一种基于角色访问控制模型的约束规范描述*)

许春根[1,2] 严 悍[2] 刘凤玉[2]

(南京理工大学应用数学系[1] 南京理工大学计算机系[2] 南京210094)

Constraint Specification for the Model of Role-Based Access Control

XU Chun-Gen YAN Han LIU Feng-Yu

(Dept. Applied Mathematics[1], Dept. of Computer[2], Nanjing University of Science and Technology, Nanjing 210094)

Abstract The access control is very important for the information management of modern enterprises. This paper proposes a framework of well-defined constraint specifications for developers to build application-level access control based on users' roles. They ensure that each role is configured with consistent privileges, each actor is authorized to proper roles and then each actor can activate and play his authorized roles without interest conflicts. These formal constraint specifications are described by the first-order predicate logic, and also some properties are proved. The model provides relatively independent, consistent and inferable constraint specifications for developers to design access control of large and complex enterprise systems.

Keywords Class, Access control, Model, Role, Security

传统的访问控制主要有自主型的访问控制 DAC(Discretionary Access Control)和强制型的访问控制 MAC(Mandatory Access Control)。强制型访问控制是"强加"给访问主体的,即系统强制主体服从访问控制政策。自主型的访问控制是在确认主体身份以及它们所属的组的基础上,对访问进行限定的一种方法。随着企业规模的增大,企业的信息化管理变得越来越重要,企业级访问控制和安全管理设计将是最难解决的问题之一,DAC 和 MAC 已不能满足需要。20世纪90年代以来出现的一种基于角色的访问控制 RBAC(Role-Based Access Control)[1]技术有效地克服了传统访问控制技术中存在的不足之处,可以减少授权管理的复杂性,降低管理开销,而且能为管理员提供一个比较好的实现安全政策的环境。近年来,RBAC 技术越来越受到人们的重视,美国计算机学会(ACM)从1995年开始举办第一届 RBAC 国际论坛(ACM Workshop on Role Based Access Control)以来,到2002年已举办了第七届(更名为 ACM Symposium on Access control models and technologies)。

随着以网络为中心的计算技术的发展,多用户分布式应用系统日益庞大且复杂化,使得有效的访问控制的设计更加难以实现。利用角色技术建立的基于角色的访问控制(RBAC)模型将是一个很好的解决办法。人们已经建立多种形式的 RBAC 模型[1,7,9],也给出了一些形式化的描述[8],并在部分系统中进行了实现,如操作系统 Unix 与 Windows NT、构件系统 EJB(Enterprise JavaBeans)、数据库系统 Oracle7/8等,但这些模型是相对简单的和不太完善的。我们以基于角色访问控制原理为基础,提出一种新的对象式模型及规范化约束,以支持企业级访问控制的设计。该模型的特点是对象式和规范化,适用于种类多样的大型多用户交互式应用和分布式信息处理系统。本文着重介绍其工作原理(建模元素、工作流程)和约束规范(一般关联规范、角色关联规范、服务接口规范、角色权限规范、作用者授权规范和激活角色规范),并对规范进行形式化描述。

1 模型概述

近年来基于角色访问控制 RBAC (Role based Access Control)得到重视。文[1]提出一个 RBAC 概念模型,基本原理是:以角色表示一个企业、机构或团体中的职责或职位;把操作对象的各种权力与不同角色联系起来表示各角色的权限;为用户授予合适角色;运行时用户激活角色而得到相应权限。系统中角色及其权限虽复杂但相对稳定,而人员升迁任免则频繁发生,故此 RBAC 的最大好处是避免为每个用户直接配置复杂的权限,在一定程度上简化了权限管理。为使系统设计员在开发大型复杂系统时能有效处理以上问题,我们以基于角色访问控制原理为基础,提出一种新的对象式模型及规范化约束。

图1是用 UML(Unified Modeling Language)[4]表示的该模型的类图,图中给出了5个类和1个接口及其之间的关系,图2给出该模型的规范之间关系。

为表示对象式系统中操作对象的"权力",模型中设计一个 Service 服务接口类型,包含一组操作和子接口,由特定业务对象类或构件实现。采用接口的好处在于:①接口表示系统内部对象向外部提供的一组服务,外部规范如何访问接口无需考虑其内部实现细节,从而简化访问控制设计;②接口作为一种程序设计类型,已由 CORBA 和 Java 提供,故便于实现;③接口具有层次结构和模块化,使得权限管理比平面结构更易实施。注意,接口类型不可实例化对象,Service 中元素是服务项目,并非实例化对象。相关规范见2.3节。

PA (Privilege_Authorize)类是角色(Role)和服务(Service)之间的关联类,其每个实例是一个二元组(r,s),记为 pa(r,s),表示角色 r 在特定语境条件下有权访问服务 s。该语境

*)国防科工委基金(项目编号:J1300B004)。许春根 博士研究生,讲师,主要研究方向:软件工程、对象技术及网络安全。严 悍 博士,讲师。刘凤玉 教授,博导。

