面向对象角色分析方法——OOram*)

Object-Oriented Role Analysis and Modeling

明 仲　　李师贤

(深圳大学信息工程学院　深圳518060)　(中山大学计算机科学系　广州510275)

Abstract This paper gives an overview of OOram method and illustrates the concept and notation of OOram through a concrete case. OOram is a new method developed by the Center for industrial research (SI) in Oslo and the University of Oslo. Role Model is the basic abstraction used in the OOram technology. An object can play several roles. This permits a systematic separation of concern by describing different phenomena in different role models. Several role models can be merged into one model through synthesis technology.

Keywords Object-oriented method, OOram, Analysis, Synthesis, Role model, Base model, Derived model

一、引言

OOram(Object Oriented role analysis and modeling)方法是由挪威工业研究中心和奥斯陆大学共同提出的一种新一代面向对象软件工程方法。其基本观点是朴素的,即:同一对象在其整个生命周期内可以扮演不同的角色,而同一角色在不同时期可以由不同的对象来扮演。OOram 正是基于这样的观点,并把这一观点发展成为一种面向对象的软件开发方法。与其它 OO 方法比较,其最大的特点是,以"角色"的概念作为建立系统模型的基本概念,角色模型是系统开发过程中的唯一模型;而大多数 OO 方法是以"对象"和"类"作为建模的基本概念的。

二、OOram 方法概述

OOram 方法包括三个方面:技术方面,主要描述概念、表示法、工具。这是本文的重点。过程方面,描述开发步骤及每个步骤产生的结果。组织方面,描述软件开发的组织,即关于人的活动的描述。

OOram 把感兴趣的现象作为一个交互对象系统,建立角色模型,角色模型把关注以外的角色作为环境角色,这样分析人员在某一时刻可集中精力于系统的某一部分,即只要确定了关注域,就可对系统的当前被关注部分建立角色模型。这就是角色模型的"事务分离"(separation of concern)。为了充分揭示现象各个方面的特性,OOram 从三种不同的角度对一个角色模型进行研究,从而得出不同的视图(view)。①环境角度:把观察者放在系统环境中,能观察与环境角色交互作用的系统;②外部角度:当观察者被放在角色之间时,能观察角色之间的消息流并间接地推导出角色的属性;③内部角度:当观察者被定位在角色内部时,能观察角色的实现。

OOram 视图是一个 OOram 模型的不同表现。每种视图都表达了角色系统的某个方面而抑制了其它方面。详细地说,OOram 分析在一个角色模型中支持10种不同的视图:

