清华
电脑学堂

- 总结了作者UML建模经验和教学心得
- 系统讲解了UML建模应用的要点和难点
- 实例众多、图例丰富、实用性强
- 提供丰富的课堂练习和课后习题
- 附赠大容量、高品质素材和案例

UML建模与应用标准教程

（2018-2020版）

夏丽华 卢 旭 编著

从零开始，循序渐进

本书介绍了UML建模与应用的基础知识和各类图表绘制技巧，内容由浅入深，循序渐进，适合零基础读者快速入门。

延伸学习，深入掌握

本书赠送20节计算机基础操作视频，帮助读者延伸学习内容，同时赠送30个办公软件模板和UML建模PPT。

系统全面，易学易用

本书构筑了面向实际应用的知识体系，体现了理论的适度性、实践的指导性和应用的典型性，对难点和重点做了详细讲解和特别提示。

全程图解，快速上手

本书采用全程图解方式，图像做了大量裁切、拼合、加工，插图做了标注处理，信息丰富，阅读体验轻松，上手容易。

紧贴实际，案例导航

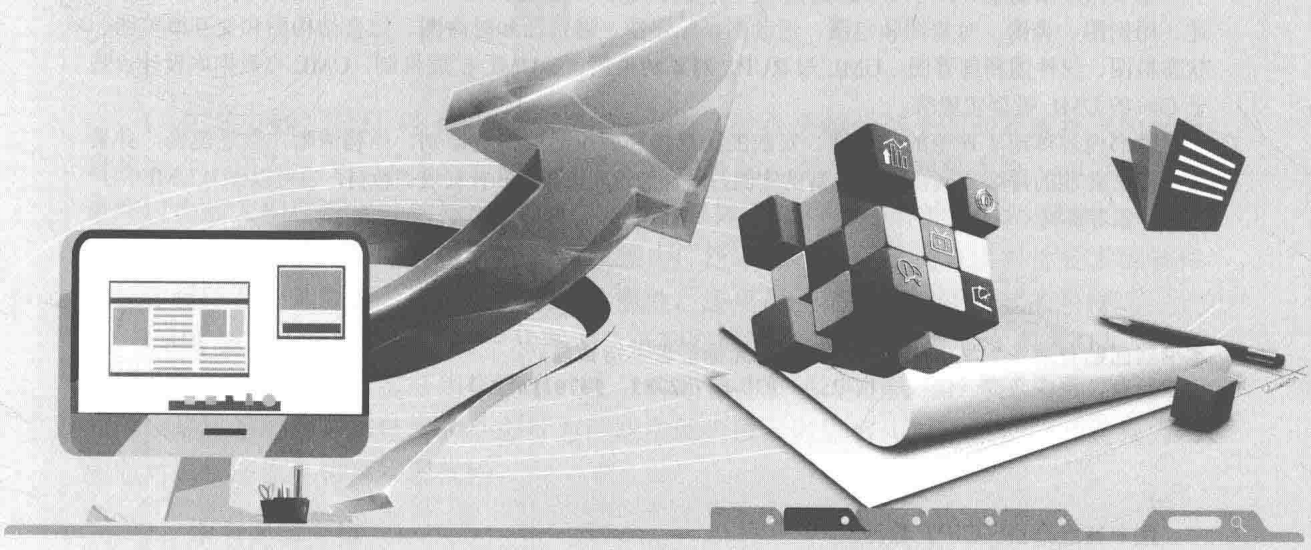
每章根据所讲内容配备精彩案例和课后练习，读者可边学边练，既可全面了解UML各种图表的绘制方法，又可快速掌握基于实际应用的项目和任务。

