高等职业教育特色精品课程规划教材
高等职业教育课程改革项目研究成果

软件工程

Software Engineering

张洪民 主 编
张金泽 邓连瑾 晏 婴 副主编

SOFTWARE

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

21世纪高等职业教育特色精品课程规划教材
高等职业教育课程改革项目研究成果

软件工程

主　编　张洪民
副主编　张金泽　邓连瑾　晏　婴
参　编　赵　震　侯小毛

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

图书在版编目（CIP）数据

软件工程/张洪民主编. —北京：北京理工大学出版社，2009.7
ISBN 978-7-5640-2490-1

Ⅰ.软…　Ⅱ.张…　Ⅲ.软件工程-教材　Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字（2009）第 117447 号

出版发行／北京理工大学出版社
社　　址／北京市海淀区中关村南大街 5 号
邮　　编／100081
电　　话／(010)68914775(办公室)　68944990(批销中心)　68911084(读者服务部)
网　　址／http：// www. bitpress. com. cn
经　　销／全国各地新华书店
印　　刷／北京圣瑞伦印刷厂
开　　本／787 毫米×1092 毫米　1/16
印　　张／11.5
字　　数／268 千字
版　　次／2009 年 7 月第 1 版　2009 年 7 月第 1 次印刷
印　　数／1～4000 册　　责任校对／陈玉梅
定　　价／23.00 元　　责任印制／边心超

前　言

在一些人眼里，今天的软件开发似乎已成为简单的事情，已有了不少好的开发工具和软件库，训练有素的软件开发人员，都强烈渴望去编写很酷的软件，可以在几天的时间里编写出一个相当复杂的软件。但为什么有一些软件能够得到用户的喜欢，而另一些则不能？为什么有些软件能够在市场上成功，而有些则受到冷落？由此可见，开发软件并不难，难就难在如何开发适用的软件。

我最大的心得是，一个产品一定要找到能够真正适用的场合，不能只是为了技术而从事技术、为了研究而进行研究，却不管用户对你所研究的技术和产品有没有需求。否则，无论你的技术是多么优秀，多么先进，恐怕你的产品在市场上都无法获得成功。

软件工程是计算机科学与技术专业的一门专业核心课程。通过本课程的学习，使学生掌握系统的软件开发理论、技术和方法，使用正确的工程方法开发出成本低、可靠性好并在机器上能高效运行的软件，为今后从事软件开发和维护打下坚实的基础。

软件工程是一门内容很多，很复杂的学科。期间要涉及的知识，从软件的知识，到进行需求分析，概要设计，详细设计，软件测试，软件工程标准化，软件文档，都有很多的内容，因此，在一本书中，将这些内容要全面的介绍，并且要有逻辑性，顺序性，突出重点，讲明原理，并不是一件很容易的事情。

本书共 12 章。第 1 章为软件工程概述；第 2、第 3、第 4 章是在介绍软件需求分析，可行性分析等步骤，是软件设计的前提工作；第 5 章是概要设计，讲述了设计的基本流程；第 6、第 7 章是详细设计与编码部分，讲述了软件设计的详细过程；第 8 章简述软件测试技术，也是软件工程学科中的一个很重要的部分；第 9 章讲述了软件维护；第 10、第 11 章内容是软件工程标准化，软件工程文档，软件工程质量等知识，也都是学习软件工程的人一定要知道的知识；第 12 章为软件工程项目管理。

我们在编写这本书的时候，希望能够尽量多地讲述软件工程方面的知识，可以作为计算机专业学生的教材或者自学用书，也可以作为计算机爱好者充实自己的读物。

由于时间仓促，虽然编者们都很努力，仍难免有不足的地方，希望我们能与读者共同提高。

编　者

目　录

第1章 软件工程概述

1.1 软件的概念

1.1.1 软件的定义

软件是为了特定目的而开发的程序、数据和文档的集合。

（1）程序：能够执行特定功能的计算机指令序列。

（2）数据：执行程序所必需的数据和数据结构。大量的数据都是按照一定的数据结构由用户在使用软件的过程中积累起来的。

（3）文档：与程序开发、维护和使用有关的图文资料。

1.1.2 软件特征

要理解软件的含义，首先要了解软件的特征是很重要的，当人们制造硬件时，是以物理的过程形式出现的（分析、设计、制造、测试）。而软件是一种逻辑实体，而不是具体的物理实体。因此，软件具有与硬件完全不同的特征。