①关注域视图:该视图是由角色模型对现象的文字描述。

②激发/响应视图:该视图表示环境角色通过发送激发消息,激发角色模型中的某种活动,并显示该活动的整体结果。

③角色表视图:该视图显示所有角色的一个列表以及这些角色的解释和属性。

④语义视图:该视图表明如何把语义加到角色上、以及语义与角色之间的关系如何。

⑤协作视图:该视图表明角色的模式以及角色之间的消息路径。

⑥界面视图:该视图定义沿着消息路径所能发送的所有消息。

⑦脚本视图:该视图显示角色之间的消息流的一个样板时间序列。

⑧过程视图:该视图显示数据如何在角色之间流动以及角色如何处理数据。

⑨状态图视图:每个角色有一个状态图,该视图描述一个角色的可能的状态、每个状态所能接受的消息、消息激发的操作及该操作完成后的状态。

*)国家自然科学基金资助项目(批准号79840761843)。收稿日期:2000-05-04

而方法规格说明视图：一对象收到消息后，触发执行对象所包含的方法，方法规格说明视图描述方法发送的消息和接收消息的角色。

不是每个角色模型都必须包含这10种视图，分析人员可根据需要选择其中的几种。下面以一个例子来说明 OOram 的基本概念和表示法。

Peter 要出公差到某地的过程如图1所示。

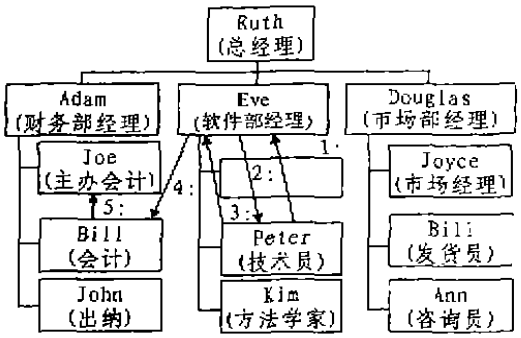

图1

1. Peter 把一个 TravelPermissionRequest 的消息发送到他的经理 Eve；

2. Eve 检查她的预算和计划，发 TravelPermission 消息给 Peter；

3. Peter 购买必要的车票并旅行后准备一个开支报告并发送一个 Expense Report 消息给 Eve；

4. Eve 检查开支报告，加上她的批准意见，发一个 AuthorizeExpenseReport 消息给会计(Bookkeeper)；

5. Bill 更新其帐目并发一个 RaymentRequest 消息给主办会计(Paymaster)Joe；

6. Joe 记录请求并记下将来某时给 Peter 付款的需求。

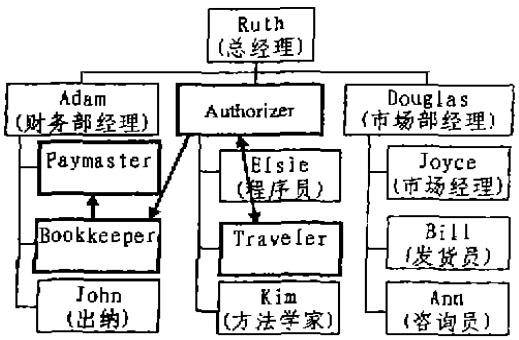

图2

图2粗线所示的是该例子所建立的模式(Pattern)。其协作视图如图3所示。图1中的 Eve 对象在模式中扮演 Authorizer 角色，Peter 扮演 traveler 角色，Bill 扮演 Bookkeeper 角色，Joe 扮演 Paymaster 角色，脚本视图如图4所示。方法视图如图5所示。

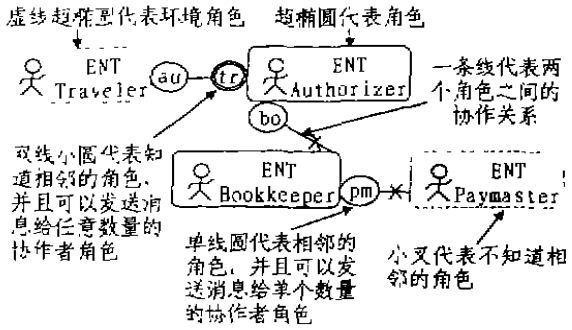

图3 协作视图

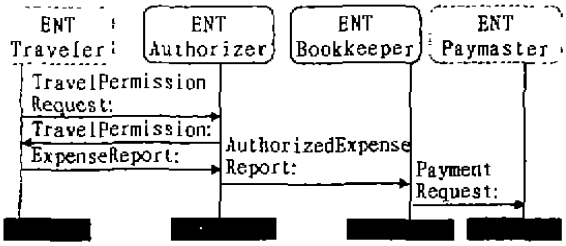

图4 脚本视图

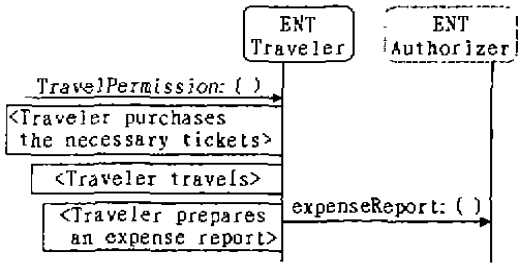

图5 旅行方法视图

一个系统经过分析，可得到多个角色模型，这些基模型可通过"综合"得到派生模型，综合时必须考虑模型之间的关系：①一般与特殊的关系。即一个模型描述一个普遍现象，其他模型描述该现象的特殊性；②聚集关系。即在某个抽象层次上的一个角色，在另一个详细的抽象层上被扩展为几个角色；③对象/主题关系。即在一个角色模型中的论域被指定为另一个角色模型的一个角色。

如果派生模型并不能产生新的信息，则"综合"操作可推迟到实现阶段完成。角色模型综合把继承的概念提升到模型的高度，派生模型继承基模型的特性。分析和综合是两种操作，使我们在研究复杂问题的时候，既可以进行概貌研究，又可以进行细节分析。我们可以

