摘要

传统的工作流安全模型或者忽视了安全控管的便利性，或者没有对任务执行权限做出有效控制，它们都不能很好地满足现代工作流管理系统的需要。有文献将任务概念引入基于角色的访问控制模型以期解决上述安全问题，但是，却没有改变基于角色访问控制模型的三层访问控制结构(用户－角色－权限)，从而导致了最小权限约束假象、数据冗余、动态性差等安全问题。另外，传统工作流安全模型没有论及对执行任务时所需要的资源访问权限的控制，从而使得用户可以绕过任务获得访问权限。

本文提出了一个新的访问控制模型，较好地解决了上述安全问题。本文首先介绍工作流技术及传统访问控制技术，然后分析了传统工作流安全模型的缺陷，在此基础上提出了一种基于任务和角色的访问控制模型。该模型包含 4 个形式化模型，特点是在基于角色的访问控制模型的三层访问控制结构的基础上明确引入了任务的概念，构造了 4 层访问控制结构(用户－角色－任务－权限)，对系统中的任务进行了较好的访问控制，有效克服了传统工作流安全模型的缺陷。该模型引入了任务分配策略、任务指派方式、任务层次、私有任务、双重验证、按级别授权等概念。这些概念使管理员的安全控管工作更加灵活、简便。

本文所提出的新模型已成功地应用于我们自主研发的分布式工作流基础件系统中。由于新模型具有自然的任务一级的最小权限约束粒度，引入了任务分配策略、任务层次等概念，所以，克服了数据冗余、动态适应性差的缺点，与传统工作流安全模型相比，采用新的访问控制模型可以提高工作流系统的安全性、灵活性和实用性。

关键词：工作流管理系统，访问控制，角色，任务，任务层次，任务上下文，任务分配策略

ABSTRACT

Traditional workflow security models often ignore the convenience of security management and can not effectively control the accessing permissions of a task. Therefore, they can not suitably meet the requirements of modern Workflow Management Systems. Although the concept of task has been introduced into Role Based Access Control models to solve the security issues in some articles, it does not change the 3-layer structure (user-role-permission) of traditional RBAC models, which results in many security problems, such as the gloss of minimal permissions constraint, data redundancy and lack of dynamic ability. Moreover, the traditional workflow security models doesn't mention the control over the accessing permissions required to execute tasks, thus making it possible for users to acquire the permissions without task assignment.

This paper presents a new access control model to solve the issues as we mentioned above. Firstly, the workflow technology and traditional access control technologies are introduced, and then the main disadvantages of traditional workflow security models are analyzed. Accordingly a series of Task and Role Based Access Control models are elaborately designed, then a model family is established which contains 4 formal models. The remarkable feature of this new model is that it imports the concept of task into traditional RBAC model and sets up reasonable 4-layer architecture (user-role-task-permission) to solve issues in traditional workflow security. The concept of level enhances the security of the model, and the introduction of some concepts makes security management jobs more convenient and more flexible such as context constraint, task assigning policy, task hierarchies and private task.

This new model has been successfully implemented in our own distributed workflow infrastructure system. Comparing to traditional workflow security model, the new model has increased the security, flexibility and practicability of our workflow system.

Key Words: WfMS, Access Control, Role, Task, Task Hierarchy, Task Context, Task Assigning Policy

独 创 性 声 明

本人声明所呈交的学位论文是我本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表和撰写过的研究成果，也不包含为获得国防科学技术大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文题目：工作流系统中的安全技术研究

学位论文作者签名：付松涛　　日期：2003 年 11 月 20 日

学位论文版权使用授权书

本人完全了解国防科学技术大学有关保留、使用学位论文的规定。本人授权国防科学技术大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档，允许论文被查阅和借阅；可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密学位论文在解密后适用本授权书。）

学位论文题目：工作流系统中的安全技术研究

学位论文作者签名：付松涛　　日期：2003 年 11 月 20 日

作者指导教师签名：谭庆平　　日期：2003 年 11 月 28 日

图目录

第一章 绪论

§1.1 课题背景

随着时代的发展，工作流技术引起越来越多企业的兴趣，工作流产品的市场和应用领域也迅速扩大。企业在应用工作流系统的同时也异常关注其安全性，但是工作流系统的安全问题却一直没有得到很好的解决。本文作者所在的课题组承担的 863 项目（面向金融领域的分布式工作流关键技术及应用框架）也同样对工作流系统中的安全性有很高的要求。作者在这种背景下提出了一种基于任务和角色的访问控制模型，克服了传统工作流安全模型中的诸多缺陷，并将其应用于课题组自主研发的工作流基础件系统中。实践证明，该模型能很好地适用于工作流系统。

1.1.1 工作流系统及其安全需求简介

工作流的概念起源于生产组织和办公自动化领域[1]。它是针对日常工作中具有固定流程的一系列活动而提出的一个概念。其目的是通过特定的工作流建模机制将日常工作中的固定流程抽象出来，然后再按照一定的规则对这个抽象出来的工作流程模板进行管理监控。工作流管理系统（Workflow Management System，WfMS）使流程尽可能的自动化执行，只有需要人工处理的地方才推给相应的人员处理，这样就达到了提高工作效率、降低生产成本的目的，从而使得企业更具有竞争力。

20 世纪 90 年代初工作流管理联盟（Workflow Management Coalition，WfMC）的成立标志着工作流技术开始进入相对成熟的阶段。WfMC 为工作流技术制定了一系列规范。根据 WfMC 的定义[2]，工作流管理系统是一个支持流程定义、工作流建立、管理和监视的工具集。这些工具集非常适用于企业重组、业务流程自动化，以及跟踪人工型和自动型业务流程，其应用已经扩展到医疗业、教育业、电信业、制造业、金融业、银行业和办公自动化等诸多领域中，大大提高了企业工作流程执行的效率[3]。

