普通高等教育
软件工程 “十二五”规划教材

12th Five-Year Plan Textbooks
of Software Engineering

工业和信息化普通高等教育
“十二五”规划教材

UML系统分析与设计教程（第2版）

冀振燕 ◎ 编著

人民邮电出版社
POSTS & TELECOM PRESS

普通高等教育
软件工程
“十二五”规划教材

12th Five-Year Plan Textbooks
of Software Engineering

工业和信息化普通高等教育
“十二五”规划教材

UML系统分析与设计教程（第2版）

冀振燕 ◎ 编著

System Analysis and Design with UML

人民邮电出版社
北京

图书在版编目（CIP）数据

UML系统分析与设计教程 / 冀振燕编著. -- 2版. -- 北京 : 人民邮电出版社, 2014.8
普通高等教育软件工程“十二五”规划教材
ISBN 978-7-115-34990-3

Ⅰ. ①U… Ⅱ. ①冀… Ⅲ. ①面向对象语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2014)第061422号

内 容 提 要

本书介绍了UML语言的基础知识以及UML在面向对象的软件系统分析与设计中的应用，并通过实例讲解了面向对象分析与设计过程，以及如何用UML语言为系统建模。

本书通过丰富的实例启发读者如何将所学到的面向对象技术应用于软件系统的分析、设计与开发中。

本书可作为高等院校计算机相关专业UML建模、面向对象分析与设计等课程的教材，也可作为软件设计与开发人员的参考用书。

◆ 编　　著　冀振燕
责任编辑　张立科
执行编辑　刘　博
责任印制　彭志环　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：17.25　　2014年8月第2版
字数：454千字　　2014年8月北京第1次印刷

定价：39.00元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前　言

在 20 世纪 90 年代初，不同的面向对象方法具有不同的建模符号体系，这些不同的符号体系极大地妨碍了软件的设计人员、开发人员和用户之间的交流。因此，有必要在分析、比较不同的建模语言以及总结面向对象技术应用实践的基础上，建立一个标准的、统一的建模语言，UML 就是这样的建模语言。UML 1.1 于 1997 年 11 月 17 日被对象管理组织（OMG）采纳成为基于面向对象技术的标准建模语言。UML 2.0 对 UML1.x 进行了很多重大修改，并于 2005 年被 OMG 采纳。

统一建模语言 UML 不仅统一了 Grady Booch、James Rumbaugh 和 Ivar Jacobson 所提出的面向对象方法中的符号表示，而且在此基础上进一步发展，并最终统一为被相关专业人员所接受的标准建模语言。

UML 是可视化（Visualizing）、规范定义（Specifying）、构造（Constructing）和文档化（Documenting）的建模语言。可视化意味着系统的 UML 模型是图形化的，可视化模型的建立为软件的设计人员、开发人员、用户和领域专家之间的交流提供了便利；规范定义意味着用 UML 建立的模型是准确的、无歧义的、完整的；构造意味着可以将 UML 模型映射到代码进行实现；文档化意味着 UML 可以为系统的体系结构以及系统的所有细节建立文档。

UML 目前已成为面向对象软件系统分析与设计的必要工具，是软件设计人员、开发人员的必备知识。

本书由 17 章组成。第 1 章介绍了 UML 的历史、内容、特点、功能、组成和工具，还对 Rational 的统一过程——RUP 进行了较详细的介绍。第 2 章介绍了常用的面向对象分析与设计方法，包括 OOA/OOD 方法、OMT 方法、Booch 方法、OOSE 方法和 Fusion 方法。第 3 章讲解了 UML 的 4 种关系，即依赖关系、类属关系、关联关系和实现关系的语义、符号表示、衍型和应用。第 4 章讲解了 UML 各种符号的语义、符号表示以及其应用，还讲解了 UML 的扩充机制。第 5 章介绍了视与图的关系，从而指导读者选择 UML 图对系统 5 个视图的静态方面和动态方面进行建模。第 6 ~ 11 章详细描述了 UML2.x 中的用例图、类图、对象图、包图、顺序图、通信图、活动图、状态机图、组件图和部署图，并详细介绍了它们的语义、功能和应用。第 12 章讲解了从 UML 类模型到对象数据库、对象关系数据库或关系数据库的逻辑模型的映射。第 13 章以“图书管理系统”的面向对象分析与设计过程为例，讲解了如何用 UML 为系统建模。第 14 章以“银行系统”的面向对象分析与设计过程为例，讲解了如何用 UML 为系统建模。第 15 章以“便携式心电记录仪”的面向对象分析与设计过程为例，讲解了如何用 UML 为嵌入式系统建模。第 16 章讲解了如何用 UML 为 Web 应用程序建模。第 17 章对 Rational Rose 中 UML 模型与 C++、VisualC++/VisualBasic、Java 语言之间的前向工程和逆向工程进行了详解。

