实现的环境，从技术、经济、操作和法律等方面研究并论证软件系统的可行性。

3. 需求分析

需求分析阶段的任务是确定“目标系统必须做什么”。需求分析在可行性分析的基础上，对用户进一步做深入细致的调研，对目标系统提出完整、准确、清晰、具体的要求，以便顺利进行后续阶段的工作。

在该阶段系统分析员必须和用户密切配合、充分交流信息，以便得出经过用户确认的系统需求，并撰写出需求规格说明书。

1.4.2 软件开发时期

软件开发时期的任务是具体设计和实现软件定义部分所定义的软件。软件开发时期又分为概要设计、详细设计、编码与单元测试、综合测试四个阶段。

概要设计和详细设计又称为系统设计，编码与单元测试和综合测试又称为系统实现。

1. 概要设计

概要设计也称为总体设计。概要设计阶段的任务是：确定目标系统必须怎样做，概括的提出解决问题的办法。

概要设计要完成的基本任务：根据软件需求规格说明书建立系统的总体结构和模块间的关系，定义各个功能模块的接口，设计数据库或数据结构，规定设计约束，制定组装测试计划。编写概要设计说明书。

2. 详细设计

详细设计也称为模块设计、物理设计。详细设计阶段的任务是：怎样具体地实现目标系统。把概要设计的解法具体化，将概要设计产生的功能模块逐步细化，形成若干个可编程的模块，并用某种过程设计语言（PDL）设计程序模块的内部细节。撰写详细设计说明书。

3. 编码与单元测试

编码阶段的任务是把每个模块的控制结构写成计算机可接受的程序代码，并对编写出的每个模块代码进行认真细致地的测试。

4. 综合测试

综合测试阶段的任务是通过各种类型的测试使软件达到预期的效果。综合测试包括组装测试和确认测试。组装测试也叫集成测试，是将测试好的各模块按一定顺序组装起来进行的测试，主要查找各模块之间接口上存在的问题；确认测试也叫系统测试，是按照软件需求规格说明书的规定，由用户对目标系统进行验收测试，决定开发的软件是否合格。

1.4.3 软件运行与维护时期

运行与维护部分的任务是使软件永久地满足用户的需求。软件在使用过程中会出现各种各样的错误，需要对软件进行维护。

（1）修正性维护：改正软件在使用过程中发现的错误。

（2）适应性维护：修改软件以适应新运行环境。

（3）完善性维护：改进软件满足用户新的需求。

（4）预防性维护：为了给未来的改进提供更好的基础或改善软件未来的可维护性或可靠性而做出的修改。

软件退役是软件生存周期中的最后一个阶段，终止对软件产品的支持，软件停止使用。

1.4.4 软件生存周期模型

软件生存周期模型也叫软件生命周期模型，是描述软件开发过程中各种活动如何执行的模型。软件生存周期模型确立了软件开发和演绎中各阶段的次序以及各阶段活动的准则，确立开发过程所遵守的规定和限制，便于各种活动的协调和人员通信，有利于活动重用和活动管理。常见的软件生存周期模型有瀑布模型、原型模型、增量模型、螺旋模型、喷泉模型等。

1. 瀑布模型

瀑布模型（Waterfall Model）是将软件生存周期规定为依线性顺序联接的若干阶段的模型。如图 1-3 所示。它包括问题定义、可行性分析、需求分析、概要设计、详细设计、编码、测试和维护。它规定了由前至后、相互衔接的固定次序，如同瀑布流水，逐级下落。

瀑布模型为软件开发提供了一种有效的管理模型。根据这一模式制定开发计划，进行成本预算，组织开发力量，以项目的阶段评审和文档控制为手段，有效地对整个开发过程进行指导。因此它是以文档作为驱动、适合于需求很明确的软件项目开发的模型。

