RBAC 模型的扩充及其应用

蔡国永,林煜明
CAI Guo-yong,LIN Yu-ming

桂林电子科技大学 计算机与控制学院,广西 桂林 541004
School of Computer and Control,Guilin University of Electronic Technology,Guilin,Guangxi 541004,China
E-mail:ccgycai@guet.edu.cn

CAI Guo-yong,LIN Yu-ming.RBAC extension and its application.Computer Engineering and Applications,2008,44(3):228-233.

Abstract: RBAC(Role Based Access Control) is a widely accepted model suitable for access control of organizational information system.However there is still a gap need to be filled whenever considering practical application of RBAC in development of trustable organizational system,especially in combination of organizational business collaborative model with RBAC.Based on RBAC model,elements of obligation and reward/sanction in business collaborative model are proposed to extend RBAC to a new model called RBAO (Role Based Access and Obligation).RBAO specifies not only the authorizations of roles in an organization, but also their obligations and associated sanctions or rewards in collaborative business.RBAO is more applicable in trustable business collaborative system development comparing with RBAC.The analysis and modeling process of using RBAO in trustable collaborative business system development is illustrated through a case study.

Key words: access control;trustable collaboration;interactive obligation;organization modeling

摘 要:RBAC(Role Based Access Control)是一种被广泛认可的信息系统访问安全规范管理模型,但 RBAC 访问安全规范模型如何与组织系统的业务过程规范模型融合,从而更有效地服务于可信业务协同系统的开发实践还值得进一步研究改进。在 RBAC 模型的基础上,融合协同业务规范中的义务及奖惩元素,提出 RBAO(Role Based Access and Obligation)模型。RBAO 模型不仅能描述角色在组织中可拥有的访问权力,还能描述角色在组织中可能要承担的义务及义务违反时将受到的处罚。这使得 RBAO 模型更适合用于组织可信规范业务协同系统的管理建模与开发。以具体实例说明了基于 RBAO 模型的可信业务协同系统管理的分析与建模方法。

关键词:访问控制;可信协同;交互义务;组织建模
文章编号:1002-8331(2008)03-0228-06 **文献标识码**:A **中图分类号**:TP311

1 引言

分布开放系统的安全是一个重要问题,对此也进行了许多研究,提出了一些重要的安全管理模型,特别是访问安全模型,如早期的 MAC 和 DAC[1],直到最近的 RBAC 模型[2]。在 RBAC 中,操作访问权限被关联到角色,用户通过角色映射成为一定角色的成员,从而间接地获得一定的权力。角色是 RBAC 中的一个十分重要的概念, 与组织管理的一般认知模型相一致,容易理解和应用。角色的丰富多样性及角色的层次分类体系为角色的灵活管理提供了有效组织手段。用户通过角色而间接获得授权,因而提供了访问控制管理的灵活性。然而角色如何被配置到用户,却是与用户要承担的业务职责、与组织系统的管理政策和管理制度相关联,RBAC 模型本身并不能够解决这一问题。另外 RBAC 模型研究中,主要关注模型本身的一些元素及其关系特性,而对 RBAC 模型如何建立这一问题还缺乏比较明晰的研究成果。对于这后一个问题,一些关于角色工程的研究为此提供了一些基本的成果。

文献[3]从 RBAC 模型中角色继承可能引起的问题出发,提出了一些角色的分类、聚类等措施,对角色层次体系与组织层次体系的融合或建立映射关系进行了分析。文献[4]在关于 agent 的研究中,从目标分解的角度,提出了 agent 的角色定义框架。但该框架里的角色定义完全是基于逻辑形式的,与 RBAC 模型中的操作性角色概念不同。对于第一个问题,即如何把 RBAC 模型直接与组织的业务过程模型融合,从而建立安全可信的组织协同业务管理模型的研究尚未见报道。本文从 RBAC 模型出发, 通过义务的方式来描述角色之间的业务协同,从而扩展 RBAC 模型为 RBAO(Role Based Access and Obligation)模型。为促进角色的义务履行,进一步引入奖罚机制,为开放式组织业务系统的可信履行提供更好的保障。另外,对如何分析建立 RBAO 模型的方法也进行了初步的探讨,并以具体实例来说明其应用。

本文其余内容组织如下:在第 2 章,对 RBAC 模型及其基本要素进行概述; 第 3 章讨论 RBAC 模型的扩充模型 RBAO;第 4 章介绍 RBAO 模型分析与建立的基本方法步骤;第 5 章以一个具体的实例来阐述基于 RBAO 模型的应用开发方法;最后是结束语。

基金项目:广西自然科学基金(the Natural Science Foundation of Guangxi of China under Grant No.0542036)。
作者简介:蔡国永(1971-),男,博士,副教授,主要研究领域为可信自治计算、多主体系统;林煜明(1978-),男,硕士,主要研究领域为网络协议测试。

