UML与Rational Rose 2003

软件工程

统一建模原理与实践教程

国 刚 周 峰 孙更新 编著

本书的主要特点

◎—理论与实例结合，内容繁简得当、由浅入深，使读者能十分容易地入门并逐步提高软件建模的能力。

◎—详细论述了 UML 的 9 个图所涉及的关键概念、术语和技术，同时还包括了数据建模、业务建模等 UML 具体应用的介绍。

◎—从理论的高度阐述了面向对象分析和设计的思想，使读者能够真正掌握系统架构设计的精髓。

◎—通过图书借阅管理系统对使用 Rational Rose 进行 UML 建模的全过程进行了深入剖析，加强读者的实践经验。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

UML与Rational Rose 2003
软件工程统一建模原理与实践教程

国 刚 周 峰 孙更新 编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书主要介绍统一建模语言UML的基础知识及Rational Rose 2003工具的使用方法。全书内容丰富，包括对软件工程思想、面向对象思想、UML相关概念、Rational Rose工具等方面内容的详细介绍，本书最后是一个图书馆借阅系统的研究实例，通过该综合实例，对使用Rational Rose进行UML建模的全过程进行了深入剖析。此外，本书每章后面配有一些习题，完成这些习题可以使读者加深对UML的认识。

本书可以作为大专院校计算机软件工程专业学生学习UML和面向对象技术的教材，也可作为广大软件开发人员和系统架构分析设计人员自学UML的参考和指导用书。

图书在版编目（CIP）数据

UML与Rational Rose 2003软件工程统一建模原理与实践教程/国刚，周峰，孙更新编著.—北京：电子工业出版社，2007.4

ISBN 978-7-121-03836-5

Ⅰ.U… Ⅱ.①国… ②周… ③孙… Ⅲ.①面向对象语言，UML—程序设计—教材 ②实时操作系统—程序设计—教材 Ⅳ.TP31

中国版本图书馆CIP数据核字（2007）第013862号

责任编辑：徐云鹏
特约编辑：卢国俊
印　　刷：北京天竺颖华印刷厂
装　　订：三河市金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编：100036
　　北京市海淀区翠微东里甲2号　邮编：100036
开　　本：787×1092 1/16　印张：18.5　字数：470千字
印　　次：2007年4月第1次印刷
定　　价：28.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系电话：（010）68279077。邮购电话：（010）88254888。
质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。
服务热线：（010）88258888。

前言

本书以介绍面向对象的统一建模语言UML为主，目的是使读者了解面向对象技术的基本概念和理论，掌握面向对象的分析和系统设计方法，以及与面向对象技术相关的一些软件开发技术，同时掌握使用Rational Rose 2003集成开发工具进行UML的分析和设计。

UML是一种定义良好、易于表达、功能强大且适用于各种应用领域的建模语言，已经被OMG采纳为标准。目前UML已经成为面向对象技术领域内占主导地位的标准建模语言。掌握UML语言，不仅有助于理解面向对象的分析与设计方法，也有助于对软件开发全过程的理解。

