UML基础与Rose建模案例

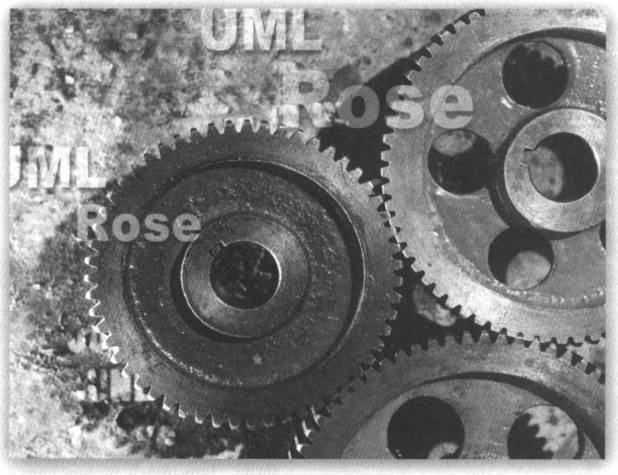
（第2版）

吴建 郑潮 汪杰 编著

人民邮电出版社
POSTS & TELECOM PRESS

UML基础与Rose建模案例

（第2版）



吴建 郑潮 汪杰 编著

人 民 邮 电 出 版 社
北 京

图书在版编目（CIP）数据

UML 基础与 Rose 建模案例 / 吴建，郑潮，汪杰编著．—2 版．
—北京：人民邮电出版社，2007.4

ISBN 978-7-115-15891-8

Ⅰ．U…　Ⅱ．①吴…②郑…③汪…　Ⅲ．面向对象语言，UML—程序设计　Ⅳ．TP312

中国版本图书馆 CIP 数据核字（2007）第 025031 号

内 容 提 要

本书介绍了使用 UML（统一建模语言）进行软件建模的基础知识以及 Rational Rose 工具的使用方法。

本书在第 1 版的基础上，充分吸取了读者宝贵的反馈意见和建议，更新了大部分案例。书中前 9 章是基础部分，对软件工程思想、UML 的相关概念、Rational Rose 工具、RUP 软件过程，以及 UML 的双向工程等进行了详细的介绍；后 3 章是案例部分，通过档案管理系统、BBS 论坛系统和新闻中心管理系统 3 个综合实例，对 UML 建模（以 Rose 为实现工具）的全过程进行了剖析；最后的附录中给出了 UML 中常用的术语、标准元素和元模型，便于读者查询。

本书是一本基础与实例紧密结合的 UML 书籍，可以作为从事面向对象软件开发人员的学习指导用书，也可以作为高等院校计算机或软件工程相关专业的教材。

UML 基础与 Rose 建模案例（第 2 版）

◆ 编　著　吴　建　郑　潮　汪　杰
责任编辑　汤　倩

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：17.25
字数：417 千字　2007 年 4 月第 2 版
印数：1－6 000 册　2007 年 4 月北京第 1 次印刷

ISBN 978-7-115-15891-8/TP

定价：28.00 元

读者服务热线：(010)67132692　印装质量热线：(010)67129223

前　言

面向对象的建模语言出现在20世纪70年代，随着编程语言的多样化以及软件产品在更多领域的应用，当时的软件工程学者开始分析与设计新的软件方法论。在这期间出现了超过50种的面向对象方法，对于这些不同符号体系的开发方法，软件设计人员和程序员往往很难找到完全适合他们的建模语言，而且这也妨碍了不同公司，甚至是不同项目开发组间的交流与经验共享。因此，有必要确立一款标准统一的，能被绝大部分软件开发和设计人员认可的建模语言，UML应运而生。1997年11月17日，UML1.1被OMG（对象管理组织）采纳，正式成为一款定义明确、功能强大、受到软件行业普遍认可的、可适用于广泛领域的建模语言。

如今，UML已经成为面向对象软件系统分析设计的必备工具，也是广大软件系统设计人员、开发人员、项目管理员、系统工程师和分析员必须掌握的基础知识。

本书是《UML基础与Rose建模案例》的第2版。在第1版面市后的两年多时间里，本书受到了广大读者的欢迎。许多热心读者向我们提出了宝贵的意见和建议；很多高校将第1版选作计算机或软件工程相关专业的教材，并在实践中总结出Rose建模方面的教学经验，反馈给我们，在此向他们表示衷心的感谢。

