工作流授权控制模型

Workflow Authorization Model

版本：1.0

作者　　：胡长城
网上游名：银狐 999 ；james999
个人主页：http://www.javafox.org
完成时间：2003-12-14
联系信箱：james-fly@vip.sina.com
MSN　　：fcxiao2000@hotmail.com

本文大部分参考文档，可以在 http://www.javafox.org/workflow/index.html 下载

目 录

1. 引文

一个月以前就想写有关工作流权限控制方面的内容，那时候也才刚刚开始写“ 工作流模型分析 1.0 ”版。因为以前开发工作流产品的缘故，对 Role-based 的模型还是比较清楚，但是针对 Task-based 模型就接触很少（因为开发的产品，仅仅针对基于角色的权限控制）。所以这一个月来，抽空都在查阅相关资料，弥补理论和实践的不足。本篇中，对 Task-based 模型分析的不够深刻，也源于此。所以有关这方面的不足和错误之处，请指出。也欢迎大家与我探讨这方面的内容。

全文写完了，才回头来写引文（有些颠倒次序了），不过在这里想谈谈自己写这篇文档的感受。为了完成这篇文档，中途花了很大的精力在查阅国外的文档，虽然现在文档写完了，留在心里面的还依然是“ 汗颜 ”，当然也有几分“ 庆幸 ”。

庆幸我们站在了巨人的肩膀上。在过去的二三十年里，有非常多的研究者们，用他们的钻研，抽象出了各式各样的模型，为我们解决问题寻找到方便快捷的途径。

但是我们也不得不深感自己的“ 落后 ”。在搜索中文工作流方面的理论性文档的时候，竟然寥寥无几。这落后并不是因为技术，而更多的是源自对“ 理论 ”的研究。

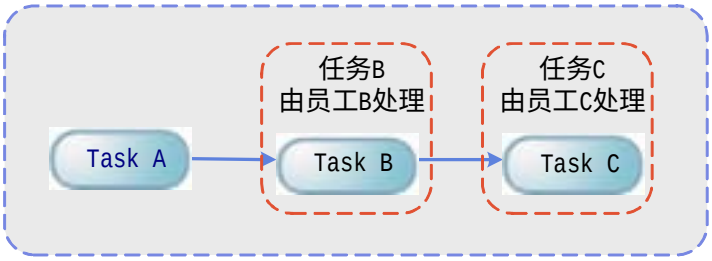
我们可以用几个月学会一门技术，但是我们要用多久，去领悟一套思想呢？

2. 概述

在“ 工作流模型分析（version 1.1）”中，我给大家介绍了工作流应用中常见的几类模型。但是，在工作流模块体系中，模型不是独立存在。特别是在人为因素的干扰下，流程会因为各种各样的受控或者受限，而发生变化。而流程所运载的数据，也会随着受控限制的不同，而表现出不同的状态。

工作流的权限控制，也就是为了让工作流运转更好的和人的操作有序、有规章的结合起来。对外表现起来就是：A 应该完成 A 所应该完成的工作，也仅仅能够完成其可操作范围内的工作。

最粗劣的表示就如下图所示。



图中员工 B（员工 B 可能属于一个角色 B）需要处理任务 B，而员工 C 需要处理任务 C。有必要说明一下，这里写的是员工，而非通常意义下的角色。是想简单的说一个观点：任何

角色仅仅是一个赋予的含义，其真正落实还是“人”。一个工作授予一个角色，代表的是属于这个角色的人，都有权限操作这项工作；但是操作这个工作的，是一个“人”，而非一个“角色”。

目前，工作流的权限控制层，比较普遍的方式是两种：

以任务为基础的授权控制模型(Task-Based Authorization Model)

以角色为基础的访问控制模型(Role-Based Access Control Model)

不过，在介绍这两种模型之前，有必要简单的说说 subject-object view of access control 模型。虽然 subject-object 在工作流应用中很少，但是作为最原始的访问控制雏形，有必要说一下。

3. Subject-Object View

在说到 subject-object view 控制模型中，大多都会有几个定语，像传统的（traditional），或被动的（passive）等等。这几个词语，就说明了其存在一定的局限性，过时性。subject-object 是在上个世纪七、八十年代比较流行的一种权限控制模型，那个时候，比较流行采用“访问矩阵（access matrix）”方式进行资源的访问控制。