海量资源，轻松获取

本书附赠海量资源，已经上传到“益阅读”空间，读者只需扫描封底二维码，便可轻松获取丰富的学习资源。

清华大学出版社

清華
电脑学堂



UML建模与应用标准教程

（2018–2020版）

夏丽华 卢 旭 编著

清華大学出版社
北京

内 容 简 介

本书循序渐进地介绍了 UML 建模、分析与开发的基础知识，全书共分 15 章，内容涉及 UML 概述、用例图、类图、对象图和包图、活动图、顺序图、通信图和时序图、组合结构图和交互概览图、状态机图、组件图和部署图、UML 与 RUP、对象约束语言、UML 扩展机制、UML 与数据库设计、基于 C++的 UML 模型实现等。

本书内容体现了理论的适度性、实践的指导性和应用的典型性原则，结构清晰，叙述流畅，并采用了图文并茂的排版方式，结合丰富的实例，适合作为高校教材和社会培训教材，也可以作为 UML 用户的自学参考资料。

图书在版编目（CIP）数据

UML 建模与应用标准教程（2018—2020 版）/夏丽华，卢旭编著. —北京：清华大学出版社，2018
（清华电脑学堂）
ISBN 978-7-302-47471-5

Ⅰ. ①U…　Ⅱ. ①夏… ②卢…　Ⅲ. ①面向对象语言-程序设计-教材　Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2017）第 133575 号

责任编辑：冯志强　陈绿春
封面设计：杨玉芳
责任校对：徐俊伟
责任印制：王静怡

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张：**19.5　　**字　　数：**463 千字
版　　次：2018 年 1 月第 1 版　　**印　　次：**2018 年 1 月第 1 次印刷
印　　数：1～3000
定　　价：59.80 元

产品编号：070268-01

前　言

UML 在 1997 年 11 月 17 日被对象管理组织 OMG 采纳为基于面向对象技术的标准建模语言，它不仅统一了面向对象方法中的符号表示，而且在其基础上进一步发展，并最终成为被人们广泛接受的标准。UML 适合于以体系结构为中心的、用例驱动的、迭代式和渐增式的软件开发过程，其应用领域颇为广泛，除了可用于具有实时性要求的软件系统建模以及处理复杂数据的信息系统建模外，还可用于描述非软件领域的系统。

UML 适用于系统开发过程中从需求分析到完成测试的各个阶段：在需求分析阶段，可以用用户模型视图来捕获用户需求；在分析和设计阶段，可以用静态结构和行为模型视图来描述系统的静态结构和动态行为；在实现阶段，可以将 UML 模型自动转换为用面向对象程序设计语言实现代码。