本书的最大特点是结合丰富的实例介绍 UML 基础知识，使得本书对理论的讲

解生动具体、直观易懂。在读者完成基础知识的学习后，可以通过对书中综合实例的进一步学习，将所学到的面向对象技术应用于软件系统的分析、设计与开发中。

本书作者曾在瑞典的 Mid Sweden University 做过 8 年的基于 UML 的面向对象技术教学工作，目前在北京交通大学做软件工程相关教学工作，本书就是根据作者多年教学的丰富经验精心编写而成的。本书兼顾了理论知识与实际应用，是一本内容全面的面向对象技术与 UML 建模著作。本书在内容结构上的安排符合学生学习新知识的习惯。

本书可作为大专院校 OOAD 或 UML 建模课程的教材，还可作为软件设计人员、开发人员的参考用书。

作者

2013 年 8 月

目　录

第 1 章 绪论

1.1 统一建模语言 UML

面向对象分析与设计（Object-Oriented Analysis and Design，OOA&D 或 OOAD）方法的发展曾在 20 世纪 80 年代末至 20 世纪 90 年代中出现过一个高潮，UML 就是这个高潮的产物。UML 不仅统一了 Grady Booch、James Rumbaugh 和 Ivar Jacobson 所提出的面向对象方法中的符号表示，而且在此基础上进一步发展，并最终被统一为广大开发者所接受的标准建模语言。

1.1.1 UML 的背景

公认的面向对象建模语言出现于 20 世纪 70 年代中期。从 1989 年到 1994 年，面向对象建模语言的数量从不到 10 种增加到了 50 多种，这些不同的面向对象建模语言具有不同的建模符号体系，且各有优劣，使用户很难找到一个完全满足自己要求的模型语言。另外，由于采用不同的建模语言，极大地妨碍了软件设计人员、开发人员和用户之间的交流。因此，有必要在分析、比较不同的建模语言以及总结面向对象技术应用实践的基础上，博采众长，建立一个标准的、统一的建模语言。

20 世纪 90 年代，3 个最流行的面向对象方法是 OMT 方法（由 James Rumbaugh 提出）、Booch 方法（由 Grady Booch 提出）和 OOSE 方法（由 Ivar Jacobson 提出），且每个方法都有自己的价值和重点。OMT 方法的强项是分析，弱项是设计；Booch 方法的强项是设计，弱项是分析；OOSE 方法擅长行为分析，而在其他方面表现较弱。

在 20 世纪 90 年代中期，Grady Booch、Ivar Jacobson、James Rumbaugh 开始借鉴彼此的方法，其中 Grady Booch 采用了 James Rumbaugh 和 Ivar Jacobson 所提出的许多很好的分析技术，而 James Rumbaugh 的 OMT-2 也采用了 Booch 所提出的很好的设计方法。但是，不同符号体系的使用仍然给软件市场带来了混乱。因为，同一个符号对于不同的人可能意义不同，而同一个事物对于不同的人也可能用不同的符号表示，因此引起了很多混乱，人们用“方法大战”形象地描述了这种混乱局面。

统一建模语言 UML 的诞生结束了符号方面的“方法大战”。UML 统一了 Booch 方法、OMT 方法、OOSE 方法的符号体系，并采纳了其他面向对象方法关于符号方面的许多好的概念。

1.1.2 UML 的发展

UML 的建立开始于 1994 年 10 月，那时 James Rumbaugh 加入了 Grady Booch 所在的 Rational 公司，他们的 UML 项目主要统一了 Booch 方法和 OMT 方法，并于 1995 年 10 月发布了 Unified