（1）软件是由开发而成的，而不是制造产生的，它具有抽象性。

（2）软件的生产与硬件不同，它没有明显的制造过程。对软件的质量控制，必须着重在软件开发方面，大多数软件是自定的。

（3）在软件的运行和使用期间，没有硬件那样的机械磨损和老化问题。

如图 1.1 所示，图（a）显示的是硬件的失效率曲线，它存在老化与磨损的特点；图（b）显示的是软件的失效率曲线，它存在退化问题，必须要多次修改（维护）软件。图（b）中突出的部分是由于修改而产生的副作用造成故障率的提高，之后的章节会介绍软件的维护。

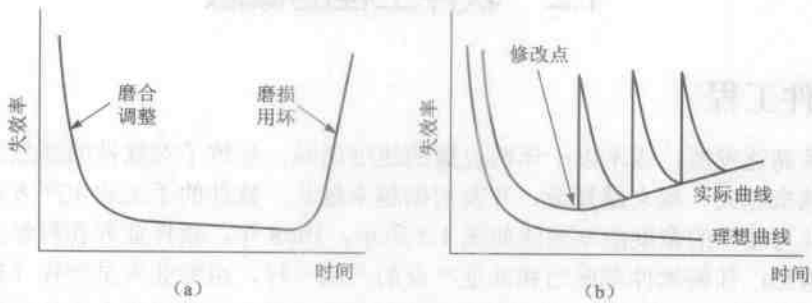

图 1.1 失效率曲线

（a）硬件失效率曲线；（b）软件失效率曲线

1.1.3 软件的发展

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。其后经过几十年的发展，计算机软件经历了以下3个发展阶段。

（1）程序设计阶段，为20世纪50～60年代。

（2）程序系统阶段，为20世纪60～70年代。

（3）软件工程阶段，为20世纪70年代以后。

几十年来计算机软件最根本的变化体现在以下几个方面。

（1）人们改变了对软件的看法。20世纪50～60年代时，程序设计曾经被看做是一种任人发挥创造才能的技术领域。当时人们认为，写出的程序只要能在计算机上得出正确的结果，程序的写法可以不受任何约束。随着计算机的广泛使用，人们要求这些程序容易看懂、容易使用，并且容易修改和扩充。于是，程序便从个人按自己意图创造的“艺术品”转变为能被广大用户接受的工程化产品。

（2）软件的需求是软件发展的动力。早期的程序开发者只是为了满足自己的需要，这种自给自足的生产方式仍然是其低级阶段的表现。进入软件工程阶段以后，软件开发的成果具有了社会属性，它要在市场中流通以满足广大用户的需要。

（3）软件工作的范围从只考虑程序的编写扩展到涉及整个软件生存周期。

1.1.4 软件危机

软件危机是一种现象，是指由于软件复杂程度愈来愈高，在计算机软件开发和维护时所遇到的一系列问题，具体表现在以下几个方面。

（1）软件开发成本高，成本难以控制。

（2）研制周期长，软件开发进度难以控制，周期拖得很长。

（3）正确性难保证，软件质量差，可靠性难以保证。

（4）软件维护困难，维护人员和维护费用不断增长。

（5）软件发展跟不上硬件的发展和用户的要求。

更糟糕的是，许多程序的个人化特性使得它们根本不能维护，于是“软件危机”出现了。

1.2 软件工程的概念

1.2.1 软件工程

硬件技术高速发展，成本以十年两位数的速度递减，导致了对软件的强烈需求。而软件系统的规模越来越大、越来越复杂、开发周期越来越长，软件的手工业生产方式使其成本急骤上升，软件与硬件的发展成本曲线如图1.2所示。1968年，软件业界和科学工作者提出了软件工程的思想：任何软件都应当和其他产业的产品一样，由专业人员制作（软件中是系统分析员、高级程序员和程序员），以系统的、工程的方法开发，并提供全方位的售后服务管理（不能因开发者离开或调走而无人管理）。