本书在第1版的基础上做了较大改动，增加了许多新的内容，主要的特点如下。

- 在讲解基础知识的过程中，结合了大量的UML图形和实例，不同于众多纯理论的UML教材的风格，更利于读者的理解和接受。
- 以一个完整的“图书馆管理系统”的建模案例贯穿于本书的理论分析之中，并对使用Rose工具进行建模的方法进行了详细的讲解。
- 书中的案例部分给出了3个综合性案例，分别为档案管理系统、BBS论坛系统和新闻中心管理系统，体现了本书较强的实践性，便于读者举一反三，利用Rose工具创建自己的UML模型。
- 附录中给出了UML中常用的术语、标准元素和元模型，便于读者查询。

为了方便教学使用，本书配备了教学大纲和课件，如果需要，可以访问人民邮电出版社的网站www.ptpress.com.cn获取，或者发E-mail至tangqian@ptpress.cn索取。

本书主要由吴建、郑潮和汪杰编写，为本书提供资料的还有付冰、何贤辉、胡标、姜琴英、厉蒋和李功等。在编写过程中，我们力求精益求精，但难免存在一些不足之处，如果读者使用本书时遇到问题，可以发E-mail到tangqian@ptpress.cn与我们联系。

编　者

目　录

第 1 章　软件工程与 UML 概述

本章对软件工程和 UML 进行简要的介绍，共分两节，每节介绍一个主题：软件工程概述、UML 概述。通过对本章的阅读，读者可以对软件和 UML 有一个总体的认识。

1.1　软件工程概述

1.1.1　软件工程的发展历史

从 20 世纪 60 年代中期到 70 年代中期，软件业进入了一个大发展时期。这一时期软件作为一种产品开始被广泛使用，同时出现了所谓的软件公司。这一时期的软件开发方法仍然沿用早期的自由软件开发方式。但是随着软件规模的急剧膨胀，软件的需求日趋复杂，软件的性能要求相对变高，随之而来的软件维护难度也越来越大，开发的成本相应增加，导致失败的软件项目比比皆是，这样的一系列问题导致了“软件危机”。

1968 年，前北大西洋公约组织的科技委员会召集了一批一流的程序员、计算机科学家以及工业界人士共商对策。通过借鉴传统工业的成功作法，他们主张通过工程化的方法开发软件来解决软件危机，并冠以“软件工程（Software Engineering）”这一术语。30 余年来，尽管软件的一些毛病仍然无法根治，但软件的发展速度却超过了任何传统工业，并未出现真正的软件危机。如今软件工程成了一门学科。

软件工程是一门建立在系统化、规范化、数量化等工程原则和方法上的，关于软件开发各个阶段的定义、任务和作用的工程学科。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、配置管理和项目计划等。

1.1.2　软件工程的生命周期

软件开发是一套关于软件开发各阶段的定义、任务和作用的，建立在理论上的一门工程学科。它对解决软件危机，指导人们利用科学和有效的方法来开发软件，提高及保证软件开发的效率和质量起到了一定的作用。

经典的软件工程思想将软件开发分成以下 5 个阶段：需求捕获（Requirement Capture）阶段、系统分析与设计（System Analysis and Design）阶段、系统实现（System Implementation）阶段、测试（Testing）阶段和维护（Maintenance）阶段。

（1）需求捕获（Requirement Capture）阶段

需求捕获阶段是通常所说的开始阶段，但实际上，真正意义上的开始阶段要做的是选择合适的项目——立项阶段。其实，软件工程中的许多关于思想的描述都是通俗易懂的。立项阶段，顾名思义，就是从若干个可以选择的项目中选择一个最适合自己的项目的阶段。这个选择的过程是至关重要的，因为它将直接决定整个软件开发过程的成败。通常情况下，要考虑几个主要的因素：经济因素（经济成本、受益等）、技术因素（可行性、技术成本等）和管理因素（人员管理、资金运作等）。

