高等院校计算机专业及专业基础课系列教材

软件工程

（第二版）

王立福　麻志毅　张世琨　编著

北京大学出版社

高等院校计算机专业及专业基础课系列教材

软件工程

(第二版)

王立福　麻志毅　张世琨　编著

北京大学出版社
北　京

内 容 简 介

本书是在北京大学计算机科学技术系使用《软件工程》教材的基础上，根据全国高等教育自学考试指导委员会制定的《软件工程考试大纲》的要求，由主讲、主考教师编写而成的，既是北京大学计算机系本科生指定教材，也是北京市高等教育自学考试指定教材。

本书结合国内外软件工程的发展，特别是国家“八五”、“九五”攻关实践，详细地讲述了软件工程的基本内容，包括基本概念、基本模型、基本方法及相应的支持工具。本书注重基础知识的系统性，同时注意选材的先进性，内容全面、层次清楚。

图书在版编目(CIP)数据

软件工程(第二版)/王立福等编著. －北京：北京大学出版社，2002.3
ISBN 7-301-03227-7

Ⅰ.软… Ⅱ.王… Ⅲ.软件工程-概论 Ⅳ.TP311.5

书　　名：软件工程(第二版)
著作责任者：王立福　麻志毅　张世琨
责任编辑：沈承凤
标准书号：ISBN 7-301-03227-7/TP·0318
出 版 者：北京大学出版社
地　　址：北京市海淀区中关村北京大学校内　100871
网　　址：http://cbs.pku.edu.cn
电　　话：发行部 62757298　编辑部 62752038
电子信箱：zpup@pup.pku.edu.cn
排 版 者：兴盛达打字服务社　62549189
印 刷 者：中国科学院印刷厂
发 行 者：北京大学出版社
经 销 者：新华书店
787 毫米×1092 毫米　16 开本　17.25 印张　430 千字
2002 年 3 月第 2 版　2002 年 10 月第 1 次修订
2003 年 12 月第 4 次印刷
定　　价：23.00 元

序

“科教兴国”战略强调教育对国民经济的基础地位，要求高等教育“实施全面素质教育，加强思想品德教育和美育，改革教育内容、课程体系和教学方法……”。为了落实好“科教兴国”这一战略决策，北京大学计算机科学技术系与北京大学出版社合作，编审出版基础主干课和专业主干课系列教材。

目前，伴随着微电子和计算机科学技术渗透到社会的各个领域，人类正跨步迈进知识经济时代。在知识经济时代，具有创新能力的高素质人才是经济持续发展的必备条件。

计算机科学技术包括科学和技术两部分，不仅强调严谨的科学性，同时也注重工程性，是一门科学性和工程性并重的学科。信息科学技术的支柱学科是微电子、计算机、通信和软件，其中微电子是基础，计算机和通信是载体，软件是核心，它们相辅相成，共同培育了知识经济。因而，高素质的信息领域科技人才应该掌握上述学科的基础理论和专业技能。

近年来，北京大学计算机科学技术系通过跟踪、分析国际知名大学的相关课程设置、教学实施情况，借鉴国内兄弟院系的课程体系调整建议，总结北京大学计算机科学技术系集计算机软、硬件技术和微电子学于一体的人才培养经验，对课程体系进行了较大力度的梳理，形成了一系列基础主干课和专业主干课。

这一系列教材正是为配合课程体系的调整而编撰的。所选书稿主要是在我系多年的教学实践中师生反映较好的讲义和教材的基础上修编而成的。我们希望这批教材能够达到“注重基础、淡化专业(或突出交叉)、内容系统、选材先进、利于教学”的要求。