Method 0.8（这是 UML 当时的名字）。与此同时，OOSE 的创始人 Ivar Jacobson 不久也加入了 Rational 公司，他们开始在 UML 项目中加入 OOSE 方法，并于 1996 年 6 月将 UM 改名为 UML（Unified Modeling Language），并发布了 UML 0.9。1996 年 10 月，该公司又发布了 UML0.91。到 1996 年，一些软件业的相关机构将 UML 作为其商业策略已日趋明显。UML 的开发者逐渐得到了来自用户的正面回应，并倡议成立了 UML 协会，以完善、加强和促进 UML 的规范工作。在定义 UML 1.0 时，DEC、HP、I-Logix、IntelliCorp、IBM、ICON 计算（ICON Computing）、MCI Systemhouse、Microsoft、Oracle、Rational、Texas 仪器（Texas Instrumnets）、Unisys 等公司都参与了该项工作。此时，UML 在美国获得了工业界和科技界的广泛支持，已有 700 多个公司表示支持采用 UML 作为建模语言。1996 年年底，UML 已稳占面向对象技术市场 85%的份额，成为可视化建模语言事实上的工业标准。

其中，UML 1.0 是当时定义完整、富于表达、功能强大的建模语言，它于 1997 年 1 月被提交给 OMG（Object Management Group，对象管理组织），申请成为标准建模语言。

1997 年 1 月至 1997 年 7 月，Andersen 咨询、Ericsson、ObjectTime、Platinum 技术、P-Tech、Reich 技术、SoftTeam、Sterling 软件和 Taskon 均加入了 UML 组。1997 年 7 月，UML 的修正版 UML 1.1 被提交给 OMG。1997 年 11 月 17 日，OMG 采纳了 UML 1.1 作为基于面向对象技术的标准建模语言。1998 年 6 月，OMG RTF（Revision Task Force）发布了 UML 1.2。随后 OMG RTF 又相继发布了 UML1.3、UML1.4 和 UML1.5，这 3 个新的版本弥补并修改了 UML1.1 中的许多短处和缺陷。2005 年，UML 2.0 被 OMG 采纳，UML2.0 对 UML1.x 进行了很多重大修改。UML 的发展历程如图 1.1 所示。

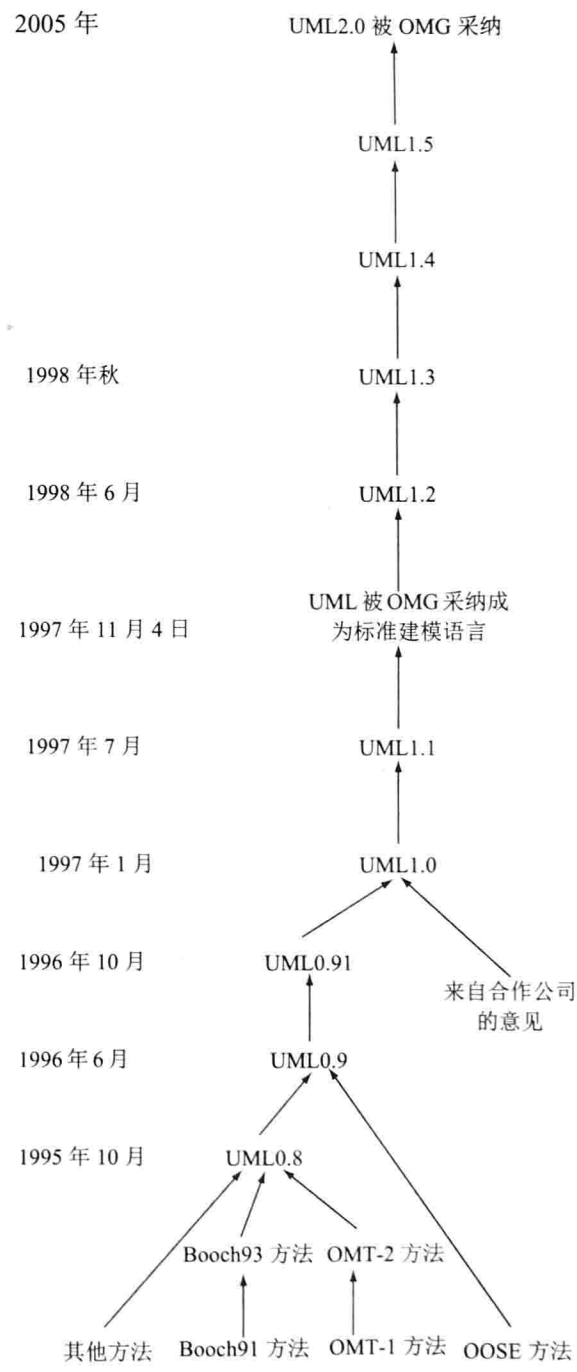

图 1.1　UML 的发展历程

统一建模语言 UML 是一种定义良好、富于表达、功能强大且普遍适用的建模语言。它融入了软件工程领域的新思想、新方法和新技术，不但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。