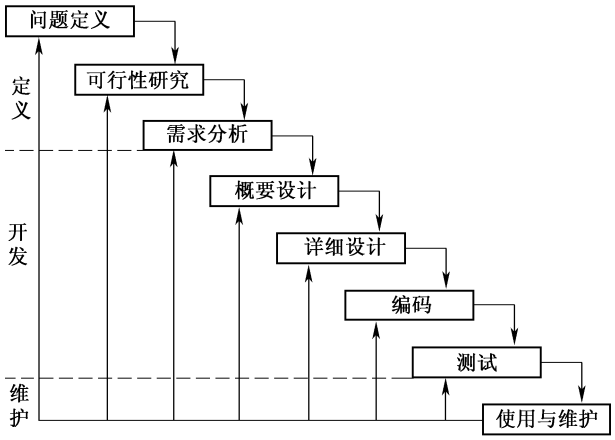

图 1-3 瀑布模型

瀑布模型有如下几个特点：

（1）阶段间具有顺序性和依赖性。

（2）推迟实现的观点。

（3）质量保证的观点。

瀑布模型的缺点：

（1）用户看到软件产品的时间靠后，因此开发的产品很可能不是建立在全面、正确认识基础上的。

（2）缺乏灵活性，修改的代价高。

2. 原型模型

原型模型（Prototyping Model）又称为快速原型模型，如图 1-4 所示。这种方法的核心思想是：在软件开发的早期，软件开发人员根据用户提出的软件需求快速建立目标系统的原型，反复让用户对原型进行评估并提出修改意见，开发人员根据用户意见对原型进行修补和完善，直到用户对所开发的系统原型满意为止。

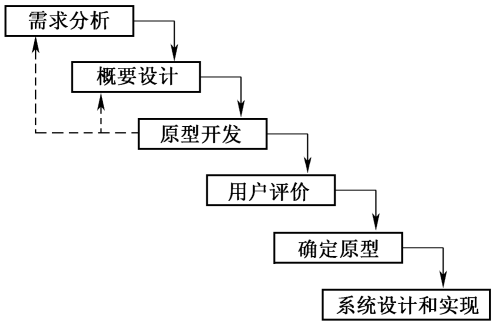

图 1-4　原型模型

原型模型有两种类型：

（1）演进型原型：方法是与客户一起工作，通过反复向客户演示原型系统并征求他们的意见，进行不断地修改，从而迭代出满足客户需求的可交付使用的最终产品。

（2）废弃型原型：其用途是为了获得用户的真正需求，一旦需求确定了，原型将被抛弃。

原型模型的适用范围：特别适用需求分析与定义规格说明；设计人机界面；充作同步培训工具。

3. 增量模型

增量模型（Increment Model）是一种非整体开发模型，如图 1-5 所示。软件在该模型中是“逐渐”被开发出来的，软件开发出一部分，就向用户展示一部分，可让用户及早看到部分软件，及早发现问题。或者先开发一个“原型”软件，完成部分主要功能，展示给用户并征求意见，然后逐步完善，最终获得满意的软件产品。

瀑布模型是一种整体开发模型。在开发过程中，用户看不到软件是什么样子，只有开发完成后，整个软件才全部展现在用户面前。这时如果用户发现有不满意的地方，为时已晚。增量模型具有较大的灵活性，适合于软件需求不明确、设计方案有一定风险的软件项目。