对于教材中的不足之处，欢迎广大读者不吝赐教。

杨芙清

一九九九年九月

北京大学计算机科学技术系

专业基础课和专业课教材编审指导小组

组　　长：杨芙清

成　　员：（按姓氏笔画序）

卢晓东　李晓明　许卓群　沈承凤　屈婉玲

张天义　赵宝瑛　袁崇义　董士海　程　旭

北京大学计算机系专业基础课名称

计算引论
数字逻辑
微机原理
计算机组织与体系结构
离散数学
数据结构
编译原理
操作系统
微电子学概论
集成电路原理与设计

北京大学计算机系专业课名称

计算机网络概论
数据库概论
软件工程
计算机图形学
面向对象技术引论

再版前言

编写一本适合本科生学习的软件工程教材，实在是一件很难的事情。其原因主要有三：一是软件工程这门课程所涉及的内容十分广泛，既涉及技术层面，又涉及管理层面；既关联实际问题的理解和描述，又关联软件工具的使用；……；其中既有哲学问题，又有方法学问题；二是在社会需求的拉动下，软件工程技术发展非常迅速，新概念、新技术、新方法不断出现，实在有些顾及不暇之势；三是作为一门学科，仅仅走过了30余年的发展历程，与其他学科相比，例如数学、物理、化学以及建筑、通信等，还是相当“年轻”的，但从另一个角度来说，仍是一门不算成熟的学科。因此，在教材内容的选取与组织方面，在有关概念的表述方面，真是一种挑战。

比较幸运的是，通过参与杨芙清院士主持的国家攻关项目，通过参与张效祥院士主编的《计算机科学技术百科全书》的编写，通过参与国家有关标准规范的制定，特别是通过几年来的教学实践，对软件工程这四个字还算有了一点领悟。因此，在本书的再版中，以原有教材为基础，进行了比较大的改动。

在教材内容的选取上，遵循以下两条原则，一是选取的内容能够有助于提高读者求解软件问题的能力，特别是提高读者直接参与软件开发实践和工程管理的能力；二是选取的内容应该是基础性的，是比较“稳定”的，但这并不意味着不能引入新的方法和技术，而是要讲清楚。

在教材内容的组织上，基本上以软件工程概念和相关的软件工程框架为“纲”，逐“目”展开，使概念与相关技术、方法有机地融为一体。

在概念的表述上，依据内容组织的特定层面，尽力引用《计算机科学技术百科全书》中有关的条目，并注重语义和概念之间关系的阐述。

由于时间仓促，更主要的是由于水平问题，再版中依然还会存在很多不足和错误，真诚地希望读者提出，并通过电子邮件(wlf@cs.pku.edu.cn)和其他方式，进行更有意义的讨论。

目　录

注：目录中带有 * 号的章节，不作为自考学生的考试内容。

第一章　软件工程概论

软件工程这一术语首次出现在1968年的NATO会议上。60年代以来，随着计算机的广泛应用，软件生产率、软件质量远远满足不了社会发展的需求，成为社会、经济发展的制约因素。当时，软件开发虽然有一些工具支持，例如编译连接器等，但基本上还是依赖开发人员的个人技能，没有可遵循的原理、原则和方法，也缺乏有效的管理。软件可靠性、可维护性较差，而且往往超出预期的开发时间要求。软件工程这一概念的提出，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"。

产生软件危机的原因很多，除了与软件本身固有的特征有关以外，还与软件开发范型、软件设计方法、软件开发支持以及软件开发管理等有关。

软件工程作为一门学科已有近30年的历史，其发展大体可划分为两个时期。

60年代末到80年代初，软件系统的规模、复杂性以及在关键领域的广泛应用，促进了软件开发过程的管理及工程化开发。这一时期主要围绕软件项目，开展了有关开发模型、支持工具以及开发方法的研究。其主要成果体现为：提出了瀑布模型；开发了诸多结构化语言（例如PASCAL语言、C语言、Ada语言等）和结构化方法（例如"自顶向下"方法），试图向程序员提供好的需求分析和设计方法，并开发了一些支持工具，例如调试工具等；开始出现各种管理方法，例如费用估算、文档复审，开发了一些相应支持工具，例如计划工具、配置管理工具等。这一时期的主要特征可概括为：前期主要研究系统实现技术，后期则开始强调管理及软件质量。

自"软件工厂"这一概念提出以来，80年代初主要围绕软件工程过程，开展了有关软件生产技术，特别是软件复用技术和软件生产管理的研究和实践。其主要成果是提出了具有广泛应用前景的面向对象方法和相关的语言（例如Smalltalk，C++，Eiffel等）；大力开展了计算机辅助软件工程（CASE）的研究与实践（例如我国在"七五"、"八五"、"九五"期间，均把这一研究作为国家重点科技攻关项目），各类CASE产品相继问世。其间，最显著的事件是过程改进项目，该项目的目标是在工业实践中，建立一种量化的评估程序，判定软件组织成熟的程度。