条件表示为 pa(r,s)对象的一个属性:context_cond。相关规范见2.4节。

语境条件(context_cond)是在运行时控制权力有效性的、依赖于语境 context 的条件。语境通常包括:可操作对象、日期、时间、网络、终端、协作者等。语境条件支持动态的或有条件的权限规范。例如,会计可查看(lookup)自己制作的凭据 Voucher,而财务经理可查看所有凭据。而 Voucher 类对 lookup 往往仅提供一种实现方法。我们可设置语境条件来约束一个操作可作用的对象范围。语境条件提供一种动态的灵活的控制机制。

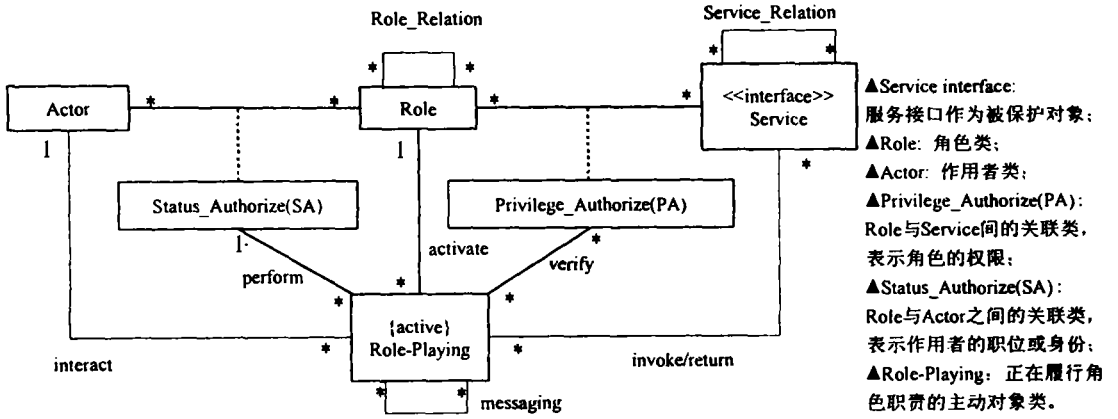

图1 基于角色访问控制模型的类图

(每个关联设置其多重性:* 表示0或多个)

通常,某操作的授权隐含其他操作的授权,以保持权限一致性,我们把这种关系称为操作隐含。譬如,对某操作的授权应自动隐含"更弱"的操作。例如,有权打印(print)某发票自动隐含有权查看(lookup)该发票;反之,若无权查看某发票隐含着无权打印该发票。对某类对象的一种操作可能隐含对其它类对象的一种操作。有权签署 sign 某发票隐含有权查看与该发票相关的帐目和凭据。隐含关系表示为 $s_1 \xrightarrow{cf} s_2$,其中服务 s_1 的授权隐含 s_2,cf 表示由 s_1 的语境条件转换到 s_2 的转换函数。规范见2.4节。

Actor 作用者是在系统外部与系统交互的实体,可以是用户、子系统或自主代理[4]。Actor 类是一个系统中所有作用者的抽象。

Role 角色是在特定语境中的实体被命名的特殊行为[4]。该模型中,角色特指作用者的角色。Role 类的一个实例是应用系统中的一个角色对象。每个角色都有特定名称和作为属性的基数约束。其中,角色的 authorized_cardinality 限定可授予该角色的作用者的数目。例如,会计角色可能有30名职员承担,而财务经理可能只有一两名。角色的 activated_cardinality 限定该角色被激活的数目,如财务系统管理员角色可由几个人承担,但在运行时,可能仅允许一人激活该角色履行管理员的职责。基数约束提供定量的控制。

SA(Status_Authorize)类是角色(Role)和作用者(Actor)之间的一个关联类,其一个实例是一个二元组(a,r),记为 $sa(a,r)$,表示作用者 a 被授权为角色 r。

角色继承(RI)类的一个实例是一个二元组(超角色 r_1,子角色 r_2),记为 $ri(r_1,r_2)$,表示在子角色和超角色之间的 isa 关系:①一个子角色是一个特殊的超角色;②一个作用者被授予子角色,将自动授予其超角色。或者说,某作用者被授予子角色的必要条件是他已授予其超角色;③子角色将自动继承其超角色的权限,除非改写,并可增加新权限。例如,公司职员(超角色)和公司会计员(子角色)之间就有角色继承关系。角色继承表示角色间的内聚性,描述了一个机构中各部门人力资源的层次结构。

职责分离(SD)类的一个实例是一个二元组$(role_1,role_2)$,记为 $sd(role_1,role_2)$,表示两角色间的利益冲突关系。有二种类型的职责分离(SD):静态职责分离(SSD)和动态职责分离(DSD)。前者规定了一个作用者不能同时授权为两种角色;后者规定了一个作用者可授权为两种角色,但在运行时不能同时激活这两种角色。例如,会计和审计员可被设计为静态职责分离(SSD)关系,而财务经理和财务系统管理员之间为动态职责分离(DSD)关系。职责分离(SD)规范确保每个作用者可安全地履行自己的职责而避免利益冲突。

对于两个角色,角色继承和职责分离是互斥的,而且静态职责分离(SSD)和动态职责分离(DSD)也是互斥的。对于三个角色,职责分离可从超角色衍生到子角色。相关规范见2.2,2.4,2.5节。