WfMC 制定的工作流安全规范[4]定义了工作流系统中的认证、授权、访问控制、数据保密性、数据完整性、可用性、匿名等概念，对工作流系统应该具有的安全特性进行了统一约定。

- ✓ 认证（Authentication）是系统对用户身份进行辨别以证明其为合法用户的过程。
- ✓ 授权（Authorization）是系统将特定权限授予某个合法用户的过程，授权必须在认证之后进行，否则授权就没有意义。
- ✓ 访问控制（Access Control）是系统对用户身份进行认证，再将其访问请求与用户授权信息进行比较以决定是否允许用户的访问请求的过程。

- ✓ 数据保密性（Data Privacy）是指数据不能被非授权泄漏。
- ✓ 数据完整性（Data Integrity）是指不能对数据进行非授权修改。
- ✓ 可用性（Availability）是指，数据对执行工作流任务的主体来说是可用的。
- ✓ 防否认（Non Repudiation）是指系统必须防止用户否认其做过的工作。
- ✓ 多级安全（multi－level security）是指在具有多个安全级别的应用领域中，要保证各级别之间的信息能无任何安全隐患地自然流通。
- ✓ 匿名（Anonymity）是指执行任务的主体（通常指用户）相互之间不知道彼此做了哪些任务。
- ✓ 审计（Audit）负责记录用户或系统在什么时间执行何种动作的信息，以备事后对可疑行为进行审查。

除访问控制外，其它安全目标的实现与是否是在工作流系统中实现关系不大，而且它们都已有许多成熟的安全模型可供使用。另一方面，工作流管理系统是一个主要应用于企业的业务自动化管理系统。整个企业的用户数量以及任务数量都比较大，企业中的业务以及各种规定（约束）千差万别且随时间而动态变化。工作流系统主要是对企业业务的处理进行尽可能的自动化管理，因此该系统非常强调任务这个概念，处处以任务为中心：任务建模、任务分配、任务调度、任务监控、任务日志等等。它的最大特点就是将总的工作任务分成更小的任务项由多人分工协作完成[5,6]，这就需要很好的访问控制机制来对这些协作人员的访问进行管理控制，既要保证不让用户执行未授权的任务，又要保证让授权用户能顺利地执行已经授权了的任务。如果系统对这些协作人员的安全防护不够，不可避免的就会出现一些人员利用工作之便进行非法操作的可能。所以，在上述安全问题中，访问控制是最重要、最困难的安全问题。

1.1.2 访问控制方法的发展历史和现状

传统的访问控制方法主要有四种：自主访问控制（Discretionary Access Control，DAC）、强制访问控制（Mandatory Access Control，MAC）、基于角色的访问控制(Role Based Access Control，RBAC)和基于任务的访问控制（Task Based Access Control，TBAC）。

自主访问控制早在20世纪60年代末就已经出现在分时系统中了。所谓的自主访问控制又叫基于主人的访问控制，客体的主人全权管理有关该客体的访问授权，有权泄漏、修改该客体的有关信息，而且主体间存在权限的转移。

后来在20世纪70年代又出现了强制访问控制机制，每个主体都有既定的安全属性，每个客体也都有既定的安全属性，主体对客体是否能执行特定的操作取决于二者安全属性之间的关系。强制访问控制中规定了两个著名的安全特性：Simple Security Condition和*-Property (Star Property)。前者定义为，主体对客体有读权限，当且仅当主体的安全等级大于或等于该客体的安全等级；后者定义为，主体对客体有写权限，当且仅当主体的安全等级小于或等于该客体的安全等级。上述两个特性保证了信息的单向流动，即信

息只能向高安全属性的方向流动。强制访问控制机制就是通过信息的单向流动来防止信息的扩散。

基于角色的访问控制机制[7-11]的概念早在20世纪70年代就已经提出，但在相当长的时间内没有得到深入的研究。进入20世纪90年代后RBAC在安全需求的推动下引起了人们的普遍关注。基于角色的访问控制主要是为了方便安全控管人员的管理工作而提出来的，该模型引入了角色这个概念，将传统的主体、客体直接授权改为通过角色来授权（三层访问控制结构：用户－角色－权限），角色代表若干权限的集合，可以进行批量授权，从而达到简化权限授予的目的。由于其方便性和实用性，RBAC为目前大多数系统（尤其是企业系统）所采用。

随着计算机应用技术的发展，出现了一些任务特征很明显的应用系统，这些系统主要强调对任务的访问控制，因此，人们又提出了一种基于任务的访问控制模型。该模型通过任务与其执行者之间的映射关系来对任务进行访问控制。

1.1.3 工作流安全模型现状

以往在开发工作流系统的时候，大多数设计者主要关注的是工作流引擎的设计和实现，对于工作流系统中安全问题的研究比较少。目前，比较常见的工作流安全模型主要采用两种安全机制：基于任务的访问控制和基于角色的访问控制。

基于任务的工作流安全模型能够解决工作流系统中任务的访问控制问题[12]，但其缺陷在于，它在关注工作流系统中的任务特性的同时忽视了安全控管的便利性，使得整个模型在真正实施时异常复杂。因此，这个模型只适用于集中式、小规模的工作流系统。基于角色的访问控制模型[13-19]的主要优势在于其安全控管的便利性，这一点能解决现代工作流系统由于规模大而带来的问题，但是由于其固有的三层访问控制结构（用户－角色－权限）而没有对任务执行权限做出有效控制，所以它不能很好地适用于分布式环境下任务特征很强的工作流管理系统。传统的访问控制模型不再能满足现代工作流管理系统的需要。