1. 本书内容介绍

本书以渐进的顺序来介绍 UML，从需求分析开始，逐步构建和部署系统。全书共分为 15 章，各章内容概括如下。

第 1 章：对 UML 进行概述，包括面向对象开发、UML 的发展历程、UML 统一的作用、UML 体系结构、UML 核心元素等基础知识。

第 2 章：全面介绍了用例图，包括用例图的构成、泛化关系、包含关系、扩展关系、用例描述等基础知识。

第 3 章：全面介绍了类图，包括类图的概念、泛化关系、依赖关系、实现关系、二元关联、关联类、聚合关系、组合关系等基础知识。

第 4 章：全面介绍了对象图和包图，包括对象和类、对象和链、使用对象图测试类图、导入包、包图概述、包之间的关系、对象图和包图建模等基础知识。

第 5 章：全面介绍了活动图，包括活动图概述、活动图的组成元素、分支与合并、分叉与汇合等基础知识。

第 6 章：全面介绍了顺序图，包括顺序图概述、对象、生命线、消息、建模和执行等基础知识。

第 7 章：全面介绍了通信图和时序图，包括通信图概述、消息序列号与控制点、创建对象、消息迭代、时序图概述、时间约束、时序图的替代表示法等基础知识。

第 8 章：全面介绍了组合结构图和交互概览图，包括组合结构图的内部结构、端口、协作、交互概览图的组成部分、使用交互、组合交互等基础知识。

第 9 章：主要介绍了状态机图，包括状态机概述、转移、事件、动作、转移的类型、顺序状态、并发子状态、同步状态、历史状态、子状态机引用状态等基础知识。

第 10 章：全面介绍了组件图和部署图，包括组件图概述、部署图概述、组件间的关系与组件嵌套、结点和连接、部署间的关系等基础知识。

第 11 章：全面介绍了 UML 与 RUP，包括理解软件开发过程、RUP 的作用、RUP

的特点、RUP 的二维空间、核心工作流程等基础知识。

第 12 章：全面介绍了对象约束语言，包括对象约束语言概述、数据类型、集合、使用约束、对象级约束、消息级约束、约束和泛化等基础知识。

第 13 章：全面介绍了 UML 扩展机制，包括 UML 的体系结构、UML 的核心语义、构造型、标记值、约束等基础知识。

第 14 章；全面介绍了 UML 与数据库设计，包括数据库设计概述、类图到数据库的转换、完整性与约束验证、数据库实现与转换技术等基础知识。

第 15 章：全面介绍了基于 C++的 UML 模型实现，包括模型元素的简单实现、实现关联、受限关联的实现、UML 关系的实现、特殊类的实现等基础知识。

2. 本书主要特色

- **系统全面** 本书提供了全面、丰富的 UML 应用案例，通过实例分析、设计过程讲解 UML 建模与应用的知识，便于读者学习操作，同时方便教师组织授课。
- **课堂练习** 本书各章都安排了课堂练习，全部围绕实例讲解相关内容，灵活生动地展示了计算机网络组建与管理的各个应用知识点。课堂练习体现本书实例的丰富性，方便读者组织学习。每章后面还提供了思考与练习，用来测试读者对本章内容的掌握程度。
- **全程图解** 各章内容全部采用图解方式，图像均做了大量的裁切、拼合、加工，信息丰富，效果精美，阅读体验轻松，上手容易。

3. 本书使用对象

本书从UML 建模与应用的基础知识入手，全面介绍UML 建模面向应用的知识体系。本书设计了课堂练习，图文并茂，能有效地吸引读者学习。本书适合作为高职高专院校学生学习使用，也可作为计算机办公应用用户深入学习 UML 建模与应用的培训和参考资料。

参与本书编写的人员除了封面署名人员之外，还有于伟伟、王翠敏、谢华、冉洪艳、张振、吕咏、王修红、扈亚臣、孙佳星、张彬、刘红娟、程博文等人。由于作者水平有限，疏漏之处在所难免，欢迎读者朋友登录清华大学出版社的网站 www.tup.com.cn 与我们联系，帮助我们改进提高。

编 者

2017 年 3 月

目　录

第1章

UML概述

1.1 面向对象开发

1. 面向对象概述

面向对象（Objec-Oriented，OO）不仅是一些具体的软件开发技术与策略，而且是一整套关于如何看待软件系统与现实世界的关系，用什么观点来研究问题并进行求解，以及如何进行系统构造的软件方法学。

面向对象的核心是对象，它是系统中用来描述客观事物的一个实体，它是构成系统的一个基本单位。一个对象由一组属性和对这组属性进行操作的一组服务组成。从更抽象的角度来说，对象是问题域或实现域中某些事物的一个抽象，它反映该事物在系统中需要保存的信息和发挥的作用；它是一组属性和有权对这些属性进行操作的一组服务的封装体。客观世界是由对象和对象之间的联系组成的。

面向对象的软件工程方法的基础是面向对象的编程语言。一般认为诞生于 1967 年的 Simula-67 是第一种面向对象的编程语言。尽管该语言对后来许多面向对象语言的设计产生了很大的影响，但它没有后继版本。继而 20 世纪 80 年代初 Smalltalk 语言掀起了一场“面向对象”运动。随后便诞生了面向对象的 C++、Eiffel 和 CLOS 等语言。尽管在当时面向对象的编程语言在实际使用中具有一定的局限性，但它仍吸引了广泛的注意，一批批面向对象编程书籍层出不穷。直到今天面向对象编程语言数不胜数，在众多领域发挥着各自的作用，如 C++、Java、C#、VB.NET 和 C++.NET 等。随着面向对象技术的不断完善，面向对象技术逐渐在软件工程邻域得到了应用。