统一建模语言 UML 代表了面向对象软件开发技术的发展方向，具有巨大的市场前景和重大的经济价值。

1.1.3　UML 的内容

作为一种建模语言，UML 的定义包括 UML 语义和 UML 表示法两个部分。

1. UML 语义

UML 语义描述了基于 UML 的精确元模型定义。元模型为 UML 的所有元素在语法和语义上提供了简单、一致、通用的定义性说明，使开发者能在语义上取得一致，消除了因人而异的表达方法所造成的影响。此外，UML 还支持对元模型的扩展定义。

2. UML 表示法

UML 表示法定义了 UML 符号的表示方法，为开发者或开发工具使用这些图形符号和文本语法为系统建模提供了标准。这些图形符号和文字所表达的是应用级的模型，在语义上它是 UML 元模型的实例。

1.1.4　UML 的主要特点

UML 的主要特点归纳如下。

（1）UML 统一了 Booch 方法、OMT 方法、OOSE 方法和其他面向对象方法的基本概念和符号。同时，UML 还汇集了面向对象领域中很多人的思想，如图 1.2 所示。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面向对象方法和丰富的计算机科学实践经验综合提炼而成的。

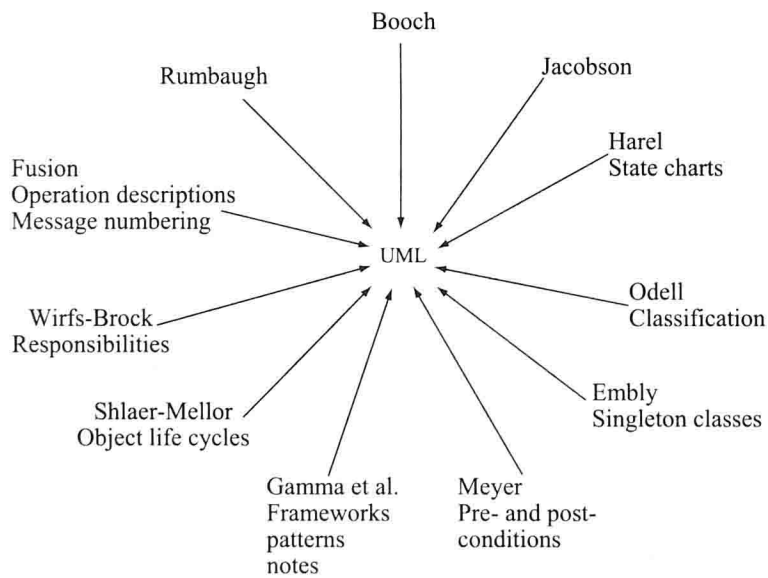

图 1.2　UML 的形成

（2）目前可以认为，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 的发展尚存一个进化过程。

（3）UML 是一种建模语言，而不是一种方法。这是因为 UML 中没有过程的概念，而过程正是方法的一个重要组成部分。UML 本身独立于过程，这意味着用户在使用 UML 进行建模时，可以选用任何适合的过程。过程的选用与软件开发过程中的若干不同因素有关，如所开发软件的种类（如实时系统、信息系统和桌面产品）、开发组织的规模（如单人开发、小组开发和团队开发）等，用户可根据不同的需要选用不同的过程。UML 只是一种语言，是独立于过程的，最好将它应

用于用例驱动的、以体系结构为中心的、迭代的、递增的过程。

1.1.5 UML 的功能

UML 是一种建模语言，该语言具有如下功能。

1. 为软件系统的产物（Artifacts）建立可视化模型

对于大多数程序员来说，在脑海中设想一个软件与用代码来实现这个软件是没有距离的，怎么想，就怎么用代码来实现它。事实上，有些东西最好直接转化为代码，因为文本是写表达式和算法的最直接的方式。过去，即使在这种情况下，程序员仍就要做一些简单的建模工作，例如在白板或纸上画一些简单的模型，但这种做法会产生下列问题。