所谓最小权限约束是指用户执行任务时只能拥有执行该任务所必需的权限，不能拥有必需权限外的其它任何权限。2002年，Savith Kandala和Ravi Sandhu[15]提出了一种工作流安全模型，这个模型直接将RBAC引入工作流系统，同时引入了任务的概念。但是，该模型在引入任务概念的同时却没有改变RBAC三层访问控制结构的本质，导致了最小权限约束假象、数据冗余、动态性差等安全问题。另外，基于角色的工作流安全模型没有论及对执行任务所需要的资源访问权限的控制，使得用户可以绕过任务获得访问权限，从而产生安全隐患。对该模型的缺陷分析详见4.1节。

国内研究人员也提出了一些工作流安全模型，但他们大都只是将RBAC模型直接用于工作流[20-24]，或对其时间特性进行约束[25,26]，或对其角色授权进行约束[27]。这些都没有改变RBAC模型固有的三层访问控制结构，没有对任务执行权限做出较好的访问控制，因此不能很好地适应当代工作流管理系统的安全需要。

§1.2 本文主要工作

在硕士课题期间，本文作者主要致力于如下工作：

1. 研究工作流管理系统并找出其安全问题的症结所在；
2. 研究传统的工作流安全模型；
3. 提出了一种新的工作流安全模型，克服了传统工作流安全模型的诸多缺陷，并对其进行形式化建模；
4. 实现这种新的工作流安全模型，并将其应用于本文作者所在的课题组自主研发的工作流基础件系统中。

§1.3 论文组织结构

本文共分 5 部分。

第 1 部分为第一章，概述本文工作。

第 2 部分为第二、三章，主要介绍工作流管理系统的基本概念（第二章）和传统的访问控制的基本概念（第三章）。

第 3 部分为第四章，它是本文的主体，详细讨论了传统访问控制的缺陷，提出了新的访问控制模型，并给出了其形式化模型。

第 4 部分为第五章，这一章首先简单介绍工作流基础件系统及其安全需求，然后描述新的访问控制模型在该系统中的应用。

第 5 部分为结束语，总结全文并探讨进一步的工作。

§1.4 本文的研究成果

本文对工作流系统中的安全技术进行研究，在传统的工作流安全模型的基础上提出了基于任务和角色的访问控制模型，并为该模型建立了一个形式化模型族。该模型引入按级别授权、双重验证等概念增强了模型的安全性，且首次引入了任务分配策略、任务指派方式、任务层次、私有任务等概念。这些概念使管理员的安全控管工作更加灵活、简便。本文作者已将基于任务和角色的访问控制模型成功地应用到作者所在课题组自主研发的分布式工作流基础件系统中。实践表明，本文所提出的新模型能更好地适用于分布式工作流系统。

在课题研究过程中，本人以第一作者身份在《国防科技大学学报》和《计算机应用》上各发表学术论文一篇（见本文附录）。

第二章 工作流管理技术基础

限于篇幅，这一章只是简要介绍工作流技术中的一些基本概念和其参考模型。详细的技术细节请参阅参考文献[1]以及工作流管理联盟的相关技术标准[2]。

§2.1 工作流导论

工作流的概念起源于生产组织和办公自动化领域。它是针对日常工作中具有固定流程的一系列活动而提出的一个概念。其目的是通过特定的工作流建模机制将日常工作中的固定流程抽象出来，然后再按照一定的规则对这个抽象出来的工作流程模板进行管理和监控。工作流使流程尽可能地自动化执行，只有需要人工处理的地方才推给相应的人员处理，这样就达到了提高工作效率、降低生产成本、提高企业竞争力的目的。

工作流管理联盟给出的工作流定义[2]为：全部或者部分由计算机支持或自动处理的业务流程。在业务流程中，文档、信息或者任务按照定义好的规则在参与者间传递，从而完成整个业务目标。

工作流管理系统（Workflow Management System，WfMS）通过管理工作活动序列，调用与各种活动步骤相关的人员、IT 资源，对业务流程提供自动化处理。其定义[2]为：工作流管理系统是这样的一个系统，它详细定义、管理工作流程，并通过运行一些软件来执行这些工作流程，这些软件的执行顺序（也即工作流程的内部执行顺序）由工作流逻辑的计算机表示形式（计算机化的业务规则——流程定义）驱动。

工作流管理系统可以给企业带来如下好处[1]：

- ✓ 提高企业管理的规范化程度；
- ✓ 更好地与上下游企业形成快速响应市场的供应链网络；
- ✓ 提高企业生产效率，缩短产品上市周期；
- ✓ 降低管理成本，提高工作质量；
- ✓ 使业务过程重组的实施更加方便。通过对已经完成的工作流实例的分析，找出存在的不足，进而不断改进工作流程。

§2.2 工作流参考模型

这一节介绍工作流管理联盟制定的工作流参考模型规范[2]，包括工作流系统的特征、通用模型以及组件和接口模型等。

2.2.1 工作流系统的特征

尽管实现的方法多种多样，但所有的 WfMS 都表现出某种共同的特征，这为不同产

品间的集成和协同工作奠定了基础。参考模型描述了一个工作流系统实现的公共模型，并且指出各种不同实现方法之间的联系。

在最高层，所有的 WfMS 都具有相同的特征，即它们必须对下面 3 个功能提供支持[2]：

- ✓ 建立时期（Build-time）功能：定义工作流流程及其组成活动；
- ✓ 运行时期（Run-time）控制功能：在运行环境中管理工作流流程；
- ✓ 运行时期（Run-time）交互功能：在运行时期与用户或外部应用程序交互来完成各种活动。