2 RBAC 模型概述

RBAC 模型的比较系统化的版本是基于 Sandhu 等的工作提出的 RBAC96 模型[2]。RBAC96 模型包括三个基本的数据元素集:用户、角色和权力。用户可以是物理实体,如人、设备等,也可以是软件实体,如自治主体、主动类实例对象等。角色是组织语境中对业务功能分类模块化管理的抽象,角色关联到一定的组织职位权力与职责,并由关联到角色的用户具体实施。权力是对使用某一服务或操作或数据的授权。另外模型中还包括一个会话集。每一个会话表示一个映射,其作用是把用户映射到分配给用户的角色集中当前活动的角色子集。用户可以创建一个会话,并选择激活用户角色集中的某一角色或角色子集。

RBAC96 模型框架中包括四个模型类:$RBAC_{0-3}$。$RBAC_0$ 是基本核心模型,表示对支持 RBAC 系统的最小需求。$RBAC_1$ 和 $RBAC_2$ 都包含了 $RBAC_0$,但分别增加了角色层次体系(表示角色间的权力继承)和约束(表示 RBAC 不同元素的允许配置应满足的条件)。$RBAC_1$ 和 $RBAC_2$ 间并不是包含或兼容的关系。$RBAC_3$ 是 $RBAC_{0-2}$ 模型的复合。形式的,对 $RBAC_1$ 模型可表示如下:

$RBAC_1$ 表示为七元组 $M=<U,R,PRMS,S,RH,UA,PA>$ 其中 U、R、$PRMS$、S 分别表示用户、角色、权力和会话的有限集;RH、UA、PA 分别表示三种关联关系,它们分别定义为:$RH\subseteq R\times R$;$UA\subseteq U\times R$;$PA\subseteq R\times PRMS$。RH 表示角色间的层次体系关系,UA 表示用户与角色之间的映射关系,PA 表示权力到角色的映射关系。在元组 M 上还定义了一些投影函数,如 $assignUs$、$assignRs$、$assignPms$、$userSs$、$sessionRs$、$availSesPerms$;具体表示如下:

$assignUs:R\mapsto 2^U$,即 $assignUs(r)=\{u\in U|(u,r)\in UA\}$;

$assignRs:R\mapsto 2^R$,即 $assignRs(u)=\{r\in R|(u,r)\in UA\}$;

$assignPms:R\mapsto 2^{PRMS}$,即 $assignPms(r)=\{p\in PRMS|(r,P)\in PA\}$;

$availSesPerms:S\mapsto 2^{PRMS}$,即 $\cup_{r\in sessionRs(s)}assignPms(r)$;

$userSs:U\mapsto 2^S$,即用户到会话的投影函数;

$sessionRs:S\mapsto 2^R$,即会话到角色的投影函数。

这些投影函数主要用于对模型实例的查询使用或方便对模型中一些约束的表达。在 $RBAC_2$ 中引入了约束。典型的约束类型有:职责分离(SoD)和角色关联度(role cardinality)。总体上,RBAC 模型可用扩展的实体关系图表示如图 1。其中的标记符号与上述元组定义中的符号含义相同,不再重述。

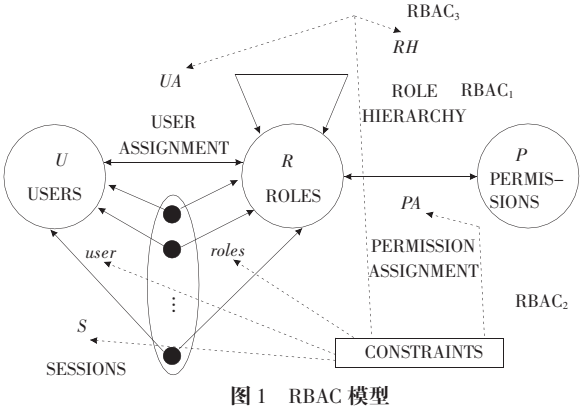

图 1 RBAC 模型

2.1 RBAC 模型中的约束类型

约束在 RBAC 模型中有重要的作用,直接涉及到安全政策的表述。在 RBAC 模型中,约束指可能发生的不期望的情形。研究人员对常见的 RBAC 模型约束形式进行了总结分类,这为实际应用 RBAC 模型提供了很好的参考。在 RBAC 模型中,主要的约束类型有[5]:

(1)关联度约束(Cardinality):指在关联关系两端的元素类型在实例个数这个量上的范围限制。如对于用户与角色之间的分配关系 UA,可限定用户可分配的角色数或角色可分配的用户数。显然关联度约束可应用到 RBAC 模型中定义的任何关联关系类型。

(2)静态职责分离(SSD):指同一元素类型的元素之间的静态制约关系。特别是指元素之间的冲突关系,如当两个角色是相互排斥(如有利益冲突)的时候,则同一个用户不能同时分配到这两个角色,这种情形称为角色的静态职责分离。如在银行业务中,负责贷款申请业务的角色和负责贷款审批业务的角色显然不能分配给同一个用户。

