计算机科学 1998Vol. 25 №. 5

# 统一建模语言 UML

The Unified Modeling Language UML

李留英 韩 冰 曹 蕾 齐治昌

(国防科学技术大学计算机系 长沙 410073)

**摘 要** UML is the third generation object oriented modeling language developed by Rational Software Corporation. It consists of traditional-functional requirement analysis method and object oriented analysis method, and will become the standard of object-oriented modeling language in the future. This paper focuses on the basic visual graphic and extension in the next higher version.

**关键词** Unified modeling, Modeling language, Object oriented, Basic visual graphic

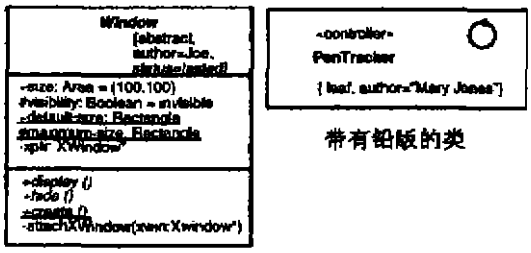
## 1 前言

统一建模语言 UML(Unified Modeling Language)是由 Rational 公司的知名专家 Gray Booch、Ivar Jacobson 和 Jim Rumbaugh 三人联合开发的第三代面向对象的建模语言。它采纳和扩展了 Booch 标记法、OMT 标记法和 OOSE 标记法，并包容了其它学者和软件厂商的建议，现已提交给 OMG，将成为标准化的面向对象建模语言。UML 是一种宽谱语言，适用于所有的应用领域，如：实时系统、client/server 等，并支持各种重要的 CASE 工具厂商。它融合了面向对象的数据驱动分析方法和行为驱动分析方法[1]，包容了软件工程的思想。其中类图和使用情况图分别描述了系统的逻辑结构和功能规范，顺序图和协作图描述各种使用情况(use case)的具体实现，部件图和配置图描述系统的物理组件和物理配置。UML 通过上述各种图元描述系统的对象模型以及其它对应信息。模型和图是两个不同的概念。模型包括系统的所有基本信息元素，而不考虑元素如何显示。图是对模型元素的可视化描述，一般只显示元素的部分信息。

## 2 静态图

UML 中的静态图主要是指类图，它描述系统的静态逻辑结构，即系统重要的抽象元素及元素间的关系，其中最基本的元素是类的图元和关系图元，分别用矩形和线段表示。

### 2.1 类 class 和铅版 stereotype




带有铅版的类

图 1 类和铅版

类是对具有相似结构、行为、关系的一组对象的描述，包括：类名、类的属性和操作。铅版是对建模元素的元级划分(meta-classification)，它通过对 UML 现有的建模元素的划分来增加 UML 语言本身的扩展能力。UML 用《》表示铅版，铅版关键字放在《》内。UML 根据现实世界的实体定义了一组铅版，并允许用户定义自己的铅版。UML 的类图元如图 1 所示，第一层是类名(黑体字)或其他性质(如 stereotype)，铅版名或铅版图标放在类名的上面或右上角；第二、三层是可选项，分别是类的属性和操作，抽象类或抽象操作采用斜体字；多个相似类组成一个包。如果引用其他包内的类，采用如下格式："包名::类名"。

李留英 博士生，感兴趣领域：分布计算、软件测试、软件重用。韩冰 硕士生，感兴趣领域：软件工程、CASE。曹蕾 硕士生，感兴趣领域：软件工程、办公自动化。齐治昌 教授、博士生导师，研究领域：分布计算、CASE、办公自动化、软件工程。

### 2.2 类型 type 和接口 interface

类型是对具有抽象状态、具体外部操作规格的对象的描述。类型没有描述对象的实现方法，只提供外部行为规格，而类提供了方法的过程实现。类型可能包括属性和操作，属性定义类型的抽象状态，也可以定义类型操作的效果。