在立项之后，真正进入了软件开发阶段（当然，这里所说的是广义的软件开发，狭义的软件开发通常指的是编码）。需求捕获是整个开发过程的基础，也直接影响着后面的几个阶段的进展。纵观软件开发从早期纯粹的程序设计到软件工程思想的萌发产生和发展的全过程，不难发现，需求捕获的工作量在不断增加，其地位也随之不断提升。这一点可以从需求捕获在整个开发过程中所占的比例（无论是时间、人力，还是资金方面）不断地提高上可以看出。

（2）系统分析与设计（System Analysis and Design）阶段

系统分析与设计包括分析和设计两个阶段，而这两个阶段是相辅相成、不可分割的。通常情况下，这一阶段是在系统分析员的领导下完成的，系统分析员不仅要有深厚的计算机硬件与软件的专业知识，还要对相关业务有一定的了解。系统分析通常是与需求捕获同时进行，而系统设计一般是在系统分析之后进行的。

（3）系统实现（System Implementation）阶段

系统实现阶段也就是通常所说的编码（Coding）阶段，在软件工程思想出现之前，这基本上就是软件开发的全部内容。而在现代的软件工程中，编码阶段所占的比重正在逐渐缩小。

（4）测试（Testing）阶段

测试阶段的主要任务是通过各种测试思想、方法和工具，使软件的 Bug 降到最低。微软（Microsoft）宣称他们采用零 Bug 发布的思想确保软件的质量，也就是说只有当测试阶段达到没有 Bug 时他们才将产品发布。测试是一项很复杂的工程。

（5）维护（Maintenance）阶段

在软件工程思想出现之前，这一阶段是令所有与之相关的角色（包括客户和发方）头疼的。可以说，软件工程思想很大程度上是为了解决软件维护的问题而提出的。因为，软件工程的 3 大目的——软件的可维护性、软件的可复用性和软件开发的自动化，可维护性就是其中之一，而且软件的可维护性是复用性和开发自动化的基础。在软件工程思想得到迅速发展的今天，虽然软件的可维护性有了很大的提高，但目前软件开发中所面临的最大的问题仍是维护问题。每年都有许多软件公司因为无法承担对其产品的高昂的维护成本而宣布破产。

值得注意的是，软件工程主要讲述软件开发的道理，基本上是软件实践者的成功经验和失败教训的总结。软件工程的观念、方法、策略和规范都是朴实无华的，一般人都能领会，关键在于运用。不可以把软件工程方法看成是诸葛亮的锦囊妙计——在出了问题后才打开看看，而应该事先掌握，预料将要出现的问题，控制每个实践环节，防患于未然。

1.2 UML 概述

1.2.1 UML 的历史

面向对象的分析与设计（OOA&OOD）方法的发展在 20 世纪 80 年代末至 90 年代中出现了一个高潮，UML 是这个高潮的产物。它不仅统一了 Booch、Rumbaugh 和 Jacobson 的表示方法，而且对其做了进一步的发展，并最终统一为大众所接受的标准建模语言。

公认的面向对象建模语言出现于 20 世纪 70 年代中期。从 1989～1994 年，其数量从不到 10 种增加到了 50 多种。在众多的建模语言中，语言的创造者努力宣传自己的产品，并在实践中不断完善。但是，使用面向对象方法的用户并不了解不同建模语言的优缺点及相互之间的差异，因而很难根据应用特点选择合适的建模语言，于是爆发了一场“方法大战”。20 世纪 90 年代中期，一批新方法出现了，其中最引人注目的是 Booch 1993、OOSE 和 OMT-2 等。