（1）不利于交流。如果某个公司或项目组使用自己的语言来描述模型，则这个模型很难为其他公司或项目组外的人员所理解。

（2）如果不建立模型，软件系统中的有些东西很难用文本编程语言来表达清楚。

（3）如果程序员在修改代码时，没有将他脑海中的模型记录下来，这个信息就可能会永远丢失，不便于软件维护。

而统一建模语言 UML 所具有的如下优点很好地解决了这些问题。

（1）UML 符号具有定义良好的语义，不会引起歧义。UML 是一个标准的、被广泛采用的建模语言，因此，用 UML 建模有利于交流。

（2）UML 是可视化的建模语言，它为系统提供了图形化的可视模型，使系统的结构变得直观，易于理解。

（3）用 UML 为软件系统建立模型，不但有利于交流，还有利于对软件的维护。

2. 规约软件系统的产物

规约，即规范定义（Specifying），意味着建立的模型是准确的、无歧义的、完整的。UML 定义了在开发软件系统过程中所做的所有重要的分析、设计和实现决策的规格说明。

3. 构造软件系统的产物

UML 不是可视化的编程语言，但它的模型可以直接对应各种各样的编程语言，也就是说，可以从 UML 模型生成 Java、C++、Visual Basic 等语言的代码，甚至还可以生成关系数据库中的表。

从 UML 模型生成编程语言代码的过程称为前向工程（Forward Engineering），从代码实现生成 UML 模型的过程称为逆向工程（Reverse Engineering）。目前，大多数 CASE 工具，如 Rational Rose、Visual Paradigm for UML、Prosa 等，都既支持前向工程又支持逆向工程。

4. 为软件系统的产物建立文档

在软件的开发过程中为软件系统建立清晰、完整、准确的文档是非常重要的。UML 可以为系统的体系结构及其所有细节建立文档。UML 还可以为需求、测试、项目规划活动和软件发布管理活动进行建模。

软件系统的产物包括如下元素（但不限于这些元素）。

- 需求。
- 体系结构。
- 设计。
- 源代码。
- 项目计划。
- 测试。

- 原型。
- 发布。

1.1.6　UML 的组成

UML 词汇表包括 3 种构造模块，即元素、关系和图。元素是模型中重要的抽象，关系将这些元素连接起来，而图则将元素的集合进行分组。

1. 元素

UML 中的元素，可分为结构元素、行为元素、分组元素和注释元素 4 种。

- 结构元素。

结构元素是 UML 模型中的名词，也是模型中主要的静态部分，代表了模型中概念的或物理的元素。在 UML 中，有 7 种最常见的结构元素，即类（Class）、接口（Interface）、协作（Collaboration）、用例（Use Case）、活动类（Active Class）、组件（Component）和节点（Node）。

- 行为元素。

行为元素是 UML 模型中的动态部分，是模型中的动词，代表了跨越时间和空间的行为。在 UML 中，有两种主要的行为元素，即交互作用（Interaction）和状态机（State Machine）。

交互作用是一种行为，由特定上下文中为完成特定目的而在对象间交换的消息集组成。交互作用包括许多其他元素，如消息、动作序列（由消息激活的行为）、连接（对象间的连接）等。

状态机也是一种行为，这种行为规定了对象在其生命周期内为响应事件而经历的状态序列，以及对事件的响应。状态机也包括许多其他元素，如状态、跃迁、事件和活动等。

- 分组元素。

分组元素是 UML 模型中用来组织元素的元素。UML 中的主要分组元素是包（Package）。

- 注释元素。

注释元素是 UML 模型中的解释性部分，可以用于描述、例解、注解（Note）UML 模型中的任何元素。UML 中的主要注释元素是注解。

2. 关系

在 UML 模型中，主要有 4 种关系。

- 依赖关系（Dependency）。
- 关联关系（Association）。
- 类属关系（Generalization）。
- 实现关系（Realization）。

3. 图

UML1.x 定义了 9 种图，UML2.0 又补充了 4 种图，总共定义了 13 种图。这 13 种图可以分为两类，即结构建模图（Structural Modeling Diagrams）和行为建模图（Behavioral Modeling Diagrams）。结构建模图描述了系统的静态结构，它包括 6 种图：类图、对象图、组件图、组合结构图、包图和部署图。行为建模图（Behavioral Modeling Diagrams）描述了系统的行为，它包括 7 种图：用例图、活动图、状态机图、顺序图、通信图、定时图和交互概览图。其中顺序图、通信图、定时图和交互概览图又称为交互作用图，这些图侧重描述对象间的交互作用。