本书共分13章。

第1章　软件工程与UML概述

本章重点是对软件工程中所涉及的一些知识点的介绍。

第2章　面向对象技术与UML

本章重点介绍面向对象技术与UML的关系，并且介绍基于UML的面向对象分析、设计过程。

第3章　UML——统一建模语言

本章主要目的是让读者对UML有一个总体的印象。

第4章　Rational Rose 2003基础与操作

本章主要介绍Rational Rose 2003的使用。

第5章　用例图

本章介绍UML中一个非常重要的概念——用例图。

第6章　静态视图

本章重点讲解了UML中的静态视图。

第7章　动态视图——状态图

本章重点讲解了UML中的状态图。

第8章　动态视图——活动图

本章重点讲解了UML动态视图中的活动图。

第9章　动态视图——时序图

本章重点讲解了UML动态视图中的时序图。

第10章　动态视图——协作图

本章重点讲解了UML中的协作图。

第11章　UML实现与部署——组件图与配置图

本章重点讲解了UML中的组件图与配置图。

第12章　Rose中的双向工程

本章重点讲解了UML中双向工程技术。

第13章　图书管理系统

本章是一个实际的开发案例，通过对案例的模型内部结构进行深入剖析，并使用双向工程自动生成实际代码。

本书在编写过程中得到了青岛大学国际学院在技术上的大力支持，唐小松老师负责了本书第2章、第3章的编写工作，此外宫生文等老师为本书的编写提出过宝贵意见，并参与了本书的部分资料搜集工作，感谢北京美迪亚电子信息有限公司的各位老师，谢谢你们的帮助和指导。

尽管我们尽了最大努力，但由于时间仓促，加之水平有限，本书难免有不妥之处，欢迎各界专家和读者朋友批评指正。

目　录

第1章　软件工程与UML概述

课前导读

面向对象技术出现于20世纪70年代末期，它具有强大的生命力。UML（Unified Modeling Language，统一建模语言）是一个通用的可视化建模语言，用于对软件进行描述、可视化处理、构造和建立软件系统制品的文档。它是由Booch、Rumbaugh和Jacobson发起，在Booch方法、OMT方法和OOSE方法的基础上，集众家之长，几经修改而成的。

重点提示

本章讲解了软件工程相关概念，包括软件工程的发展历史和生命周期，及现代软件工程；建模的目的、意义与原理；UML的历史、定义、内容及应用领域。

- ▲ 软件工程的生命周期
- ▲ 软件开发模型
- ▲ 瀑布模型
- ▲ 原型模型
- ▲ 螺旋模型
- ▲ 喷泉模型
- ▲ 建模的重要性
- ▲ 建模原理
- ▲ 面向对象建模
- ▲ UML的定义
- ▲ UML包含的内容

1.1　软件工程概述

1.1.1　软件工程的发展历史

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”。

1.1.1.1　软件与软件危机

软件是计算机系统中与硬件相互依存的另一部分，它是包括程序、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：

- 程序设计阶段，约为20世纪50至60年代；
- 程序系统阶段，约为20世纪60至70年代；
- 软件工程阶段，约为20世纪70年代以后。

20世纪60年代中期以后，一些开发大型软件系统的要求提了出来。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：

（1）用户需求不明确、变更过多；
（2）软件成本日益增长；
（3）开发进度难以控制；
（4）软件质量差；
（5）软件维护困难。

软件危机产生的原因如下：

（1）软件开发无计划性；
（2）软件需求不充分；
（3）软件开发过程无规范；
（4）软件产品无评测手段。

软件是逻辑部件：试制阶段难衡量；开发质量较难评价；开发过程较难管理和控制；运行过程中才能暴露没有检测出来的事故，相当于修改设计，软件维护困难；对于大型软件系统，软件规模庞大，产生软件危机既有技术问题，也有管理方法问题；早期开发的个体化，忽视需求分析；认为软件开发就是写程序，轻视维护，对用户不了解。因此，设计软件系统时，前期工作不能忽视，做好软件定义时期的工作，这是降低成本、提高软件质量的关键。

1.1.1.2 软件工程的出现

“软件危机”使得人们开始对软件及其特性进行更深一步的研究，人们改变了早期对软件的不正确看法。早期那些被认为是优秀的程序常常很难被别人看懂，通篇充满了程序技巧。现在人们普遍认为优秀的程序除了功能正确、性能优良之外，还应该容易看懂、容易使用、容易修改和扩充。

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，北大西洋公约组织在原西德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、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。