(3)动态职责分离(DSD):指同一元素类型的实例之间的动态制约关系。主要指角色间的动态冲突关系。即在一个会话期间,对激活的角色有某种限制(如用户不能同时激活所分配的但是相互排斥的角色)。如在贷款业务处理中,为使系统管理具有更好的灵活性,有时也允许同一个用户同时具有处理贷款申请和贷款审批操作的权力,但为保证公正性,需要保证对同一笔业务,同一个用户不能同时执行贷款申请和审批。这就是动态职责分离约束机制。

在实际应用 RBAC 到某一领域时,需要保证什么样的约束,采用什么样的约束类型与具体的应用需求和实施策略有关。另外还可能需要一些新的约束类型以适应新型政策的需要。有些约束也可能只与特定领域有关系,并不具有普遍性。因此 RBAC 模型必须结合实际领域的业务模型来分析并进行建模。

3 RBAC 模型的扩展

实际应用中 RBAC 模型有较大的局限性,如在 RBAC 模型中,权力表示赋予角色(或用户)某种能力(如读写文件、审批贷款等),但这种能力是否实施并没有定义,即这种权力可能实施了,也可能没有实施。在实际分布协同系统中,除了有权利之外,常常需要明确地说明权力的实施,即有权利的用户(或角色)必须实施某种义务(如在接到顾客的投诉后,投诉处理机构有义务在 5 个工作日内做出答复),这样才能有效地保证协作业务目标的完成。另外在 RBAC 模型中,对权力操作之间的交互关系也没有说明。实际应用中,权力之间有明显的关系存在,要获得某种权力可能直接与另一权力的实施或已获得有关。如要审批贷款,显然应该有申请贷款的操作已经实施。因此有必要扩充 RBAC 以能够处理这种情况,从而使 RBAC 更好适应实际应用系统的需要。由于操作权力采用角色分组管理,因此操作之间的关系可以通过角色之间的交互来体现。因此下面通过引入角色义务这种关系元素,来补充 RBAC 的不足。

义务表示有能力/权力的角色用户必须执行(或实施)某一操作活动。义务可以是基于条件或事件的,即条件引发的义务或事件引发的义务。义务非常适合于描述自治主体之间的交互,相应的已提出了不少进行义务推理的逻辑系统,如动态义务逻辑 DDL[6]。由于自治主体有一定的自由决策能力,义务可能不被实施,因此义务通常和奖罚机制联系在一起,以促进义务的履行,从而保证系统协同业务功能的安全可信实施,达到系统的总体目标。

在 RBAC 四种基本元素(用户、角色、权利、会话)的基础上,扩充一个角色义务和奖惩元素。角色义务表示角色在某一

事件发生时,通告与该角色相关联的用户或自治主体必须执行某一操作。如果不执行,就导致违约,会受到相应的违约处罚。如果义务实施了,就进行相应的义务履行奖励处理。扩展RBAC模型(以下称为RBAO模型)表示为多元组 $M=<U,R,PRMS,OBLS,RSancs,S,RH,UA,PA,OA,RSA>$,其中 $OBLS$ 表示义务集,$obl=<event|condition,operation,object>$,$obl\in OBLS$,表示义务是一个三元组,三个元组元素分别表示事件/条件、操作及实施对象。OA 表示角色到义务的分配关系,即 $OA\subseteq R\times OBLS$。$RSancs$ 表示奖罚操作,$rs=<event|condition,RSoperation,RSobject>$,表示奖,$rs\in RSancs$ 惩操作事件/条件及实施对象。RSA 表示角色到奖励或处罚的分配关系,即 $RSA\subseteq R\times RSancs$,其它符号含义同RBAC模型。RBAO模型可用扩展的实体关系图表示如图2。

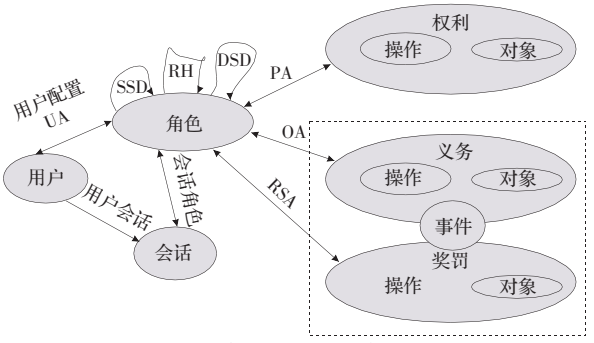

图2 RBAO模型

类似于RBAC模型,在RBAO模型中也可以定义义务、奖惩相关的投影函数,定义如下:

$assignObls:R\mapsto 2^{OBLS}$,即 $assignObls(r)=\{u\in OBLS|(r,u)\in OA\}$;

$assignRSs:R\mapsto 2^{RSancs}$,即 $assignRSs(r)=\{u\in RSancs|(r,u)\in RSA\}$;

$availSesObls:S\mapsto 2^{OBLS}$,即 $\cup_{r\in sessionRs(s)}assignObls(r)$;