在运行时,作用者激活被授予的角色,以履行其职责。模型中提出 Role-Playing 表示被激活的角色,并建模为一个主动类 RP。其一个实例是一个主动对象,表示一个作用者 a 在特定语境中激活角色 r 的一次执行,记为 $rp(a,r)$,其语境记为 rp(a,r).context_cond。一个角色 r 被激活当且仅当至少有一个 $rp(a,r)$ 存在。作用者与 RP 对象交互,以履行角色职责,并实现协同工作和动态控制,规范见2.6节。

总之,该模型涉及到的关键技术有服务接口(表示对象式系统中被保护的权力)、语境条件(动态的有条件的权限规范)、操作隐含(保持权限一致性)、基数约束(实现定量控制)、角色继承(角色间的内聚性)、职责分离(角色间的利益冲突)及动态控制(在运行过程中监控作用者的状态和行为)。

2 约束规范的描述

约束规范提供一组规范,以约束模型中各元素之间的关系。其中包含6大类、30项:一般关联规范、角色关联规范、服务接口规范、角色权限规范、作用者授权规范和激活角色规范。我们为了保证各规范之间的独立性,很多能推理的规范就作为性质出现,并给予证明。这些规范之间也无矛盾性,具有一致性的特点。在关联规范和服务接口规范中我们采用 UML 的关联和接口规范。规范采用一阶谓词逻辑表示,符号"$\neg$,$\wedge$,$\vee$"分别表示"否定,合取,析取";"$\forall$,$\exists$"分别表示全称量

词、存在量词;"→,↔"分别表示"蕴涵,双向蕴涵"。对于 $a\in A$ 的语义,有以下约定:①A 作为一个集合,a 是其中一个元素;②若 A 作为一个类,a 是其中一个对象;③若 A 是 Service 接口,a 是其中一项服务操作。有些符号我们也采用了简记,例如,$\forall r(r\in Role)$ 简记为 $\forall r\in Role$,$\exists r_1\exists r_2(r_1\in Role\wedge r_2\in Role)$,简记为 $\exists r_1,r_2\in Role$ 等。

2.1 一般关联规范

规范1 两个对象之间属于同一个关联类中的联接至多一个:

$$\forall C\forall t\forall t'(C\in AssoC(A,B)\wedge t(a,b)\in C\wedge t'(a',b')\in C\wedge(a=a')\wedge(b=b')\rightarrow t=t')$$

其中 $AssoC(A,B)$ 表示类 A 和类 B 之间所有关联类的集合,$t(a,b)$ 表示对象 $a\in A$ 和对象 $b\in B$ 之间的联接。

规范2 若某关联类存在一个联接,那么该联接所涉及的对象必须在被关联的类中存在:

$$\forall C\forall t(C\in AssoC(A,B)\wedge t\in C)\rightarrow\exists a\in A\wedge\exists b\in B\wedge(t=(a,b))$$

规范3 若类 A 和类 B 关联,且该关联的 A 端多重性的最小值为1,那么对于类 B 中的任一对象,类 A 中一定存在一个对象与其相联接:

$$\forall C\forall b(C\in AssoC(A,B)\wedge(\min(C\rightarrow A.multip)=1)\wedge b\in B\rightarrow\exists a\in A\wedge link(a,b)\in C)$$

其中 $C\rightarrow A.multip$ 表示关联 C 的 A 端多重性,$Link(a,b)$ 表示对象 a 与对象 b 之间的一个联接。

规范1和规范2适用于关联类 PA,RI,SD 和 SA。规范1确保关联类对象的一致性,规范2和规范3表示关联对象存在的必要条件。图1中,当一个 Role-Playing 对象 $rp(a,r)$ 存在时,由规范3,一个 SA 对象 $sa(a,r)$ 必须存在;再由规范2,Actor 类中定存在对象 a,且 $Role$ 类中定存在 r。规范2和规范3确保在关联类对象被撤销之前,被关联的对象不可能被撤销,从而确保关联关系的一致性,并简化约束规范。

2.2 角色关联规范

规范4 角色继承(RI)关系 $ri(super\text{-}role,\ sub\text{-}role)$ 是强偏序关系,即非自反的、反对称的和传递性:

$$\forall r\in Role\rightarrow\neg(ri(r,r)\in RI)$$

$$\forall r_1,r_2\in Role(\exists ri_1(r_1,r_2)\in RI\rightarrow\neg(\exists ri_2(r_2,r_1)\in RI))$$

$$\forall r_1,r_2,r_3\in Role(\exists ri_1(r_2,r_2)\in RI\wedge\exists ri_2(r_2,r_3)\in RI\rightarrow\exists ri_3(r_1,r_3)\in RI)$$

规范5 职责分离(SD)关系 $sd(r_1,r_2)$ 是非自反的、对称的和不传递的:

$$\forall r\in Role\rightarrow\neg(sd(r,r)\in SD)\quad\forall r_1,r_2\in Role(\exists sd_1(r_1,r_2)\in RI\rightarrow\exists sd_2(r_2,r_1)\in SD)$$

$$\forall r_1,r_2,r_3\in Role(\exists sd_1(r_1,r_2)\in SD\wedge\exists sd_2(r_2,r_3)\in SD\rightarrow\exists sd_3(r_1,r_3)\in SD\vee\neg(\exists sd_3(r_1,r_3)\in SD))$$