软件工程的方法、工具、过程构成了软件工程的三要素。

软件工程有七条基本原理，如下所示。

1. 用分阶段的生存周期计划严格管理

这条基本原理指出，应该把软件生存周期划分成若干个阶段，并相应地制定出切实可行的计划，然后严格按照计划对软件开发与维护工作进行管理。应该制定的计划有项目概要计划、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等。各级管理人员都必须严格按照计划对软件开发和维护工作进行管理。据统计，不成功的软件项目中，有一半左右是由于计划不周造成的。

2. 坚持进行阶段评审

据统计，在软件生存周期各阶段中，编码阶段之前的错误约占63%，而编码错误仅占37%。另外，错误发现并改正得越晚，所花费的代价越高。坚持在每个阶段结束前进行严格的评审，就可以尽早发现错误，从而以最小的代价改正错误。因此，这是一条必须坚持的重要原理。

3. 实行严格的产品控制

决不能随意改变需求，只能依靠科学的产品控制技术来顺应用户提出的改变需求的要求。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（又称为变动控制），即凡是修改软件的建议，尤其是涉及基本配置的修改建议，都必须按规程进行严格的评审，评审通过后才能实施。

这里的“基准配置”是指经过阶段评审后的软件配置成分，即各阶段产生的文档或程序代码等。

4. 采用现代程序设计技术

实践表明，采用先进的程序设计技术既可以提高软件开发与维护的效率，又可以提高软件的质量。多年来，人们一直致力于研究新的“程序设计技术”。比如，20世纪60年代末提出的结构程序设计技术；后来又发展出各种结构分析（SA）和结构设计（SD）技术；之后又出现了面向对象分析（OOA）和面向对象设计（OOD）技术等。

5. 结果应能清楚地审查

软件产品是一种看不见、摸不着的逻辑产品。因此，软件开发小组的工作进展情况可见性差，难于评价和管理。为了更好地进行评价与管理，应根据软件开发的总目标和完成期限，尽量明确地规定软件开发小组的责任和产品标准，从而使所得到的结果能清楚地被审查。

6. 开发小组的人员应少而精

软件开发小组人员素质和数量是影响软件质量和开发效率的重要因素。实践表明，素质高的人员与素质低的人员相比，开发效率可能高几倍至几十倍，而且所开发的软件中的错误也要少得多。另外，开发小组的人数不宜过多，因为随着人数的增加，人员之间交流情况、讨论问题的通信开销将急剧增加，这不但不能提高生产率，反而由于误解等原因可能增加出错的概率。

7. 承认不断改进软件工程实践的必要性

遵循上述六条基本原理，就能够较好地实现软件的工程化生产。但是，软件工程不能停留在已有的技术水平上，应积极主动地采纳或创造新的软件技术，要注意不断总结经验，收集工作量、进度、成本等数据，并进行出错类型和问题报告的统计。这些数据既可用来评估新的软件技术的效果，又可用来指明应优先进行研究的软件工具和技术。

软件工程的目标是在给定成本、进度的前提下，开发出具有可修改性、有效性、可靠性、可理解性、可维护性、可重用性、可适应性、可移植性、可追踪性和可互操作性且满足用户需求的软件产品。

“可靠性”这个目标在软件工程中有着重要的意义。广义上讲，它涉及到产品设计的一系列问题，从而使产品能在相当长的时期内稳定工作。狭义上讲，可靠性是软件成功运行的概率度量，可靠性分析和可靠性测试可作为衡量软件质量和其他开发过程的最重要的方法之一。