图 2.1 描述 WfMS 的基本特征以及上述功能间的关系。

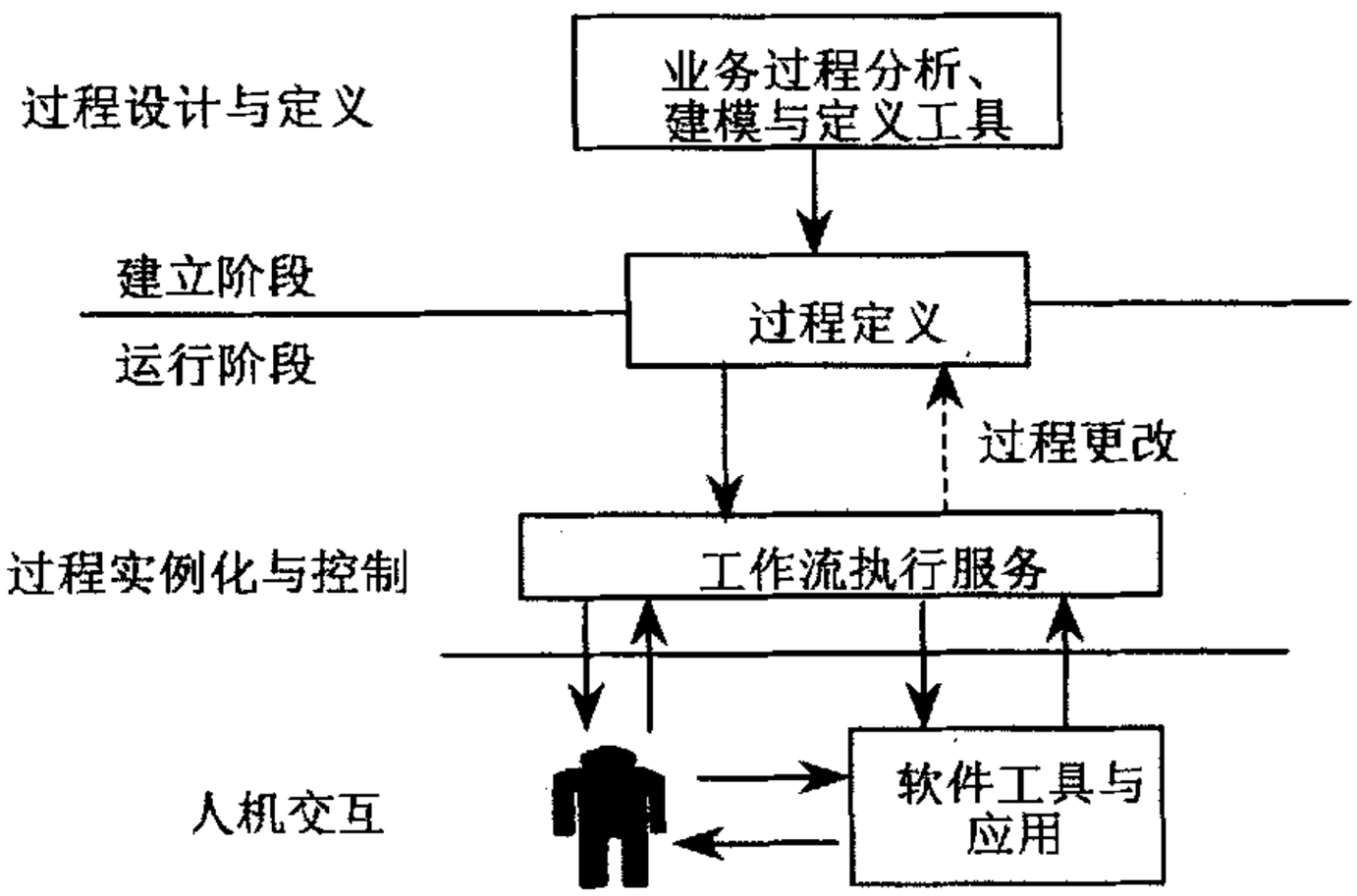

图 2.1　工作流系统特征

建立时期的功能负责产生业务流程的计算机化定义。在这个阶段，通过使用一个或多个分析、建模和系统定义工具，把实际的业务流程转变成形式的、计算机可以处理的定义。通常把定义的结果称为流程模型、流程模板、流程元数据或者流程定义。

在运行时期，流程定义由负责创建和控制流程实例的软件解释，这个软件还负责安排流程中各个活动的执行时间，调用适当的人员、外部应用程序等。这些运行时期的控制功能把流程定义中描述的流程与现实中的实际流程联系起来。实现运行时期控制功能的核心组件是基本的工作流管理控制软件（通常称为引擎）。引擎负责流程的创建与删除，控制流程中活动的执行时间安排，以及与人和应用程序资源进行交互。

典型地，工作流中的活动与人员的操作有关，这样的活动需要人机交互才能完成。这里的人机交互通常是指人为地进行某些信息处理操作（如填写表单、更改数据等）。工作流中的活动通常还与外部的应用程序资源有关，引擎通过与特殊的外部应用程序交互来处理信息。制定标准的框架来支持这些交互有如下好处：在多工作流系统中使用一致的接口，有利于系统间的协同；可以开发与不同工作流产品协同工作的通用应用工具。