首先建立一个模型，然后将所有相关模型加进来得到整体模型。整体模型类似于数据库中的概念模式，单个模型类似于数据库中的外部模式。

角色模型综合分为安全综合和非安全综合。如果在派生模型中基模型中的静态和动态正确性都能保持，这种综合称为是安全的；如果派生模型中只保持基模型的静态正确性，其基模型的动态正确性必须在派生模型中重新研究才能确定，则这种综合称为非安全综合，无论哪种综合，派生模型都必须正确地反映基模型的语义。要保持静态正确性，必须遵守以下原则：

①基模型中的所有角色都映射到派生模型的相应角色上。

②基角色的属性仍保留为相应派生角色的属性。

③基模型中的所有端口都被映射到派生模型的相应端口，派生模型端口的势应与相应基模型端口的势一致，即派生端口的最小势应大于等于相应基端口的最小势，派生端口的最大势应小于等于相应基端口的最大势。

④基端口发出的消息界面应与相应派生端口发出的消息界面一致。

安全综合的本质是在派生模型中保持基模型活动的完整性。这要求基模型活动的激发消息在派生模型中仍为激发消息，或激发消息是发自派生模型的某个方法的消息。有两种方法实现安全综合：

①活动超位(Activity superposition)，即基模型的激发消息在派生模型中仍为激发消息，基模型活动在派生模型中保持不变，它独立于派生模型中的其它活动。

②活动聚集(Activity aggregation)，即基模型活动细节在派生模型方法中详细定义。这很象一个在方法中定义的一个封闭的子程序，方法发送一个激发消息给基模型，相应的基模型活动被激发且其完成过程不被干扰。基模型活动结束，方法继续运行。

聚集是一种重要的建模思想，可以用它来集成不同层次上的各种模型。当考虑外部系统时，聚集用单个角色来表示；当考虑部分系统时，聚集用一个角色模型来表示，外部系统的单个角色作为部分系统的环境角色。

OOram 支持三种不同类型的聚集：(1)封装(Encapsulation)，这是一种从外部聚集角色看不到被封装的部件的方式。(2)嵌入(embeding)，这是聚集的部件部分可见的方式。(3)虚拟，即用一个虚拟角色来代表部分角色结构。

下面考虑 TravelExpense 例子中的"订飞机票"问题。其"关注域"如图6所示；激发/响应视图如图7所示；脚本视图如图8所示；协作视图如图9所示。消息界面视图如图10所示。

Airline tickets are ordered by a booking clerk and paid directly to the travel agent. The traveler must show the cost of the tickets on the expense report as an expense. And as an advance, since he or she did not pay for them.

图6 关注域视图

Stimulus	Response	Comments
ABTraveler≫	ticketWithCost≫	Ticket cost retained in
OrderTicket	ABTraveler	attribute of ABTraveler role