$availSesRSs:S\mapsto 2^{RSancs}$,即 $\cup_{r\in sessionRs(s)}assignRSs(r)$。

其它投影函数与RBAC模型相同。通过投影函数可以实现对模型信息的查询。

在RBAO模型中,通过角色义务及其奖罚机制,以事件作为连接纽带,显式地说明了角色关联主体之间的义务交互关系,从而把角色关联主体之间的交互关系也整合到基于角色的安全访问模型中,为角色访问控制权力的分配提供了合理的依据,而义务的分配则直接取决于系统的协同业务功能需求。通过如此扩充之后,RBAO中的角色与RBAC中的角色已经有显著的不同,在RBAC中,角色只是一种表示权力配置的被动的关联元素,而在RBAO模型中,角色除表示权力配置的关联关系外,还可表示角色主体间的社交规范关系,从而更能促使开放环境下用户(或自治主体)间协同业务的可信实现。类似于RBAC,对RBAO模型中的约束也可以进行总结分类,以利于RBAO模型的分析应用。

3.1 RBAO模型中的约束与冲突类型

RBAC中约束的关注焦点主要是针对权力配置中可能的不安全问题,即主要表现为各种分离机制,而在RBAO模型中,除了权力需求的正确配置外,还要考虑义务关系的正确配置及义务违反的处置,以促使协同任务的可信履行。由于义务是交互关系的规则,所以其交互过程的终止性、进展性、公平性等也是模型验证分析中重要问题。因此下面把RBAO中的约束分为如下几个方面:

(1)不期望的约束类型:这主要对应于RBAC模型中的职责分离约束(SSD和DSD)、关联度约束。该类约束主要从可能的安全威胁角度,考虑可能存在的安全问题,然后建议相应的分离措施以通过权力分配的机制消除这些威胁;

(2)期望的约束类型:这一类型主要从业务功能活性的角度考虑,描述为达到业务的目标,每一个角色期望与其交互的角色应承担的义务和自身的义务。如贷款办理的角色期望与之交互的处理贷款审批的角色在一个工作日内返回审批结果;

(3)奖罚约束类型:为促进具有一定自治性的用户履行义务,对一些业务关键的操作采取奖罚方式,说明义务实施将获得的收益或违反义务将受到的处罚。

另外,在RBAO模型中,同时从互补的两个方面(期望与不期望,奖励与处罚)来描述角色的权力和义务关系。因此模型本身可能会存在不一致性,引起冲突。这种冲突如果是语法上的,则可以通过语法分析进行判别。如果冲突是语义上引起的,则只能通过模型动态推理或模型检查的方式才能判定。语义上的冲突超出了本文研究的范围,将另文讨论。这里只对前一类冲突进行总结。前一类冲突也称为模态冲突,主要的模态冲突类型有:

(1)权力义务模态冲突:即对同样的操作,可能存在授权与义务的不一致描述。如可能存在这样的模型描述:有义务但无权利或有权利但无义务。因此需要检查这种情况的出现,确认问题的根源并予以消除,使系统授权尽可能满足最小权利原则;

(2)奖励处罚模态冲突:即对同样的操作,可能存在奖励与处罚的不一致描述;

(3)义务模态冲突:指某一角色分配的义务操作本身之间存在有资源冲突或目标冲突等,从而导致义务的不确定性。这种角色义务分配本身的冲突可能导致不应该出现的奖罚操作,因此要对之重点进行分析检查;

(4)权利模态冲突:指如果权力分配时,允许同时使用否定语句和肯定语句,则可能同时存在一个允许的授权说明语句和一个禁止的授权说明语句;

(5)交互模态冲突:指角色之间交互的权利和义务关系的不一致。即某角色有权利请求另一角色提供某一服务,而另一角色并没有提供该服务的义务。

3.2 RBAO模型中的模态冲突检测算法框架

在RBAO建模中,如果采用合适的形式描述语言,则可以采用与文献[8]类似的做法,对相关的冲突或约束进行自动检查。在检查是否存在可能违反管理或业务需求的模态冲突时,最基本的方面是计算冲突的角色。冲突角色的产生其根源是操作及操作对象的在安全、义务和奖惩管理上的冲突。因此如果给定操作及操作对象的基本冲突定义,依据RBAO模型中权利义务到角色的分配关系,及角色之间的层次体系关系,就可以计算出有冲突的角色集,从而建立起正确的RBAO模型,避免出现不恰当的配置关系。下面给出冲突角色计算的一个基本算法框架。

定义 角色冲突,设 r_1、r_2 为角色,r_1、r_2 的规约集分别为 P_1、P_2,r_1 与 r_2 冲突当且仅当存在 $p_1\in P_1,p_2\in P_2$,使得 p_1 与 p_2 冲突。

计算冲突的角色分两种情况:一是角色之间如果没有定义层次体系关系,则可以直接根据其授权的操作集判断;如果角色之间允许有层次体系关系,则角色就可能不是显式定义的,而可能是通过继承等方式间接构造出来的,则角色之间是否存在冲突,就需要考虑角色之间的继承关系。冲突角色计算的基本计算框架步骤如下:

(1)定义基本操作及对象的冲突的可能模式,并作为算法的输入;把此模式与基本操作和对象实例库进行模式匹配得到

具体的冲突操作实例集;