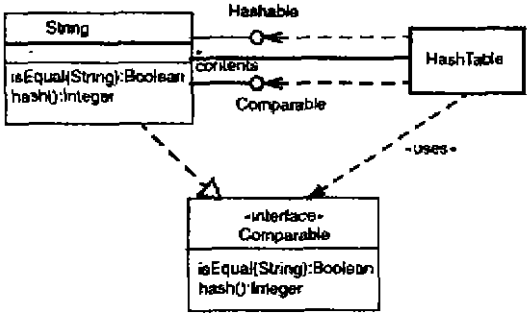



图 2 接口

接口是指使用类型(type)描述类、部件或其它实体的外部可见的行为。可以用一个带有类型名的小圆圈显示接口，也可以用实线将圆圈附加到类或高层的容器(如包含类的包)，说明该类支持接口内声明的所有操作，也可以用带《interface》的矩形图元列出所有的操作。图 2 中的 Comparable 为一接口，支持的操作包括：isEqual(String)和 hash()。如果一个类需要接口提供的操作，用虚箭头连接到该圆圈。接口为一种类型，也可以用矩形图元表示。

### 2.3 模板 template

模板是描述带有一个或多个未绑定的形式参数类。模板不是类，而是一组类的抽象，只有将参数绑定到具体的值才能产生类，所以又称为参数化类。其形式参数可以是属性类型、数据类型、甚至操作。模板内的属性和操作由形式参数定义。图 3 中带虚

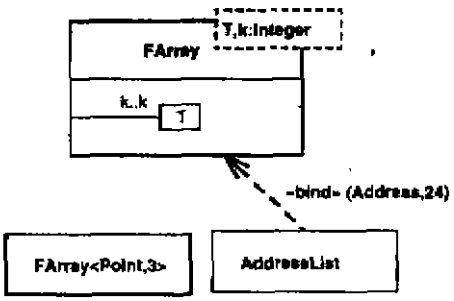



图 3 模板

框的矩形图元为模板，虚框为形式化参数表，是由逗号隔开的、采用 name:type 格式的参数表。name 是指模板内的参数标志符，type 是参数的类型表达式。

### 2.4 对象和对象图

对象是类的实例，它的图元表示和类的十分相似。上层显示对象名和类，格式为"对象名:类名"。并在类名和对象名下面加一下划线。如果省略对象名，则代表类内满足要求的任意对象。第二层描述对象的属性以及值的列表，格式为"属性名:类型=值"，类型可以省略，值采用数值方式。

对象图是一个实例图，分为静态对象图和动态对象图，静态对象图是类图的实例，它反映系统在某一时刻的详细状态信息，动态对象图显示系统在某段时间内的状态信息以及在该段时间内发生的变化，通常用协作图(Collaboration Diagram)来刻画。

### 2.5 关系 relationship

单个类提供的信息比较有限，各个类之间只有相互协同、才能完成复杂的功能。所以，UML 定义了几种关系，描述类之间的关系。

●关联 association 定义了类之间的一种语义联系，是对类实例间链接的抽象。关联用带修饰符的直线表示，用带菱形的直线表示多值关联。与类相连的关联的终点称为关联角色，每个关联可以有两个或多个角色。关联的多数信息，如关联的阶(multiplicity)、角色名(rolename)、限定符(qualifier)、遍历性(navigability)等均附加在关联角色上，阶说明参

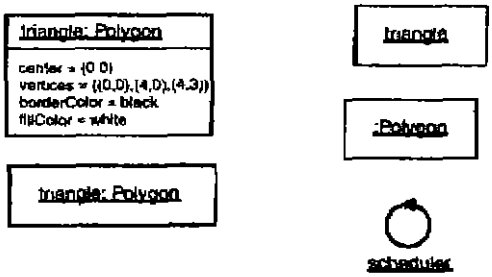



图 4 对象

与该关联实例的数量范围，如 0、1、0…1、0…*(零到多个)等；角色名指明类在该关联中扮演什么角色；限定符对参与关联的类的实例进行划分；遍历性指明关联是单向的还是双向的。如果没有特殊标记，一般是双向关联，反之亦然。

●聚集 aggregation 是一种特殊形式的关联，用来说明对象间的整体/部分关系。聚集关系分为引

用(by reference)和传值(by value),分别用空心菱形和实心菱形表示,图5的关联Contains说明Point的实例是Polygon实例的一部分,同时也可以属于其它的对象;而GraphicsBundle只能是一个Polygon的组成部分,而不能为其他对象所共享。

