统一建模语言 UML 及其建模实例

王瑞金[1]，段会川[2]，Martin Gogolla[3]
(1. 山东大学 管理学院，山东 济南 250100；2. 山东师范大学 计算机系，山东 济南 250014；3. 不莱梅大学 计算机系，德国)

摘 要：以统一建模语言(Unified Modeling Language，UML)的九种图为重点，简要介绍了 UML 的基本概念、历史、内容及其应用；并通过对一个超级市场销售管理系统的实际建模，简要演示了应用 UML 建模的思路。

关键词：统一建模语言；面向对象；系统分析与设计
中图法分类号：TP311.52 **文献标识码**：A **文章编号**：1001-3695(2002)08-0080-05

Unified Modeling Language and Its Application for Modeling

WANG Rui-jin[1], DUAN Hui-chuan[2], Martin Gogolla[3]
(1. *School of Management, Shandong University, Jinan Shandong* 250100, *China*; 2. *Dept. of Computer Science, Shandong Teacher's University, Jinan Shandong* 250014, *China*; 3. *Dept. of Computer Science, University of Bremen, Germany*)

Abstract: In the present paper we introduced briefly the main concepts of the Unified Modeling Language (UML), concentrating on the nine kinds of diagrams. In order to show a clue to the application of the UML, we built a simple model for the sales management system in a supermarket using the UML.

Key words: Unified Modeling Language(UML); Object-Oriented(OO); System Analysis and Design

引言

在管理信息系统的开发过程中，为系统建模恰似为一个建筑描绘一个蓝图一样重要。随着面向对象分析与设计(OOA&D)技术的发展，出现了多种用于系统建模的 OO 建模语言。其中最具代表性并得到广泛应用的有 OOSE[1]（面向对象的软件工程）、OMT[2] 和 Booch[3]。它们各自都是一套完备的方法，并且具有某方面的优势。OOSE 是一种面向用例的方法，比较适合于商业工程和需求分析；OMT 适用于描述分析与以数据为中心的信息系统；Booch 特别适用于系统的设计与构造。在如此众多的 OO 建模语言中，并没有一种居于支配地位，且除了一些细微差别外，都共享一些公认的概念，从而为用户在应用中的选择造成了困难，在进入 OO 市场和采用 OO 建模方法时产生了犹豫。因此，有必要在比较不同建模语言优缺点和总结面向对象技术应用实践的基础上，统一建模语言。UML 正是在这种背景下诞生的。

2 UML 简介[4~8]

统一建模语言(Unified Modeling Language，UML)试图建立一种统一的、面向对象的建模方法，是一种用于面向对象和基于构件的、系统建模的、定义明确的、被广泛接受的可视化建模语言。UML 是由包括 Rational Software 公司在内的联盟开发的，它继承了 Booch，OOSE，OMT 和其它方法的思想。许多公司已将 UML 作为一个商业策略而纳入他们的开发过程和产品中，涵盖许多领域，比如商业建模、需求管理、分析和设计、编程和测试等。

2.1 UML 的发展

UML 的发展开始于 1994 年 10 月，Rational Software 公司的 Grady Booch 和 Jim Rumbaugh 开始致力于统一 Booch 和 OMT 方法的工作。鉴于 Booch 和 OMT 方法在发展过程中已相互借鉴，并且在世界范围内被共同认定为具有代表性的面向对象的方法，Booch 和 Rumbaugh 联合起来致力于两者的统一。1995 年发布了第一个公开版本，称为统一方法 UM0.8(Unitied Method)。在 1995 年秋季，Ivar Jacobson 和他的 Objectory 公司加盟这项工作，融合进 OOSE 方法。

Booch，Rumbaugh 和 Jacobson 努力的结果就是 1996 年 6 月和 10 月 UML 0.9 和 UML 0.91 的推出。在同一年，许多组织开始将 UML 作为他们的商业策略。由 Object Management Group(OMG)发起的 Request For Proposal (RFP)为这些组织联合力量创建一个共同 RFP 响应提供了契机。Rational 建立了 UML 成员协会，由许多希望为定义 UML 而贡献资源的组织构成。合作者包括：Digital Equipment Corp.，HP，I-Logix，IntelliCorp，IBM，ICON Computing，MCI Sustemhouse，Microsoft，Oracle，Rational

