田林琳 李鹤 / 主编

UML软件建模

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

普通高等院校电气信息类专业项目驱动系列规划教材

UML 软件建模（项目教学版）

主　编　田林琳　李　鹤
副主编　李　莹　杨　玥　梁　爽　吴晓艳

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

内 容 简 介

统一建模语言（Unified Modeling Language，UML）是面向对象软件的标准化建模语言。UML是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。本书通过案例——图书借阅系统贯穿始终，让学生掌握面向对象程序分析设计的过程，并学会用Rational Rose建模工具创建UML图和图中模型元素。最后通过一个集课程资源共享、机考平台、论坛等实用功能于一体的综合实验——网络学习平台，帮助学生固化理论知识，提升实践动手能力。

本书适用于UML和Rational Rose的初、中级用户，可以作为本科院校计算机和软件相关专业的教学用书或参考书，也适合作为各类软件开发人员的学习和参考用书。

图书在版编目（CIP）数据

UML软件建模/田林琳，李鹤主编. —北京：北京理工大学出版社，2018.6（2018.7重印）

ISBN 978-7-5682-5811-1

Ⅰ.①U…　Ⅱ.①田…②李…　Ⅲ.①面向对象语言-程序设计　Ⅳ.①TP312.8

中国版本图书馆CIP数据核字（2018）第145233号

出版发行／北京理工大学出版社有限责任公司
社　　址／北京市海淀区中关村南大街5号
邮　　编／100081
电　　话／（010）68914775（总编室）
　　（010）82562903（教材售后服务热线）
　　（010）68948351（其他图书服务热线）
网　　址／http：//www.bitpress.com.cn
经　　销／全国各地新华书店
印　　刷／
开　　本／787毫米×1092毫米　1/16
印　　张／12.5　　责任编辑／王玲玲
字　　数／308千字　　文案编辑／王玲玲
版　　次／2018年6月第1版　2018年7月第2次印刷　　责任校对／周瑞红
定　　价／33.80元　　责任印制／李志强

理 论 篇

前　言

UML是当前比较流行的一种建模语言，该语言以图形化的方式对软件系统进行分析和设计，是一款定义明确、功能强大、受到软件行业普遍认可、可适用于广泛领域的建模语言。

Rational Rose是目前最受业界瞩目的可视化软件开发工具，通过Rational Rose能用一种统一的方式设计各种项目的UML图。

本书包含了UML的基础知识、UML的基本元素及UML的使用方法，在讲述UML的使用过程时，是结合学生熟悉的项目图书借阅系统进行讲解的。学生在项目案例教学的过程中，体会如何进行面向对象的分析设计，如何将分析和设计的结果转化为UML模型。书中详细讲解了Rational Rose的使用方法，学生能从中感受到Rational Rose开发UML的便捷性和高效性。此外，在每个章节后面还提供了一定数量的习题，帮助学生固化理论知识。最后通过一个集课程资源共享、机考平台、论坛等实用功能于一体的综合实验——网络学习平台，帮助学生提升实践动手能力。

本书特点：

（1）本书通过先进的建模工具、完整的软件模型、系统的UML知识，由浅入深、循序渐进地让读者学会应用UML知识，构思软件模型，绘制UML图。本书选用Rational Rose 2007进行软件建模。第1章和第2章介绍了软件工程和面向对象基础，对UML和Rational Rose进行了概述。第3～10章详细介绍了UML用例图、类图、对象图、序列图、协作图、活动图、状态图、包图、构件图、配置图的作用、模型元素和绘制方法。最后是综合实验网络学习平台的UML建模。

（2）项目教学，全过程设置了6个合理的教学环节：项目背景、项目任务、预备知识、项目实施、同步训练、单元习题。通过图书借阅项目贯穿始终，能让学生学会在现实项目中如何应用UML建模。最后通过一个完整的案例网络学习平台，让学生亲身体验系统分析设计的过程，完成UML建模，更扎实地构建自己的UML知识体系。

本书由沈阳工学院教师田林琳、李鹤主编，李莹、杨玥、梁爽、吴晓艳（沈阳理工大学）参与部分章节的编写。此外，参与本书编写工作的还有郝雪燕、王羚伊、高晶、袁田、孙瑜等。

在本书的编写过程中，借鉴了许多现行教材的宝贵经验，在此，谨向这些教材的作者表示诚挚的感谢。由于时间仓促，加之编者水平有限，书中难免有不足之处，敬请广大读者批评指正。

编　者

目　　录

理 论 篇

实 践 篇

理 论 篇

第 1 章

概　　述

1.1　软件工程概述

从 20 世纪 60 年代中期到 70 年代中期，软件行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”。