1.1.2 软件生命周期及软件开发模型

软件生命周期表明软件从功能确定、设计到开发成功投入使用，并在使用中不断地修改、增补和完善，直至被新的需要所替代而停止使用该软件的全过程。

我们可以将软件生存周期划分为3个过程共9个阶段：

3个过程是：软件定义过程、软件开发过程、软件使用与维护过程。

9个阶段有：可行性研究、需求分析、概要设计、详细设计、实现、组装测试、验收测试、使用与维护、退役。

它们之间的关系如图1-1所示。

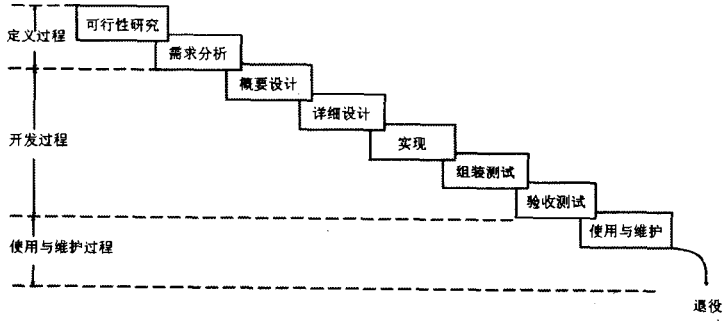

图1-1 软件生存周期阶段的划分

1.1.2.1 软件定义过程

软件定义的基本任务是确定软件系统的工程需求，也就是要搞清“做什么”。软件定义过程可通过软件系统的可行性研究和需求分析两个阶段来完成。

1. 可行性研究

本阶段的任务是根据用户提出的工程项目的性质、目标和规模，进一步了解用户的要求及现有的环境及条件，从技术、经济和社会等多方面研究并论证该项目的可行性。即该项目是否值得去解决，是否存在可行的解决办法。

此时，系统分析人员应在用户的配合下对用户的要求和现有的环境进行深入调查并写出调研报告。进而进行可行性论证。可行性论证包括经济可行性、技术可行性、操作可行性、法律可行性等。在此基础上还要制定初步的项目计划，包括需要的软硬件资源、定义任务、风险分析、成本/效益分析以及进度安排等。

可行性研究的结果是使用部门负责人做出是否继续进行该项目决定的重要依据。

2. 需求分析

（1）需求分析的任务

需求分析的任务是确定待开发的软件系统“做什么”。

具体任务包括确定软件系统的功能需求、性能需求和运行环境约束，编制软件需求规格说明书、软件系统的验收测试准则和初步的用户手册。

（2）需求分析的实现途径

软件系统需求一般由用户提出。系统分析员和开发人员在需求分析阶段必须与用户反复讨论、协商，充分交流信息，并用某种方法和工具构建软件系统的逻辑模型。为了使开发方与用户对待开发软件系统达成一致的理解，必须建立相应的需求文档。有时对大型、复杂的软件系统的主要功能、接口、人机界面等还要进行模拟或建造原型，以便向用户和开发方展示待开发软件系统的主要特征。确定软件需求的过程有时需要反复多次，最终得到用户和开发者的确认。

（3）需求分析的阶段成果

需求分析阶段的主要成果有软件需求规格说明、软件验收测试计划和准则、初步的用户手册等。其中，软件需求规格说明（Software Requirements Specification，即SRS）是一个关键性的文档。多数场合，面向开发者的软件需求用需求规格说明语言来描述，它是软件开发人员进行软件设计的依据；另一方面，从某种意义上讲，SRS又起到与用户签定合同的合同书的作用。因此，在SRS中应包括软件系统的全部功能需求、性能需求、接口需求、设计需求、基本结构、开发标准和验收准则等。

1.1.2.2 软件开发过程

软件开发过程由概要设计、详细设计、实现（即编码与单元测试）、组装测试、验收测试5个阶段组成。其中，概要设计和详细设计统称为设计；编码即编程；单元测试、组装测试和验收测试统称为测试。开发者通常可提出多种设计方案，并对各种方案在功能、性能、成本、进度等方面进行比较，从中选出一种“最佳方案”。