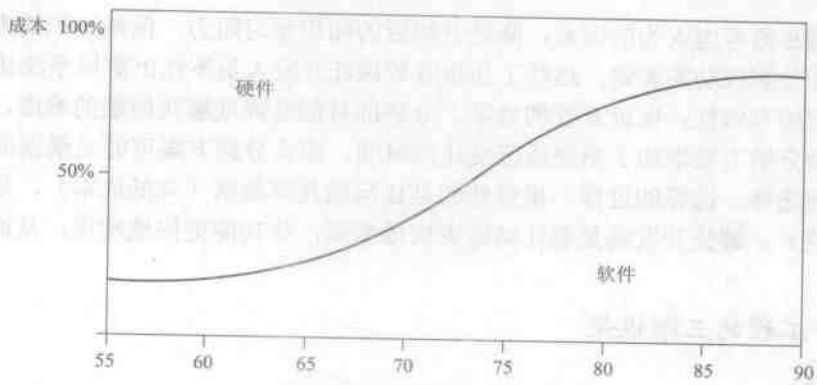

图 1.2　软件与硬件的发展成本曲线

软件工程是一类求解软件的工程。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、数学用于构造模型与算法；工程科学用于制定规范、设计范型、评估成本及确定权衡；管理科学用于计划、资源、质量和成本等的管理。

软件工程是一门指导计算机软件开发和维护的交叉学科，比较著名的定义有以下几个。

（1）Boehm：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。

（2）IEEE：软件工程是开发、运行、维护和修复软件的系统方法。

（3）Fritz Bauer：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。

1.2.2　软件工程的基本原理

1. 软件工程的三要素

（1）工具评估（包括开发平台，开发语言，开发工具以及Frameworks）。

① 用好的工具提高生产效率，使人关注于有效工作内容，从而减少不必要的工作量，降低成本。特别是对于分工合作下的团队开发尤为重要。典型的分工是流水线式的，一步接着一步。减少上一个环节的工作量，可以提前进入下一个环节。

② 用好的工具保证质量——另外一种生产效率。

保证质量有利于减少工作上的反复，尤其是用于测试的工作量，在提高生产效率的同时也可以保证士气。

（2）开发方法。解决问题的办法就是分开而治。要被分解的问题域是：数据（模型），计算和流程；而如何分解的问题便是架构师的任务，流行的有 OOD 和 AOSD 两种。在大比例结构中必须考虑的是：抽象分层，技术分层以及模块划分。抽象分层（包括模型，计算以及流程的抽象）以及模块划分是基于业务的纵向以及横向分解，我们也称之为解耦。而技术分层则是对于业务逻辑的技术分类，分类本身还可能涉及平台的技术限制。所有分解都涉及上下文的边界建立——不仅是业务逻辑的区分同时也是技术的区分。

分解问题必须考虑人为的因素，降低分解后的知识学习阻力，保持知识的内聚以及有效的组织是保证分解成功的关键。这些工作将有效保证开发人员不作出破坏系统边界的行为。

考核分解的有效性：保证开发的效率。分解的目的是降低解决问题的难度，从而提高生产效率，如果分解方案增加了系统适应变化的时间，那么分解方案可能是错误的。

（3）过程选择。选择的过程，最重要的是让问题及早暴露（降低成本），尽早让用户使用（创造价值）。敏捷开发就是要让问题更快地暴露，让功能更快地实现，从而早早进入下一个循环。

2. 软件工程的三维框架

（1）目标。

① 可用性：软件基本结构的实现及文档的可用程度。

② 正确性：软件产品达到预期功能要求的程度。

③ 合算性：软件开发、运行的整个开销满足用户要求的程度。

（2）原则。

① 采用适当的开发模型作为开发的指导，以控制软件开发的易变性。

② 运用良好的设计方法，提高软件质量和软件生产率。

③ 有效的工程支持（工具支持）。

④ 有效的管理。

（3）活动。

软件生命周期的各个阶段。

3. 七条基本原理

（1）用分阶段的生命周期计划严格管理。经统计发现，在不成功的软件项目中有一半左右是由于计划不周造成的，可见把建立完善的计划作为第一条基本原理是吸取了前人的教训而提出来的。