面向对象的软件工程方法包括面向对象的分析（OOA）、面向对象的设计（OOD）、面向对象的编程（OOP）等内容。

1）面向对象的分析

OOA 就是应用面向对象方法进行系统分析。OOA 是面向对象方法从编程领域向分析领域发展的产物。从根本上讲，面向对象是一种方法论，不仅仅是一种编程技巧和编程风格，而且是一套可用于软件开发全过程的软件工程方法，OOA 是其中的第一个环节。OOA 的基本任务是运用面向对象方法，从问题域中获取需要的类和对象，以及它们之间的各种关系。

2）面向对象的设计

OOD 指面向对象设计，在软件设计生命周期中发生于 OOA 后期或者之后。在面向对象的软件工程中，OOD 是软件开发过程中的一个大阶段，其目标是建立可靠的、可实现的系统模型；其过程是完善 OOA 的成果，细化分析。其与 OOA 的关系为：OOA 表达了“做什么”，而 OOD 则表达了“怎么做”，即分析只解决系统“做什么”，不涉及“怎么做”，而设计解决“怎么做”的问题。

3）面向对象的编程

OOP 就是使用某种面向对象的语言，实现系统中的类和对象，并使得系统能够正常运行。在理想的 OO 开发过程中，OOP 只是简单地使用编程语言实现了 OOA 和 OOD 分析和设计模型。

2. 面向对象开发概述

面向对象开发方法的原则是鼓励软件开发者在软件生命周期内应用其概念来工作和

思考。只有较好地识别、组织和理解了应用领域的内在概念，才能有效地表达出数据结构和函数的细节。

面向对象开发只有到了最后几个阶段才不是独立于编程语言的概念过程。所以可以将面向对象开发看作是一种思维方式，而不是一种编程技术。它的最大好处在于帮助规划人员、开发者和客户清晰地表达抽象的概念，并将这些概念互相传达。它可以充当规约、分析、文档、接口以及编程的一种媒介。

为了加深读者对面向对象开发的理解，下面将它与传统的软件开发作比较。面向对象的开发方法把完整的信息系统看成对象的集合，用这些对象来完成所需要的任务。对象能根据情况执行一定的行为，并且每个对象都有自己的数据。而传统开发方法则把系统看成一些与数据交互的过程，这些数据与过程隔离保存在不同的文件中，当程序运行时，就创建或修改数据文件。图 1-1 显示了面向对象开发与传统软件开发之间的区别。

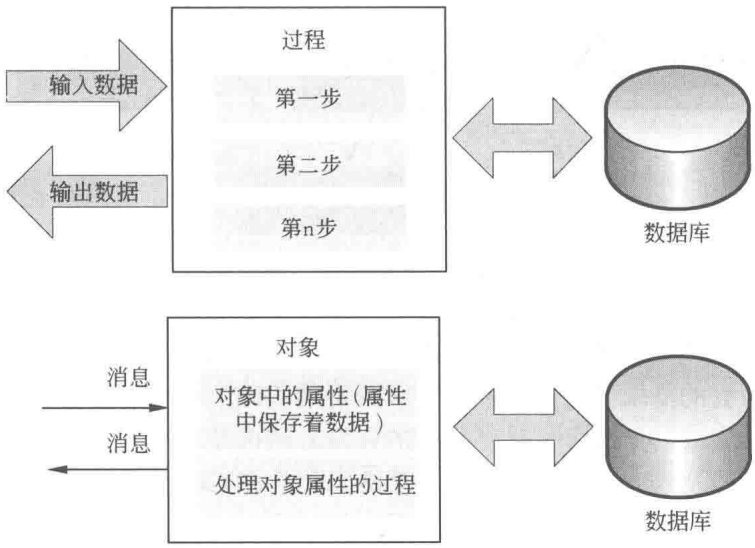


图 1-1　传统方法与面向对象方法的比较

过程通过接收输入的数据，然后对它进行处理，随后保存数据或输出数据。面向对象则是通过接收消息来更新它的内部数据。这些差别虽然看起来简单，但对于整个系统的分析、设计和实现来说却非常重要。