图7 激发/响应视图

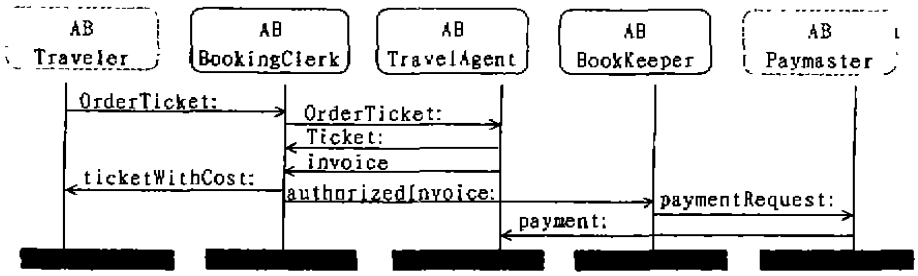

图8 脚本视图

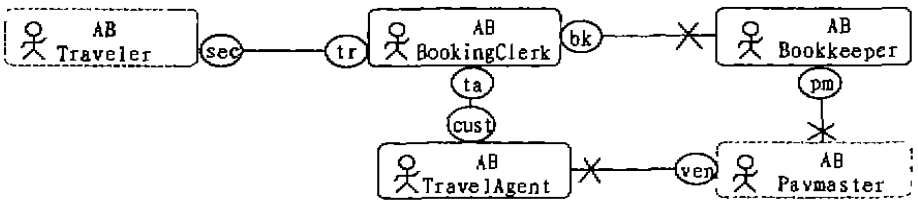

图9 协作视图

Interface 'ABBookingClerk<ABTraveler'
message 'orderTicket:' explanation "Purchase ticket(s)."
Param. 'ticketSpecification' type 'String'
Interface 'ABTraveler<ABBookingClerk'
message 'ticketWithCost:' explanation "Transmitting the ticket(s) together with cost information."
Param 'package' type 'String'
Interface 'ABTravelAgent<ABBookingClerk'
message 'orderTicket:' explanation "Reserve specified passages and issue ticket(s)."
Param 'ticketSpecification' type 'String'
Interface 'ABBookingClerk<ABTravelAgent'
Message 'ticket:' explanation "Transmittal of ticket(s)."
Param 'aTicket' type 'String'
Message 'invoice:' explanation "Transmittal of invoice."
Param 'anInvoice' type 'String'
Interface 'ABBookkeeper<ABBookingClerk'
Message 'authorizedInvoice:' explanation "Pay this authorized ticket invoice."
Param 'anInvoice' type 'String'
Interface 'ABPaymaster<ABBookkeeper'
Message 'paymentRequest:' explanation "Pay this invoice."
Param 'anInvoice' type 'String'
Interface 'ABTravelAgent<ABPaymaster'
Message 'payment:' explanation "Transmittal of payment."
Param 'aCheque' type 'String'

图10 消息界面视图

将 TravelExpense 模型与 AB 模型综合后得到派生模型 DTE，DTE 的关注域视图如图11所示；图12为 DTE 的环境协作视图，其中带阴影的超椭圆代表一个虚拟角色；图13为 DTE 协作视图综合；图14为 DTE 综合表，它反映了基模型角色和派生模型角色之间的映射关系；图15为 DTE 的脚本视图；图16为 Traveler 方法的第一部分，图17为第二部分。

The area of concern is the procedure for travel management, including the purchase of tickets.

图11 关注域视图

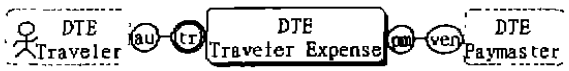

图12 环境协作视图

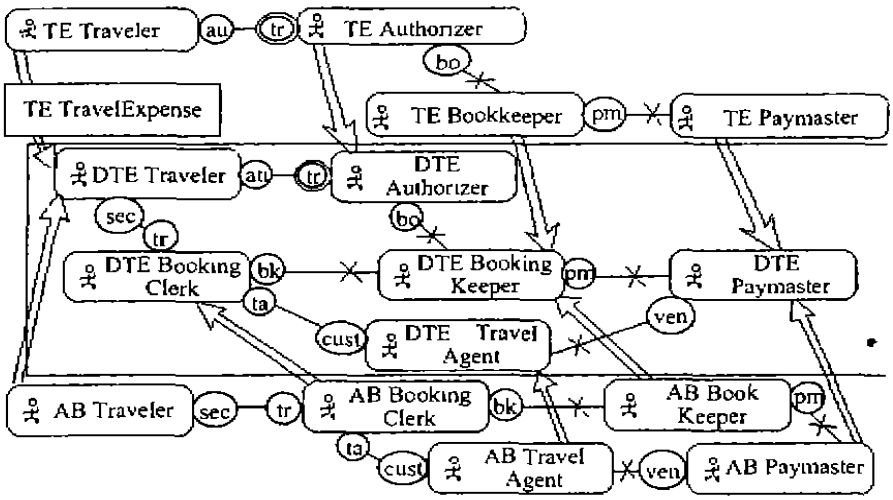

图13 协作视图综合

Derived model DTE	Base model TE	Base model AB
DTE-Traveler	TE-Traveler	AB-Traveler
DTE-Authorizer	TE-Authorizer	
DTE-Bookkeeper	TE-Bookkeeper	AB-Bookkeeper
DTE-BookingClerk		AB-BookingClerk
DTE-TravelAgent		AB-TravelAgent
DTE-Paymaster	TE-Paymaster	AB-Paymaster