有关它的历史，有必要追溯到1969年。那时候B.W. Lampson 通过形式化表示方法运用主体（subject）、客体（object）和访问矩阵（access matrix）的思想第一次对访问控制问题进行了抽象。

主体是访问操作中的主动实体，客体是访问操作中被动实体，即，主体对客体进行访问。访问矩阵是以主体为行索引、以客体为列索引的矩阵，矩阵中的每一个元素表示一组访问方式，是若干访问方式的集合。矩阵中第i 行第j 列的元素Mij 记录着第i 个主体Si 可以执行的对第j 个客体Oj 的访问方式。比如Mij等于{read, write}表示Si 可以对Oj 进行读和写访问。

	Object A	Object B
Subject A	R	RW
Subject B	W	R

上图，简单的描绘了 subject-object 访问控制模型。其中绿色字体则表示了主体（Subject）对客体（Object）的访问权限（Right）。同样的，我们也很清晰的看到，在 subject-object view 访问控制模型中，有三个基本的元素：Subject（S），Object（O），Right（R）。

subject-object 访问控制模式在早期的安全应用中被广泛的采用，但随着应用逐渐系统化，流程化。被动的和静态的访问控制方式已经无法满足动态的应用。在 so 中，任何一

个访问许可都是独立的，与其它许可之间没有任何的关联关系。

4. 基于任务的授权控制模型

在上个世纪八十年代中后期，基于任务的访问控制被大量的应用。用于解决基于流程任务的，或基于分布式程序的，以及基于事务型的应用安全控制问题。那时候比较流行“ 主动式安全模型（Active security model）” 概念。

不同于传统的访问控制模型（比如上面的 subject-object 等），TBAC 更加倾向于基于应用层的控制，而且是基于主动式安全模型（Active security model）。

主动式安全模型：这种安全模型，更多的从活动或任务的角度看待安全和实施的控制问题，它抽象出了一种控制机制，用于任务运行期间的管理。其同时也抽象出任务之间的一些访问/授权关联关系，比如只有当任务 A 被授权访问后，任务 B 才能被授权访问。

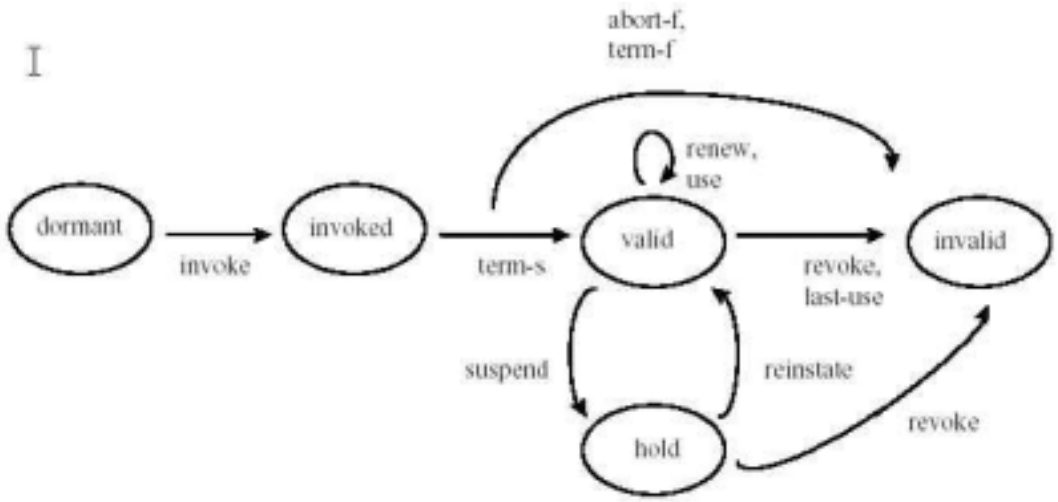
所以，在主动式的安全模型中，许可（Permission）就不是一个固定的状态，授权和许可等行为都在被不断的监控，许可状态（Permission State）也随着不断的变化。

在 TBAC 中，更多体现出授权许可状态，以及任务与任务之间的关联关系（Dependency）。其中，许可状态则是 AS（Authorization Step）的结果反映。