在软件开发与维护的漫长的生命周期中，需要完成许多性质各异的工作。这条基本原理意味着，应该把软件生命周期划分成若干个阶段，并相应地制订出切实可行的计划，然后严格按照计划对软件的开发与维护工作进行管理。Boehm 认为，在软件的整个生命周期中应该制订并严格执行 6 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、验证计划和运行维护计划。不同层次的管理人员都必须严格按照计划各尽其职地管理软件开发与维护工作，绝不能受客户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，这样说至少有两个理由：第一，大部分错误是在编码之前造成的，例如，根据 Boehm 等人的统计，设计错误占软件错误的 63%，编码仅占 37%；第二，错误发现与改正得越晚，所需付出的代价也越高。因此，在每个阶段都要进行严格的评审，以便尽早发现在软件开发过程中所犯的错误，是一条必须遵循的重要原则。

（3）实行严格的产品控制。在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，但是，在软件开发过程中改变需求又是难免的，由于外部环境的变化，相应地改变用户需求是一种客观需要，显然不能硬性禁止客户提出改变需求的要求，而只能依靠科学的产品控制技术来顺应这种要求。也就是说，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。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动控制：一切有关修改软件的建议，特别是涉及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。绝对不能谁想修改软件（包括尚在开发过程中的软件），就随意进行修改。

（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。20 世纪 60 年代末提出的结构程序设计技术，已经成为绝大多数人公认的先进的程序设计技术。以后又进一步发展出各种结构分析（SA）与结构设计（SD）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。软件产品不同于一般的物理产品，它是看不到摸不着的逻辑产品。软件开发人员（或开发小组）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于评价和管理。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，从而使得所得到的结果能够清楚地审查。

（6）开发小组的人员应该少而精。这条基本原理的含义是，软件开发小组组成人员的素质应该较高，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。素质高的人员的开发效率比素质低的人员的开发效率可能高几倍至几十倍，而且素质高的人员所开发的软件中的错误明显少于素质低的人员所开发的软件中的错误。此外，随着开发小组人员数目的增加，因为交流情况讨论问题而造成的通信开销也会急剧增加。当开发小组人员数为 N 时，可能的通信路径有 $N(N-1)/2$ 条，可见随着人数 N 的增大，通信开销将急剧增加。因此，组成少而精的开发小组是软件工程的一条基本原理。

（7）承认不断改进软件工程实践的必要性。遵循前 6 条基本原理，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，仅有前 6 条原理并不能保证软件开发与维护的过程能赶上时代前进的步伐，能跟上技术的不断进步。因此，Boehm 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 7 条基本原理。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，例如，收集进度和资源耗费数据，收集出错类型和问题报告数据等。这些数据不仅可以用来评价新的软件技术的效果，而且可以用来指明必须着重开发的软件工具和应该优先研究的技术。

1.3　软件过程

软件过程

软件过程，是指软件的整个生命周期，是从需求获取，需求分析，设计，实现到测试，发布和维护一个过程模型。一个软件过程定义了软件开发中采用的方法，但软件工程还包含该过程中应用的技术——技术方法和自动化工具。

过程是活动的集合，活动是任务的集合，任务是将输入变换为输出的操作。按照不同人员的工作内容来分类，软件过程可分为 3 类：基本过程、支持过程和组织过程。

（1）基本过程：指与软件生产直接相关的过程，包括获取过程、供应过程、开发过程、

运行过程和维护过程。

（2）支持过程：是有关各方按他们的支持目标所从事的一系列相关活动。支持过程有助于提高系统或软件产品的质量，有助于系统的运行。包括文档过程、配置管理过程、质量保证过程、验证过程、确认过程、联合评审过程、审计过程和问题解决过程。