Booch 是面向对象方法最早的倡导者之一，他提出了面向对象软件工程的概念。1991 年，他将以前面向 Ada 的工作扩展到整个面向对象设计领域。Booch 1993 比较适合于系统的设计和构造。Rumbaugh 等人提出了面向对象的建模技术（OMT）方法，采用了面向对象的概念，并引入各种独立于语言的表示符。这种方法用对象模型、动态模型、功能模型和用例模型，共同完成对整个系统的建模，所定义的概念和符号可用于软件开发的分析、设计和实现的全过程，软件开发人员不必在开发过程的不同阶段进行概念和符号的转换。OMT-2 特别适用于分析和描述以数据为中心的信息系统。Jacobson 于 1994 年提出了 OOSE 方法，其最大特点是面向用例（Use Case），并在用例的描述中引入了外部角色的概念。用例的概念是精确描述需求的重要武器，但用例贯穿于整个开发过程，包括对系统的测试和验证。OOSE 比较适合支持商业工程和需求分析。此外，还有 Coad/Yourdon 方法，即著名的 OOA/OOD，它是最早的面向对象的分析和设计方法之一，该方法简单、易学，适合于面向对象技术的初学者使用，但由于该方法在处理能力方面的局限，目前已很少使用。

概括起来，首先，面对众多的建模语言，用户由于没有能力区别不同语言之间的差别，因此很难找到一种比较适合其应用特点的语言；其次，众多的建模语言实际上各有千秋；最后，虽然不同的建模语言大多雷同，但仍存在某些细微的差别，极大地妨碍了用户之间的交流。因此在客观上，有必要在精心比较不同的建模语言优缺点及总结面向对象技术应用实践的基础上，组织联合设计小组，根据应用需求，取其精华，去其糟粕，求同存异，统一建模语言。

1994 年 10 月，Grady Booch 和 Jim Rumbaugh 首先将 Booch 93 和 OMT-2 统一起来，并于 1995 年 10 月发布了第一个公开版本，称之为统一方法 UM 0.8（Unitied Method）。1995 年秋，OOSE 的创始人 Jacobson 加盟到这一工作中。经过 Booch、Rumbaugh 和 Jacobson 3 人的共同努力，于 1996 年 6 月和 10 月分别发布了两个新的版本，即 UML 0.9 和 UML 0.91，并将 UM 重新命名为 UML（Unified Modeling Language）。UML 的开发者倡议并成立了 UML 成员协会，以完善、加强和促进 UML 的定义工作。当时的成员有 DEC、HP、I-Logix、Itellicorp、IBM、ICON Computing、MCI Systemhouse、Microsoft、Oracle、Rational Software、TI 以及 Unisys。

UML 成员协会对 UML 1.0 及 UML 1.1 的定义和发布起了重要的促进作用。

1.2.2 UML 包含的内容

首先，UML 融合了 Booch、OMT 和 OOSE 方法中的基本概念，而且这些基本概念与其他面向对象技术中的基本概念大多相同，因而，UML 必然成为这些方法以及其他方法的使用者乐于采用的一种简单一致的建模语言；其次，UML 不是上述方法的简单汇合，而是在这些方法的基础上广泛征求意见，集众家之长，几经修改而完成的，UML 扩展了现有方法的应用范围；最后，UML 是标准的建模语言，而不是标准的开发过程。

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

描述基于 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、一致和通用的定义性说明，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

定义 UML 符号的表示法，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。这些图形符号和文本所表达的是应用级的模型，在语义上它是 UML 元模型的实例。

UML 的重要内容可以由下列 5 类图来定义。

第 1 类是用例图（Use Case Diagram），从用户角度描述系统功能，并指出各功能的操作者。

第 2 类是静态图（Static Diagram），包括类图、对象图和包图。其中类图描述系统中类的静态结构。不仅定义系统中的类，表示类之间的联系（如关联、依赖和聚合等），也包括类的内部结构（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中都是有效的。对象图是类图的实例，使用与类图几乎完全相同的标识。它们的不同点在于对象图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。由于对象存在生命周期，因此对象图只能在系统某一时间段内存在。包由包或类组成，表示包与包之间的关系。包图用于描述系统的分层结构。

第 3 类是行为图（Behavior Diagram），描述系统的动态模型和组成对象间的交互关系，包括状态图和活动图。其中状态图描述类的对象所有可能的状态以及事件发生时状态的转移条件。通常，状态图是对类图的补充。在实际上并不需要为所有的类画状态图，只需为那些有多个状态且其行为受外界环境的影响并且发生改变的类画状态图。而活动图描述满足用例要求所要进行的活动以及活动间的约束关系，有利于识别并行活动。