（1）结构建模图。

结构建模图定义了模型的静态结构，一般用来为构成模型的类、对象、接口、物理组件以及元素之间的关系进行建模。

- 类图（Class Diagram）。

类图描述系统中各个类的静态结构，它不仅定义了系统中的类，表示了类之间的联系（如关联、依赖、聚合等），还描述了类的内部结构（类的属性和操作）。类图描述的是一种静态关系，在系统的整个生命周期都是有效的。

- 对象图（Object Diagram）。

对象图是类图的实例，使用与类图类似的标识。它们的不同点在于对象图只是显示类的多个对象实例，而不是实际的类。一个对象图是类图的一个实例。对象图只在系统整个生命周期的某一时间段存在。

- 组件图（Component Diagram）。

组件图描述代码组件的物理结构及各组件之间的依赖关系。一个组件可能是源代码组件、二进制组件或可执行组件。组件图包含逻辑类或实现类的有关信息，有助于程序员分析和理解组件之间的相互影响程度。

- 组合结构图（Composite Structure Diagram）。

组合结构图描述了分类器（例如，类、组件或用例）的内部结构，它包括分类器与系统其他部分的交互作用点。

- 包图（Package Diagram）。

包图描述了包与包之间的关系。

- 部署图（Deployment Diagram）。

部署图定义系统中软硬件的物理体系结构。部署图不但描述了实际的计算机和设备（用节点表示）以及它们之间的连接关系，还描述了连接的类型以及组件之间的依赖性。在节点内部，可放置可执行组件和对象以显示节点与可执行软件单元的对应关系。

（2）行为建模图。

行为建模图描述了系统的动态结构、系统对象间的交互关系以及对象的瞬时状态。

- 用例图（Use Case Diagram）。

用例图从用户角度描述系统功能，并指出各功能的操作者。

- 活动图（Activity Diagram）。

活动图描述了为满足用例要求所要进行的各类活动以及活动间的约束关系，活动图有利于识别并行活动。

- 状态机图（State Machine Diagram）。

状态机图描述了类的对象所有可能的状态，以及事件发生时状态的跃迁条件。状态机图通常是对类图的补充。在实际应用中，开发人员并不需要为所有的类画状态机图，而只需为那些有多个状态且其行为受外界环境影响并发生改变的类画状态机图。

- 顺序图（Sequence Diagram）。

顺序图描述了对象之间的动态合作关系，它强调对象之间消息发送的时间顺序，同时显示对象之间的交互。

- 通信图（Communication Diagram）。

通信图描述了类的实例、实例间的相互关系以及实例间的消息流。通信图跟顺序图相似，描述了对象间的动态协作关系。除显示信息交换外，通信图还描述对象以及对象之间的关系。

- 定时图（Timing Diagram）。

定时图用于描述一个或多个对象在给定时间段的行为。

- 交互概览图（Interaction Overview Diagram）。

交互概览图是活动图和顺序图的混合，它概括描述了系统或商业过程中的控制流。

1.2　RUP

一个项目的成功需要 3 个方面的支持，即符号（Notation）、过程（Process）和工具（Tool）。前面，已经介绍了符号——统一建模语言 UML，现在介绍过程——Rational 统一过程（Rational Unified Process，RUP）。

1.2.1　RUP 的发展

RUP 经过了多年发展才逐渐变得成熟，它结合了很多个公司和人的集体经验。

图 1.3 描述了 RUP 的发展历史。从时间上回顾，RUP 是 Rational Objectory 过程的直接继承者。RUP 吸收了数据工程、商业建模、项目管理和配置管理等许多领域的工程实践经验。其中，RUP 2000 还吸收了由 ObjectTime 公司的创立者所开发的实时面向对象方法中的元素。RUP 与 Rational 软件工具包是紧密集成在一起的。