4.1. Authorization-step

授权步骤抽象出最基本的认证过程。有关其具体的介绍，可以参看【R.S97】文档。

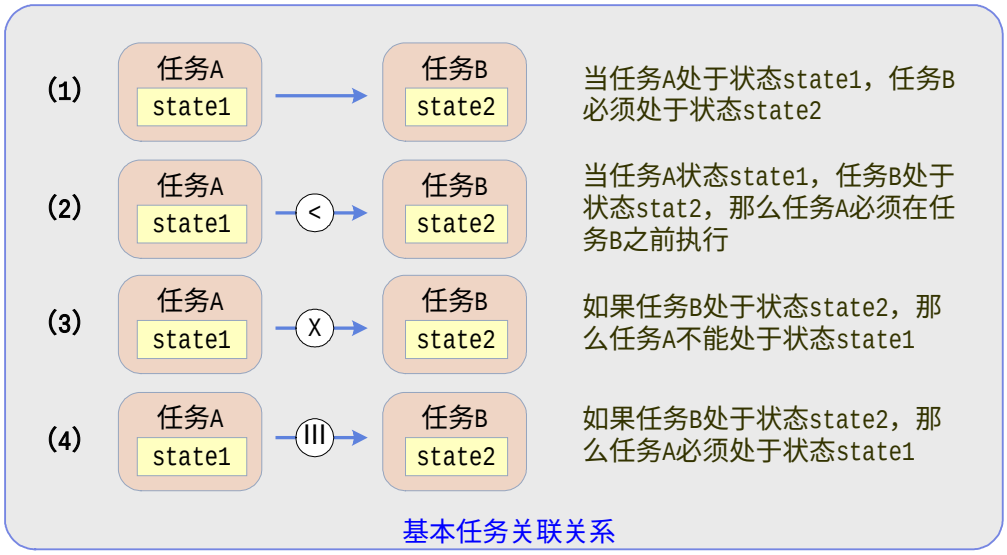
下图显示了授权过程的生命周期（life-cycle of authorizations）。



4.2. 基本任务关联关系

按照 TBAC0 模型中，任务之间的关联关系大致有 4 种。当然还另外存在一种关系：无关联（Separation Dependency）。

接下来，给大家简要介绍一下Existential和Temporal，以及concurrency（存在两种关联）这三类四种关联关系。其中Existential和Temporal是一种正向分析，而concurrency则是逆向分析。参看下图：



基本任务关联关系

第(1)(2)两种方式，就是通常被称为Existential Dependency and Temporal Dependency的依赖关系。注意，这里的state表达的是“ permission state ”，也就是认证被通过（执行）或不通过（不执行）。

基于任务的授权控制模型就简单介绍到这里，因为以前我也没有具体应用过 TBAC，很多细微的内容也无法深入的谈论。所写的这点内容，也是搜刮了很资料后，自己总结的。这些日子虽然加紧弥补这方面的空缺，但依然感觉“ 仅窥皮毛 ”。如果哪为对 TBAC 有深入研究，希望不吝赐教。我的 email 和 msn 第一页有，也希望大家就这方面问题与我交流。

5. 基于角色的访问控制模型

随着应用复杂的度的不断提高，传统的访问控制（比如基于 Subject-Object 的 MAC 和 DAC 方式）越来越显现出其局限性，其几乎静态化的个体——资源控制已经无法适应复杂多变的应用系统——资源的变动，人员的变动，以及系统/任务之间相互协调和关联关系。在这种情况下，基于角色和基于任务的等一些较为主动型的访问控制方式，被广泛的采纳。