任何一种开发方法，在开发任何系统之前，开发人员都会对此项目先进行分析。系统需求分析就是对系统需求的研究、了解和说明。系统需求定义了系统要为从事业务活动的用户完成的任务。这些需求一般用图表来描述，对图表进行规范化后就构成了该系统的基本需求模型。系统分析过程中建立的模型被称为逻辑模型，因为它仅仅描述了系统的需求，而不涉及到如何实现此需求。系统设计就是建立一个新模型，该模型展示了组成软件系统所使用的技术。系统设计过程中所建立的模型也称为物理模型。

在传统的结构化分析和设计中，开发人员也使用图形模型，如数据流图（DFD）用来表示输入、输出和处理，还要建立实体关系图（ERD）以表示有关存储数据的详细资料。它的设计模型主要由结构图等构成。

在 OO 开发中，因为需要描述不同的对象，所以 OO 开发中所建立的模型不同于传

统的模型。例如，OO 开发不仅需要用数据和方法来描述建模，还需要用模型来描述对象之间的交互。OO 开发中使用 UML 来构造模型。

OO 开发方法不仅在模型上与传统的开发方法不同，在系统开发生命周期也有不同。系统开发生命周期是开发一个项目的管理框架，它列出了开发系统时的每个阶段和在每个阶段所要完成的任务。几个主要阶段有计划、分析、设计、实现和支持。系统开发生命周期最初用在传统的系统开发中，但它也能用于 OO 开发中。OO 开发人员经常使用迭代开发方法来分析、设计和实现。

迭代开发方法就是先分析、设计，编写部分程序完成系统需求的一部分。然后再分析—设计—实现完成其他需求。图 1-2 演示了迭代开发方法。

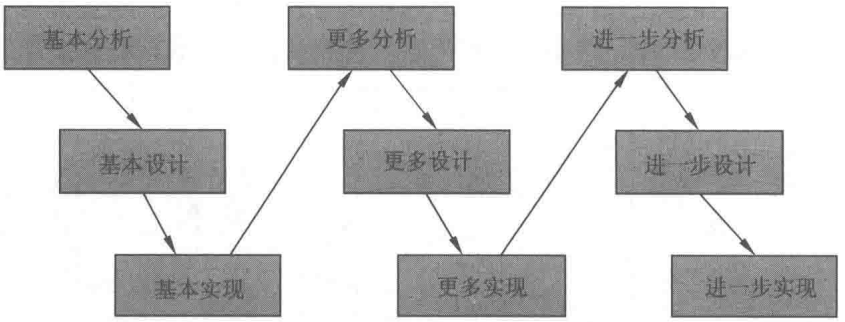


图 1-2 迭代开发

迭代开发方法和早期瀑布开发方法形成了鲜明的对比。在瀑布开发方法中，在开始设计前要完成所有的需求分析，然后在需求分析的基础上进行系统设计，编程工作要在系统分析和设计完成后才进行。虽然传统的开发方法也使用了迭代开发方法，但是因为每个迭代过程都涉及到改进和增加模块的功能，而且在 OO 开发过程中每次迭代增加一个类，所以 OO 开发比传统的开发更适用于迭代开发。

OO 方法在建造系统模型和系统如何工作方面和传统的编程不同，在系统开发生命周期和项目管理中，OO 开发仍然和传统的系统开发有相似之处。

1.1.2 面向对象的主要特征

为了进一步理解面向对象的内涵，还需要进一步了解面向对象的主要特征。

1. 抽象

抽象（Abstract）就是忽略事物中与当前目标无关的非本质特征，更充分地注意与当前目标有关的本质特征，从而找出事物的共性，并把具有共性的事物划为一类，得到一个抽象的概念。