（3）组织过程：是指那些与软件生产组织有关的过程。包括管理过程、基础设施过程、改进过程和培训过程。

软件工程过程通常包含4种基本过程活动。

（1）P（Plan）：软件规格说明。规定软件的功能及其运行的限制。

（2）D（Do）：软件开发。产生满足规格说明的软件。

（3）C（Check）：软件确认。确认软件能够完成客户提出的要求。

（4）A（Action）：软件演进。为满足客户的变更要求，软件必须在使用的过程中演进。

1.4 软件工程生存周期

和其他事物一样，软件也有一个孕育、诞生、成长、成熟以及衰亡的生存过程，称为计算机软件的生存周期。软件生存周期是从软件的产生直到报废的过程，周期内有问题定义、可行性分析、总体描述、系统设计、编码、调试和测试、验收与运行、维护升级以及废弃等阶段。这种按时间分程的思想方法是软件工程中的一种思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、形成文档以供交流或备查，以提高软件的质量。但随着新的面向对象的设计方法和技术的成熟，软件生命周期设计方法的指导意义正在逐步减少。根据这一思想，把上述基本的过程活动进一步展开，可以得到软件生存周期的6个步骤。

（1）制订计划：确定要开发软件系统的总目标，给出它的功能、性能、可靠性以及接口等方面的要求；研究完成该项软件任务的可行性，探讨解决问题的可能方案；制定完成开发任务的实施计划，连同可行性研究报告，提交管理部门审查。

（2）需求分析：对待开发软件提出的需求进行分析并给出详细的定义。编写出软件需求说明书及初步的用户手册，提交管理机构评审。

（3）软件设计：把已确定了的各项需求转换成一个相应的体系结构，进而对每个模块要完成的工作进行具体的描述。编写设计说明书，提交评审。

（4）程序编写：把软件设计转换成计算机可以接受的程序代码。

（5）软件测试：在设计测试用例的基础上检验软件的各个组成部分。

（6）运行／维护：已交付的软件投入正式使用，并在运行过程中进行适当的维护。

1.4.1 瀑布模型

1970年温斯顿·罗伊斯（Winston Royce）提出了著名的“瀑布模型”，直到20世纪80年代早期，它一直是唯一被广泛采用的软件开发模型。

瀑布模型将软件生命周期划分为制订计划、需求分析、软件设计、程序编写、软件测试和运行维护6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。从本质来讲，它是一个软件开发架构，开发过程是通过一系列阶段顺序展开的，从系统需求分析开始直到产品发布和维护，每个阶段都会产生循环反馈，因此，如果有信息

未被覆盖或者发现了问题，那么最好“返回”上一个阶段并进行适当的修改，开发进程从一个阶段“流动”到下一个阶段，这也是瀑布模型名称的由来。

瀑布模型核心思想是按工序将问题化简，将功能的实现与设计分开，便于分工协作，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。采用瀑布模型的软件过程如图 1.3 所示。