(2)计算角色的操作对象集。如果是基本角色,则可以直接得出;如是复合角色,则可通过追踪其复合方式,直到基本角色,然后由基本角色操作对象集的并得到复合角色的操作对象;

(3)计算角色对的冲突性。对任意两个角色,应用第(2)步计算出的操作对象集,如果它们的并包含了冲突操作集中的元素,则这两个角色冲突。否则它们不存在冲突。

4　RBAO 的建模方法

如何根据系统的需求规范叙述,抽取并建立起恰当的 RBAO 模型是 RBAO 应用的关键。针对开放组织协同业务功能域及其安全规范要求,提出以下的 RBAO 建模方法。该方法从 3 个模型层次,4 个语境范畴和 3 个约束政策维度去抽取并逐步细化需求规范叙述,进而建立 RBAO 模型的具体形式描述。用图 3 表示 RBAO 建模的基本范畴元素。横轴表示 4 个语境范畴:组织、产品、任务和交互;纵轴表示 3 个模型层次:元模型、概念模型和实例模型,它们一起构成模型信息语义域;3 个不同颜色的平面分别表示 3 类约束政策平面,表示系统需求规范语义。

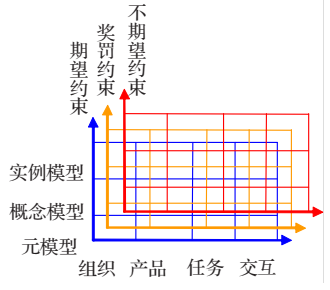

图 3　RBAO 建模框架模型

组织表示 RBAO 描述的系统针对的应用域机构,任务表示该应用机构依赖系统开展的业务,产品表示机构业务涉及的基本对象,交互表示机构中各职位/角色的基本职权及相互关系。基于这个多维度模型抽取框架,RBAO 模型建立的基本步骤是:

(1)以 RBAO 元组表示中的基本元素作为元模型的基本元素;

(2)从组织、产品、任务、交互 4 个语境去(具体化)扩充元模型,建立业务域的概念模型;

(3)实例化概念模型中的元素,建立需求规范叙述所需的具体概念实例表示;

(4)在这 3 个层次模型的基础上,分别建立 3 个政策平面的需求规范的形式表达;

(5)重复前面 4 个步骤,直到所有的需求规范叙述都可在信息语义模型的基础上得到形式的表达。

通过这几步的操作,初步的 RBAO 模型就建立起来了,然后可以分析并识别可能的安全威胁,并定义基本的冲突操作和对象,进而计算冲突的角色,并根据具体的应用领域,考虑这种冲突是否必须消除,最终建立起初始正确的 RBAO 模型规范。

4.1　RBAO 概念元模型

RBAO 概念元模型由上面介绍的 RBAO 基本模型扩充而来,它为建立具体的 RBAO 概念模型提供基本的框架。RBAO 概念元模型可用类关系图表示如图 4。其中核心元概念有角色、操作、对象和事件,表示结构关系的基本元概念有组织、任务、产品、交互及授权、奖罚。在这些基本元概念的基础上,根据具体的组织系统及其业务领域、安全需求等,可以进一步扩充,

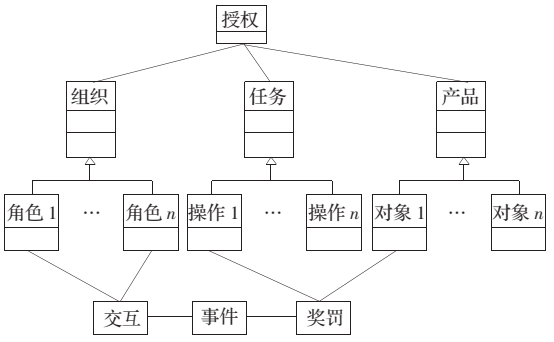

图 4　RBAO 元模型类关系图

建立起组织业务系统的概念模型、实例模型。下节将通过具体实例来说明概念模型及实例模型的建立。

5　RBAO 模型的应用案例

现以一学术会议组织/机构业务协同处理系统(ACOS)为例来说明如何基于 RBAO 模型逐步细化 ACOS 的需求。设 ACOS 的需求叙述如下:

ACOS 系统支持会议组织机构开展会议组织业务。其基本工作过程如下:首先由会议组织者负责选定会议主席和会议秘书,组成某个会议的工作程序组。会议主席和秘书负责会议论文相关的业务工作,主要有:(1)发布会议征文广告;(2)招聘会议论文评阅人;(3)接收作者提交的论文;(4)组织论文的评阅,并决定录用的论文;(5)通知作者论文的评阅结果;(6)通知录用论文的作者按时支付会务费;(7)组织会议的召开。