收稿日期：2001-08-14；修返日期：2001-12-22

Software, TI 和 Unisys。这种联合产生了 UML 1.0,一种定义良好、易于表达、功能强大、普遍适用的建模语言,并被作为 RFP 早期的响应提交给 OMG。

1997 年 IBM, ObjectTime, Platinum Technology, Ptech, Taskon & Reich Technologies 和 Softeam 也向 OMG 提交了各自的 RFP 响应。这些公司也成为 UML 的合作者,并提出了他们的意见和想法,并且和其他合作者一同制定了改进的 UML 1.1。UML 1.1 主要是改进了 UML 1.0 语义的清晰度,并融合进了新合作者的贡献。UML 1.1 在 1997 年被 OMG 采纳。随后 UML 的维护由 OMG Revision Task Force (RTF)接管。RTF 在 1999 年推出了 UML 1.3,目前将要推出 UML 2.0。

UML 的发展如图 1 所示。

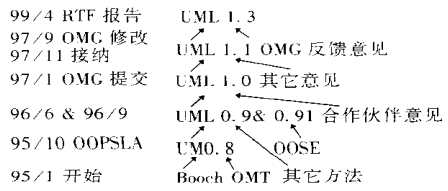

图 1　UML 发展历史

UML 为使用者提供了一个便于使用、表达丰富的可视的建模语言,使之能够开发和交流有意义的模型。它也提供了扩展和特化机制来扩展它的核心概念。UML 独立于详细的编程语言和开发过程,为了解建模语言提供了规范化的基础。它激励了 OO 工具市场的成长,并且支持诸如合作(Collaborations)、架构(Frameworks)、模板(Patterns)和构件(Components)等高层次的开发概念。

UML 与 Booch, OMT, OOSE 等其它先进的 OO 方法具有相似的语义和定义,并且融合了 UML 合作者的意见和公众的反馈。UML 达到了两个方面的统一:第一,它有效地结束了先前方法中建模语言之间许多通常不必要的差异;第二,而且也许更重要的是,它统一了不同系统间的观点、开发阶段(需求分析、设计和实现)和内部概念。因此可以预见,UML 将在全世界范围内被普遍使用,许多组织已将 UML 确定为他们的组织标准,因为它是建立在具有主导性的 OO 方法的建模语言基础上的。

2.2　UML 的内容

UML 的发展方向是简化和巩固大量已经存在的面向对象(OO)的建模方法,主要是集成 Booch, OOSE 和 OMT 的概念。UML 的大部分基本概念和现有的 OO 方法的概念类似。正是这些特性使 UML 成为最受使用者青睐的建模语言。

UML 定义了九种图,这些图被用来建立系统的静态(结构)和动态(行为)模型。

(1)结构图

UML 的结构图用于描述建立系统模型时在问题域中遇到的主要事物。

- 类图(Class Diagram)　它包括一组类、界面、合作以及它们之间的关系,例如联系、依赖和继承。它被用来建立系统静态模型,并在系统的整个生命周期中有效。
- 对象图(Object Diagram)　一个对象就是一个类的实例。一个对象图表示一组对象和它们之间在某一时刻的关系。它被用来为类图中的实例建立模型。
- 构件图(Component Diagram)　构件图描述一组构件和它们之间的关系。它被用来展示一个系统实现的静态视图。一个构件可以是一个源代码、一个二进制或是一个可执行单元。
- 配置图(Deployment Diagram)　配置图由一组节点以及它们之间的关系组成。它被用来表示一个包括软、硬件系统的物理结构的静态配置视图。配置图中的一个节点通常包含一个或多个构件。