例如，用增量模型开发一个字处理软件。

增量 1：基本的文字输入和编辑功能；

增量 2：格式处理功能；

增量 3：拼写检查功能；

增量 4：排版和打印功能。

增量模型优点：

（1）能在较短的时间内向用户提供一些已完成且有用的工作产品。

（2）逐步增加的产品功能使用户有充足时间学习适应新产品。

使用增量模型应注意：在把每个新的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。

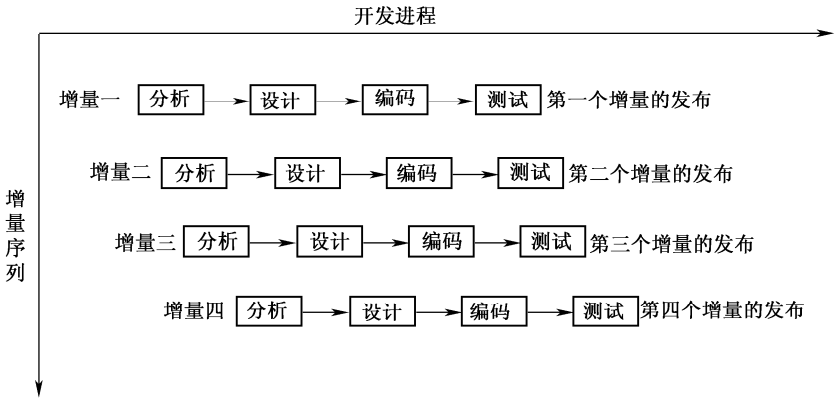

图 1-5　增量模型

4. 螺旋模型

螺旋模型（Spiral Model）是 Boehm 于 1988 年提出来的。螺旋模型是在原型模型的基础上扩展而成的，它结合了瀑布模型的特点，并在原有的基础上加入了风险分析的机制。螺旋模型的基本思想是使用原型及其他方法来尽量降低风险。

螺旋模型通常用来指导大型软件项目的开发，它将开发划分为制定计划、风险分析、实施工程和客户评估四类活动。

如图 1-6 所示，螺旋模型沿着螺线旋转，每转一圈，表示开发出一个更完善的新的软件版本。四个象限分别表达了四个方面的活动，即：

制定计划——确定软件目标，选定实施方案，弄清项目开发的限制条件。

风险分析——分析所选方案，考虑如何识别和消除风险。

实施工程——实施软件开发。

客户评估——评价开发工作，提出修正建议。

螺旋模型优点：对软件开发风险有充分认识，因此适用于内部开发的大规模软件项目。但进行风险评估需要开发人员具有丰富的开发经验和各方面的专业知识。

5. 喷泉模型

喷泉模型（Water Fountain Model）是 B.H.Sollers 和 J.M.Edwards 于 1990 年提出的一种软件开发模型。喷泉模型是以面向对象的软件开发技术为基础，以用户需求为动力，以对象来驱动的模型。它克服了瀑布模型不支持软件重用和生存周期中多项开发活动集成的局限性，使得软件开发过程具有迭代和无缝的特性。如图 1-7 所示，各个阶段不但没有明显的界限，而且还