近几年来，软件工程的研究已从过程（管理）转向产品（开发），更加注重新的程序开发范型和软件生产。其中，围绕网络，特别是Internet网的广泛应用，以软件复用技术研究为基础，在软件构件技术及软件"平台"技术研究方面，在需求工程技术及领域分析技术研究方面，以及在软件体系结构、应用框架、面向应用的语言研究方面，均取得了非常有影响的成果，有力地促进了软件工程学科和软件产业的发展。

1.1　软件工程概念

计算机系统中的程序及其文档称为软件。其中，程序是计算机任务的处理对象和处理规则的描述；文档是为了理解程序所需的阐述性资料。细言之，软件一词具有三层含义。一为个体含义，即指计算机系统中的程序及其文档；二为整体含义，即指在特定计算机系统中所有上述个体含义下的软件的总称，亦即计算机系统中硬件除外的所有成分；三为学科含义，即指在

研究、开发、维护以及使用前述含义下的软件所涉及的理论、方法、技术所构成的学科。一般而言，工程是将科学论理和知识应用于实践的科学。在了解了“软件”和“工程”两个概念的基础上，软件工程可定义如下：

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、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、成本等管理。软件工程是一门指导计算机软件开发和维护的工程学科。

1.2 软件工程框架

软件工程与其他工程(例如土木工程)一样，有其自己的目标、活动和原则。软件工程框架如图 1.1 所示。

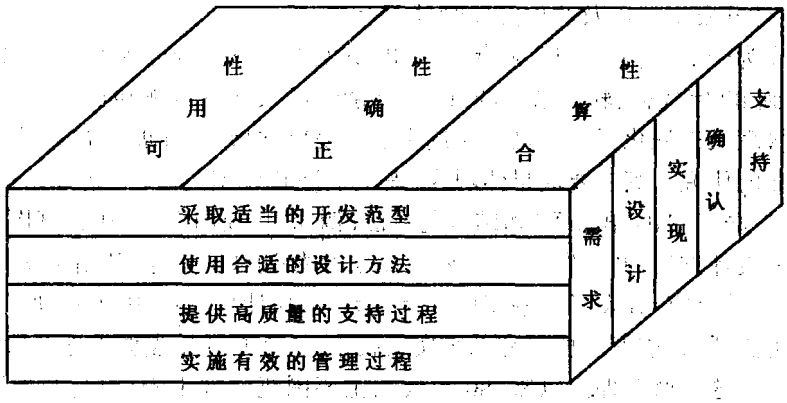

图 1.1 软件工程框架

软件工程的目标可概括为“生产具有正确性、可用性以及开销合宜的产品”。正确性意指软件产品达到预期功能的程度。可用性意指软件基本结构、实现及文档为用户可用的程度。开销合宜是指软件开发、运行的整个开销满足用户要求的程度。这些目标的实现不论在理论上还是在实践中均存在很多问题有待解决，它们形成了对过程、过程模型及工程方法选取的约束。

软件工程活动是“生产一个最终满足需求且达到工程目标的软件产品所需要的步骤”。主要包括需求、设计、实现、确认以及支持等活动。需求活动包括问题分析和需求分析。问题分析获取需求定义，又称软件需求规约。需求分析生成功能规约。设计活动一般包括概要设计和详细设计。概要设计建立整个软件体系结构，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。详细设计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。实现活动把设计结果转换为可执行的程序代码。确认活动贯穿于整个开发过程，实现完成后的确认，保证最终产品满足用户的要求。支持活动包括修改和完善。伴随以上活动，还有管理过程、支持过程、培训过程等。

围绕工程设计、工程支持以及工程管理，提出了以下四条基本原则：

第一条原则是选取适宜的开发模型。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。

第二条原则是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。

第三条原则是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。

第四条原则是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，仅当软件过程予以有效管理时，才能实现有效的软件工程。

在这一框架中，软件工程活动主要包括需求、设计、实现、确认和支持。其中，

(1) 需求的任务就是定义问题，即通过需求获取，得到一个需求陈述，继之，以需求陈述为基础，采用一种形式化或半形式化途径，给出被建系统的模型，进而按照一定的标准编制该系统的需求规格说明书，或曰需求规约；最后，还要验证需求陈述和需求规约之间的一致性、完整性、可跟踪性等。