2.2.2 工作流通用模型

尽管市场上的工作流产品是各种各样的，但是仍然可以构建一个通用的工作流系统实现模型，这个模型可以适用于市场上的大多数产品，为开发协同工作的工作流系统奠定了基础。

通用模型有 3 种类型的组件[2]：

- ✓ 为工作流系统的各种功能提供支持的软件组件；
- ✓ 被一个或多个软件组件使用的各种类型的系统定义数据和控制数据；
- ✓ 应用程序和应用程序数据库：虽然不是工作流产品的一部分，但是它们会被工作流产品调用，因而作为整个工作流系统的一部分。

通用工作流系统的主要功能组件如图 2.2 所示。

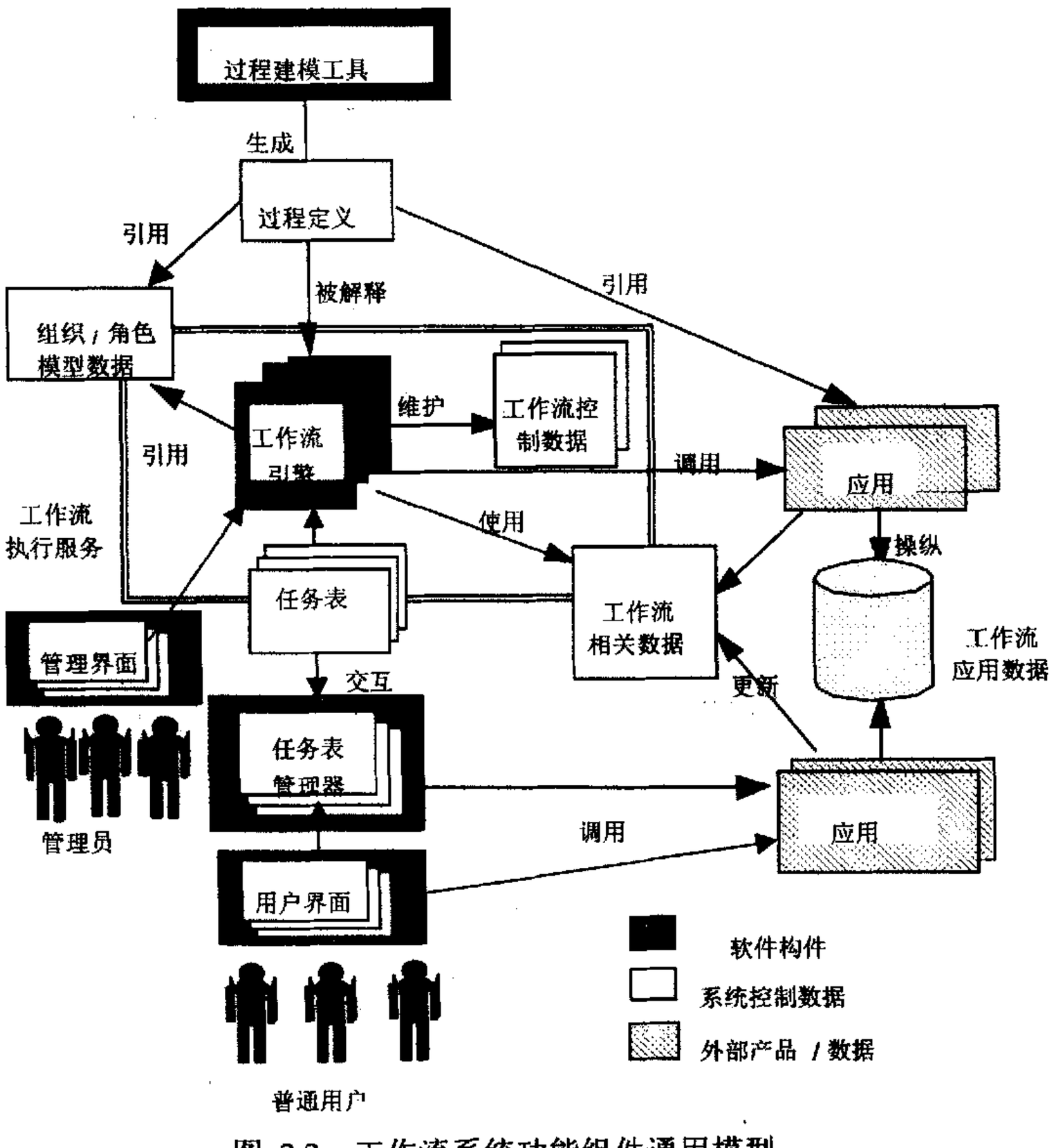

图 2.2 工作流系统功能组件通用模型

2.2.3 工作流组件与接口模型

工作流参考模型来源于对普通工作流程序结构的分析。结构中的标准接口可以使不同产品在不同的结构层次上协同工作。所有工作流系统都包含一系列的公共组件，组件间采用一套预定义的方法进行协作。