存在重叠区域，这样各项工作可以开展，进而提高开发效率。

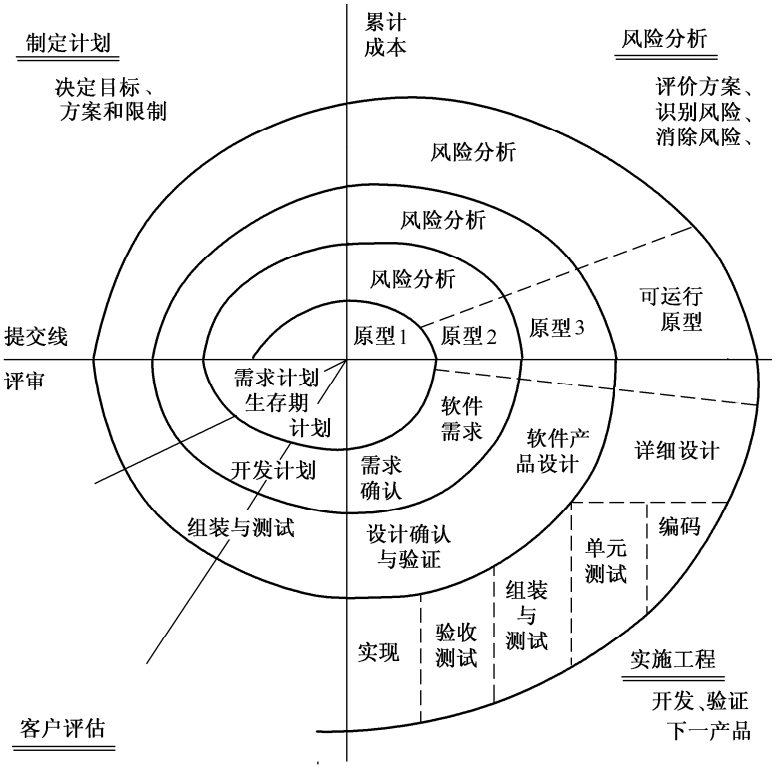

图 1-6　螺旋模型

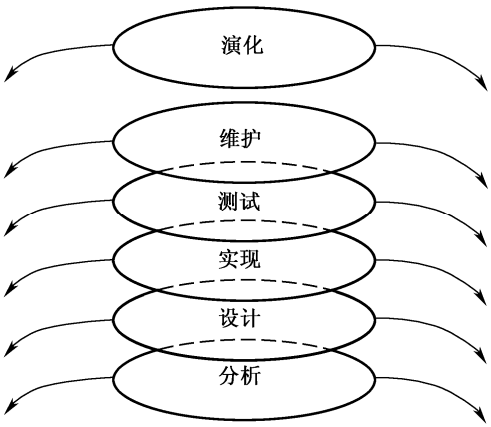

图 1-7　喷泉模型

习题一

一、选择题

1. 下列活动不属于软件开发阶段的是（　　）。
   A. 需求分析　　B. 概要设计　　C. 详细设计　　D. 测试
2. 以下对软件工程原理的叙述不正确的是（　　）。
   A. 用分阶段的生命周期计划严格管理
   B. 采用现代程序设计技术
   C. 开发小组的人员应该足够多
   D. 承认不断改进软件工程实践的必要性
3. 在以下软件开发模型中，最常用在软件需求难以完全明确的情况下的是（　　）。
   A. 瀑布模型　　B. 原型模型　　C. 螺旋模型　　D. 增量模型
4. 对于喷泉模型，下列说法错误的是（　　）。
   A. 是一种面向对象的开发模型
   B. 具有迭代性
   C. 具有无缝性
   D. 各阶段之间具有顺序性和依赖性
5. 软件是一种（　　）产品。
   A. 有形　　B. 逻辑　　C. 程序　　D. 数据
6. 下列关于瀑布模型的描述正确的是（　　）。
   A. 瀑布模型的核心是按照软件开发的时间顺序将问题简化
   B. 瀑布模型具有良好的灵活性
   C. 瀑布模型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
   D. 利用瀑布模型，如果发现问题修改的代价很低
7. 软件工程的出现主要是由于（　　）。
   A. 程序方法学的影响　　B. 其他工程学科的影响
   C. 计算机的发展　　D. 软件危机的出现
8. 瀑布模型本质上是一种（　　）。
   A. 线性顺序模型　　B. 顺序迭代模型
   C. 线性迭代模型　　D. 及早见到产品模型
9. 具有风险分析的软件生存周期模型是（　　）。
   A. 瀑布模型　　B. 螺旋模型
   C. 增量模型　　D. 喷泉模型

二、填空题

1．软件是计算机中与硬件相互依存的部分，它是_____、_____和_____的完整集合。

2．在软件生存期中，软件定义阶段包括_____、_____和_____三个阶段。

3．瀑布模型适合于_____的项目开发，它的主要特点是各阶段之间具有_____和_____。

4．螺旋模型包含了四个方面的活动，它们分别是_____、_____、_____和_____。

5．软件工程是从_____和_____两个方面研究如何运用工程学的基本原理和方法来更好地开发和维护计算机软件的一门学科。

三、简答题

1．什么是软件？软件有哪些特点？

2．什么是软件危机？软件危机有哪些表现？软件危机产生的原因有哪些？如何消除软件危机？

3．软件生存期包括哪些时期？各个时期包括哪些阶段？

4．什么是软件工程？软件工程包括哪些要素？

5．软件工程的目标有哪些？

6．试述软件工程的原则。

7．试述软件工程的基本原理。

8．软件开发的主要方法有哪些？

9．简述软件生存周期模型及其特点。