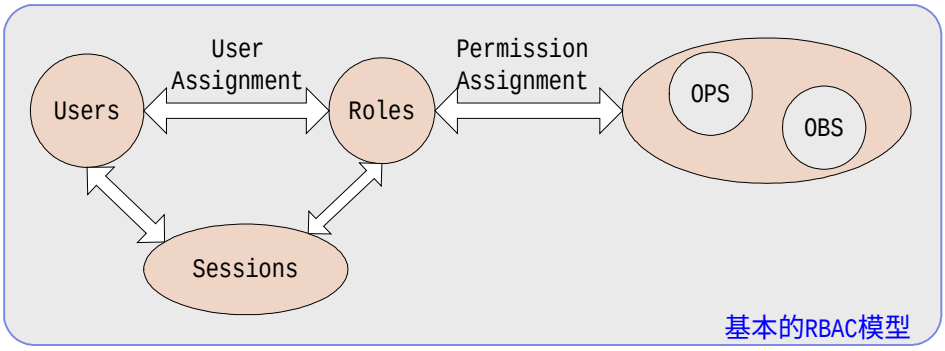
有关 Role-based 的访问控制虽然早在八十年代就被广泛的采纳，但其相关的抽象模型和理论模型架构确较为空缺。直到九十年代中期，Ravi Sandhu 等人才提出了相关抽象模型，并于 2000 提出 RBAC 的标准草案【RAV00】。

请参考 http://csrc.nist.gov/rbac/RBAC-std-draft.doc 文档。

上面的一些 RBAC 参考文档，可能更多的是基于 OS 级别的，而不是基于应用级别的。本篇既然是谈论有关工作流的权限控制模型，当然更多是从应用角度来看待问题。

5.1. Core RBAC

首先来看看最基本也最简单的 RBAC 模型，我们称为“ Core RBAC ”。参看下图在基本的 RBAC 模型中，包含五个基本元素：使用者（Users）、角色（Roles）、许可（permissions）、资源客体（objects（OBS））、操作（operations（OPS））。当然，额外的还包含了两个概念：分配（assigment）和会话（session）。



基本的RBAC模型

5.1.1. User 和 Role

User 和 Role 在此就不多解释了，估计大家都比较理解。但有必要补充一点，用户（User）和角色（Role）是没有等级概念的。在现实生活中，我们可以根据用户信息或者角色来分辨其的级别。比如一个处长（角色），我们可以理解其级别，其比一个科员（角色）要大。但是，对于计算机来说，其并不能理解一个叫“ 处长 ”的角色和一个叫“ 科员 ”的角色，有什么级别的大小。这只能通过对不同角色的授权（授予访问不同资源权限）来控制。当然，如果你非要给角色加一个“ 级别 ”的属性，我也不能说什么。

5.1.2. 许可

许可（permission）则描述了角色对 OP 或 OB 所具有的权限，其反映的是授权的结果。比如授予角色 A 对资源 B 有读的权限，则代表了一个许可的存在，这个许可表示：角色 A 获取了对资源 B 读的许可。

针对 object（OB）来说，其描述的是 permission 和 object 之间的一种关联关系，而这层关系则表示了某一角色对某一资源客体（object）所具有的权限及权限状态；针对 operation（OP）来说，其描述的是 permission 和 operation 之间的一种关联关系，而这层关系则表示了某一角色对某一操作（operation）所具有的权限及权限状态

5.1.3. 分配

分配（assignment）则包含两个方面，用户分配（user assignment）和许可分配（permission assignment）。

用户分配表示的是，将用户或用户组分配给特定的角色。

许可分配表示的是，为角色分配访问许可（访问 object 和 opertiaon 的许可）

5.1.4. 会话（session）

这里的会话（session）和 web 应用中的 session 概念有所不同。这里的 session 表示的是一个用户和多个（或一个）角色之间的关系。从这里可以看出来，一个用户可以被分配给多个角色，说的通俗一点，就是用户 A，属于角色 A，同时也属于角色 B。这会涉及到角色权限叠加问题，将在后面说明。

系统安全的根本目标是数据资料的安全，而数据资料是由它的使用者进行存取和维护的。传统的数据存取和维护，都是以新的数据覆盖旧的数据，对于数据的无心错误，或有心的更改数据，一个管理者并无法有效地查出蛛丝马迹。因此，对数据的存取控制是系统安全的一个重要方面。为此，Sandhu 等学者提出了一套以角色为基础的存取控制（Role-based Access Control，RBAC）理论，其基本组件包括使用者（User）、角色（Role）、授权（Authorization）及会话（Session）。以及权限（Rights），用户组（User Group），资源（Object）

5.2. Core RBAC 扩展

RBAC 还有另外的几种模型，比如 Hierarchal RBAC，Constrained RBAC 等，这些在 Ravi Sandhu 的 RBAC 的标准草案【RAV00】中都有详细的叙述，在此不再重复的说明。因为这些内容就有些超出本篇的主题。