(2) 软件设计的任务是在需求的基础上，给出被建系统的软件设计方案。一般来说，软件设计包括总体设计和详细设计。总体设计给出被建系统的软件体系结构，例如层次模块体系结构，消息总线体系结构，以数据库为中心的体系结构等。详细设计的任务是定义体系结构中的每一模块或构件，即给出它们的实现算法。由于在软件设计中采用的方法不同，因此一般来说其设计结果的表示风格也是不同的。

(3) 实现的任务是在软件设计的基础上，编码被建系统软件体系结构中的每一模块或构件，其方式可以是，或直接采购所需的模块和构件，或使用一种特定的语言或程序设计环境(例如 VB, VC)直接编码，或修改、包装已有的模块或构件。

(4) 确认工作贯穿软件开发的整个过程，主要包括需求复审、设计复审以及程序测试。其中应当注意的是，软件错误大部分出现在需求阶段，其次是出现在设计阶段，而在编码阶段出现的错误与前两个阶段相比是出现错误最少的阶段。

(5) 支持的任务是为系统的运行提供纠错性维护和完善性维护，即对系统运行中发现的错误进行改正，对系统运行中发现的欠缺进行修改。

综上所述，这一软件工程框架告诉我们，软件工程目标是可用性、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)以及软件经济学等。

习 题 一

1. 解释以下术语：

软件　　工程　　软件工程

2. 简述以下问题：

(1) 软件工程目标；

(2) 软件工程原则；

(3) 软件与程序之间的关系；

(4) 软件工程目标、原则和活动三者之间的关系。

3. 概要叙述软件工程各活动的主要任务和目标。

4. 简要叙述软件工程学科研究的内容。

软件工程活动：

第二章　软件开发模型

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、直观地表达软件开发全部过程，明确规定要完成的主要活动和任务，它用来作为软件项目工作的基础。模型都应该是稳定和普遍适用的。

软件开发包括需求、设计、编码和测试等阶段，有时也包括维护阶段。软件开发模型对于不同的应用系统，允许采用不同的开发手段和方法，使用各种不同的程序设计语言以及各种不同技能的人员参与工作，还应允许采用不同的软件工具或各种不同的软件工程环境。

最早出现的软件开发模型是 1970 年 W. Royce 提出的瀑布模型，而后随着软件工程学科的发展和软件开发的实践，相继提出了演化模型、螺旋模型、增量模型、喷泉模型等。

2.1　瀑布模型

瀑布模型将软件生存周期的各项活动规定为依固定顺序连接的若干阶段工作，形如瀑布流水，最终得到软件产品。

瀑布模型可追溯到 50 年代末期，当时人们已感到必须先确认“做什么”，才能编制程序将其实现，即使是比较简单的小型问题也不例外。最简单的两级瀑布模型如图 2.1 所示。

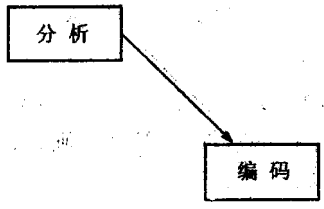

图 2.1　两级瀑布模型

对于较大软件项目，问题更加复杂，两级模型已不能满足软件开发的实际需要，一个更精确的软件开发步骤可按需要解决问题的顺序依次为：做什么—如何做—制作—检测—使用，于是一个反映软件过程的基本框架如图 2.2 所示。

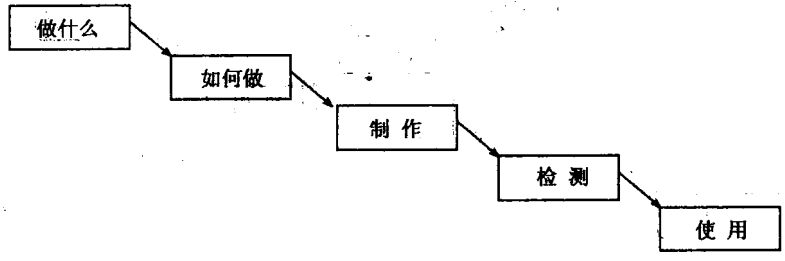

图 2.2　瀑布模型雏型

图 2.2 表明，首先应给出软件的目标，确定要做什么；然后要决定如何达到这一目标，给出策略、方法和步骤；继而加以实现，制作出所需要的软件；经过适当的检测，判定符合初始目标以后，方可投入运行和使用。可以说这是瀑布模型的雏型。