(2)行为图

UML 行为图用于描述建立系统的动态模型。

- 用例图(Use Case Diagram)　一个用例描述系统的功能,并且从用户的角度指出每个功能的执行者(Actor)。一个用例图详细描述一个系统或者系统的一部分的行为。
- 状态图(Statechart Diagram)　状态图表现一个状态机器,包括状态(State)、转移(Transition)、事件(Event)和活动(Activity)。它被用来描述一个系统的动态视图,例如,一个对象所有可能的状态和触发状态转移的条件。它们在描述一个界面、类或者合作的行为时尤其重要。状态图强调一个对象的事序行为。
- 活动图(Activity Diagram)　活动图描述一系列活动、活动之间的顺序流动或分支流动,以及活动的发出或承受对象。它们在为一个系统的功能建模时尤其重要。活动图强调对象之间的控制流。
- 顺序图(Sequence Diagram)　顺序图是一个交互图,它强调消息(Message)的时间顺序。一个顺序图描述一组对象以及由这些对象发送和接收的消息。这些对象通常是命名或匿名的类的实例,但也可以代表其它事物的实例,比如合作、构件和节点。
- 合作图(Collaboration Diagram)　合作图也是一个交互图,它强调接收和发送消息的对象的结构(空间)组织。一个合作图描述一组对象、这些对象之间的连接以及这些对象发送和接收的消息。

状态图和活动图在语义上是等价的,也就是说一个状态图和一个活动图可以互相转换而不丢失任何信息。顺序图和合作图之间也存在着类似的等价关系。

根据图在系统开发过程中不同阶段的应用,这九种图还可以被分为以下五类:

- 用例图
- 静态图　类图和对象图
- 行为图　状态图和活动图
- 交互图　顺序图和合作图
- 实现图　构件图和配置图

2.3　UML 的应用领域

UML 是软件开发中设计系统蓝图使用的标准语言。UML 用可视化方法详细描述和构造一个软件系统。事实上,它并不只局限在软件建模,它还可以为非软件系统建模,如法律系统的工作流程、健康保健系统的结构与行为,以及硬件设计等。

UML 已经有效地应用在以下领域：

- 企业信息系统
- 银行和金融服务
- 电讯
- 交通
- 国防/太空工业
- 零售
- 医学电子数据处理
- 科研
- 基于 Web 的网络服务

3 UML 建模的应用实例[4,10]

UML 在很大程度上是过程无关的建模语言，也就是说它没有局限于任何特定的软件开发周期。为最大限度地发挥 UML 的优势，我们应该考虑的过程是：

- 用例驱动(Use Case Driven)
- 以架构为中心(Architecture-centric)
- 迭代和递增(Iterative and Increment)

这种用例驱动、架构为中心、迭代和递增的过程可以分为四个阶段：

- 开始(Inception)
- 详细描述(Elaboration)
- 构建(Construction)
- 移交(Transition)

开始和详细描述包括开发周期的工程活动；构建和移交组成它的产品。

在每个阶段内都有一些迭代。一个迭代代表一个完整的开发周期，从需求分析到实现和测试，结果是一个可执行项目的发布。每个迭代都要通过各种各样的工作流程，虽然每个工作流程有不同的侧重点，具体情况取决于不同阶段。在开始阶段，焦点是需求的获得；在详细描述阶段，重点是转向分析和设计；在构建阶段，实现是中心任务；移交阶段的重心则在于配置。

通常有九种工作流：

- 商业建模　描述组织的结构和动态性质。
- 需求　通过基于用例的方法来描述需求。
- 分析和设计　描述多种架构视图。
- 实现　软件开发、单元测试和集成。
- 测试　描述测试事例、步骤和错误跟踪方案。
- 配置　可移交的系统构型。
- 构型管理　控制项目组件的变化，并维持其一体性。
- 项目管理　描述采用迭代方法的不同工作策略。
- 环境　包括开发系统必要的各种基础设施。

在以上讨论的内容中，每个工作流程中最重要的是一系列应用 UML 建立的模型。为了对在软件系统开发中应用 UML 有一个直观而清晰的了解，我们考虑一个非常简单、理想化的软件开发周期，包括三个阶段：需求、分析和设计、实现。图 2 表示了 UML 的图和各种图所应用的阶段之间的关系[9]。