虽然不再谈论 Hierarchal RBAC 这些扩展的模型，但有必要提下有关 Composite Roles 的概念。单从字面就可以理解，其表达的是一个复合角色。

Composite Roles 是由多个角色组成。复合角色的存在，对于解决一个用户同时属于多个角色的问题比较容易。比如原本一个用户 Ua，需要被赋予角色 Ra 和 Rb，这就需要两次分配操作。但如果将 Ua 分配给一个复合角色 CRa，这个 CRa 包含 Ra 和 Rb，则 Ua 就同时具有了角色 Ra 和 Rb 的权限。

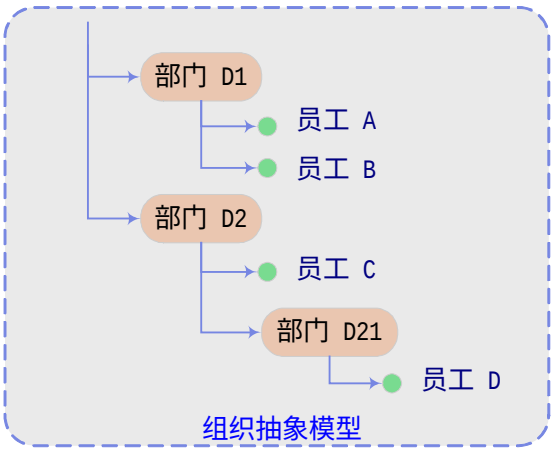
到目前为止，本篇花了大量个段落在说明基本的几种访问控制权限模型，而对于其在工作流中的应用却较少提及。因为只有对这些基本的几个模型有了较为清晰和基本的概念认识，才能够谈论其在工作流系统中的应用。而实际上，如果对这些访问控制模型很清楚了，再谈论工作流上的应用，也就轻松多了。

至此，也算插几句额外话语，同时也对一些有至于工作流方面研究的朋友说几句：工作流涉及到多方面的知识，不单单只是“ 工作流 ”这一个概念。也许换个方式可能会更好理解——多方面的知识会在工作流的应用中体现出来。

从访问控制本身来说，就是一门很深厚的学问。所以谈论工作流的权限控制模型，不如说，如何将访问控制模型，更恰当，更简洁的在工作流系统中应用。

5.3. 组织模型

组织模型是部门和人员的层次关系的一个抽象模形。如下图所示



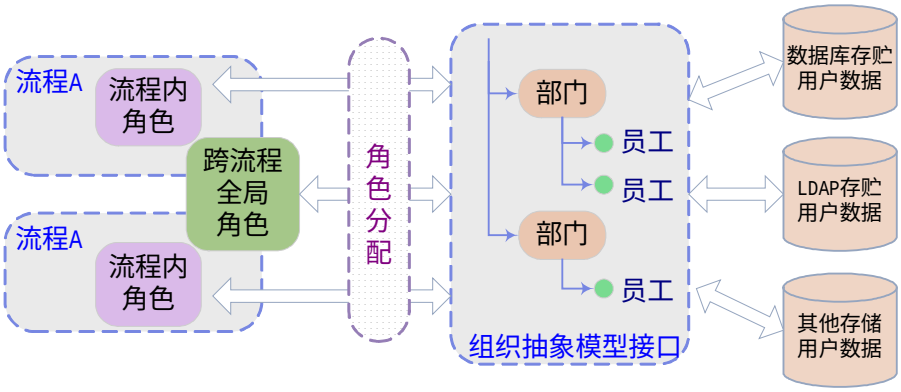
在上面抽象的组织模型图中，我们需要明确几点

(1) 处于同一分支线内的部门是有层次关系的，这个层次可以理解为级别等级。比如，部门 D2 明显要比部门 D21 级别高。

(2) 对于同一部门的员工，其在组织模型中，没有等级分别。对于计算机来说，其均代表一个 User。

(3) 因为部门之间存在层次区别，所以处于同一分支线内的员工，会因为部门层次级别，而相应的存在级别等级。比如员工 C，明显要比员工 D 等级高。但对于员工 A 和员工 D，在没有强制性说明的情况下，一般是没有等级可比性的。