1970 年 W. Royce 首先将这一模型精确化，提出了具有多个开发阶段的瀑布模型，如图 2.3所示。

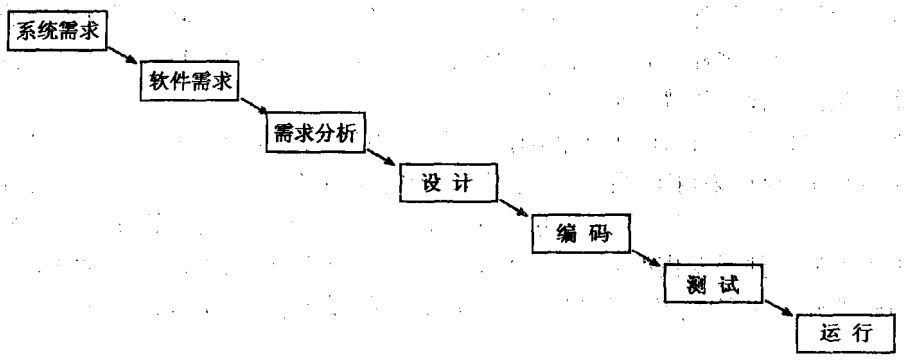

图 2.3 初始瀑布模型

这一模型规定了各开发阶段的活动为：提出系统需求、提出软件需求、需求分析、设计、编码、测试和运行，并且还规定了自上而下相互衔接的固定顺序，于是构成了人们熟知的瀑布模型。然而实践表明，各开发阶段间的关系并非完全是自上而下的线性图式，软件开发的实际情况是，每个开发阶段均具有以下特征：

(1) 从上一阶段接受本阶段工作的对象，作为输入；

(2) 对上述输入实施本阶段的活动；

(3) 给出本阶段的工作成果，作为输出传入下一阶段；

(4) 对本阶段工作进行评审，若本阶段工作得到确认，则继续下阶段工作；否则返回前一阶段，甚至更前阶段。

为表达向前阶段的反馈，在模型图中增加了虚线表示的箭头，构成了具有反馈回路的瀑布模型，如图 2.4 所示。

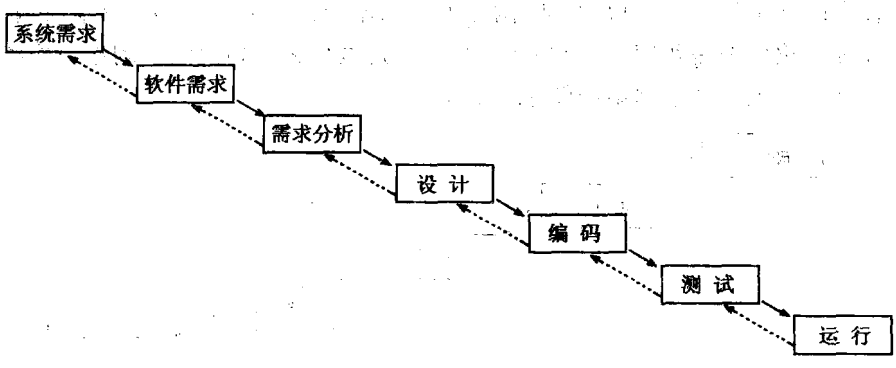

图 2.4 初始瀑布模型

瀑布模型有着不同形式的变种，比如，另一常见的具有反馈回路的瀑布模型包括的七个阶段是：可行性研究、需求分析和规约、设计和规约、编码和单元测试、集成测试和系统测试、交付、维护。不同形式瀑布模型的变种之间并无本质差别，选择哪一种形式可由软件项目特性及开发组织决定。

许多采用瀑布模型的开发组织为有效地组织实施，制定了软件开发规范或开发标准，其中明确规定了各个开发阶段应交付的产品。这就为严格控制软件开发项目的进度，最终按时交付产品以及保证软件产品质量创造了有利条件。