图 2　UML 图与系统开发阶段的关系

在下面的部分中，我们将通过一个超级市场销售管理系统来描述 UML 的建模过程。

3.1 功能需求

我们以一个超级市场管理系统的前台销售分系统为例，这个分系统被简化以最基本的方式处理销售业务。这个系统的功能需求如下：

- 为顾客选购的商品计价、收费、打印清单。
- 记录每种商品的编号、单价和现有数量。
- 帮助商家找出哪种商品将脱销，从而及时补充货源。
- 随时按上级系统的要求报告当前的款货数量、增减商品的种类或修改商品定价。
- 交接班时结算货款数目和商品数目。

由于系统较小，我们省略了构件图和配置图。

3.2 用例图

图 3 表示用例图，它包括一系列用例和从系统中抽象出的执行者。

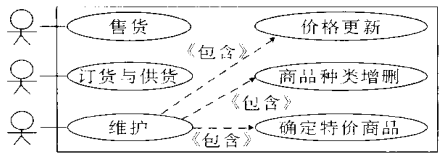

图 3　用例图

围绕系统的售货员、采购员、管理者是人在与系统交互中扮演的执行者。用例售货、订货和供货，以及维护模拟系统的功能需求。用例价格更新、确定特价商品、商品种类增删包含在用例维护里。用例和执行者之间的联系表示了执行者对用例的责任。例如，执行者售货员负责卖出商品，这是由用例售货所描述的功能。

3.3 类图

图 4 表示类图，它包括一组由所讨论的系统中抽象出的类和它们之间的联系。从图 4 左上面开始，你可以找到名为收款机、销售事件和账册的类。

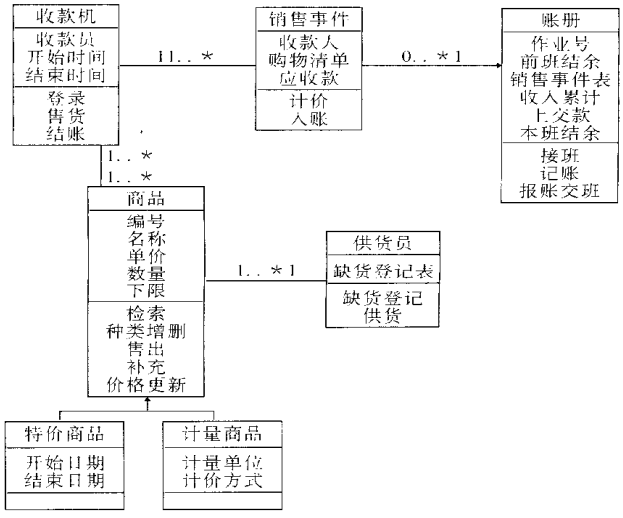

图 4　类图

类收款机的属性本班收款员表示哪一个收款员在收款机上工作；开始时间和结束时间规定了收款员什么时候开始和结束工作。类收款机有三个操作：①登录，接收收款员的登记，并且使他/她能开始工作；②售货，

计算金额并收钱;③结账,在收款员下班或交班时结算本班的账目。

类销售事件有三个属性:①收款人,记录新的销售事件是在哪个收款机上由哪个收款员处理;②购物清单,记录顾客所购商品的编号、名称、数量和价格;③应收款,表示发生在一次购买事件中累加起来的总钱数。操作计价逐条记录商品清单,并累计应收款数。入账将本次销售事件的信息加入账册。账册的属性销售事件表和收入累计记录所有的销售事件和一个收款员在一天内的销售总量。前班节余、上交款和本班节余描述一次交接班的信息。操作接班记录接收前一班收款员的货款;报账交班是向上级系统报账,记录上交的和留给下一班收款员的款数。

商品的属性包括编号、名称、单价、数量和下限。下限描述什么时候一种商品的架上数量小于这个值,就提醒供货员补充。商品有五种操作:(1)检索。当商品从收款机得到信息后查找将被售出的商品。(2)种类增删。修改商品的目录。(3)售出。从架上数量减去一种已售出商品的数量。(4)补充。加上被补充的商品的数量。(5)价格更新。改变商品的价格。