大多组织模型是与工作流平列的，是独立应用的。参看下图，将不同的用户存储数据（数据存储的，LDAP 存储的等等），以统一个组织模型接口对外展现。



可以看出，一个组织模型可以服务于多工作流应用。在引入的跨工作流的全局角色概念后，角色也实现了跨工作流的应用。

5.4. 角色定义

角色本身仅仅只是一个名词，其本身并不能代表权限的大小。比如，我们可以定一个“ Project Manager ”的角色，也可以定一个“ Team Leader ”的角色。对于现实中我们来说，看到这样两个角色，就清楚 PM 的权限要比一个 TL 的权限级别高。但是对计算机来说，这两个角色仅仅是两个“ 词语 ”，是等同的。

当然，你可以采用 Hierarchal RBAC 模式，在角色上实现层次化。也可以采用 Composite Roles 模式，对角色实现一定的分组和复合，以便于权限分配。

大多数工作流产品是仅采取 Core RBAC 中的角色定义模式，通过 permission assignment 来决定各个角色的权限大小。但是 Core RBAC 没有确定角色的层次级别，仅仅描述出来各个角色所应该具有的权限。

5.5. 角色类型

在用户组织模型中，我们提到了“ 跨流程全局角色 ”概念。在这一节，让门来看看具体有多少角色类型。所有的分类，都是为了便于管理和应用，当然角色的分类，也不例外。

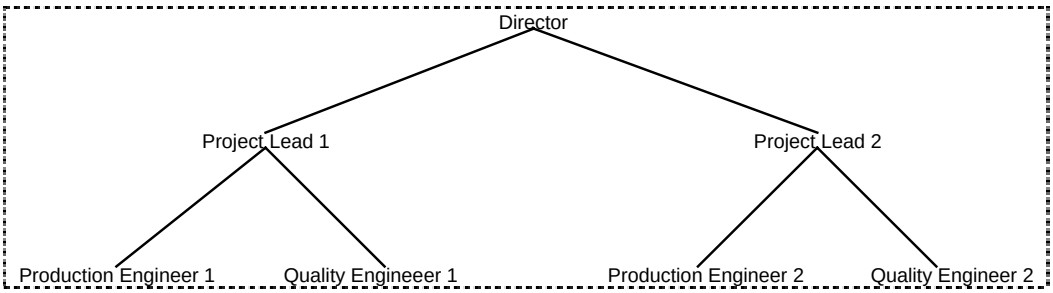
5.5.1. 从形式上分

普通角色

实际上就是一般意义上角色。只是为了与下面的几个角色类型区分，特意加上“ 普通 ”一词。

分等级角色

分等级角色（Hierarchical Roles）可能更加符合现实的需求。参看下图，可以清楚的看到一个 Project Lead 要在 Production Engineer 级别高。



图片来自：【RAV00】

复合角色

在 Core RBAC 扩展这一节中给大家简要介绍了复合角色（Composite Roles）概念。其表示的就是一个角色组。

动态角色

在工作流运转中，未必所有的静态角色都可以很良好的解决所有问题，注意我用的是“ 良好 ”一词。既然工作流中已经存在一些预定的角色，应该可以满足大部分的权限控制之类的问题。但是，预定的所有静态角色都是很不灵活的。

工作流本身是一个动态的产物，为因为其动态的运转而产生一些动态的需求。动态角色的也就是为了在角色这一层，尽量解决好一些动态用户问题。

解决动态的问题，一般都需要一些“ 适当规则 ”来约束，从而使动态问题可以按照一定原则抽象解决。

有必要提一下，动态角色，一般仅用来解决动态用户集问题，而不会为动态用户专门分配权限。流程运转中，属于动态角色中的用户，会依据其所属的静态角色，而获取相应的权限。

5.5.2. 从应用范围上分

流程内角色

> 在流程定义的时候，才会根据流程的中会存在有哪些执行人（执行角色）来的来定义。这些角色仅仅只能在当前流程内使用。

跨流程全局角色

同一个工作流引擎中，可存在多个工作流并行，对于流程内的角色，各个工作流之间时独立的，透明的。但可能存在某些角色，对于所有工作流均是同一个角色，其是在系统级定

义，而非某个特定工作流内部定一。

这样就存在两种全局角色：

第一种：仅仅只是角色名称在全局定义，但不进行 User Assignment 操作。这意味着所有的工作流均包含这一角色。但属于这个角色的用户（组），确必须在各个工作流中重新分配。