例如，在设计一个学生管理系统的过程中以学生李华为例时，就只关心他的学号、班级、成绩等，而忽略他的身高、体重等信息。因此，抽象性是对事物的抽象概括和描述，实现了客观世界向计算机世界的转化。将客观事物抽象成对象及类是比较难的过程，也是面向对象方法的第一步。例如，将学生抽象成对象及类的过程如图 1-3 所示。

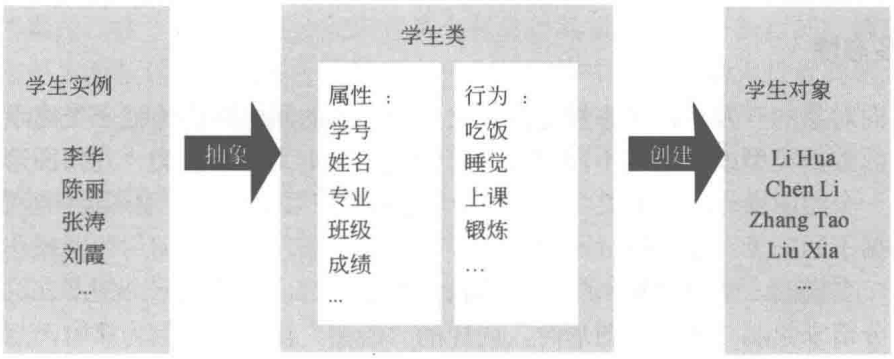


图 1-3 抽象过程示意图

2. 封装

封装是面向对象的一个重要原则。封装指将对象属性和操作结合在一起，构成一个独立的对象。它的内部信息是隐蔽的，不允许外界直接存取对象的属性，而只能通过指定的接口与对象联系。

封装使得对象属性和操作紧密给合在一起，这反映了事物的状态特性与动作是事物不可分割的特征。系统中把对象看成其属性和操作的结合体，就使对象能够集中而完整地描述一个事物。这避免了将数据和功能分离开进行处理，使得系统的组成与现实世界中的事物具有良好的对应性。

封装的信息隐蔽作用反映事物的独立性。这样使得对于对象外部而言，只需要注意对象对外呈现的行为，而不必关心其内部的工作细节。封装可以使软件系统的错误局部化，因而大幅减少了查错和排错的难度。另一方面，当修改对象内部时，由于它只通过操作接口对外部提供服务，因此大大减少了内部修改对外部的影响。

3. 继承

继承是指子类可以拥有父类的全部属性和操作。继承是 OO 方法的一个重要的概念，并且是 OO 技术可以提高软件开发效率的一个重要原因。

在建造系统模型时，可以根据所涉及到的事物的共性抽象出一些基本类，在此基础上再根据事物的个性抽象出新的类。新类既具有父类的全部属性和操作，又具有自己独特的属性和操作。父类与子类的关系为一般与特殊的关系。

继承机制具有特殊的意义。由于子类可以自动拥有父类的全部属性和操作，这样使得定义子类时，不必重复定义那些在父类中已经定义过的属性和操作，只需要声明该类是某个父类的子类，将精力集中在定义子类所特有的属性和操作上，这样就提高了软件的可重用性。

断承具有传递性。如果子类 B 继承了类 A，而子类 C 又继承了类 B，则子类 C 可以继承类 A 和类 B 的所有属性和操作。这样子类 C 的对象除了具有该类的所有特性外，还具有全部父类的所有特性。