图14 综合表

三、角色模型与其实现的关系

有两种情况综合被推迟到实现阶段完成。①派生模型中的角色数大于9。②派生模型没有产生新信息。因此，一个程序往往对应多个模型。

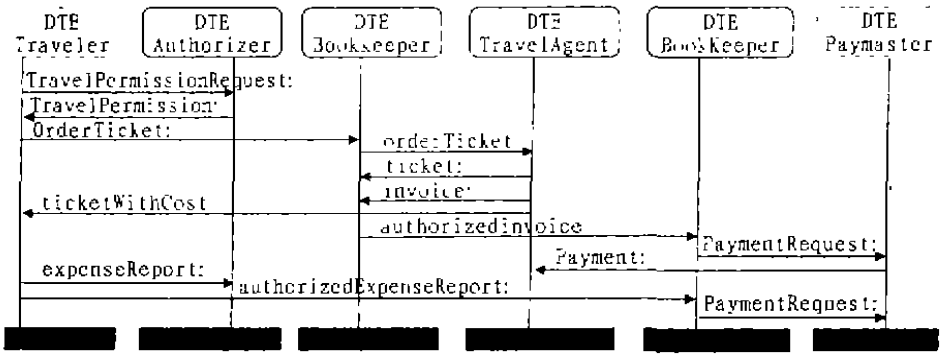

图15 脚本视图

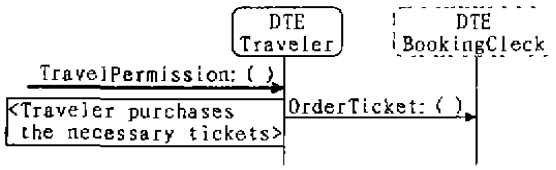

图16 方法视图(1)

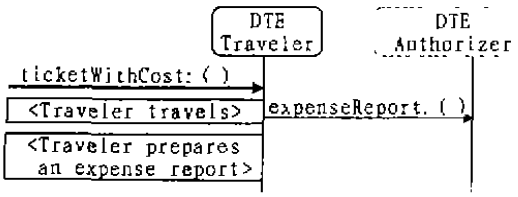

图17 方法视图(2)

角色模型描述了协作对象模式的静态特性和动态特性。而程序中的类则准确地规定了类的实例的静态属性和动态属性。两种描述都是定义对象，且它们之间有清楚明确的关系。很容易把角色模型中的信息变换到类中的相应的信息。

一个 OOram 角色模型可以提升为一个对象规格说明模型，在该模型中，角色被提升为对象规格说明，虚拟角色必须在提升前转化为具体的角色，因此一个虚拟角色代表一个对象簇而不是单个的对象。

表1 OOram 概念到程序语言的映射

OOram	Smalltalk	C++
Role Model	—	—
Role	Object	Object
Object specification, type	Class	Class
Port	Variable	Pointer data member
Interface	Protocol	Abstract data or protocol class
Message	Message	Function call
Method, action	Method	Member function
Derived Model	Subclass	Derived class
Base model	Superclass	Base class

表1显示的映射能帮助程序员理解 OOram 术语，但这种术语映射并不是等价。因为 OOram 方法的焦点是角色，而编程语言的焦点是类。

对象规格说明描述一个将被实现的对象，当我们把角色提升为一个对象规格说明的时候，角色将变得更具体，对应于一个对象规格说明的编程语言的概念是类，在对象规格说明和类之间默认有一对一的关系，但是一般而言，它们之间有多到多的关系。

需要强调的是 OOram 方法并不要求一定要建立完整的派生模型和对象规格说明。相反，OOram 方法只构造理解系统所必需的模型，而模型与源代码一起构成系统的文档。大多数定义良好的模型都是充分独立的，使得正式的综合操作变得多余了，程序员直接把一些基模型实现为类，以便类的实例扮演所有所需的角色。

结束语 用 OOram 方法分析系统，只需建立一种模型，即角色模型，用各种方法对角色模型的各个侧面进行研究，可得到角色模型的多种视图，这些视图深刻地揭示了系统的静态特性和动态特性。

角色模型的继承不仅体现了类之间的继承关系，而且解释了派生模型中的角色是如何描述整体现象的，OOram 的继承是在角色模型的上下文中进行的，派生模型继承基模型的特性。