规范6 对于任两个角色,职责分离(SD)和角色继承(RI)是互斥的:

$$\forall r_1,r_2\in Role(\exists sd(r_1,r_2)\in SD\rightarrow\neg(\exists ri_1(r_1,r_2)\in RI)\wedge\neg(\exists ri_2(r_2,r_1)\in RI))$$

$$\forall r_1,r_2\in Role(\exists ri(r_1,r_2)\in RI\rightarrow\neg(\exists sd_1(r_1,r_2)\in SD)\wedge\neg(\exists sd_2(r_2,r_1)\in SD))$$

规范7 若超角色与某角色有 SD 关系,那么子角色与其也有 SD 关系:

$$\forall r,r_1,r_2\in Role(\exists ri(r_1,r)\in RI\wedge\exists sd(r_1,r_2)\in SD\rightarrow\exists sd'(r,r_2)\in SD)$$

规范8 职责分离(SD)有静态职责分离(SSD)和动态职责分离(DSD)两种子类型:

$$\forall r_1,r_2\in Role(\exists sd(r_1,r_2)\in SD\leftrightarrow\exists ssd(r_1,r_2)\in SSD\vee\exists dsd(r_1,r_2)\in DSD)$$

规范9 对于任两个角色,静态职责分离(SSD)与动态职责分离(DSD)是互斥的:

$$\forall r_1,r_2\in Role(\exists ssd(r_1,r_2)\in SSD\rightarrow\neg(\exists dsd(r_1,r_2)\in DSD))$$

$$\forall r_1,r_2\in Role(\exists dsd(r_1,r_2)\in DSD\rightarrow\neg(\exists ssd(r_1,r_2)\in SSD))$$

性质1 没有角色能继承两个具有 SD 关系的角色:

$$\forall r,r_1,r_2\in Role(\exists ri_1(r_1,r),ri_2(r_2,r)\in RI\rightarrow\neg(\exists sd(r_1,r_2)\in SD))$$

证明:假设 $sd(r_1,r_2)$ 存在,由规范7,$sd(r,r_2)$ 存在;再由规范5,$sd(r_2,r)$ 存在;而此时由已知得 $ri(r_2,r)$ 存在,那么 $sd(r_2,r)$ 和 $ri(r_2,r)$ 同时存在。这与规范6相矛盾,故假设不成立。

2.3 服务接口规范

规范10 服务 Service 接口中的一个元素是一个操作或一个接口:

$$\forall s(s\in Service\leftrightarrow s\in Operation\vee s\in Interface)$$

规范11 每个操作都在特定接口中说明:

$$\forall op\in Operation\rightarrow\exists i\in Interface\wedge op\lhd i$$

其中 $op\lhd i$ 表示操作方法 op 在接口 i 中说明。

规范12 接口之间的一般化 Genaralization 关系具有强偏序关系,即非自反的、反对称的和传递性:

$$\forall i\in interface\rightarrow\neg(\exists g(i,i)\in Generalization)$$

$$\forall i_1,i_2\in Interface(\exists g_1(i_1,i_2)\in Generalization\rightarrow\neg(\exists g_2(i_2,i_1)\in Generalization))$$

$$\forall i,i_1,i_2\in Role(\exists g_1(i,i_1)\in Generalization\wedge\exists g_2(i_1,i_2)\in Generalization\rightarrow\exists g_3(i,i_2)\in Generalization)$$

其中,$g(super\text{-}interface,sub\text{-}interface)$ 表示接口间的一个一般化 Generalization 关系。

2.4 角色权限规范

规范13 一个 PA 对象存在即表示在特定语境条件下一个角色能有权访问某项服务:

$$\forall r\forall s(r\in Role\wedge s\in Service\wedge\exists pa(r,s)\in PA\leftrightarrow(CC(pa(r,s).context\_cond)\rightarrow(r\xrightarrow{cc}s)))$$

其中,$CC(pa(r,s).context\_cond)$ 表示当前语境满足条件;$pa(r,s).context\_cond.r\xrightarrow{cc}s$ 表示角色 r 在语境条件 cc 下有权访问服务 s。

规范14 若某角色有权访问某个操作,那么该角色也有权访问该操作所在的接口:

$$\forall r\in Role,\forall s\in Operation,\forall i\in Interface(\exists pa(r,s)\in PA\wedge s\lhd i\rightarrow\exists pa'(r,i)\in PA,pa'(r,i).context\_cond=pa(r,s).context\_cond)$$

规范15 若某角色有权访问某接口,那么该角色也有权访问该接口的超接口:

$$\forall r\in Role,\forall s,s'\in Interface(\exists pa(r,s)\in PA\wedge\exists g(s',s)\in Generalization\rightarrow\exists pa'(r,s')\in PA,pa'(r,s').context\_cond=pa(r,s).context\_cond)$$

规范16 若某角色有权访问某操作,而该操作隐含另一

个操作，那么该角色在转换的语境条件下也有权访问被隐含的操作：

$\forall r\in Role,\forall s_1,s_2\in Operation((s_1\xrightarrow{cf}s_2)\wedge\exists\ pa_1(r,s_1)\in PA\rightarrow\exists\ pa_2(r,s_2)\in PA, pa_2(r,s_2).context\_cond=cf(pa_1(r,s_1).context\_cond))$