1965 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、计算机科学家及工业界人士共商对策，他们主张通过借鉴传统工业的成功做法，通过工程化的方法开发软件来解决软件危机，这一主张被冠以“软件工程”的名称。50 余年来，尽管软件行业的一些毛病仍然无法根治，但软件行业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。如今软件工程已成了一门学科。

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。

经典的软件工程思想将软件开发分成需求捕获、系统分析与设计、系统实现、测试和维护五个阶段。

1. 需求捕获（Requirement Capture）阶段

需求捕获阶段就是通常所说的开始阶段。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。涉及项目的选取、可行性分析、需求定义等过程。需求捕获的过程非常重要，是整个开发过程的基础，直接影响着后面各阶段的发展。

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，这两个阶段是相辅相成、不可分割的。通常情况下，这个阶段是在系统分析员的领导下完成的，系统分析员不仅要具备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的，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 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，软件工程有三大目的——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，并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，不可以在出了问题后才想到查阅软件工程的知识，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 面向对象设计概述

面向对象技术现在已经逐渐取代了传统的技术，成为当今计算机软件工程学中的主要开发技术，随着面向对象技术的不断发展，越来越多的软件开发人员加入了它的阵营之中。面向对象技术之所以会被广大的软件开发人员所青睐，是由于它作为一种先进的设计和构造软件的技术，使计算机以更符合人的思维方式去解决一系列的编程问题。使用面向对象技术编写的程序极大地提高了代码复用程度和可扩展性，使编程效率也得到了极大的提高，同时减少了软件维护的代价。

1.2.1 面向对象基本概念

面向对象技术是一种以对象为基础，以事件或消息来驱动对象执行处理的程序设计技术。从程序设计方法上来讲，它是一种自下而上的程序设计方法，它不像面向过程程序设计那样一开始就需要使用一个主函数来概括出整个程序，面向对象程序设计往往从问题的一部分着手，一点一点地构建出整个程序。面向对象设计是以数据为中心，使用类作为表现数据的工具，类是划分程序的基本单位。而函数在面向对象设计中成了类的接口。以数据为中心而不是以功能为中心来描述系统，相对来讲，更能使程序具有稳定性。它将数据和对数据的操作封装到一起，作为一个整体进行处理，并且采用数据抽象和信息隐藏技术，最终将其抽象成一种新的数据类型——类。类与类之间的联系及类的重用出现了类的继承、多态等特

性。类的集成度越高，越适合大型应用程序的开发。另外，面向对象程序的控制流程运行时，是事件进行驱动的，而不再由预定的顺序进行执行。事件驱动程序的执行围绕消息的产生与处理，靠消息的循环机制来实现。更加重要的是，可以利用不断成熟的各种框架，比如 . NET 中的 . NET Framework 等。在实际的编程过程中，使用这些框架能够迅速地将程序构建起来。面向对象的程序设计方法还能够使程序的结构清晰简单，能够大大提高代码的重用性，有效地减少程序的维护量，提高软件的开发效率。

在结构上，面向对象程序设计和结构化程序设计也有很大的不同。结构化程序设计首先应该确定的是程序的流程怎样走，函数间的调用关系怎样，也就是函数间的依赖关系是什么。一个主函数依赖于其子函数，这些子函数又依赖于更小的子函数，而在程序中，越小的函数处理的往往是细节的实现，这些具体的实现又常常变化。这样的结果，就使程序的核心逻辑依赖于外延的细节。程序中本来应该是比较稳定的核心逻辑，也因为依赖于易变化的部分，而变得不稳定起来，一个细节上的小小改动，也有可能在依赖关系上引发一系列变动。可以说这种依赖关系也是过程式设计不能很好地处理变化的原因之一，而一个合理的依赖关系，应该是倒过来的，由细节的实现依赖于核心逻辑才对。而面向对象程序设计是由类的定义和类的使用两部分组成的，主程序中定义数个对象并规定它们之间消息传递的方式，程序中的一切操作都是通过面向对象的发送消息机制来实现的。对象接收到消息后，启动消息处理函数完成相应的操作。

1.2.2　面向对象三大要素

封装、继承、多态是面向对象程序的三大特征，这些特征保证了程序的安全性、可靠性、可重用性和易维护性。随着技术的发展，把这些思想用于硬件、数据库、人工智能技术、分布式计算、网络、操作系统等领域，越来越显示出其优越性。

1. 封装（Encapsulation）

封装就是把对象的状态和行为绑到一起的机制，使对象形成一个独立的整体，并且尽可能地隐藏对象的内部细节。封装有两个含义：一是把对象的全部状态和行为结合在一起，形成一个不可分割的整体。对象的私有属性只能够由对象的行为来修改和读取；二是尽可能隐蔽对象的内部细节，与外界的联系只能够通过外部接口来实现。