●继承 Inheritance　描述一个类共享其它类的结构和行为,如图6所示。

●依赖 Dependency　表明某个元素的变化对其他元素产生的影响。如果一个类为其它类提供服务,UML采用依赖关系描述客户类对服务提供者要求的服务。图7的ClassA是ClassB的友员。

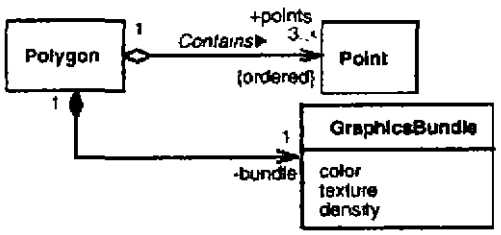



图5　关联

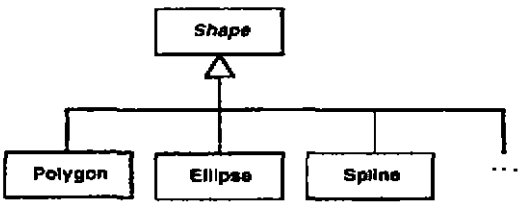



图6　继承

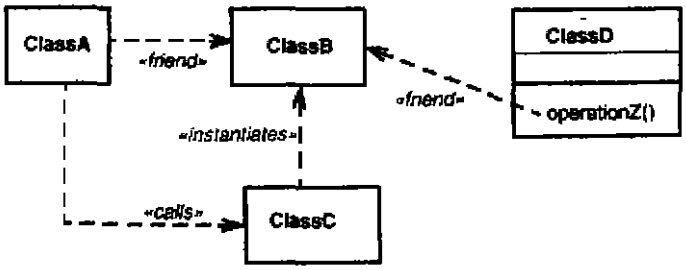



图7　依赖

## 3　动态图

传统的面向对象需求分析方法只给出了系统的对象模型,没有描述系统的功能需求,而系统的功能是用户最关心的问题,为此,UML采用使用情况(use case)图描述用户所关心的系统功能,用顺序图和协作图从不同侧面描述各种功能的具体实现。

### 3.1　use case 图

use case图由一组被系统边界包围的use case、系统边界外的actor、actor与use case间的通讯组成,主要刻画系统的功能需求和环境的约束。如图8所示,大矩形表示系统边界,排列在系统外部的Customer,Saleperson代表与系统交互的外部实体,这些实体均为各种类,它们具有actor的属性和功能。椭圆check status、place order等为use case。UML用2.5节提供的关系图元描述actor和use case、use case和use case之间的各种依赖。use case图存在如下关系:

● Communicate:actor通过一条路径和use case通讯。

● Extends:如果有从use case A指向use case B的箭头,表示A的实例扩展了B的行为。

● Uses:如果有从use case A指向use case B的箭头,表示A的实例包含B的行为。

use case实例化后产生的实例—场景(scenario)用来描述use case的动作序列的执行。可以用不同的场景刻画同一个use case实例。多个use case的多个场景建立系统的行为模型,为了刻画这些场景,UML使用交互图来对场景进行建模。

### 3.2　交互图

类图和use case图刻画了系统的结构特性和功能需求,却无法刻画实时需求,而交互图却能应用于时序需求。

交互图分为顺序图和协作图,它们的功能类似,但侧重点不同。为实现特殊目的而进行的特定消息交换称为交互。为了刻画交互,必须指定某一上下文和可能的交互顺序。由于顺序图不能显示协作的上下文,只能显示协作的行为,如:消息的时间顺序以及方法激活,协作图刻画了几乎整个上下文,以及与之有关的对象关系。