其中，$s_1\xrightarrow{cf}s_2$表示操作方法 s_1在语境条件转换函数 cf 下隐含操作方法 s_2。

性质2 对任何角色 r 和服务 s，若由规范14、规范15、规范16或直接定义得到多个对象 $pa(r,s)$，那么最终仅有一个 PA 对象，其语境条件是这些 $pa(r,s)$对象的语境条件的并集：

$\forall r\in Role,\forall s\in Service(\exists\ pa_1(r,s)\in PA\vee\exists\ pa_2(r,s)\in PA\rightarrow\exists\ pa(r,s)\in PA, pa(r,s).context\_cond=pa_1(r,s).context\_cond\vee pa_2(r,s).context\_cond)$

证明：(1)$\exists\ pa_1(r,s)\in PA\vee\exists\ pa_2(r,s)\in PA$，由规范13知：

$(2)(CC(pa_1(r,s).context\_cond)\rightarrow(r\xrightarrow{cc}s))\vee(CC(pa_2(r,s).context\_cond)\rightarrow(r\xrightarrow{cc}s))$

$(3)(CC(pa_1(r,s).context\_cond)\vee CC(pa_2(r,s).context\_cond))\rightarrow(r\xrightarrow{cc}s)$

$(4)CC(pa_1(r,s).context\_cond\vee(pa_2(r,s).context\_cond))\rightarrow(r\xrightarrow{cc}s)$

$(5)\exists\ pa(r,s)\in PA, pa(r,s).context\_cond=pa_1(r,s).context\_cond\vee pa_2(r,s).context\_cond$

规范17 若某角色 r 未定义对某服务 s 的权限，那么该角色继承其超角色对该服务的权限规范。继承算法表示为以下三步：

第一步，首先得到角色 r 的所有对服务 s 有权的超角色集合：

$A=\{r'\in Role|\exists\ ri(r',r)\in RI\wedge\exists\ pa(r',s)\in PA\}$

若 A 为空，表示角色 r 可能没有超角色，或者所有的超角色都没有对服务 s 的权限；又由角色 r 自己也未定义对服务 s 的权限，故角色 r 对服务 s 无权。

第二步，再从 A 中过滤掉所有间接的超角色：

$B=A-\{r'\in A|\exists\ ri(r',r'')\in RI, r''\in A, r''\neq r'\}$。通俗地说，B 是 A 中角色 r 的每个分枝上对服务 s 有权的最上层的超角色。

若 A 不为空，则 B 也不为空。

第三步，最后从 B 中由性质2得到一个继承而来的权限规范：

$B\neq\phi\rightarrow\exists\ pa(r,s)\in PA, pa(r,s).context\_cond=\bigvee_{r'\in B}pa'(r',s).context\_cond$

规范18 若某角色 r 和其超角色均定义了对某服务 s 的权限规范，那么该角色的权限规范将改写(override)从其超角色继承而来的规范，且新规范应具有更大权力：

$\forall r,r'\in Role,\forall s\in Service(\exists\ ri(r',r)\in RI\wedge\exists\ pa'(r',s)\in PA\wedge\exists\ pa(r,s)\in PA\rightarrow(CC(pa(r,s).context\_cond)\rightarrow(r\xrightarrow{cc}s)))$

2.5 作用者授权规范

规范19 一个 SA 对象存在表示在特定语境下某作用者有权扮演某角色：

$\forall r\in Role\forall a\in Actor(\exists\ sa(a,r)\in SA\leftrightarrow(CC(sa(a,r).context\_cond)\rightarrow a\xrightarrow{cc}r))$

其中，$CC(sa(a,r).context_cond)$表示当前语境满足条件：$sa(a,r).context\_cond$。$a\xrightarrow{cc}r$ 表示作用者 a 在语境条件 CC 有权扮演角色 r。

规范20 若某角色 r 未定义对某作用者 a 的授权规范，该角色将从其子角色对该作用者的授权规范中派生而来。派生算法类似规范17的继承算法如下：

令 $A=\{r'\in Role|\exists\ ri(r,r')\in RI\wedge\exists\ sa(a,r')\in SA\}$

再令 $B=A-\{r'\in A|\exists\ ri(r'',r')\in RI, r''\in A, r''\neq r'\}$。

若 A 为空，则角色 r 对作用者 a 无授权。若 A 不为空，则 B 也不为空。最后从 B 中得到一个派生而来的授权规范：

$B\neq\phi\rightarrow\exists\ sa(r,s)\in SA, sa(a,s).context\_cond=\bigvee_{r'\in B}sa'(a,r').context\_cond$

规范21 抽象角色不能被直接授予作用者，即某抽象角色被授予某作用，必要条件是其一个角色被授予该作用者：

$\forall r\in Role\wedge(r.isAbstract=true)\wedge sa(a,r)\in SA\rightarrow\exists\ r'\in Role(\exists\ ri(r,r')\in RI\wedge\exists\ sa'(a,r)\in SA)$

规范22 若某角色 r 和其子角色均定义了对作用者 a 的授权规范，那么该角色的授权规范将改写(override)从其子角色派生而来的规范，且新规范应具有更大权力：