图 2.3 描述了工作流系统框架结构中的主要组件与接口[2]。

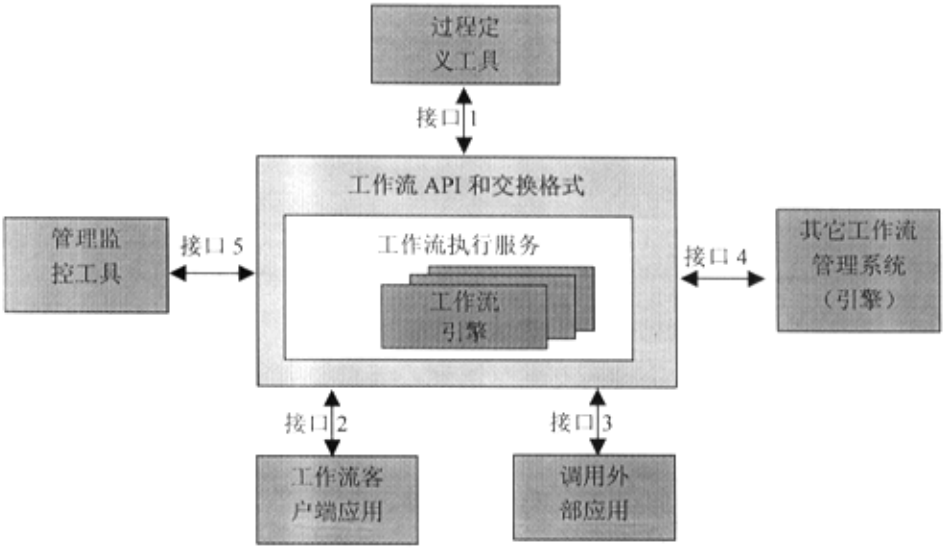

图 2.3 工作流参考模型－组件与接口

工作流执行服务器周围的接口称为 WAPI（Workflow APIs）。通过这些接口可以访问工作流系统的服务。这些接口还控制工作流控制软件与其他系统组件间的交互。在这 5 个接口中的许多功能，都是被 2 个或更多个接口同时拥有的，因此 WAPI 可以看作是统一的服务接口，系统开发者可以共同使用这 5 个接口来支持工作流管理功能，而不是单独地使用其中某个接口。

工作流执行服务器（Workflow Enactment Server）是由一个或多个工作流引擎构成的软件服务器，用来创建、管理、执行工作流实例。应用程序可以通过 WAPI 来与这个服务器交互。

引擎可以通过客户端应用程序接口（Client Application Interface）与任务表处理器（一种可与终端用户和引擎交互的软件实体，主要用于处理任务表）交互。终端用户负责从任务表中选择、推进任务项并控制任务项的具体执行。

应用程序调用接口（The Invoked Application Interface）允许引擎直接激活应用程序来执行一个活动。典型地，引擎调用以后台服务为主的应用程序。

客户端应用程序接口和应用程序调用接口可以相互调用。比如，当执行活动要用到的应用程序需要与终端用户交互时，通常是使用客户端应用程序接口通过应用程序调用

接口来调用这个应用程序。

在工作流引擎间需要一个标准的交换格式来实现异种产品间的调用。使用接口 4，执行服务器可以把活动或者子流程转移到另外一个执行服务器中执行。在工作流参考模型中，这被称作“工作流引擎交互（Workflow Engine Interchange）”。

WfMC 给出的参考模型规范还为管理和监视功能定义了公共的接口标准。这样，一个开发商的产品就可以用来管理其他开发商的工作流引擎的运行。通过公共的接口，多个不同的工作流执行服务器可以共享管理和监视功能。

2.2.4 工作流引擎运行示例

由于流程定义是工作流管理系统的基础环节，所以，流程定义机制的好坏直接影响到工作流管理系统性能的好坏。通常的流程定义单元包括：包、模板、任务以及任务之间的转换边。包是流程定义的最大单元，包通常是某些相关业务流程的集合，包含若干模板。模板是对企业某一业务的流程定义结果。模板中包含若干任务和任务之间的转换。

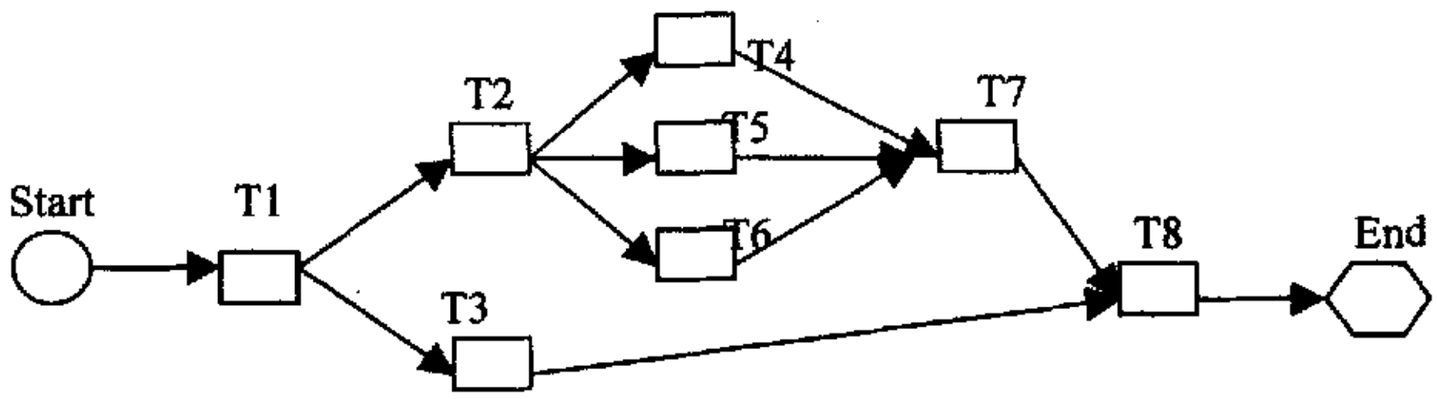

图 2.4　流程定义结果示例

一种典型的流程定义如图 2.4 所示。要想使这个流程真正运作起来，必须先将这个流程定义传给引擎，引擎解析出这个流程之后，找到其起始节点 Start，进行初始化工作。初始化工作完成之后找到第一个节点 T1，并对其实例化，生成任务实例。然后又分两种情况：如果该节点是自动节点，那么由引擎自动调用相应的内部功能模块或外部应用程序来完成这个实例；如果是手工任务节点，则找到这个任务实例的执行者，将该任务实例推给这个用户，当用户做完这个任务实例时，将结果提交给引擎。最后引擎根据已解析出来的流程，找到相应的下一个任务节点，再实例化，再执行。这样依次执行下去。直到 End 节点结束。