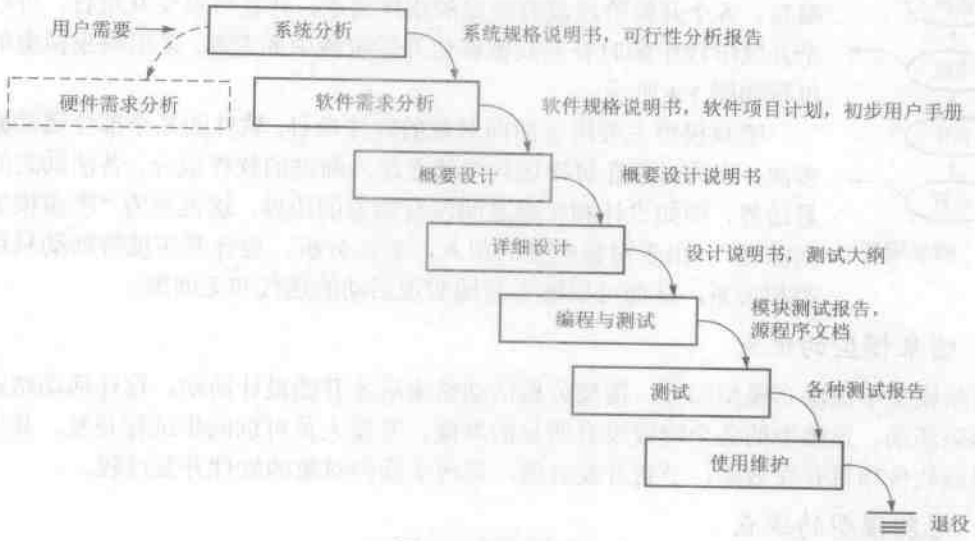

图 1.3　瀑布模型

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。

1. 瀑布模型的优点

（1）为项目提供了按阶段划分的检查点。

（2）当前一阶段完成后，只需要去关注后续阶段。

（3）可在迭代模型中应用瀑布模型。

2. 瀑布模型的缺点

（1）在项目各个阶段之间极少有反馈。

（2）只有在项目生命周期的后期才能看到结果。

（3）通过过多的强制完成日期和里程碑来跟踪各个项目阶段。

尽管瀑布模型招致了很多批评，但是它对很多类型的项目而言依然是有效的，如果正确使用，可以节省大量的时间和金钱。对于项目而言，是否使用这一模型主要取决于是否能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。对于经常变化的项目而言，瀑布模型毫无价值，这种情况可以考虑其他的架构来进行项目管理，比如螺旋模型。

1.4.2 喷泉模型

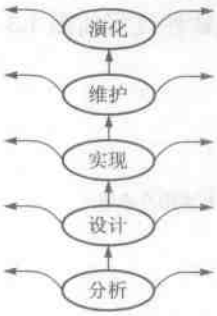

图 1.4 喷泉模型

喷泉模型是一种以用户需求为动力，以对象为驱动的模型，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周期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过程如图 1.4 所示。

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”。由于对象概念的引入，表达分析、设计及实现等活动只用对象类和关系，从而可以较容易地实现活动的迭代和无间隙。

1. 喷泉模型的优点

喷泉模型不像瀑布模型那样，需要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，开发人员可以同步进行开发。其优点是可以提高软件项目开发效率，节省开发时间，适应于面向对象的软件开发过程。

2. 喷泉模型的缺点

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，因此不利于项目的管理。此外这种模型要求严格管理文档，使得审核的难度加大，尤其是面对可能随时加入各种信息、需求与资料的情况。

1.4.3 增量模型

增量模型融合了瀑布模型的基本成分（重复应用）和原型实现的迭代特征，该模型采用随着日程时间的进展而交错的线性序列，每一个线性序列产生软件的一个可发布的“增量”。当使用增量模型时，第 1 个增量往往是核心的产品，即第 1 个增量实现了基本的需求，但很多补充的特征还没有发布。客户对每一个增量的使用和评估都作为下一个增量发布的新特征和功能，这个过程在每一个增量发布后不断重复，直到产生了最终的完善产品。增量模型强调每一个增量均发布一个可操作的产品。采用增量模型的软件过程如图 1.5 所示。

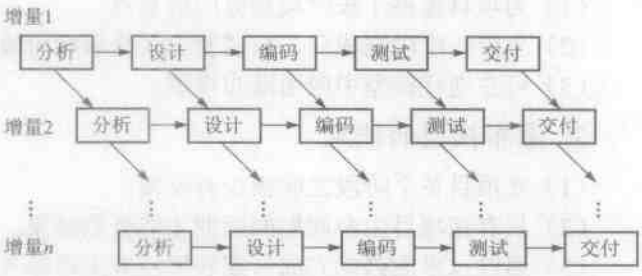

图 1.5 增量模型

增量模型与原型实现模型和其他演化方法一样，本质上是迭代的，但与原型实现不一样的是，其强调每一个增量均发布一个可操作产品。早期的增量是最终产品的“可拆卸”版本，但提供了为用户服务的功能，并且为用户提供了评估的平台。

增量模型的特点是引进了增量包的概念，无须等到所有需求都出来，只要某个需求的增量包出来即可进行开发。虽然某个增量包可能还需要进一步适应客户的需求并且更改，但只要这个增量包足够小，其影响对整个项目来说是可以承受的。