$\forall r,r'\in Role,\forall a\in Actor(\exists\ ri(r,r')\in RI\wedge\exists\ sa'(a,r')\in SA\wedge\exists\ sa(a,r)\in SA\rightarrow(CC(sa(a,r).context\_cond)\rightarrow(a\xrightarrow{cc}r)))$

规范23 任一角色授予的作用者的数目不超过该角色的授权基数：

$\forall r\in Role(|\{sa(a,r)\in SA|a\in Actor\}|\leqslant r.authorized\_cardinality)$

其中|…|表示集的基数。

规范24 任何作用者不能被授权为两个具有 SSD 关系的角色：

$\forall a\in Actor,\forall r_1,r_2\in Role(\exists\ ssd(r_1,r_2)\in SSD\wedge\exists\ sa(a,r_1)\in SA\rightarrow\neg\exists\ sa(a,r_2)\in SA)$

性质3 子角色的授权基数不超过超角色的授权基数：

$\forall r_1,r_2\in Role(\exists\ ri(r_1,r_2)\in RI\rightarrow r_2.authorized\_cardinality\leqslant r_1.authorized\_cardinality)$

证明：反证法。假设 $r_2.authorized\_cardinality>r_1.authorized\_cardinality$)。

而且设 $r_2.authorized\_cardinality=n$，因存在 $ri(r_1,r_2)$，由规范20可知

$|\{sa(a,r_1)|sa(a,r_1)\in SA,a\in Actor\}|\geqslant|\{sa(a,r_2)|sa(a,r_2)\in SA,a\in Actor\}|$

当 $|\{sa(a,r_2)\in SA|sa(a,r_2)\in SA,a\in Actor\}|=n$ 时，则有 $|\{sa(a,r_1)\in SA|sa(a,r_1)\in SA,a\in Actor\}|\geqslant n$。

再由规范23，有 $r_1.authorized\_cardinality\geqslant|\{sa(a,r_1)\in SA|sa(a,r_1)\in SA,a\in Actor\}|$，从而有 $r.authorized\_cardinality\geqslant n$，这与假设矛盾。

2.6 激活角色规范

规范25 一个 $rp(a,r)$对象在当前语境 CC 中存在，在当前语境 CC 满足角色 r 对某服务 s 的权限规范时，$rp(a,r)$对象有权调用服务 s：

$\forall a\in Actor\forall r\in Role(\exists\ rp(a,r)\in Role\text{-}Playing\wedge CC$

$(rp(a,r).context\_cond)$

$\rightarrow \forall s \in Service(r \xrightarrow{CC} s \rightarrow rp(a,r) \xrightarrow{CC} s))$

其中，$CC(rp(a,r).context_cond)$表示 $rp(a,r)$在当前语境 CC 中存在；$r \xrightarrow{CC} s$ 表示角色 r 在语境 CC 有权访问服务 s；$rp(a,r) \xrightarrow{CC} s$ 表示 $rp(a,r)$在当前语境 CC 中有权调用服务 s。

规范26 抽象角色不能被直接激活，即某抽象角色被激活的必要条件是其一个非抽象的子角色被激活：

$\forall r \in Role \wedge ((r.isAbstract = true) \wedge rp(a,r) \in Role\text{-}Playing \rightarrow \exists r' \in Role \wedge \exists ri(r,r') \in RI \wedge \exists rp'(a,r) \in Role\text{-}Playing \wedge (r.isAbstract = false)$

规范27 任一角色被激活的数目不超过它的激活基数：

$\forall r \in Role(|\{rp(a,r) | rp(a,r) \in Role - Playing, a \in Actor\}| \leqslant r.activated\_cardinality)$

规范28 一个作用者不能同时激活两个具有 DSD 关系的角色：

$\forall a \in Actor, \forall r_1, r_2 \in Role(\exists dsd(r_1,r_2) \in DSD \wedge \exists rp\{a,r_1\} \in Role - Playing \rightarrow \neg(\exists rp\{a,r_2\} \in Role - Playing))$

规范29 若某作用者在某语境条件下激活一个子角色，那么在同样的语境条件下也激活其超角色：

$\forall a \in Actor, \forall r_1, r_2 \in Role(\exists ri(r_1,r_2) \in RI \wedge \exists rp\{a,r_2\} \in Role - Playing \rightarrow \exists rp'\{a,r_1\} \in Role - Playing \wedge rp'\{a,r_1\}.context\_cond = rp\{a,r_2\}.context\_cond)$

规范30 一个作用者在同样的语境条件下激活某一角色至多一次：

$\forall a \in Actor, \forall r \in Role(|\{rp\{a,r\} | rp(a,r) \in Role - Playing \wedge CC(rp(a,r).context\_cond| \leqslant 1\})$

性质4 子角色的激活基数不超过超角色的激活基数：

$\forall r_1, r_2 \in Role(\exists ri(r_1,r_2) \in RI \rightarrow r_2.activated\_cardinality \leqslant r_1.activated\_cardinality)$