为保证公平性和业务的正常开展,要求业务支持系统 ACOS 要保证如下的会议制度或规定得到实施:(1)不允许同一用户投递并评阅同一论文;(2)同一用户不能投递 2 篇以上的论文;(3)同一评阅人不能评阅 10 篇以上的论文;(4)评阅人每次只能选择一篇论文评阅,在上一篇论文没有完成评阅前,不能再申请论文评阅;(5)如果一用户投递论文在录用后没有按时交付会务费,则作放弃出版权处理;(6)如果评阅人没有按规定时间给出论文的评阅结果,则其在该系列会议中的信誉将会降低;如果按时递交评阅结果,则其信誉值将会增加。当信誉值增加到一定程度后,将会成为高级评阅人。高级评阅人在本系列会议中将拥有更多的权利。

5.1　ACOS 系统基本 RBAO 概念模型的建立

根据上面 ACOS 系统的基本需求叙述,分别从组织角色、组织业务基本操作,业务对象三个方面初步考虑 ACOS 系统中的基本概念。对 ACOS 系统,针对其基本需求叙述,基本概念列于表 1。表中第 1 列为组织角色、第 2 列为基本业务操作,第 3 列为基本业务对象。

5.2　基本概念模型的扩充

根据 ACOS 系统的基本制度规范,可以对上述表格进一步扩充。由于评阅人要按信誉进行管理和分类,因此把评阅人进行一步细化,分为初级评阅和高级评阅人。另外在制度规范中,还涉及奖惩操作,因此,在表格中增加一列,表示奖罚相关的操作,并补充与奖罚操作有关的对象与角色。扩充的结果增加到表 1 中,如下划线所示部分。为表达方便,把 ACOS 中的有关组织制度约束也增加到表格 1 的最下部,表示组织制度约束。

构造好表 1 后,可以进行角色到业务功能操作的基本授权,建立起角色的职责权利。假设业务操作按角色是这样分配的:会议组织者负责创建 ACOS 的基本信息中心结构,创建主席和秘书角色实例及制度约束处理。会议主席的职责/权利有:

表 1　ACOS 系统模型规约

ACOS 用户角色	ACOS 业务操作	ACOS 业务对象	ACOS 奖罚操作	备注
会议组织者	选定会议主席和秘书	会议征文	增加信誉	
会议主席	发布征文广告	论文	减少信誉	
会议秘书	招聘评阅人	评阅意见	通知处罚	
论文评阅人	接收论文	录用通知		
(高级评阅,初级评阅)	论文评阅	会务费		
	决定论文的录用	处罚通知		
作者	通知评阅结果	信誉		
	通知及支付会务费			
	组织会议的召开			
组织制度约束	(1)任何同时扮演作者及评阅人的用户不能评阅自己的论文;(2)任何作者不能投递 2 篇以上的论文;(3)任一评阅人不能评阅 10 篇以上的论文;(4)任一评阅人每次只能选择一篇论文评阅,直到上一篇论文完成评阅,才能再申请论文评阅。			

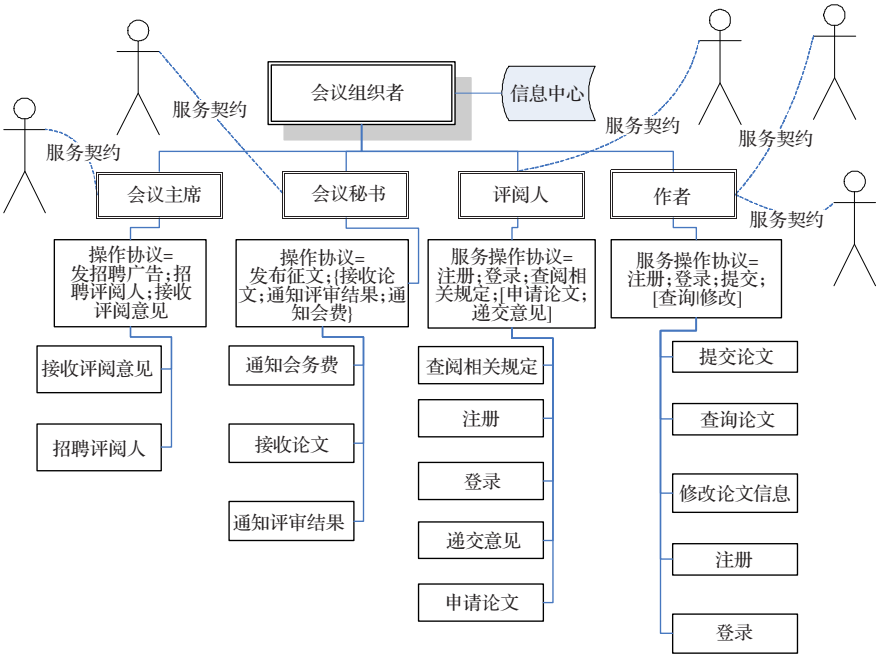

图 5　ACOS 角色配置图

发布评阅人招聘广告,选择论文评阅人,分配论文稿件的评阅,决定论文的录用。秘书的职责/权利有:发布会议征文通知,接收及管理论文作者及其投稿,通知作者论文评阅结论和录用结果。论文评阅人的职责/权利有:选择论文评阅,递交论文评阅意见。论文作者的职责权利活动有:投递自己的会议论文,查询自己的论文评阅结果,提交会务费用,出席会议等。则系统基本角色(组织者角色除外)功能配置表示如图 5。