供货员的属性缺货登记表用来登记缺货商品。供货员有两个操作:缺货登记,输入缺货商品的编号和名称到缺货登记表;供货,供货员补充商品后向商品发送消息以更改数量,并删除缺货登记表中相应的条目。

类特价商品和计量商品是商品类的子类,表示两个子类继承了商品的所有属性和操作,并且又有自己特殊的属性和操作。

收款机和销售事件之间有一种联系,描述收款机处理销售事件。而且,每个收款机还可以处理任何数目的销售事件,但是一个销售事件只能由一个收款机处理。

商品和供货员之间也有联系,描述供货员为超级市场提供商品。而且一个供货员可提供一种或多种商品,但是一种商品只能由一个供货员来提供。

在账册和销售事件之间有一种组成关系,它表示账册由许多销售事件组成。

3.4　对象图

图 5 表示对象图,它包含一系列的对象和它们之间的联系。图 5 中只表示了每个对象的重要属性。为了简化,虽然类销售事件的很多对象可能和类收款机的一个对象相关,但我们只画出它们之间的一个一对二联系,没有给出对象图中其它联系的多重性。

3.5　顺序图

图 6 表示了一个顺序图,描述了处理一个销售事件的控制流。

流程由收款机对收款员的指令响应开始。首先,收款机向商品发送消息请求检索操作来查找将被出售的商品;然后发送消息请求售出操作。如果这种商品的数量少于下限,则向供货员发送消息进行缺货登记。收款机的售出操作将向销售事件发送信息请求销售计价和入账操作。最后,销售事件发送信息给账册中的记账,并且控制流程返回收款机等待下一次销售。

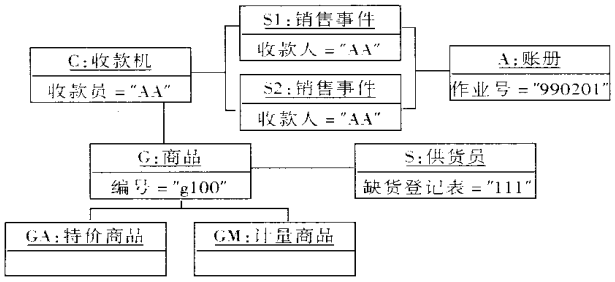

图 5　对象图

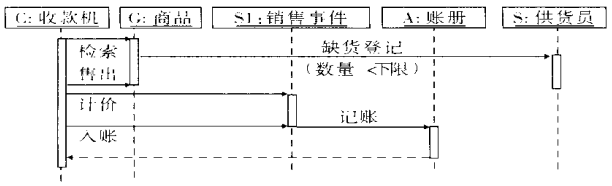

图 6　顺序图

3.6　合作图

众所周知,合作图在语义上等同于顺序图。图 7 表示合作图,它详细描述了一个销售时间过程中的控制流程,并强调这些对象之间的结构关系。

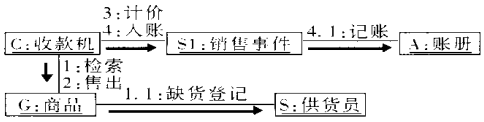

图 7　合作图

3.7　状态图

图 8 给出了对象收款机的状态图。限于篇幅,我们没有为其它对象绘制状态图。

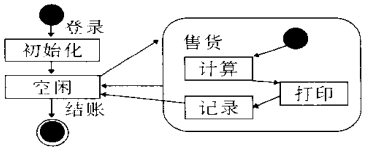

图 8　状态图

我们现在描述一下对象收款机的状态转移。在对象收款机的生命周期中有三个状态:初始化状态(收款机开始工作)、空闲状态(收款机准备接收收款员的命令)和售货(一个组合的状态,收款机处理一个销售事件)。当收款机对象开机接收收款员的登录后进入初始化状态进行初始化,然后处于空闲状态等待接收指令。接收收款员的指令后控制由空闲状态转到售货状态。进入售货状态时,控制先到计算状态(计算商品和钱的总数),然后到打印状态(打印顾客的发票),最后到记录状态(为对象销售事件记录销售的细节)。