证明：由以规范27，规范29按性质3的过程可证明。

性质5 一个作用者不能激活两个具有 SD 关系的角色：

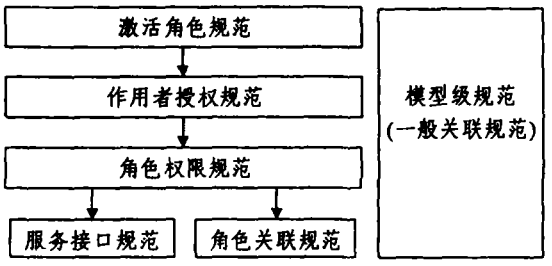

图2 模型的元素级规范具有层次结构，上层规范依赖下层规范

$\forall a \in Actor, \forall r_1, r_2 \in Role(\exists rp(a,r_1) \in Role - Playing \wedge \exists rp(a,r_2) \in Role - Playing \rightarrow \neg(sd(r_1,r_2) \in SD))$

证明：反证法。假设 $sd(r_1,r_2)$存在，由规范8可知，$ssd(r_1,r_2)$或 $dsd(r_1,r_2)$存在。若 $dsd(r_1,r_2)$存在，则与规范28矛盾；若 $ssd(r_1,r_2)$存在，又 $rp(a,r_1)$和 $rp(a,r_2)$存在，则 $sa(a,r_1)$与 $sa(a,r_2)$同时存在，这与规范24矛盾。故假设不成立。

性质6 一个作用者 a 在当前语境 CC 中对某项服务 s 有权，其充分条件是存在一个被 a 激活的角色 r，该角色在语境 CC 中对服务 s 有权调用：

$\forall a \in Actor, \forall s \in Service(\exists r \in Role \wedge \exists rp(a,r) \in Role - Playing \wedge CC(rp(a,r).context\_cond)) \wedge (r \xrightarrow{CC} s) \rightarrow (a \xrightarrow{CC} s))$

证明：因为 $CC(rp(a,r).context_cond)$和 $r \xrightarrow{CC} s$ 成立，由规范25可知，$rp(a,r) \xrightarrow{CC} s$ 成立，说明 $rp(a,r)$在当前语境 CC 中有权调用服务 s，也即作用者 a 在当前语境 CC 中对服务 s 有权，故 $a \xrightarrow{CC} s$ 成立。

3 约束规范的结构

我们建立的约束规范，根据所约束的对象不同，可分为模型级规范和元素级规范。模型级规范主要包括一般关联规范，我们的模型中包含了大量的关联结构，而且每个关联都表示不同的语义，但它们都遵循相同规范。我们给出一组简单的一般关联规范（共3项）作为模型级规范，而忽略各关联在语义上的差异，可简化约束规范的复杂性。我们根据各模型元素的语义建立的一组规范称为元素级的规范，主要包括角色关联规范、服务接口规范、角色权限规范、作用者授权规范和激活角色规范。

这些规范相互之间既有相对独立性又有依赖关系，且依赖关系呈现层次结构，上层规范依赖下层规范（如图2）。其中下层活动中规范的实施依赖上层。除了规范中直接依赖外，上层规范常使用一些形参量，由下层规范实施时提供具体实参量。例如，在角色权限规范中，确定会计角色可查看“自己”制作的凭据和帐款；而只有当用户激活会计角色后，即激活角色（Role-Playing）后才提供“自己”的实参量。规范相互之间具有相对独立性，且能保持系统的稳定性。当修改系统中某些角色的权限规范时，仅影响一部分用户的权限，而不影响其他；当发生工作人员的升迁任免时，管理员仅需在作用者授权规范中调整用户所扮演的角色，避免了为每一个用户重新设置复杂的权限规范，这简化了安全管理工作的复杂性，而且减少了出错概率。

结语 从大型分布式应用开发运行的需求来看，系统级的访问控制，如操作系统 Unix 与 Windows NT、构件系统 EJB（Enterprise JavaBeans）、数据库系统 Oracle7/8等，虽提供了简化的基于角色访问控制规范，但不能满足复杂多样的应用系统的需求。针对大型复杂系统的访问控制和安全管理，建立基于角色的企业级访问控制规范是必要的。我们介绍了一种新的对象式规范化模型，该模型结构简洁而语义丰富，并给出一组具有相对独立性的、一致性的、可推理的约束规范，比已有的模型及规范更完备，并具伸缩性，适用于种类繁多的多用户交互式和分布式信息处理系统。文[1]提出 RBAC 的概念模型，它不是一个对象式模型。本文提出的模型是采用 UML 规范的完全对象式模型，把角色、角色权限和关联都做为对象处理，简化了建模元素。对于大型系统，更易设计和实现。文[2]提出 RBAC 的部分规范，其中不涉及权力之间角色与权力之间规范，因而不完备，而且过于复杂，难以验证和实现。本文提出建立服务接口作为被保护的权力，而且对权限之间的操作隐含关系进行建模，并提出语境条件作为角色与权力之间的动态权限规范，使访问控制更具灵活性、更高效。已有模型和规范忽视了被激活的角色的动态控制，仅把被激活的角色作为一种状态。我们提出被激活的角色也是一种对象，并按 UML 规范建模为主动对象类 Role-Playing。本文给出了