第二种：不光角色名称在全局定义，而且进行 User Assignment 操作。这意味着不仅所有的工作流均包含这一角色，而且这个角色中的用户，在各个工作流中都是一样的。这样在各个工作流定义中，就不允许在对全局角色进行重新分配用户操作。

系统内置角色

系统内置角色，大多都是为了方便管理。比如会默认的存在一些角色名称，如：系统管理，流程管理员等等。当然这些角色可就本身在系统初始化的时候，就被赋予一些高级权限；也可能在系统定义或流程定义时候，在赋予一些管理权限。

5.6. 角色分配

参考“ Core RBAC ”中的 User Assignment。

角色分配，就是将用户赋予相应的角色。当然为了操作的方便，工作流 Design 都会采用为角色选取用户的方式。虽然表达的不一样，不过结果总归是一样的：用户都有了属于自己的角色，当然，存在某些用户，其属于两个或多个角色。

5.7. 角色授权

参考“ Core RBAC ”中的 Permission Assignment。

角色授权，就是对角色，授予对特定资源或操作的访问权限。在工作流系统中，这些资源主要针对任务 Task，文档或表单，文档或表单的元素。

在 Core RBAC 中，用到了 Permission 这个名词。这个名词反映了一个结果，而这里所说的“ 授权 ”则更倾向于动作。

5.7.1. 授权对象

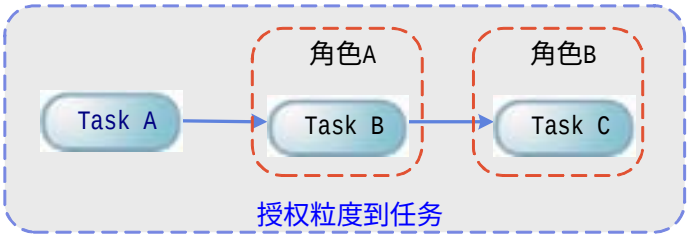
在 Core RBAC 中，就已经说到了两种授权对象：资源客体（objects（OBS））和操作（operations（OPS））