下面将简单地介绍软件开发过程中各阶段的任务、实现的途径和阶段成果。

1. 概要设计——总体设计

- 任务：是对需求规格说明中提供的软件系统逻辑模型进行进一步的分解，从而建立软件系统的总体结构和各子系统之间、各模块之间的关系，定义各子系统接口界面和各功能模块的接口，设计全局数据库或数据结构，规定设计约束，制定组装测试计划，进而给出每个功能模块的功能描述、全局数据定义和外部文件定义等。
- 实现的途径：选择某种方法和工具。设计的软件系统应具有良好的总体结构，尽量降低模块接口的复杂度，并力争做到各功能模块之间的低耦合度，而功能模块内部具有较高的内聚度。
- 阶段性成果包括：概要设计说明书、数据库或数据结构说明书、组装测试计划等文档。

2. 详细设计

- 任务：是将概要设计产生的功能模块进一步细化，形成可编程的程序模块，然后设计程序模块的内部细节，包括算法、数据结构以及各程序模块间的接口信息，并设计模块的单元测试计划。
- 途径：可以采用结构化的设计方法，采用结构化的程序流程图、N-S图、过程设计语言（PDL，Procedure Design Language）等工具进行描述，也可以采用面向对象设计方法等。
- 阶段成果：应提供“详细设计规格说明”（或称“模块开发卷宗”）和单元测试计划等详细设计文档。

3. 实现编码和单元测试

- 编码的主要任务是根据详细设计规格说明，用某种选定的程序设计语言把详细设计的结果转化为机器可运行的源程序模块，这是一个编程和调试程序的过程。一般来说，对软件系统所采用的分析方法、设计方法、编程方法以及所选用的程序设计语言应尽可能保持一致。编码阶段应注意遵循编程标准、养成良好的编程风格，以便编写出正确的便于理解、调试和维护的程序模块。
- 单元测试：每编写出一个程序模块的源程序，调试通过后，即对该模块进行测试，这称为单元测试。
- 实现阶段的成果：按一定规则存在盘上的通过单元测试的各功能模块的集合、详细的单元测试报告等文档。

4. 组装测试

- 组装测试：根据概要设计提供的软件结构、各功能模块的说明和组装测试计划，把经过单元测试检验的模块按照某种选定的策略逐步进行组装和测试。
- 主要任务：测试系统各模块间的连接是否正确，系统或子系统的正确处理能力、容错能力、输入/输出处理是否达到要求。
- 阶段成果：满足概要设计要求、可运行的软件系统和源程序清单、组装测试报告等文档。

5. 验收测试——确认测试

- 任务：按照验收测试计划和准则对软件系统进行测试，看其是否达到了需求规格说明中定义的全部功能和性能等方面的需求。
- 确认测试结束时，应生成验收测试报告、项目开发总结报告，并向用户提交源程序清单、最终用户手册、操作手册等文档资料。
- 最后，由专家、用户负责人、软件开发和管理人员组成的软件评审小组要对软件验收测试报告、测试结果和软件进行评审，通过后，软件产品正式通过验收（即完成了开发合同），可以交付用户使用了。

1.1.2.3 软件使用与维护过程

- 任务：通过各种维护活动使软件系统持久地满足用户的需求。
- 每项维护活动实质上都是一次压缩和简化了的软件定义和软件开发过程。都要经历提出维护要求、分析维护要求、提出维护方案、审批维护方案、确定维护计划、修改软件设计、修改程序、测试程序、评审、验收等步骤。

软件在使用的过程中，应及时收集被发现的软件错误，并定期撰写“软件问题报告”；而每一项维护活动都应该准确地记录下来，并作为正式的文档资料保存。据统计，软件维护人员为了分析和理解原软件系统所花费的工作量约占整个维护工作量的60%以上。在软件开发的过程中应重视对软件可维护性的支持。

1.1.2.4 软件开发模型

软件开发模型（又称为软件生存周期模型）是从软件项目需求定义开始，直至软件经使用后废弃为止，跨越整个生存期的系统开发、运作和维护所实施的全部过程、活动和任务的结构框架。

它指出了软件开发过程各阶段之间的关系和顺序，是软件开发过程的概括。它为软件开发过程提供原则和方法，并为软件工程管理提供里程碑和进度表。因此，软件开发模型也是软件工程的重要内容。