3.2.1 顺序图 Sequence diagram 顺序图以

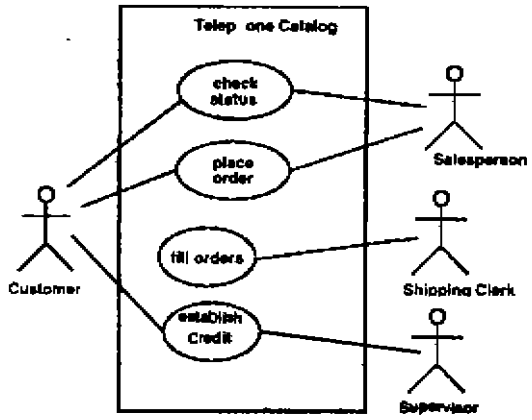



图 8 Use Case 和 Actor

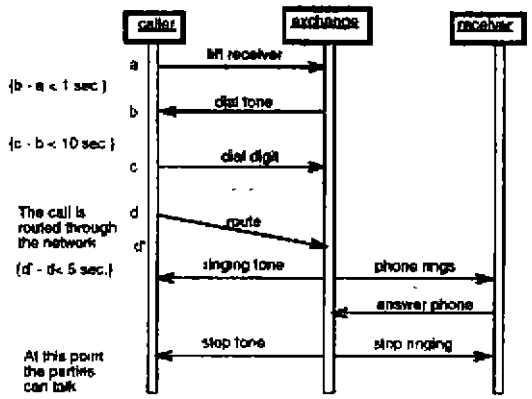



图 9 顺序图

时间顺序显示对象在其生命周期内的交互活动。它只显示参与的对象，而不刻画对象间的关系或对象的属性。如图 9 所示，顺序图采用两维坐标：垂直轴表示时间，水平轴表示不同的对象。水平有向线代表对象间交互的消息，并用消息名标记，或者用顺序号显示交互顺序。而且可以在交互附近附加各种补充信息，如：定时标记、动作等。图左边的脚本是对时间信息的补充说明。

垂直虚线表示对象的生存时间段。如果对象在该段时间内被创建或删除，对应的生命线就在该点结束，并用"X"标记对象被删除。生命线可以被分割为多个并发的生命线，每一段代表消息流的一个分支，这些线在某一点会合。激活描述在一定的时间段内对象的执行。在图内，用一个长瘦矩形代表激活，其两端分别与初始时间和完成时间相连。动作标记符号放在激活符号的旁边或左边。对过程控制流，激活符号的两端分别是输入消息和返回消息。在包含控制线程的并发进程，激活代表对象执行操作的持续时间。

3.2.2 协作图 Collaboration Diagram 主要描述对象间的关系，用顺序号决定消息的顺序和并发线程的顺序。协作图更强调内部结构。协作是描述类型间的交互活动的建模元素，由两部分组成，对象的静态结构的描述，如：对象的关系、属性、操作，又称为上下文 context；执行行为的描述——对象间交换消息的顺序；这两个方面构成了完整的行为规格。

在执行期间创建、删除或执行后删除的对象用《new》《destroyed》《transient》标记。协作图的交互的启动者可以是 actor，对象间的链接关系通常是关联的实例，而且铅版也可以附加在链接角色上，指明实现的方式。常用的铅版有：《association》关联、《parameter》过程参数、《local》过程内的局部变量、《global》全局变量和《self》自链接（对象可以给自己发送消息）。

一个协作可以附加在某个类型、操作或 use case 上，描述它们的效果，也可以附加到类、方法或 use case 的实现（通过《implements》细化关系），说明它们的内部实现。

消息流描述对象间传递的各种消息，通常采用过程调用、主动线程间的信号发送、事件的引发等方式实现消息。图 10 内的各种链接旁边附加的各种小箭头，指明消息传递的方向和消息交互方式。附加在小箭头上的消息标记符号，指明要发送的消息、消息参数和返回类型、大型交互中的消息顺序等。

## 4 状态机

UML 提供状态机来刻画系统动态特性。状态机是规定对象或交互作用在其生命期内对事件的响应所经历的状态序列，以及对象或交互作用的响应和动作。状态机包括状态顶点和状态转换，描述系统状态在外界事件作用下的变迁，状态顶点可以是单个状态，也可以是复合状态；转换是关系的一个子类型，指明从一个状态到另一个状态的通路。状态转换是由事件激发的，通过动作来完成，UML 采用如下方式描述转换：event(argument)[condition]^ target. sendevent(argument)/operation(argument)，其中 condition 表示事件发生所满足的条件；target. sendevent 是指引发的事件；operation 代表事件引发时的操作。状态机实例描述协作实例的所有潜在