1. 增量模型的优点

采用增量模型的优点是人员分配灵活，刚开始不用投入大量人力资源。如果核心产品很受欢迎，则可增加人力实现下一个增量。当配备的人员不能在设定的期限内完成产品时，它提供了一种先推出核心产品的途径。这样即可先发布部分功能给客户，对客户起到镇静剂的作用。此外，增量能够有计划地管理技术风险。

2. 增量模型的缺点

增量模型存在以下缺陷。

（1）由于各个构件是逐渐并入已有的软件体系结构中的，所以加入构件必须不破坏已构造好的系统部分，这需要软件具备开放式的体系结构。

（2）在开发过程中，需求的变化是不可避免的。增量模型的灵活性可以使其适应这种变化的能力大大优于瀑布模型和快速原型模型，但也很容易退化为边做边改模型，从而是软件过程的控制失去整体性。

（3）如果增量包之间存在相交的情况且未很好处理，则必须做全盘系统分析，这种模型将功能细化后分别开发的方法较适应于需求经常改变的软件开发过程。

1.4.4　螺旋模型

1988 年，Barry Boehm 正式发表了软件系统开发的“螺旋模型”，它将瀑布模型和快速原型模型结合起来，强调了其他模型所忽视的风险分析，特别适合于大型复杂的系统。

如图 1.6 所示，螺旋模型沿着螺线进行若干次迭代，图中的 4 个象限代表了以下活动。

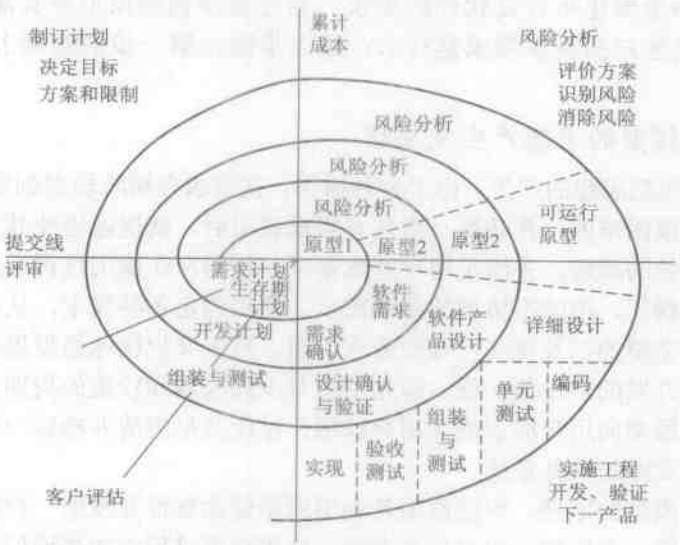

图 1.6　螺旋模型

（1）制订计划：确定软件目标，选定实施方案，弄清项目开发的限制条件。

（2）风险分析：分析评估所选方案，考虑如何识别和消除风险。

（3）实施工程：实施软件开发和验证。

（4）客户评估：评价开发工作，提出修正建议，制订下一步计划。

螺旋模型由风险驱动，强调可选方案和约束条件从而支持软件的重用，有助于将软件质量作为特殊目标融入产品开发之中。但是，螺旋模型也有一定的限制条件，具体如下。

（1）螺旋模型强调风险分析，但要求许多客户接受和相信这种分析，并作出相关反应是不容易的，因此，这种模型往往适应于内部的大规模软件开发。

（2）如果执行风险分析将大大影响项目的利润，那么进行风险分析毫无意义，因此，螺旋模型只适合于大规模软件项目。

（3）软件开发人员应该擅长寻找可能的风险，准确地分析风险，否则将会带来更大的风险。

一个阶段首先是确定该阶段的目标，完成这些目标的选择方案及其约束条件，然后从风险角度分析方案的开发策略，努力排除各种潜在的风险，有时需要通过建造原型来完成。如果某些风险不能排除，该方案立即终止，否则启动下一个开发步骤。最后，评价该阶段的结果，并设计下一个阶段。