2. 继承（Inheritance）

继承是一种连接类与类之间的层次模型。继承是指特殊类的对象拥有其一般类的属性和行为。继承意味着“自动地拥有”，即在特殊类中不必重新对已经在一般类中定义过的属性和行为进行定义，而是自动、隐含地拥有其一般类的属性和行为。继承对类的重用性提供了一种明确表述共性的方法。即一个特殊类既有自己定义的属性和行为，又有继承下来的属性和行为。尽管继承下来的属性和行为在特殊类中是隐式的，但无论在概念上还是在实际效果上，都是这个类的属性和行为。继承是传递的，当这个特殊类被它更下层的特殊类继承的时候，它继承来的和自己定义的属性与行为又被下一层的特殊类继承下去。我们把一般类称为基类，把特殊类称为派生类。通过继承可以让代码更加简洁、易读，易于重用和扩展，易于维护和修改。

3. 多态（Polymorphism）

多态是指两个或多个属于不同类的对象对于同一个消息或方法调用所做出不同响应的能

力。面向对象设计也借鉴了客观世界的多态性，体现在不同的对象可以根据相同的消息产生各自不同的动作。具体到面向对象程序设计，多态性是指在两个或多个属于不同类的对象中，同一函数名对应多个具有相似功能的不同函数，可以使用相同的调用方式来调用这些具有不同功能的同名函数。

面向对象技术发展的重大成果之一就是出现了统一建模语言——UML。面向对象技术领域内占主导地位的标准建模语言，统一了过去相互独立的数十种面向对象的建模语言共同存在的局面，通过统一语义和符号表示，系统地对软件工程进行描述和构造，形成了一个统一的、公共的、具有广泛适用性的建模语言。

1.3 UML 概述

1.3.1 UML 发展历史

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，其数量从不到十种增加到了五十多种。在众多的建模语言中，语言的创造者们都在努力推崇自己的产品，并在实践中不断完善。但是，OO 方法（Object - Oriented Method，面向对象的方法）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，一批新方法出现了，其中最引人注目的是 Booch 1993、OOSE 和 OMT - 2 等。

Grady 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 比较适用于系统的设计和构造。

James Rumbaugh 等人提出了面向对象的建模技术（OMT，一种软件开发方法）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，至 2013 年已很少使用。

综上所述，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，极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 开始致力于这一工作。他们首先将 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，称为统一方法 UM 0.8（United Method）。1995 年秋，OOSE 的创始人 Ivar Jacobson 加盟到这一工作。经过 Booch、Rumbaugh 和 Jacobson 三人的共同努力，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）。

1996 年，一些机构将 UML 作为其商业策略已日趋明显。UML 的开发者得到了来自公众的正面反应，并倡议成立了 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 及 Unisys。这一机构对 UML 1.0（1997 年 1 月）及 UML 1.1（1997 年 11 月 17 日）的定义和发布起了重要的促进作用。

2001 年，UML 1.4 这一版本被核准推出。2003 年，UML 2.0 标准版发布，它建立在 UML 1.x 基础之上，大多数 UML 1.x 模型在 UML 2.0 中都可用。但 UML 2.0 在结构建模方面有一系列重大的改进，包括结构类、精确的接口和端口、扩展性、交互片段和操作符，以及基于时间建模能力的增强。

UML 版本变更得比较慢，主要是因为建模语言的抽象级别更高，所以，相对而言，实现语言如 C#、Java 等版本变化更加频繁。2010 年 5 月发布了 UML 2.3。UML 2.4 所有的技术环节于 2012 年 1 月完成，2017 年 12 月发布了 UML 2.5。同时，UML 也被 ISO 吸纳为标准：ISO/IEC19501 和 ISO/IEC19595。

1.3.2　UML 的内容

UML 是一种定义良好、易于表达、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。它的作用域不限于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。

UML 包含了 3 个方面的内容：模型的概念和表示法、语言的公共机制、对象约束语言。

1. UML 模型的概念和表示法

UML 有三类基本的标准模型建筑块：事物、联系和图形。

UML 规定了 4 种事物表示法：结构事物、动作事物、分组事物和注释事物。

①结构事物指模型的静态部分，如对象类、Use Case、接口、组件等；

②动作事物指模型的动态部分，如交互、状态机等；

③分组事物指模型的组织部分，如包；

④注释事物指模型的解释说明部分，如注释。

UML 提供的模型建筑块之间的基本联系有 4 种：依赖、关联、泛化、实现。依赖是指模型建筑块之间的一种语义联系，其中一个独立事物发生改变将影响另一个依赖事物的语义。关联是指模型建筑块之间的结构联系，两者存在结构性的连接。聚合是一种特殊的关联，表示结构的整体与部分的关系。泛化是指模型建筑块之间的一般与特殊的联系。实现是指模型建筑块之间的一种语义联系，其中一个规定了一组约定，另一个负责实现它们。例如，接口和实现接口功能的类或组件之间的联系就是实现。