角色模型聚集有很重要的实践意义。通过"事务分离"，复杂的现象可被描述为多个简单的角色模型。普遍的现象可被描述为普遍的角色模型，可在构造应用系统的模块时，重用该模型，如果这些普遍的角色模型被实现为一些相关的类，那么在实现应用程序时，可以以一种安全可控的方式重用这些框架。

OOram 角色模型提供了系统重用技术，可以以一种崭新方式来组织应用系统。可扩展的和系统的重用使我们能从小的项目中生成大的应用程序。

一个基于层次结构的DSM模型*)

A Hierarchical DSM Model

李 翼 郭建新 陈贵海 谢 立

(软件新技术国家重点实验室 南京大学计算机系 南京210093)

Abstract We present a hierarchical DSM scheme named GDSM. In our solution, sub-tasks are to be mapped to groups, which apply different techniques based on the distinct characteristics of inter-group and intra-group data sharing. A serial of alternatives are used with respect to the intra-group and inter-group property of this mechanism: (1)Consistency model: release consistency vs scope consistency; (2) Coherence protocol: multiple-writer and write-update protocol vs. single-writer and write-invalidate protocol; (3)Granularity: fine-grain vs. coarse-grain. Our novel strategy to combine grouping with consistency, data conversion and granularity switch cuts down the overhead of consistency maintenance, increases the parallelism between groups and thus promotes system performance. We also put forward the architecture of Protocol Engine implementing the GDSM scheme, the engine that achieves message-forwarding transparency, clear hierarchical structure and encapsulation of group.

Keywords DSM, Group, Consistency model, Granularity, Protocol Engine

一、引言

分布式共享内存(Distributed Shared Memory, DSM)是并行处理中的一种关键技术。它为程序员提供了一个逻辑上统一的虚拟地址空间,任何一个处理机都可以对这一地址空间直接进行读写访问。其中一致性模型、一致性协议、粒度是影响DSM性能的重要因素。

DSM中物理内存分布在多个节点上,数据一般采用复制的方式来实现共享内存的抽象,因此如何在保证内存一致性前提下尽量提高效率成为DSM的一个重要研究内容。一致性模型[5]本质上是内存系统与并行程序之间关于内存访问的协议。严格一致性模型包括原子一致性、顺序一致性等,严格的一致性模型不利于开发程序的并行性,因此出现了很多松懈一致性模型,包括弱一致性、释放一致性、域一致性(Scope Consistency)和单项一致性(Entry Consistency)等。

一致性模型可以用不同程度的惰性协议来实现一致性维护操作和数据修改的传播,这样就需要区分单写和多写协议、写更新和写无效。单写协议允许多个读者同时存取某个页面,但只有一个写者拥有修改权,单写协议易于实现,但通信代价较大,多写协议同时允许多个进程对共享数据拥有写权限。这种方式减少了假共享和对全局带宽的要求,但需要额外的处理代价。写更新和写无效的区别在于系统出现不一致时的处理方式。写更新要求在某处被修改的数据拷贝立即在其他节点得到更新,而写无效仅仅使其他节点的数据拷贝无效,因此,在写更新协议中,如果该节点在数据修改前拥有该数据,则修改后该数据依然有效。在写无效中,远处的节点必须从修改节点处获得更新过的数据。

共享数据的粒度也是一个重要的问题。共享粒度指一致性维护的单位,即系统在数据存取故障时复制的数据大小。它不仅包括一次读数据要传输多少数据,还包括一次写数据会影响多少数据的有效性。粒度是

*)本文得到国家自然科学基金(NSF#69803005)的资助。

参考文献

1 Reenskaug T. Working with Objects (The OOram Software Engineering). Manning Greenwich, 1996
2 邵维忠,杨芙清. 面向对象的系统分析. 清华大学出版社,广西科学技术出版社,1998年12月
3 姜鸿飞,全炳哲,金淳兆. 面向对象开发方法的最新进展. 计算机科学,1998,25(2)
4 周之英,王淳. IDEF4与面向对象设计方法. 计算机科学,1995,22(2)
5 李师贤,李文军,周晓聪. 面向对象程序设计基础. 高等教育出版社,1998
6 袁晓东,陈家骏,郑国梁. 基于角色分类的子类型关系. 计算机研究与发展,1997(11)