5.7.2. 授权粒度

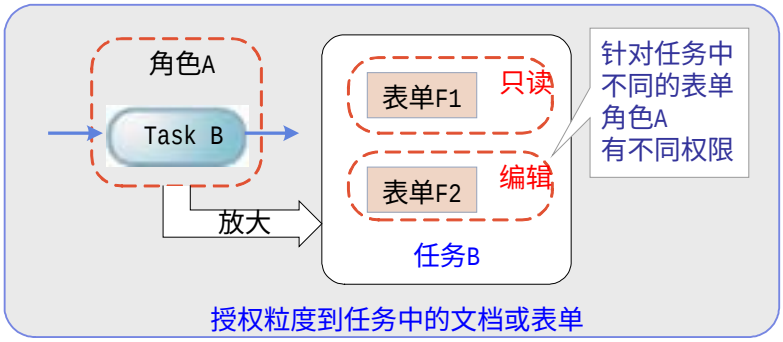
授权粒度表达的是，某一角色，所能够赋予权限的纵深深度。下面显示了工作流中常用

的三种粒度。对于某些工作流管理员的权限（比如最工作流管理，工作流监控）不再此列

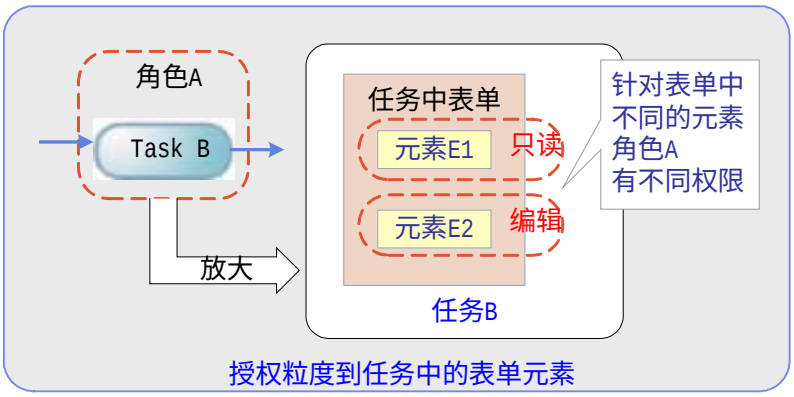
（1）控制到对任务的访问



（2）控制到对文档或表单的访问



（3）控制到对表单中元素的访问



相当一部分工作流产品，仅仅实现了（1）的访问粒度，也就是对工作流中任务（或活动）的访问控制。标准的 XPDL 规范，也仅仅控制到这一层。

对于（2）和（3）实现起来都有一定的难度，主要是因为文档和表单对于某个工作流系统来说是“变动”的。这就大大的增加了 Degine 和 Engine 的复杂度。特别是（3）这种控制方式，据我所知，目前国内没有几家工作流产品做到，其中 RiseSoft 公司的 RiseOffice 产品是实现了“行权限列权限”的控制（即（3）这种方式）。

5.7.3. 角色重叠

在大多数工作流系统中，都会遇到角色重叠的问题。主要表现在：

用户 Ua，即属于角色 Ra，又属于角色 Rb。其中 Ra 对资源 O 享有读权限，Rb 对资源有编辑写权限。那么 Ua 对资源 O 采用什么权限。

大多产品会采用“权限叠加”的方式，这样 Ua 就对资源 O 具有了读写的权限。当然，部分工作流回采用“回避”方式，限定一个用户只属于一个角色，从而回避了角色重叠的问题。

在 Hierarchal RBAC 模型中，还会涉及到“权限继承”方式：高层角色自然而然就继承了低层角色所具有权限。

6. 参考文档

【R.S97】R. K. Thomas，R. S. Sandhu《Task-based Authorization Controls：A Family of Models for Active and Enterprise-oriented Authorization Management》

【ROS96】Roshan Thomas
《Task-based Authorization: A Research Project in Next-generation Active Security Models for Workflows》

【RAV00】Ravi Sandhu,David F. Ferraiolo, Serban Gavrila,《A Proposed Standard for Role-Based Access Control》

【RAV96】Ravi Sandhu,Edward J.Coyne,Hal L.Feinstein,Charles E.Youman《Role-Based Access Control Models》

【RAV97】Ravi Sandhu《Rationale for the RBAC96 Family of Access Control Models》

【RAV97】 Vijayalakshmi Atluri,Wei-Kuang Huang
《An Authorization Model for Workflows》

【SHE02】Shengli Wu

《Authorization and Access Control of Application Data in Workflow Systems》

《安全操作系统研究的发展》石文昌
http://sonata.iscas.ac.cn/wshi/papers/Secure_OS_Overview.pdf

《系统安全的最小特权原则》作者不祥
http://www.yesky.com/SoftChannel/72356682675519488/20031031/1740846.shtml

《安全模型的一种模式语言》Eduardo B. Fernandez 等著，Happy 译，选自：UMLChina
http://www.uml.org.cn/sjms/sjms52.htm

7. 日志

2003-12-9

读了 R. K. Thomas 的《Task-based Authorization Controls：A Family of Models for Active and Enterprise-oriented Authorization Management》受益匪浅。

2003-12-10

看的资料越多，越是汗颜啊。我们不光技术落后国外很多，思想和理论积累，更是差距遥远。实在不想用“遥远”这个词，可是没有办法啊，要走的路，还很长很长。

2003-12-11

这几天都在为 Subject-Object 和 TBAC 烦心。SO 到是很容易理解，可是怎么用恰当的中文翻译 SO 这个词语，实在是费了不少心思。TBAC 以前接触的很少，没想到已经有十几年的发展历史了。基于任务的授权控制模型写了几行字，发现思绪还是很乱，于是决定放一放，先写基于角色的。

2003-12-12

完成了 Core RBAC 的介绍。不过这些内容大多是对【RAV96】和【RAV97】的理解。

2003-12-13

忙活了一天，总算写完了基于角色的访问控制模型。

2003-12-14

上午，回头补写了基于任务的授权控制模型。总觉得要写的东东很多，也很杂。于是就决定以简单点为主。

下午对全文进行了修改，补上了引文。总算结束了。