（下转第129页）

同编辑，无法提供多个编辑者对节内的文本同时编辑[1]。我们的算法除了具有 tickle 锁的功能外，还具有更小范围的实时分布式协同编辑的并发控制与一致性维护能力。

JCE 中的协同编辑采用了 floor 控制方法，每个协作编者只有获得 floor 才能进行编辑操作，较适合讨论式的协作编辑，不适合实时无约束的协作编辑。而我们的算法具有实时性和无约束性。

操作转换算法具有良好的实时响应性、收敛性和无约束性，但是不能保证上下文的语义完整性，该算法的提出者和实现者在后来的实现中也集成了锁机制的并发控制算法[8]。我们的算法在实时响应、收敛和无约束等方面的性能与它相似，此外还能保证具体上下文的语义完整性。

结论 并发控制问题一直是分布式系统研究的重点和难点。本文提出的基于相对位置的乐观锁机制的并发控制策略是结合操作转换的思想，对传统的乐观锁机制及 tickle 锁方法进行了改进。我们已经将该并发控制算法和一致性维护技术成功地运用到了实时分布式协同编辑系统开发中。经原型系统实际运行和多个协作编者操作观察表明：该算法实时响应性和并发性好。该算法经进一步改进和扩充，还可应用到如线性表的分布式共享对象操作的并发控制中。

参 考 文 献

1 Greif I, Seliger R, Weihl W. A case Study of CES: A Distributed Collaborative Editing System Implemented in Argus. IEEE Transaction on software Engineering, 1992, 18(9): 827～839

2 Abde-Wahab H, Kvande B, Kim O, Pavreau J P. An Internet Collaborative Environment for Shared Java Applications. In: proc of 5th IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems (FTDCS'97), Los Alamitos: IEEE Computer Society Press, 1997. 112～117

3 Choudhary R, Dewan P. A General Multi-User Undo/Redo Model. In: Marmolin H, Sundblad H and Schmidt K, eds. Proc. of European Conf. on Computer Supported Work, Dordrecht: Kluwer Academic Publishers, 1995. 231～246

4 Sun C Z. Undo Any Operation at Any Time in Group Editor. In: Proc. of 2000 ACM Conf. on Computer Supported Cooperative Work, New York: ACM Press, 191～200

5 Ellis C A, Gibbs S J, Rein G L. Groupware: some issues and experiences. Communications of the ACM, 1991, 34(1): 39～58

6 Sun C Z, Jia X, Zhang Y, Yang Y. A Generation Transformation Scheme for Consistency Maintenance in Real-time Cooperative Editing Systems. In: Proc. of Intl. ACM SIGGROUP Conf. on Supporting Group Work, New York: ACM Press, 1997. 425～434

7 Sun C Z, Ellis C A. Operational Transformation in Real-Time Group Editors: Issues, Algorithms, and Achievements. In: Proc. of the ACM Conf. on CSCW, New York: ACM Press, 1998. 59～68

8 Sun C Z, Sosic R. Optional Locking Integrated with Operational Transformation in Distributed Real-time Group Editors. In: Proc. of ACM 18th Symposium on Principles of Distributed Computing, New York: ACM Press, 1999. 43～52

（上接第121页）

形式化的约束规范，这对进一步的机器演证和软件开发将有很重要的作用。

参 考 文 献

1 Sandhu R S, Coyne E J, Feinstein H L, Youman C E. Role-Based Access Control Models. IEEE Computer, 1996, 29(2): 38～47

2 Gavrila S I, Barkley J F. Formal specification for role based access control user/role and role/role relationship management. RBAC '98. In: Proc. of the third ACM workshop on Role-based access control, Fairfax, VA, 1998. 81～90

3 Yan Han, Zhang Hong, LiuFengyu. Modeling and Implementing Active Behavior Control Based on Roles. In: Proc. of Fourth Intl. Workshop on CSCW in Design, Compiegne, France, 1999

4 Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide. USA, Addison Wesley, 1999

5 Dewan D, Shen H. Controlling access in multiuser interface, ACM Transactions on Computer-Human Interaction, 1998, 5(1): 34～62

6 Shin M E, Ahn G-J. UML-based Representation of Role-based Access Control. In: Proc. of 5th IEEE Intl. Workshop on Enterprise Security (WETICE 2000), June NIST, MD, 2000

7 Moffett J D. Controb principles and role hierarchies, RBAC '98. In: Proc. of the third ACM workshop on Role-based access control, Fairfax, VA, 1998. 63～69

8 Ahn G-J, Sandhu R S. The RSL99 Language for Role-Based Separation of Duty Constraints. RBAC'99. In: Proc. of the Fourth ACM Workshop on Role-Based Access Control, Fairfax, VA, USA, 1999. 43～54

9 Swift M M, Brundrett P, Dyke C V. Improving the granularity of access control in Windows NT. In: Proc. of the Sixth ACM Symposium on Access control models and technologies. Chantilly, VA USA, 2001. 87～96

10 严悍，张宏，许满武. 基于角色访问控制对象式建模及实现. 计算机学报，2000，23(10)

11 陆钟万著. 面向计算机科学的数理逻辑. 北京：科学出版社，1998

12 丁铁凡，吉逸. 企业外部网的访问控制技术与系统的研究. 计算机集成制造系统，2001，7(1)：7～10