3.8　活动图

图 9 表示一个销售事件的活动图,它详细描述了在收款员对收款机发出指令后的工作流程。活动开始于收款机接收指令然后流向商品(检索)。如果将被出售的商品数量小于下限,流程将流向供货员(缺货登记),然后回到收款机的执行售货来销售商品。销售过程包

括销售事件(计算然后入账)、账册(记账)和收款机(打印)。最后结束在收款机(等待)等待下一个销售事件。

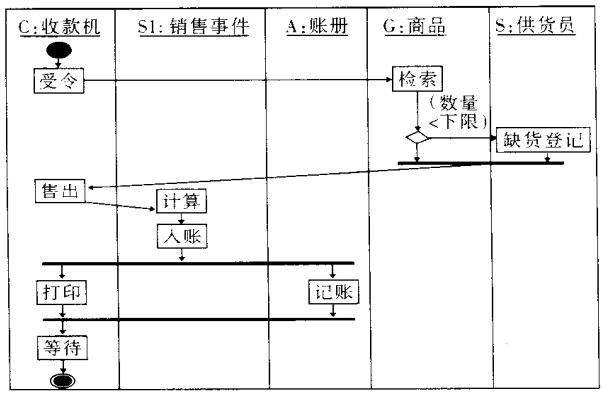

图 9 活动图

4 结论

UML 是一种图形语言,它以可视化的方法描述和构建一个以软件为中心的系统产品。我们用 UML 为一个系统建模。通常,绝大部分问题的 80% 可以只用 UML 的 20% 就能解决,因此,在本文中只介绍了最重要的部分(图)。九种图从不同角度和层次上为系统分析、设计和实现提供支持。

基本的结构元素,例如类、属性、操作、构件和包等,同诸如依赖、继承和联系等基本的结构关系一起,足以为任何问题域的应用创建静态模型。利用列出的基本动态建模元素,如简单的状态图和交互图,我们可以建模描述一个系统的多种动态特征。

参考文献:

[1] I Jacobson, M Christerson, P Johnsson, et al. Object-Oriented Software Engineering: A Use Case Driven Approach [M]. Prentice-Hall, 1992.

[2] J Rumbaugh, M Blaha, W Premerlani, et al. Object-Oriented Modeling and Design [M]. Englewood Cliffs (NJ): Prentice-Hall, 1991.

[3] Grady Booch. Object-Oriented Design with Application [M]. Benjamin-Cummings, 1991.

[4] Grady Booch, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language User Guide [M]. Reading, Massachusetts: Addison-Wesley, 1999.

[5] James Rumbaugh, Ivar Jacobson, Grady Booch. The Unified Modeling Langrage Reference Manual [M]. Reading, Massachusetts: Addison-Wesley, 1999.

[6] Grady Booch, Ivar Jacobson, James Rumbargh. UML Notation Guide (Version 1.1) [M]. Santa Clara: Rational Software Corporation, 1997.

[7] Grady Booch, Ivar Jacobson, James Rumbaugh. UML Semantics (Version 1.1) [M]. Santa Clara: Rational Software Corporation, 1997.

[8] Grady Booch, Ivar Jacobson, James Rumbaugh. UML Summary (Version 1.1) [M]. Santa Clara: Rational Software Corporation, 1997.

[9] Martin Gogolla. UML for the Impatient [R]. Research Report 3/98. University of Bremen, 1998.

[10] 邵维忠, 杨芙清. 面向对象的系统分析 [M]. 北京: 清华大学出版社, 1998.152.

作者简介:

王瑞金(1964-),男,副教授,博士生,研究方向为信息管理与信息系统;段会川(1967-),男,副教授,硕士生,研究方向为计算机软件;Martin Gogolla,男,博士,教授,德国不莱梅大学计算机系数据库研究组负责人,研究方向为面向对象的软件设计、数据库技术。