瀑布模型20多年来之所以广泛流行，是因为它在支持结构化软件开发、控制软件开发的复杂性、促进软件开发工程化等方面起着显著作用。与此同时，瀑布模型在大量软件开发实践中也逐渐暴露出它的缺点。其中最为突出的缺点是该模型缺乏灵活性，无法通过开发活动澄清本来不够确切的软件需求，这些问题可能导致开发出的软件并不是用户真正需要的软件，无疑要进行返工或不得不在维护中纠正需求的偏差，为此必须付出高额的代价，为软件开发带来了不必要的损失。并且，随着软件开发项目规模的日益庞大，该模型的不足所引发的问题显得更加严重。

2.2 演化模型

演化模型主要针对事先不能完整定义需求的软件开发。用户可以给出待开发系统的核心需求，并且当看到核心需求实现后，能够有效地提出反馈，以支持系统的最终设计和实现。软件开发人员根据用户的需求，首先开发核心系统。当该核心系统投入运行后，用户试用之，完成他们的工作，并提出精化系统、增强系统能力的需求。软件开发人员根据用户的反馈，实施开发的迭代过程。每一迭代过程均由需求、设计、编码、测试、集成等阶段组成，为整个系统增加一个可定义的、可管理的子集。如图2.5所示。

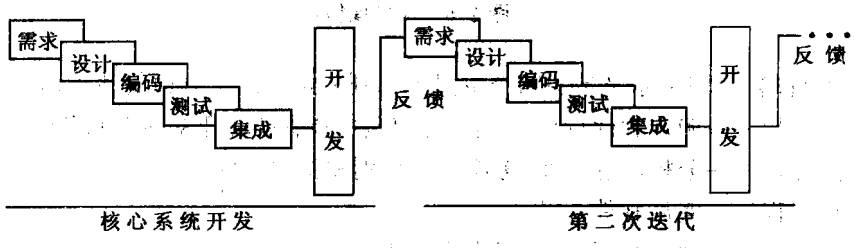

图2.5 演化模型

如果在一次迭代中，有的需求不能满足用户的要求，可在下一次迭代中予以修正。

演化模型在一定程度上减少了软件开发活动的盲目性。

2.3 螺旋模型

螺旋模型是在瀑布模型和演化模型的基础上，加入两者所忽略的风险分析所建立的一种软件开发模型。该模型于1988年由TRW公司B·鲍姆(Barry W.Boehm)提出。

软件风险是任何软件开发项目中普遍存在的问题，不同项目其风险有大有小。在制定软件开发计划时，系统分析员必须回答：项目的需求是什么，需要投入多少资源以及如何安排开发进度等一系列问题。然而若要他们当即给出准确无误的回答是不容易的，甚至几乎是不可能的。但系统分析员又不可能完全回避这一问题。凭借经验的估计给出初步的设想便难免带来一定风险。实践表明，项目规模越大，问题越复杂，资源、成本、进度等因素的不确定性就越大，承担项目所冒的风险也越大。风险是软件开发不可忽视的潜在不利因素，它可能在不同程度上损害到软件开发过程和软件产品的质量。软件风险驾驭的目标是在造成危害之前，及时对风险进行识别、分析，采取对策，进而消除或减少风险的损害。