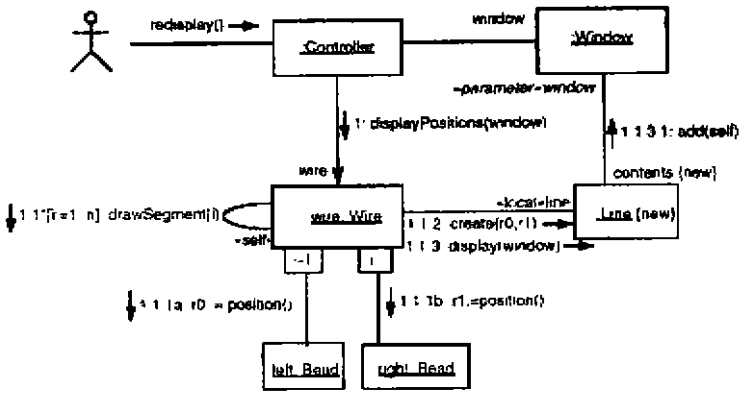



图 10　协作图

的行为。用 Booch 标记法指明同步方式，这些符号可以和消息一起表明在消息传递中如何实现并发处理的同步。

### 4.1　状态图

状态图刻画对象在其生存期内对接受的激励所经历的一系列状态。UML 的状态图以 David Harel 的状态图(StateChart)为基础，如图 11 所示。其中，圆角矩形代表状态，这三层分别为状态名，状态变量表和内部活动表，二、三层均是可选项，第三层的格式为"事件名 参数表/动作表达式"。如状态 Busy 内的事件 do 的动作为 play busy tone。箭头代表状态之间的关联，并在箭头上附加各种激活条件和激活时间。与一般的 Mealy-Moore 图相比，UML 的状态图的描述能力更强，它支持以下的描述：●转换卫士(Guard on transition)，状态转换必须满足的条件。●转换转播(Propagated transition)，一个事件激活某个状态转换的同时产生的事件。●转换动作(Action on transition)状态转换时，引发的行为。●进入状态的动作(Action on Entry state)进入某个状态时，引发的动作。●退出状态的动作(Action on Exit state)退出状态时，引发的动作。●嵌套状态。UML 支持复合状态，允许状态的分解。●并发。

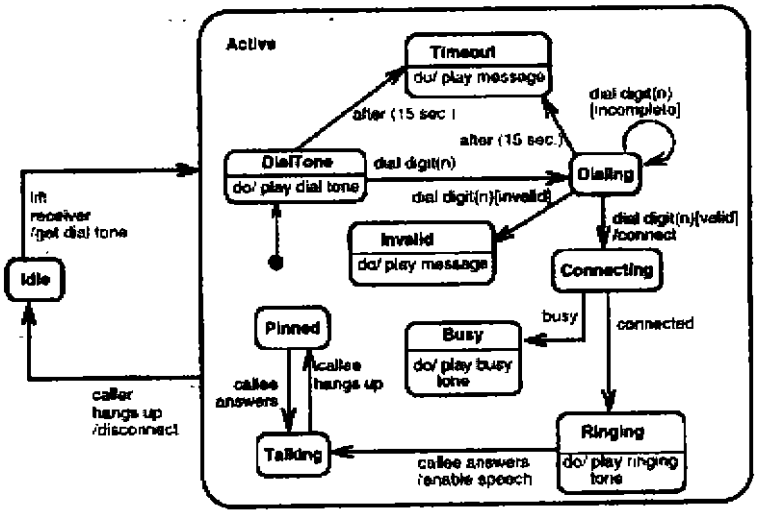



图 11　状态图

### 4.2　活动图

活动图是状态图的一种特殊形式。在状态图中，状态可以是简单状态、伪状态、复合状态、动作状态；而活动图中的状态均为动作状态。动作状态的转换只由操作的完成而引发，与外部事件无关。在 UML 中，系统交互是由一组实例协同所产生的行为。在建模时，交互和状态机可以相互翻译，但是，它们在描述行为时采用的方式不同，它们之间的主要区别是：