1.4.5 快速原型模型

原型是指模拟某种产品的原始模型，在其他产业中经常使用。软件开发中的原型是软件的一个早期可运行的版本，它反映了最终系统的重要特性。

快速原型模型又称原型模型，它是增量模型的另一种形式：它是在开发真实系统之前，构造一个原型，在该原型的基础上，逐渐完成整个系统的开发工作。快速原型模型的第一步是建造一个快速原型，实现客户或未来的用户与系统的交互，用户或客户对原型进行评价，进一步细化待开发软件的需求，通过逐步调整原型使其满足客户的要求，开发人员可以确定客户的真正需求是什么；第二步则在第一步的基础上开发客户满意的软件产品。

1. 快速原型模型的思想产生及原理

（1）快速原型模型思想的产生。由于种种原因，在需求分析阶段得到完全、一致、准确、合理的需求说明是很困难的。在获得一组基本需求说明后，就快速地使其“实现”，通过原型反馈，加深对系统的理解，并满足用户基本要求，使用户在试用过程中受到启发，对需求说明进行补充和精确化，消除不协调的系统需求，逐步确定各种需求，从而获得合理、协调一致、无歧义的、完整的以及现实可行的需求说明。后来又把快速原型思想用到软件开发的其他阶段，向软件开发的全过程扩展。即先用相对少的成本和较短的周期开发一个简单的、但可以运行的系统原型向用户演示或让用户试用，以便及早澄清并检验一些主要设计策略，在此基础上再开发实际的软件系统。

（2）快速原型模型的原理。快速原型是利用原型辅助软件开发的一种新思想。经过简单快速分析，快速实现一个原型，用户与开发者在试用原型过程中加强通信与反馈，通过反复评价和改进原型，减少误解，弥补漏洞，适应变化，最终提高软件质量。

2. 快速原型模型的运用方式

由于运用原型的目的和方式不同，在使用原型时也采取不同的策略，主要有抛弃策略和附加策略。

（1）抛弃策略是将原型用于开发过程的某个阶段，促使该阶段的开发结果更加完整、准确、一致、可靠，该阶段结束后，原型随之作废。探索型和实验型原型就是采用此策略的。

（2）附加策略是将原型用于开发的全过程，原型由最基本的核心开始，逐步增加新的功能和新的需求，反复修改反复扩充，最后发展为用户满意的最终系统，演化型快速原型就是采用此策略。

采用何种形式、何种策略运用快速原型主要取决于软件项目的特点、人员素质、可供支持的原型开发工具和技术等，这要根据实际情况的特点来决定。

3. 快速原型模型的开发步骤

（1）快速分析。在分析人员与用户的密切配合下，迅速确定系统的基本需求，根据原型所要体现的特征描述基本需求以满足开发原型的需要。

（2）构造原型。在快速分析的基础上，根据基本需求说明尽快实现一个可行的系统。这里要求具有强有力的软件工具的支持，并忽略最终系统在某些细节上的要求，如安全性、坚固性以及例外处理等，主要考虑原型系统能够充分反映所要评价的特性，而暂时删除一切次要内容。

（3）运行原型。这是发现问题、消除误解以及开发者与用户充分协调的一个步骤。

（4）评价原型。在运行的基础上，考核评价原型的特性，分析运行效果是否满足用户的愿望，纠正过去交互中的误解与分析中的错误，增添新的要求，并满足因环境变化或用户的新想法引起的系统要求变动，提出全面的修改意见。

（5）修改。根据评价原型的活动结果进行修改。若原型未满足需求说明的要求，说明对需求说明存在不一致的理解或实现方案不够合理，则根据明确的要求迅速修改原型。

1.4.6　各种模型的比较

每个软件开发组织应该选择适合于该组织的软件开发模型，并且应该随着当前正在开发的特定产品特性而变化，以减小所选模型的缺点，充分利用其优点，表 1.1 列出了几种常见模型的优缺点。

表 1.1　各种模型比较

模型	优　点	缺　点
瀑布模型	文档驱动	系统可能不满足客户的需求
快速原型模型	关注满足客户需求	可能导致系统设计差、效率低，难于维护
增量模型	开发早期反馈及时，易于维护	需要开放式体系结构，可能会设计差、效率低
螺旋模型	风险驱动	风险分析人员需要有经验且经过充分训练