第三章 传统访问控制

所谓访问控制，就是通过某种途径显式地准许或限制用户访问的能力及范围，从而限制对敏感资源的访问，防止非法用户的侵入，避免合法用户因操作不慎而造成破坏。目前，许多敏感的信息都是通过计算机来控制和管理的。如何确保这些信息不被人窃取和破坏，即如何使它们安全，是当今计算机技术的研究热点。ISO（国际标准化组织）在网络安全标准（ISO7498-2）中定义了 5 种安全服务（身份认证服务、访问控制服务、数据保密服务、数据完整性服务、不可否认服务）。访问控制服务是其中的一个重要组成部分，它以身份认证服务为基础，是实现数据保密性服务和数据完整性服务的主要手段。访问控制根据系统中已有的授权规则，对合法用户的访问请求进行判断，以决定是否允许用户享有相应的访问权限。访问控制作为提供信息安全保障的主要手段被广泛地应用于防火墙、文件访问、VPN 及物理安全等多个方面。

最简单的访问控制可通过加密方式来实现。也即，对每个客体进行加密，加密之后，只有持有相应密钥的主体才能对该客体进行访问。但是，这种方式有其局限性，主要是加密只能对数据进行加密，而访问控制中的客体可以是一个进程、一个服务等。在这种情况下，加密就无能为力了。加密通常只是作为自主访问控制、强制访问控制等访问控制中的一种辅助手段。

下面分别介绍几种传统的访问控制模型。

§3.1 自主访问控制

自主访问控制（Discretionary Access Control，DAC）最早出现在 20 世纪 60 年代末的分时系统中。自主访问控制是一种最为普遍的访问控制手段，它允许客体的属主制定针对该客体的保护策略。DAC 可以限定哪些主体针对哪些客体可以执行什么操作，这样，就可以灵活地对访问控制策略进行调整。

自主访问控制中，主体可以针对被保护客体制定自己的保护策略。

- ✓ 每个主体拥有一个主体名并属于一个组或具有一个角色；
- ✓ 每个客体拥有一个限定主体对其访问的授权列表；
- ✓ 每次访问发生时都会基于授权列表检查主体是否具有对客体的访问权限。

自主访问控制机制易于扩展和理解。大多数系统仅基于自主访问控制机制来实现访问控制，如主流操作系统(Windows NT Server、 UNIX 系统等)、防火墙等。

客体的主人（主体）全权管理有关该客体的访问授权，有权泄漏、修改该客体的有关信息。所以，自主访问控制又被称为基于主人的访问控制。

DAC 主要由主体一客体矩阵（通常称为访问控制矩阵，ACM）来实现。该访问控

制矩阵中的元素表示主体对相应客体所拥有的访问权限。该实现方式分两类：

- 基于主体的 DAC 实现

  也即权力表的实现方式，表明主体对所有客体的权限。当删除一个客体时，要检查所有主体的权力表。

- 基于客体的 DAC 实现

  也即访问控制表的实现方式，表明客体对所有主体的权限。当删除一个主体时，要检查所有客体的访问控制表。

DAC 具有简单、易用的优点，而且在一定程度上实现了多用户环境下的资源保护。但是，DAC 也有难以克服的缺陷。一方面，由于客体拥有者可以随意更改客体的访问授权，所以客体拥有者可以随意将客体访问权限授予非法主体，从而产生安全隐患。另一方面，由于系统中的客体拥有者往往比较分散且难以统一管理，导致 DAC 资源管理过于分散，从而使得只应用 DAC 机制的系统的安全难以得到有力保证。

DAC 的一种改进方法是让安全管理员来限制访问权限随意扩散，这样就形成了一种半自主式的资源管理方案。这种方案在一定程度上缓解了 DAC 资源管理过于分散带来的安全问题，但是它的本质还是一种 DAC 机制，客体拥有者还是有可能将客体访问权限授予非法主体。

§3.2 强制访问控制

强制访问控制（Mandatory Access Control，MAC）最早出现在 20 世纪 70 年代，在 20 世纪 80 年代得到普遍应用。强制访问控制是指主体（用户、进程等）与客体都有一个既定的安全属性，主体对客体是否能执行特定的操作取决于二者安全属性之间的关系。安全属性是强制性的规定，它是由安全管理员或者操作系统根据限定的规则确定的，不能被其他任何主体修改。如果系统认为具有某一个安全属性的主体不适于访问某个客体，那么任何主体（包括客体的拥有者）都无法使该主体具有访问这个客体的权限。

MAC 用于保护系统确定的客体，主体不能对此客体进行更改。也就是说，系统独立于主体行为强制执行访问控制，主体不能改变自身或客体的安全属性。这样的访问控制规则通常对主体和客体按照安全等级划分标签，访问控制机制通过比较安全标签来确定授予还是拒绝主体对客体的访问。强制访问控制进行了很强的等级划分，所以经常用于军事领域。