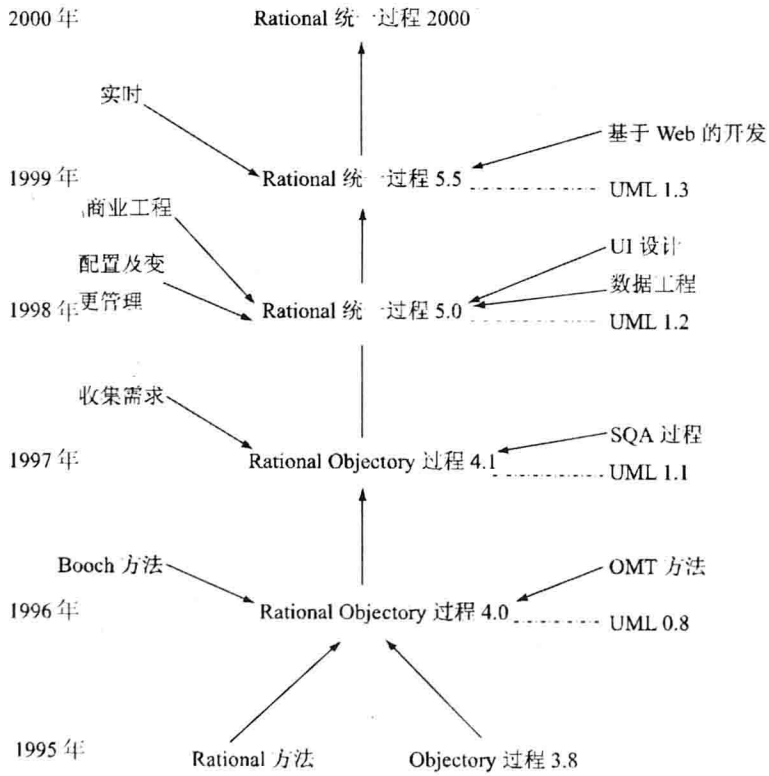

图 1.3　RUP 的发展历史

Objectory 过程是 Ivar Jacobson 于 1987 年根据他在 Ericsson 公司的经验在瑞典创建的，这个过程随后成为 Ivar Jacobson 的公司——Objectory AB 的产品。该过程以用例概念和面向对象的方法为中心，很快得到了软件工业的认可并被世界各地的许多公司采用。1992 年，Objectory 过程的简化版本作为课本被出版。1995 年，Rational 软件公司与 Objectory AB 公司合并，之后，Rational

方法与 Objectory 过程（版本 3.8）相集成，产生了 Rational Objectory 过程（版本 4.0）。

Rational Objectory 过程从 Objectory 过程中继承了过程结构和用例的中心概念，从 Rational 方法得到了迭代开发和体系结构的模式。后来，该版本还吸收了来自 Requisite 公司的需求管理部分和来自 SQA 公司的详细的测试过程。并且，该过程第一个使用了统一建模语言——UML 0.8。

Rational 软件公司被 IBM 收购后，RUP 被纳入 IBM 的 RMC（Rational Method Composer）产品中。

1.2.2 什么是 RUP

RUP 是一个软件工程化过程。它提供了在开发过程中分派任务和责任的方法，它的目标是在可预见的日程和预算前提下，确保满足最终用户需求的高质量软件的产生。

RUP 是由 Rational 软件公司开发和维护的过程产品，它的开发队伍同其顾客、合伙人、Rational 公司的其他产品组及顾问组织紧密合作，以确保开发过程中不断地对产品进行更新、改善与发展，总结新的经验，使产品融合最好的工程实践。RUP 的特点如下所述。

- RUP 提高了团队生产力。使用 RUP，无论项目组成员是进行需求分析、设计、测试、项目管理，还是配置管理，所有的成员都可享有共同的语言、过程和开发软件的视图。
- RUP 可进行活动创建并维护模型。RUP 强调开发和维护模型（模型为开发过程中的软件系统提供了语义丰富的描述），而不是侧重于产生大量的书面文档。
- RUP 为如何有效地使用统一建模语言 UML 提供了指导。
- RUP 是一个可裁剪定制的过程，而没有一个固定的开发过程能适合所有的软件开发。使用 RUP，开发者可以根据不同的情况来定制这个过程，RUP 为裁减定制适合特定需要的开发过程提供了支持。因此，RUP 不但适用于小的开发团队，也适用于大的开发机构。
- RUP 吸收了许多已经在商业上得到证明的软件开发的最佳实践经验，使其能够适用于范围广泛的项目和组织。RUP 为每个项目组成员提供了必要的准则、模板和指导工具，使整个开发团队可以充分利用这些最佳工程实践经验，这为开发队伍提供了很多优势。