螺旋模型如图 2.6 所示。

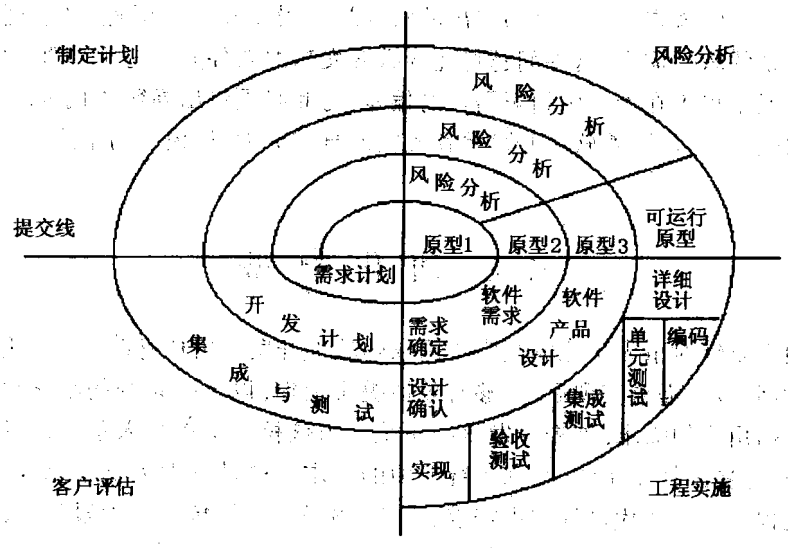

图 2.6 螺旋模型

沿着螺线旋转，在笛卡尔坐标的四个象限上分别表达了四个方面的活动，即：

(1) 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；

(2) 风险分析——分析所选方案，考虑如何识别和消除风险；

(3) 实施工程——实施软件开发；

(4) 客户评估——评价开发工作，提出修正建议。

沿螺线自内向外每旋转一圈便开发出更为完善的一个新的软件版本。例如，在第一圈，确定了初步的目标、方案和限制条件以后，转入右上象限，对风险进行识别和分析。如果风险分析表明，需求具有不确定性，那么在右下的工程象限内，所建的原型会帮助开发人员和客户，考虑其他开发模型，并把需求作进一步修正。

客户对工程成果作出评价后，给出修正建议。在此基础上需再次计划，并进行风险分析。在每一圈螺线的风险分析的终点作出是否继续下去的判断。假如风险过大，开发者和用户无法承受，项目有可能终止。多数情况下沿螺线的活动会继续下去，自内向外逐步延伸，最终得到所期望的系统。图 2.7 给出了螺旋模型的另一图示。

如果对所开发项目的需求已有了较好的理解或较大的把握，无需开发原型，便可采用普通

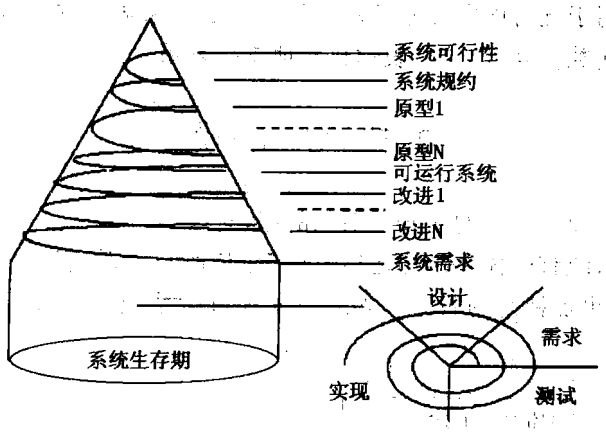

图 2.7 螺旋模型的另一种表示

的瀑布模型。这在螺旋模型中可认为是单圈螺线。与此相反，如果对所开发项目的需求理解较差，需要开发原型，甚至需要不止一个原型的帮助，那就要经历多圈螺线。在这种情况下，外圈的开发包含了更多的活动。也可能某些开发采用了不同的模型。

螺旋模型适合于大型软件的开发，它是颇为实际的方法，它吸收了 T. Gilb 提出的软件工程"演化"概念。使得开发人员和客户对每个演化层出现的风险均有所了解，并继而作出反应。和其他模型相比，螺旋模型的优越性较为明显，但要求许多客户接受和相信演化方法并不容易。本模型的使用需要具有相当丰富的风险评估经验和专门知识。如果项目风险较大，又未能及时发现，势必造成重大损失。此外，螺旋模型是出现较晚的新模型，远不如瀑布模型普及，要让广大软件人员和用户接受，还有待于更多的实践。

2.4 喷 泉 模 型

喷泉模型体现了软件创建所固有的迭代和无间隙的特征。喷泉模型如图 2.8 所示。

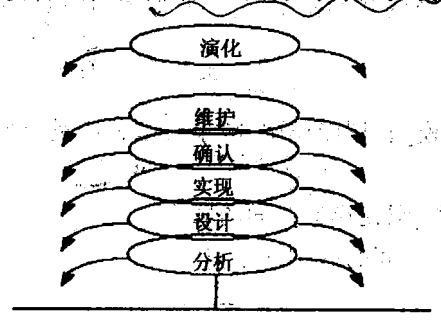

图 2.8 喷泉模型

这一模型表明了软件刻画活动需要多次重复。例如，在编码之前，再次进行分析和设计，其间，添加有关功能，使系统得以演化。同时，该模型还表明活动之间没有明显的间隙，例如在分析和设计之间没有明显的界限。