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，相继提出了原型模型、螺旋模型、喷泉模型等。在实际开发时，应根据项目的特点和现有的条件选取合适的模型，也可以把几种模型组合起来使用以便充分利用各模型的优点。下面对提到的这几种模型分别进行介绍。

1. 瀑布模型

瀑布模型（waterfall model）是由W.Royce于1970年提出来的，又称为软件生存周期模型。瀑布模型规定了各项软件工程活动，包括制定开发计划、进行需求分析和说明、软件设计、程序编码、测试及运行维护，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，如图1-2所示。

瀑布模型严格按照软件生存周期各个阶段来进行开发，上一阶段的输出即是下一阶段的输入，并强调每一阶段的严格性。它规定了各阶段的任务和应提交的成果及文档，每一阶段

的任务完成后，都必须对其阶段性产品（主要是文档）进行评审，通过后，才能开始下一阶段的工作。因此，它是一种以文档作为驱动的模型。

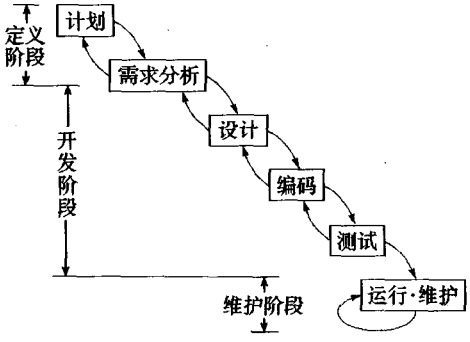

图1-2 软件生存周期的瀑布模型

然而软件开发的实践表明，上述各项活动之间并非完全是自上而下，呈线性图式。实际情况是，每项开发活动均处于一个质量环（输入－处理－输出－评审）中。只有当其工作得到确认，才能继续进行下一项活动（在图1-2中用向下的箭头表示）；否则返工（在图1-2中由向上的箭头表示）。

瀑布模型的优点：提供了软件开发的基本框架，有利于大型软件开发过程中人员的组织、管理，有利于软件开发方法和工具的研究与使用，因此，在软件工程中占有重要的地位。

瀑布模型的缺点：

（1）在软件开发的初期阶段就要求做出正确、全面、完整的需求分析，这对许多应用软件来说是极其困难的。

（2）在需求分析阶段，当需求确定后，无法及时验证需求是否正确、完整。

（3）作为整体开发的瀑布模型，由于不支持产品的演化，缺乏灵活性，对开发过程中很难发现的错误，只有在最终产品运行时才能暴露出来，从而使软件产品难以维护。

瀑布模型一般适用于功能、性能明确、完整、无重大变化的软件系统的开发。例如操作系统、编译系统、数据库管理系统等系统软件的开发。应用有一定的局限性。

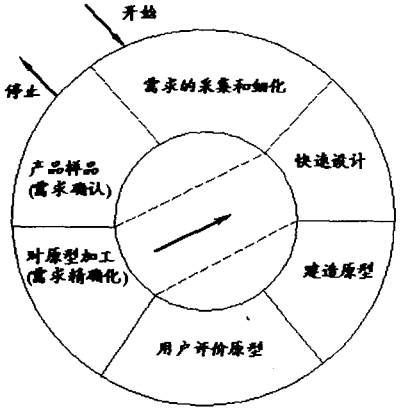

图1-3 软件生存周期的原型模型

2. 原型模型

原型模型（prototyping model）的基本框架是软件开发人员根据用户提出的软件基本需求快速开发一个原型，以便向用户展示软件系统应有的部分或全部功能和性能，在征求用户对原型的评价意见后，进一步使需求精确化、完全化，并据此改进、完善原型，如此迭代，直到软件开发人员和用户都确认软件系统的需求并达成一致的理解为止。软件需求确定后，便可进行设计、编码、测试等各个开发步骤。如图1-3所示。