RUP 所吸收的最佳工程实践经验如下。

1. 迭代式软件开发

目前，由于软件系统已经变得越来越复杂，使用瀑布模型式的软件开发过程（即先定义整个问题，再设计整个解决方案，然后编制软件，最后进行软件测试）已经变得不可能。现在需要一种迭代式的软件开发过程，使开发者在迭代的过程中通过不断地细化来逐渐加深对问题的理解，不断完善解决方案，从而最终形成有效的解决方案。RUP 支持迭代的过程，它通过将风险分散于每一次迭代，大大降低了项目的风险。并且，由于每次迭代都产生一个可执行的版本，频繁的状态检查也可以确保项目能按时进行。

2. 需求管理

用例（Case）和脚本（Scenario）已经被证明是捕获功能需求的有效方法，它们的使用为由其所驱动的软件设计、实现和测试提供了保证，所产生的最终系统更能满足最终用户的需要。

3. 使用基于组件的体系结构

组件是实现明确功能的模块或子系统，是促进更有效地软件重用的弹性结构。它设计灵活、可修改、直观且便于理解。RUP 支持基于组件的软件开发。

4. 可视化的软件建模

可视化的软件建模过程说明了如何对软件进行可视化建模以捕获系统的体系结构、组件的结

构和行为。这样的建模过程可以隐藏细节，并能使用图形化的构建模块来书写代码。可视化的建模可以帮助开发人员理解软件的不同方面，了解系统的各部分是如何协作的，确保模型与代码一致，设计与实现一致，并促进沟通。工业化的 UML 是成功地进行可视化建模的基础。

5. 验证软件质量

系统性能和可靠性通常是软件能否被接受的重要因素。开发人员可根据系统的可靠性需求、功能需求和性能需求来检查系统的质量，RUP 可以帮助开发人员计划、设计、实现、执行和评估这些测试类型。质量评估应该是内建于过程，包含在所有的活动中，并由所有有关人员参与，使用客观的标准的活动，而不是由单独的小组进行的单独活动。

6. 控制软件的变化

在软件开发过程中，需求的变化通常是不可避免的。RUP 描述了如何控制、跟踪和监控需求变更，以确保软件能够成功地迭代开发。RUP 还指导开发人员通过隔离来自其他工作空间的修改，以及控制整个软件产物（如模型、代码、文档等）的修改，来为每个开发者建立安全的工作区。另外，通过进行自动化集成和建立管理，RUP 使团队成员能够协调一致地工作。

1.2.3　过程概览

RUP 可以用二维结构（或两个轴）来描述，如图 1.4 所示。

- 时间。软件开发的生命周期被划分为阶段和迭代。
- 过程组件。

为了软件项目的开发成功，两个轴都要被考虑。

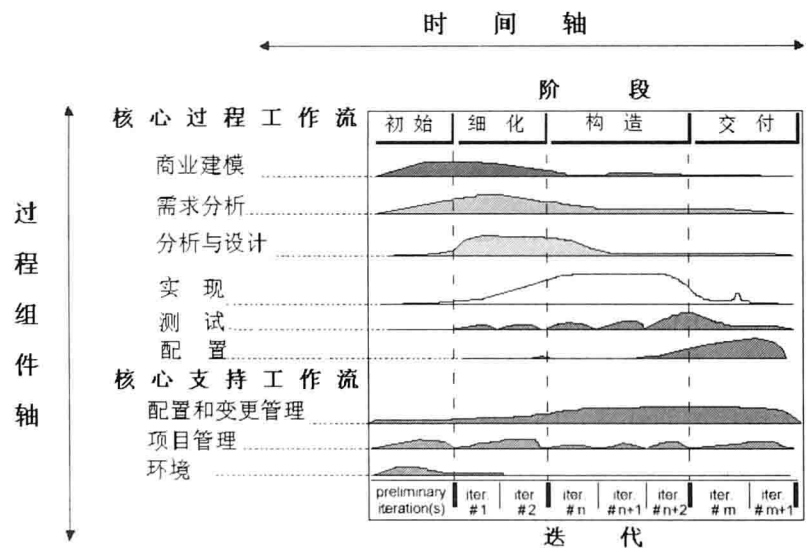

图 1.4　RUP

1.2.4　时间轴

软件生命周期可以被分解为许多阶段，每个阶段都致力于产生软件产品的新版本。RUP 将软件生命周期划分为 4 个连续的阶段：初始阶段（Inception）、细化阶段（Elaboration）、构造阶段（Construction）、交付阶段（Transition）。

每个阶段都有特定的目标，下面对这 4 个阶段进行具体地描述。