如果限定每个子类只能继承单独一个父类的属性和操作，则这种继承称为单继承。在有些情况下，一个子类可以同时继承多个父类的属性和操作，这种继承称为多重继承。

4. 多态性

在面向对象的开发中，多态性是指在父类中定义的属性和操作被子类继承后，可以具有不同的数据类型或表现出不同的行为。例如，在定义一个父类“几何图形”时，为其定义了一个绘图操作。当子类“椭圆”和“矩形”都继承了几何图形类的绘图操作时，该操作根据不同的类对象将执行不同的操作，在“椭圆”类对象调用绘图操作时，该操作将绘制一个椭圆，而当“矩形”类对象执行该操作时，将绘制一个矩形。这样当系统的其他部分请求绘制一个几何图形时，同样的“绘图”操作消息因为接收消息的对象不同，将执行不同的操作。

在父类与子类的类层次结构中，利用多态性可以使不同层次的类之间共享一个方法名，而各自有不同的操作。当一个对象接收到一个请求消息时，所采取的操作将根据该对象所属的类决定。

在继承父类的属性和操作的名称时，子类也可以根据自己的情况重新定义该方法。这种情况称为重载。重载是实现多态性的方法之一。

5. 关联

在现实世界中，事物不是孤立的、互相无关的，而是彼此之间存在着各种各样的联系。例如在一个学校中，有教师、学生、教室等事物，他们之间存在着某种特定的联系。在面向对象的方法中，用关联来表示类或对象集合之间的这种关系。在面向对象中，常把对象之间的连接称为链接，而把存在对象连接的类之间的联系称为关联。

如果在 OOA 和 OOD 阶段定义了一个关联，那么在实现阶段必须通过某种数据结构来实现它。关联还具有多重性，多重性表示参加关联的对象之间数量上的约束，有一对一、一对多、多对多等不同的情况。

6. 聚合

现实世界中既有简单的事物，也有复杂的事物。当人们认识比较复杂的事物时，常用的思维方法为：把复杂的事物分解成若干个比较简单的事物。在面向对象的技术中像这样将一个复杂的对象分解为几个简单对象的方法称为聚合。

聚合是面向对象方法的基本概念之一。它指定了系统的构造原则，即一个复杂的对象可以分解为多个简单对象。同时它也表示为对象之间的关系：一个对象可以是另一个对象的组成部分，同时该对象也可以由其他对象构成。

7. 消息

消息是指对象之间在交互中所传递的通信信息。当系统中的其他对象需要请求该对象执行某个操作时，就向其发送消息，该对象接收消息并完成指定的操作，然后把操作结果返回到请求服务的对象。

一个消息一般应该含有如下信息：接收消息的对象、请求该对象提供的服务、输入信息和响应信息。

消息在面向对象的程序中具体表现为函数调用，或其他类似于函数调用的机制。对

于一个顺序系统，由于其不存在并发执行多个任务，其操作是顺序执行的，因此其消息实现目前主要为函数调用。而在并发程序和分布式程序中，消息则为进程间的通信机制和远程调用过程等其他通信机制。

1.1.3 面向对象的层和模型

了解了面向对象的主要特征之后，还需要了解一下面向对象的层和模型，才能进一步了解并制作 UML 建模。

1. 面向对象的层

面向对象的开发中，通常把面向对象系统中相互联系的所有对象分成 3 层：数据访问层、业务逻辑层和界面表示层，区分层次的目的是为了“高内聚、低耦合”的思想。它们的作用如下。

1）数据访问层

主要是对原始数据（数据库或者文本文件等存放数据的形式）的操作层，而不是指原始数据，也就是说，是对数据的操作，而不是数据库，具体为业务逻辑层或表示层提供数据服务。

2）业务逻辑层

主要是针对具体的问题的操作，也可以理解成对数据层的操作，对数据业务进行逻辑处理，如果说数据层是积木，那逻辑层就是对这些积木的搭建。

3）界面表示层

简单来说就是展现给用户的界面，即用户在使用一个系统时的所见所得，像菜单、按钮和输入框等都属于这一层。图 1-4 显示了在图书管理系统中添加学生和借书信息操作时的 3 层过程。