5.3　概念模型的进一步细化及角色交互

为进一步明确角色之间的交互及权利义务规约,还需进一步细化上述的业务操作,因篇幅所限,这里以初级评阅人、会议主席及会议组织者间的交互为例来说明细化过程。设细化后交互过程如下:(1)用户首先向系统注册,申请成为评阅人,即把自己的信息上传到会议组织者的信息库;然后由会议主席决定是否同意接收其为评阅人, 如同意则给其分配帐号和初始密码,并传递给用户;(2)用户根据其帐号和密码登录系统,系统建立起相应的评阅人会话并激活相应的评阅人角色;(3)在评阅人会话中,评阅人可以执行相关论文目录查询,向会议主席申请评阅某一论文;(4)在主席同意论文申请后,则把论文全文传递到评阅人信箱。收到论文后,评阅人有义务在一周内返回评阅结果;(5)如评阅人按时完成其职责,得到主席认可后,系统将会给评阅人信誉和经验加分;否则系统将会给评阅人信誉减分。下面用交互图来可视化这一过程,如图 6 所示。图中对评阅人的义务以带三角线标识,奖罚以带叉子的线标识,权利则以带圆圈的线标识。

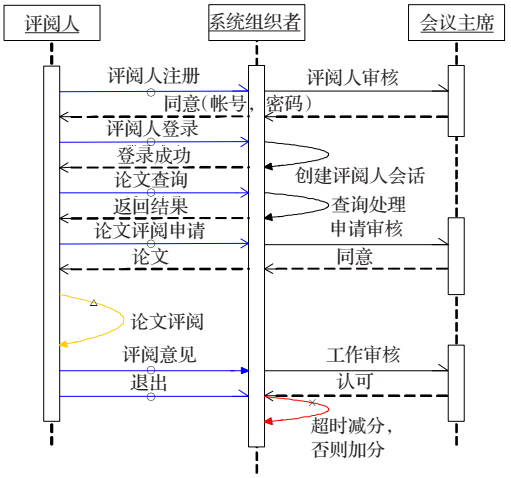

图 6　ACOS 角色交互图

5.4 基于本体的 RBAO 模型形式规约及冲突分析

为对 RBAO 模型进行形式分析，必须以形式的语言对 RBAO 模型进行描述，下面介绍以基于本体的角色政策语言 RPL 来描述 RBAO 模型。RPL 的角色相关部分的语法如下：

Role::='concept role' <name> '('['Super:'<role name>]
{'property:'<property name><property type>} {'operation:'<content3>}
{'authorization:' <content1>}{'obligation:'<content2>}')'

content1::=[<name>]'('<subject> <modality> <target><Activity><constraint>')'

content2::=[<name>]'('< subject> <modality> <target><Activity><Event><constraint>')'

content3::=[<name>]'('<Moperation>')'

name=property name=role name=actor name::=<label>

label::=<letter>{<letter>|<digit>}

subject=target::='self'|<role name>|<actor name>

modality::='+'|'-'

constraint::=<boolean expression>|'('<constraint> <op1> <constraint>')'|'not'<constraint>

op1::='and'|'or'

Moperation::=<Aoperation>|'('<Moperation> op <Moperation>')'

op::=';'|'+'|'||'

Aoperation::='?'<Event>|'! '<Event>|<attachR>|<deattachR>|<activateR>|<suspendR>|<shiftR>

Event::='ask'|'tell'|'timeout'|'success'|'failure'

RPL 主要用于对角色的权利与义务进行形式描述，以方便检查系统权利与义务等方面配置的合理性、安全性。在 RPL 中，角色之间的关系用 super 指示，权利用 authorization 指示，义务用 obligation 指示。权利内容 content1 包括 5 部分，分别表示权利主体(角色或扮演角色的演员)、权利模态(允许或禁止)、客体(操作对象或角色)、动作及约束。义务内容 content2 也包括类似的部分，但义务由事件 event 触发。另外角色可以设置自身的基本操作 operation，以辅助义务的实施，如生成某些事件等。下面对评阅人的交互图，用 RPL 描述的权利义务如下：

concept role 评阅人(Super:角色
Authorization:ja1(self,+,主席,ask(评阅注册),初始状态)
Authorization:ja2(self,+,系统组织者,ask(评阅登录),评阅人状态 1)
Authorization:ja3(self,+,系统组织者,ask(查询论文库),评阅人状态 2)
Authorization:ja4(self,+,主席,ask(申请论文)),评阅人状态 3)
Authorization:ja4(self,+,主席,tell(评阅意见)),评阅人状态 4)
Authorization:ja5(self,+,系统组织者,tell(退出),评阅人状态 4)
Obligation:jo1(self,+,扮演者,ask(评阅论文),tell(论文),期限<=一周)
)