●状态机描述了它所反映的类型或协同实例的潜在行为;交互仅描述在某个环境下某个行为的一条简单路径。

●状态机实例通常描述单个类型的行为,而交互涉及多个类型的行为。

## 5 实现图

实现图包括组件图和配置图,展示系统的源代码的结构和运行时刻的实现结构。

### 5.1 组件图 Component Diagram

组件通常为代码块、二进制代码块和可执行部件等,是对建模元素的物理实现的抽象。组件图是指用依赖关系链接起来的组件集合,可以描述与特定语言相关的编译时刻的依赖关系。如图 12 内的 Scheduler、Planner 和 GUI 均为组件。组件包括接口,因此组件图中也可以描述组件之间的接口关系和调用关系。图 12 中的 update 是 Planner 的接口,从 GUI 指向 update 的箭头,说明 GUI 和 update 之间存在调用关系。

### 5.2 配置图 Deployment Diagram

配置图主要表现运行时刻各个处理元素的配置以及位于其上的软组件、进程(主动对象的特例)、对象。配置图是由通信链接起来的节点图,节点代表具有独立的内存和处理能力的计算资源。每个节点可以包含组件实例,也可以包含若干个对象。像虚线箭头描述组件间的通讯依赖关系,stereotype 是对通讯特性作补充说明,如:用《supports》标记在该节点上运行的组件,用《becomes》标注对象从一个节点到另一个节点的运动。

**总结** 本文主要论述了 Rational 公司 1997 年 1 月发表的 UML1.0 版本的基本组成结构。1997 年 11 月发表的 UML1.1 版本对 UML1.0 进行了扩充,将 UML 语言体系结构由原先的三层扩充为四层:meta-metamodel、metamodel、model 和 suer object,即增加了 meta-metamodel 层,meta-metamodel 为语言的体系结构奠定基础,而 metamodel 是 meta-metamodel 的实例,定义了描述 meta-metamodel 的语言。Model 是 metamodel 的实例,定义了描述信息领域的语言。UserObject 是 Model 的实例,用来描述具体领域。UML1.1 版本还支持不同层次的形式化描述。

UML 语言涵盖了面向对象软件系统从分析、设计到编码实现的全过程,融合了早期面向对象建模方法和各种建模语言的优点,为面向对象系统的开发、软件生产自动化工具与环境建造提供了一个丰富、严谨、扩充性强的表达方式。

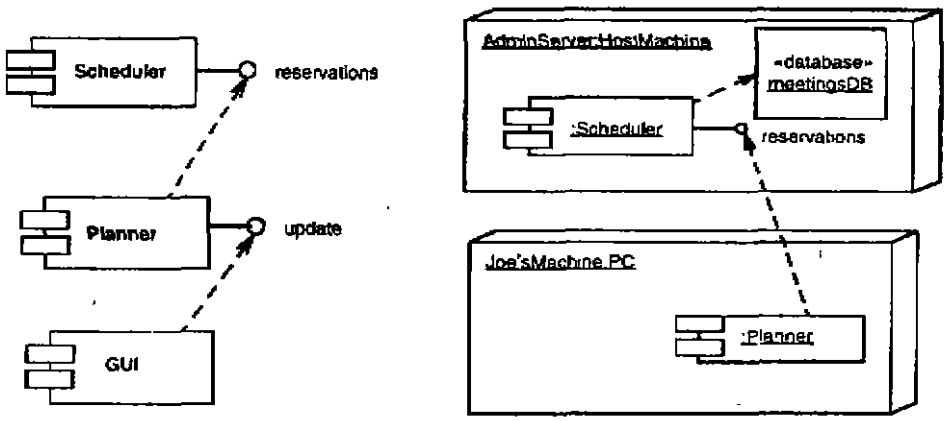



图 12 部件图和配置图

## 参 考 文 献

[1] Unified Modeling Language For Real-Time System Design,© Rational Software Corporation

[2] Unified Modeling Language Notation Guide Version 1.0,© Rational Software Corporation,Jan. 13,1997

[3] Unified Modeling Language Semantics Version 1.0, © Rational Software Corporation,Jan. 13,1997

[4] Comparison of POOM and OMT Object-oriented Analysis Methods ,Shuichiro Yamamoto,Proc. of the ISFSST-97,Oct. ,1997