第 4 类是交互图（Interactive Diagram），描述对象间的交互关系，包括时序图和协作图。其中，时序图显示对象之间的动态合作关系，它强调对象之间消息发送的顺序，同时显示对象之间的交互；协作图描述对象间的协作关系，协作图跟时序图相似，显示对象间的动态合作关系。除显示信息交换外，协作图还显示对象以及它们之间的关系。如果强调时间和顺序，则使用时序图；如果强调上下级关系，则选择协作图。这两种图合称为交互图。

第 5 类是实现图（Implementation Diagram），包括组件图和配置图。其中组件图描述代码部件的物理结构及各组件之间的依赖关系。一个组件可能是一个资源代码组件、一个二进制组

件或一个可执行组件。它包含逻辑类或实现类的有关信息。组件图有助于分析和理解部件之间的相互影响程度。配置图定义系统中软硬件的物理体系结构。它可以显示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，也可显示连接的类型及部件之间的依赖性。在节点内部，放置可执行部件和对象，以显示节点与可执行软件单元的对应关系。

从应用的角度看，当采用面向对象技术设计系统时，第 1 步是描述需求；第 2 步根据需求建立系统的静态模型，以构造系统的结构；第 3 步是描述系统的行为。其中在第 1 步与第 2 步中所建立的模型都是静态的，包括用例图、类图（包含包）、对象图、组件图和配置图等 5 个图形，是 UML 的静态建模机制。第 3 步中所建立的模型或者可以执行，或者表示执行时的时序状态或交互关系。它包括状态图、活动图、时序图和协作图等 4 个图形，是 UML 的动态建模机制。因此， UML 的主要内容也可以归纳为静态建模机制和动态建模机制两大类。

1.2.3 UML 的定义

UML（Unified Modeling Language，统一建模语言），是一种面向对象的建模语言。它的主要作用是帮助用户对软件系统进行面向对象的描述和建模(建模是通过将用户的业务需求映射为代码，保证代码满足这些需求，并能方便地回溯需求的过程)，它可以描述这个软件开发过程从需求分析直到实现和测试的全过程。UML 通过建立各种联系，如类与类之间的关系、类/对象怎样相互配合实现系统的行为状态等（这些都称为模型元素），来组建整个结构模型。UML 提供了各种图形，比如用例图、类图、时序图、协作图和状态图等，来把这些模型元素及其关系可视化，让人们可以清楚容易地理解模型。可以从多个视角来考察模型，从而更加全面地了解模型，这样同一个模型元素可能会出现在多个 UML 图中，不过都保持相同的意义和符号。

1. UML 的组成

UML 由视图（View）、图（Diagram）、模型元素（Model Element）和通用机制（General Mechanism）等几个部分组成。

视图（View）是表达系统的某一方面特征的 UML 建模元素的子集，视图并不是图，它是由一个或多个图组成的对系统某个角度的抽象。在建立一个系统模型时，通过定义多个反映系统不同方面的视图，才能对系统做出完整、精确的描述。

图（Diagram）是模型元素集的图形表示，通常是由弧（关系）和顶点（其他模型元素）相互连接构成的。UML 通常提供 9 种基本的图，把这几种基本图结合起来就可以描述系统的所有视图。

模型元素（Model Element）代表面向对象中的类、对象、接口、消息和关系等概念。UML 中的模型元素包括事物和事物之间的联系，事物之间的关系能够把事物联系在一起，组成有意义的结构模型。常见的联系包括关联关系、依赖关系、泛化关系、实现关系和聚合关系。同一个模型元素可以在几个不同的 UML 图中使用，不过同一个模型元素在任何图中都保持相同的意义和符号。

通用机制（General Mechanism）用于表示其他信息，比如注释、模型元素的语义等。另外，UML 还提供扩展机制（Extension Mechanism），使 UML 能够适应一个特殊的方法/过程、

组织或用户。

UML 是用来描述模型的，用模型来描述系统的结构或静态特征，以及行为或动态特征。

为方便起见，用视图来划分系统各个方面，每一个视图描述系统某一方面的特征。这样一个完整的系统模型就由许多视图来共同描述。

UML 中的视图大致可以分为如下 5 种。

（1）用例视图（Use Case View），强调从用户的角度看到的或需要的系统功能，是被称为参与者的外部用户所能观察到的系统功能的模型图。

（2）逻辑视图（Logical View），展现系统的静态或结构组成及特征，也称为结构模型视图（Structural Model View）或静态视图（Static View）。