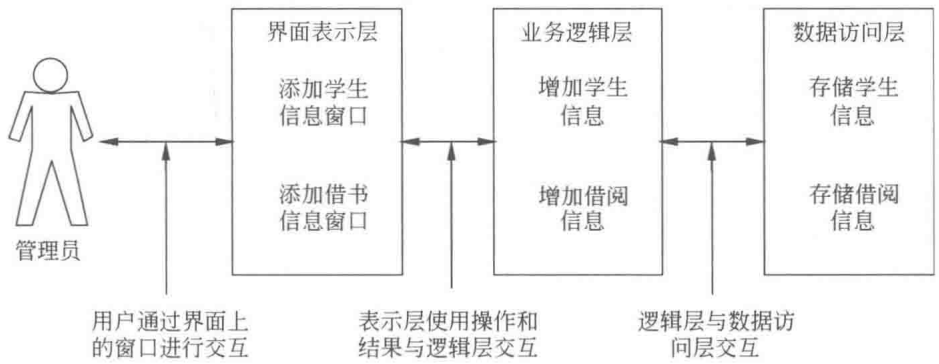


图 1-4 图书管理系统 3 层分析

从图 1-4 中可以看出，管理员和图形用户界面（表示层）交流，图形用户界面一般由包含表示对象的窗口组成，窗口中包含按钮、菜单、工具栏的窗体。用户不能直接和业务逻辑层交互，而是通过鼠标和键盘对用户界面进行操作，使表示层与业务逻辑层交互。

当业务逻辑层中的对象需要保存实现持久化时，就需要使用数据库实现对象的持久性，即保存对象中的数据。每个过程需要为每个逻辑类定义一个单独的数据访问层，以便处理数据和保存有用的信息。

提　示

这 3 层构成了系统的物理模型，在构造系统模型过程中，开发人员会使用 UML 语言作为建造模型的工具。下节将介绍与此对应的 3 种模型。

2. 面向对象的模型

使用 3 种模型从不同的视角来描述系统，它们分别是描述系统内部对象及其关系的类模型，描述对象生命历史的状态模型，以及描述对象之间交互行为的交互模型。每种模型都会在开发的所有阶段中得到应用，并随着开发过程的进行获得更多的细节。对系统的完整描述需要所有这 3 种视角的模型。

1）类模型

类模型（Class Model）描述了系统内部对象及其关系的静态结构。类模型界定了软件开发的上下文，它包含类图。类图（Class Diagram）的结点是类，弧表示类间的关系。

2）状态模型

状态模型（State Model）描述了对象随着时间发生变化的那些方面。状态模型使用状态图确定并实现控制。状态图（State Diagram）的结点是状态，弧是由事件引发的状态间的转移。

3）交互模型

交互模型（Interaction Model）描述系统中的对象如何协作以完成更为广泛的任务。交互模型自用例开始，用例随后会用顺序图和活动图详细描述。用例（Use Case）关注系统的功能，即系统为用户做了哪些事情。顺序图（Sequence Diagram）显示交互的对象及发生交互的时间顺序。活动图（Activity Diagram）描述重要的处理步骤。

上述的 3 个模型描述了一套完整的系统的相互独立的部分，但它们又是交叉相连的。类模型是最基本的，因为在描述何时及如何发生变化之前，要先描述是哪些内容正在发生变化或转化。

1.2　认识 UML

统一建模语言（UML）仅仅是一种语言。它不是系统设计的方法，而是系统建模的标准。UML 经历了多年的研究、发展并不断完善，成为现在诸多领域内建模的首选标准。开发人员主要使用 UML 来构造各种模型，以便描述系统需求和设计。

1.2.1　UML 的发展历程

由于 Booch 方法和 OMT 方法都已经独自成功地发展成为世界上主要的面向对象方法，因此 Grady Booch 和 Jim Rumbaugh 于 1994 年 10 月共同合作把他们的工作统一起来。

1995 年成为“统一方法（Unified Method）”0.8 版。之后 Ivar Jacobson 加入，吸取