可以在此基础上，以评阅人为基础角色，定义高级评阅人角色。类似的可以定义系统组织者，主席，秘书等角色的权利、义务和奖罚处理规则。当所有角色规约完成后，则可按上述算法框架检查是否存在冲突的角色。如存在，则进行相应的修正，直至初始 RBAO 模型规约满足一致性为止。如果把 RPL 描述转化为本体推理机(如 Pellet[7])等支持的语言，则参照文献[8]，可以自动进行 RBAO 模型的一致性或冲突性检查。

设有用户实例集 $U=\{u1,u2,u3,\cdots,u9\}$，其中扮演作者角色的实例集 $A=\{u1,u2,\cdots,u7\}$，评阅人实例集 $R=\{u1,u8,u9\}$，论文实例集 $P=\{p1,p2,p3,\cdots,p7\}$，分别对应 $u1\sim u7$ 的论文，设当前论文的评阅分配集 $S=\{<u1:p2,p4>,<u8:p3,p6>,<u9:p5,p7>\}$。由于投递论文和评阅论文操作可能存在冲突，因此若现评阅人 $u1$ 向主席申请论文 $p1$，主席应检查是否有存在动态职责分离约束(DSD)，如制度约束：

(1) $\forall u\in U\wedge u\in R\wedge u\in A\cdot\Box(paper(u)\notin reviewPaper(u))$，即要求总是保证评阅人不能评阅自己的论文。如约束不能满足，则申请将被拒绝。类似地，其它制度约束可如下表示并相应进行检验；

(2) $\forall u\in U\wedge u\in A\cdot\Box(|paper(u)|\leqslant 2)$；

(3) $\forall u\in U\wedge u\in R\cdot\Box(|reviewPaper(u)|\leqslant 10)$；

(4) $\forall u\in U\wedge u\in R\cdot if\forall p\in reviewPaper(u)\cdot p1\in P, then\ can(request(p1))=true$。

6 结束语

RBAC 模型对安全访问控制是一种合适的模型，然而在实际应用中，权利和义务总是相互关联的，传统的 RBAC 模型显得描述能力不足，有必要对 RBAC 模型进行扩充。文献[9]对 RBAC 进行时态扩充，提出了角色时态扩充的 TRBAC 模型，可以表达角色的时间约束或角色间的时态依赖；Hansen 等[10]通过扩充 RBAC，使得它可以表达授权时权力关系的空间约束，从而提出相应的 SRBAC 模型。本文从权利义务奖惩的角度对 RBAC 模型进行扩充，把角色作为基本的权利和义务组织单元，从而使角色成为组织的协同管理对象，可以更好的促进组织协同业务目标的实现。针对扩展的 RBAO 模型，同时还提出了基本的建模过程框架，并设计了相应的角色规范语言。相比于其它 RBAC 的扩展模型，RBAO 模型具有更大的灵活性，适合开放分布的协同环境，能对权力实施过程进行必要的监管。对正确履行的义务给予奖励，否则给予处罚。从而不仅保证访问的安全，还能促使义务的实施，达到系统安全协同的目标。RBAO 模型对于安全协同系统的开发提供了很好的分析建模基础，然而对于 RBAO 中角色的具体实现还需要实现相应的支持中间件，多代理平台技术可能是实现 RBAO 模型的较好的基础，目前的主要工作是研究如何在移动代理平台上实现基于 RBAO 模型的开放可信的协同业务系统。(收稿日期:2007 年 7 月)

参考文献:

[1] 屈延文.软件行为学[M].北京:电子工业工业社,2004.
[2] Sandhu R,Coyne E,Feinstein H,et al.Role-based access control models[J].IEEE Computers,1996(29):38-47.
[3] Lee H H,Lee Y L,Noh B N.A framework for modeling organization structure in role engineering[C]//LNCS 3732:PARA 2004,2006:1017-1024.
[4] Periorellis P,Parastatidis S.Task-based access control for virtual organizations[C]//Guelfi N.LNCS 3409:FIDJI 2004,2005:38-47.
[5] Hansen F,Oleshchuk V.Conformance checking of RBAC policy and its implementation[C]//Robert H D.LNCS 3439:ISPEC 2005,2005:144-155.
[6] Duc H N.Semantical investigations in the logic of actions and norms[D].Institut fur Logik und Wissenschaftstheorie,1995.
[7] Sirin E,Parsia B.Pellet system description[EB/OL].[2006-10].http://www.mindswap.org/2003/pellet.
[8] Zhao Chen,Heilili Nuermaimaiti,Liu Sheng-ping,et al.Representation and reasoning on RBAC:a description logic approach[C]//LNCS 2005:Second International Colloquium -Theoretical Aspects of Computing(ICTAC 2005),2005,3722:394-406.
[9] Bertino E,Bonatti P A,Ferrari E.TRBAC:a temporal role-based access control model[J].ACM Trans Inf Syst Sec,2001,4:191-223.
[10] Hansen F,Oleshchuk V.SRBAC:a spatial role-based access control model for mobile systems[C]//7th Nordic Workshop on Secure IT Systems,2003:129-141.