（3）并发视图（Concurrent View），体现了系统的动态或行为特征，也称为行为模型视图（Behavioral Model View）或动态视图（Dynamic View）。

（4）组件视图（Component View），体现了系统实现的结构和行为特征，也称为实现模型视图（Implementation Model View）。

（5）配置视图（Deployment View），体现了系统实现环境的结构和行为特征，也称为环境模型视图（Environment Model View）或物理视图（Physical View）。

视图是由图组成的，UML 提供 9 种不同的图。

（1）用例图（Use Case Diagram），描述系统功能。

（2）类图（Class Diagram），描述系统的静态结构。

（3）对象图（Object Diagram），描述系统在某个时刻的静态结构。

（4）时序图（Sequence Diagram），按时间顺序描述系统元素间的交互。

（5）协作图（Collaboration Diagram），按照时间和空间顺序描述系统元素间的交互和它们之间的关系。

（6）状态图（State Diagram），描述了系统元素的状态条件和响应。

（7）活动图（Activity Diagram），描述了系统元素的活动。

（8）组件图（Component Diagram），描述了实现系统的元素的组织。

（9）配置图（Deployment Diagram），描述了环境元素的配置，并把实现系统的元素映射到配置上。

根据它们在不同架构视图的应用，可以把 9 种图分成以下几类。

（1）用户模型视图：用例图。

（2）结构模型视图：类图和对象图。

（3）行为模型视图：时序图、协作图、状态图和活动图。

（4）实现模型视图：组件图。

（5）环境模型视图：配置图。

2．UML 的建模机制

UML 有两套建模机制：静态建模机制和动态建模机制。静态建模机制包括用例图、类图、对象图、包、组件图和配置图。动态建模机制包括消息、状态图、时序图、协作图、活动图。

对于本节中的诸多概念，读者暂时只需了解即可，在后面的章节中将会结合实例进行详细的介绍。

1.2.4 UML 的应用领域

UML 的目标是以面向对象图的方式来描述任何类型的系统。其中最常用的是建立软件系统的模型，但它同样可以用于描述非软件领域的系统，如机械系统、企业机构或业务过程，以及处理复杂数据的信息系统、具有实时要求的工业系统或工业过程等。

总之，UML 是一个通用的标准建模语言，可以对任何具有静态结构和动态行为的系统进行建模。此外，UML 适用于系统开发过程中从需求规格描述到系统完成后测试的不同阶段。在需求分析阶段，可以用用例来捕获用户需求。通过用例建模，描述对系统感兴趣的外部角色及其对系统（用例）的功能要求。分析阶段主要关心问题域中的主要概念（如抽象、类和对象等）和机制，需要识别这些类以及它们相互间的关系，并用 UML 类图来描述。为实现用例，类之间需要协作，这可以用 UML 动态模型来描述。在分析阶段，只对问题域的对象（现实世界的概念）建模，而不考虑定义软件系统中技术细节的类（如处理用户接口、数据库、通信和并行性等问题的类）。这些技术细节将在设计阶段引入，因此设计阶段为构造阶段提供更详细的规格说明。

编程（构造）是一个独立的阶段，其任务是用面向对象编程语言将来自设计阶段的类转换成实际的代码。在用 UML 建立分析和设计模型时，应尽量避免考虑把模型转换成某种特定的编程语言。因为在早期阶段，模型仅仅是理解和分析系统结构的工具，过早考虑编码问题十分不利于建立简单、正确的模型。

UML 模型还可作为测试阶段的依据。系统通常需要经过单元测试、集成测试、系统测试和验收测试。不同的测试小组使用不同的 UML 图作为测试依据：单元测试使用类图和类规格说明；集成测试使用部件图和协作图；系统测试使用用例图来验证系统的行为；验收测试由用户进行，以验证系统测试的结果是否满足在需求捕获阶段确定的需求。