在强制访问控制系统中，所有主体和客体都被分配了安全标签，安全标签标识一个安全等级。

- ✓ 主体被分配一个既定的安全标签；
- ✓ 客体也被分配一个既定的安全标签；
- ✓ 执行访问控制时对主体和客体的安全标签进行比较。

DAC 有两个重要的安全特性：

- Simple Security：主体读客体，当且仅当主体的安全等级大于或等于该客体的安

全等级。

- *-Property (Star Property)：一个主体要写一个客体，当且仅当主体的安全等级小于或等于该客体安全等级。

上述两个特性保证了信息的单向流动（下读上写），即信息只能向高安全属性的方向流动。MAC 就是通过信息的单向流动来防止信息的扩散。

MAC 具有安全性强的优点，但是，正是由于它的这种强安全特性，使得它不够灵活，应用面窄，一般只用于军事等具有明显等级观念的领域。

MAC 的一种改进为“中国墙”模型（Chinese Wall Model，CWM）[28]。CWM 将客体分为若干冲突域（conflicting domains），每个冲突域内部的各个客体都是相互冲突的（也即客体数据是相互保密的）。对每一个冲突域而言，任意主体至多只能访问其中的一个客体。

强制访问控制和自主访问控制有时会结合使用。例如，系统可能首先执行强制访问控制来检查用户是否有权访问一个文件组（这种保护是强制的，也就是说，这些策略不能被用户更改），然后再针对该组中的各个文件制定相关的访问控制列表（自主访问控制策略）。

§3.3 基于角色的访问控制

在访问控制方面，DAC 显得太弱，MAC 显得太强，而且它们的配置及计算量都过于复杂。针对这些缺点，有人提出了一种新的访问控制模型，这就是基于角色的访问控制模型（Role Based Access Control，RBAC）。

基于角色的访问控制这个概念早在 20 世纪 70 年代人们在做多用户、多应用在线系统（multi-user and multi-application on-line systems）时就提出来了，但是一直没有得到更多的关注。角色是访问控制策略形式化之后的一种语义结构，它是人员集合和权限集合的连接点，该连接点使得与某角色关联的人员集合中的任意一个人员都享有与该角色关联的权限集合中的任意一个权限。在访问控制中引入角色这个概念能大大简化管理员的访问控制管理工作。因为，对于一个企业来说，活动或业务相对固定（也即权限相对固定），所以，先将企业内某一组相关人员的权限集中起来放在一个集合中，这个集合就称之为角色，然后再将角色指派给相应人员即达到授权的目的。角色在人员和权限之间的授权关系上担当桥梁的作用，只有具有角色的人员才拥有与角色相应的权限。由于角色相对固定，所以，一旦企业内的 RBAC 模型建立起来之后，管理员所做的大部分工作都仅仅是维护人员与角色的指派关系。

角色与以往访问控制系统中的用户组的区别在于，用户组只是用户的集合，而角色则是用户集合与权限集合之间的连接点，如图 3.1 所示。

20 世纪 90 年代，Ravi S Sandhu 提出了一个基于角色的访问控制模型。下面对该模型进行详细介绍。

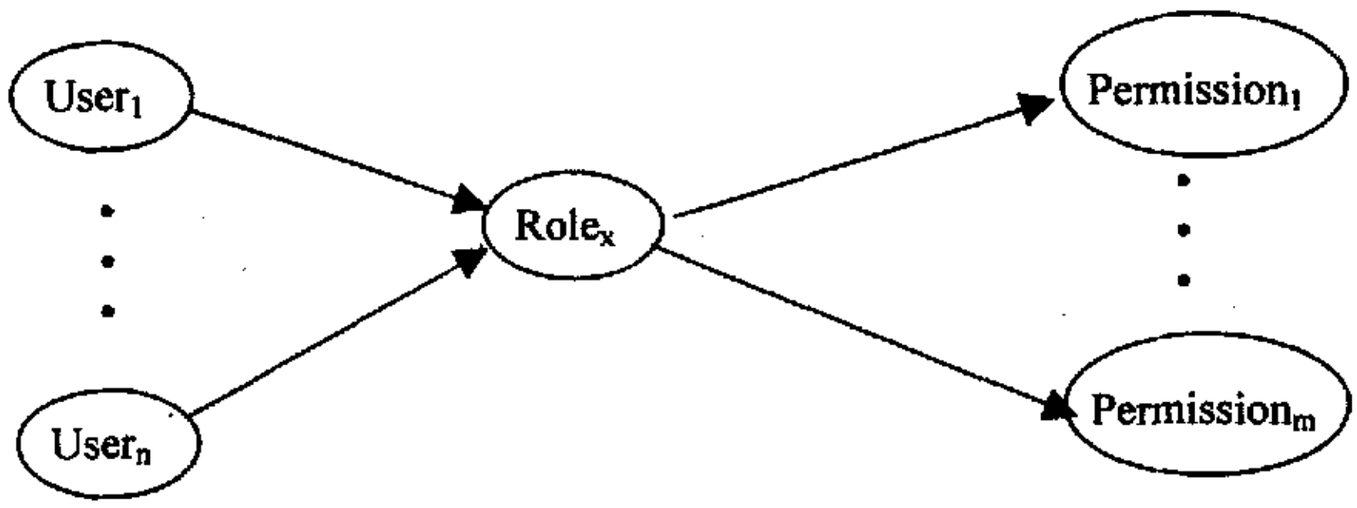

图 3.1 RBAC 中角色的关系示意图

Sandhu 将 RBAC 模型分为四个子模型，分别叫 $RBAC_0$、$RBAC_1$、$RBAC_2$、$RBAC_3$，这四个模型统称为 RBAC96 模型。其中，$RBAC_0$ 是基本模型，$RBAC_1$ 在此基础上引入了角色层次的概念，$RBAC_2$ 在 $RBAC_0$ 的基础上引入了一些约束，而 $RBAC_3$ 则是前三种模型的有机结合。它们之间的关系如图 3.2 所示。

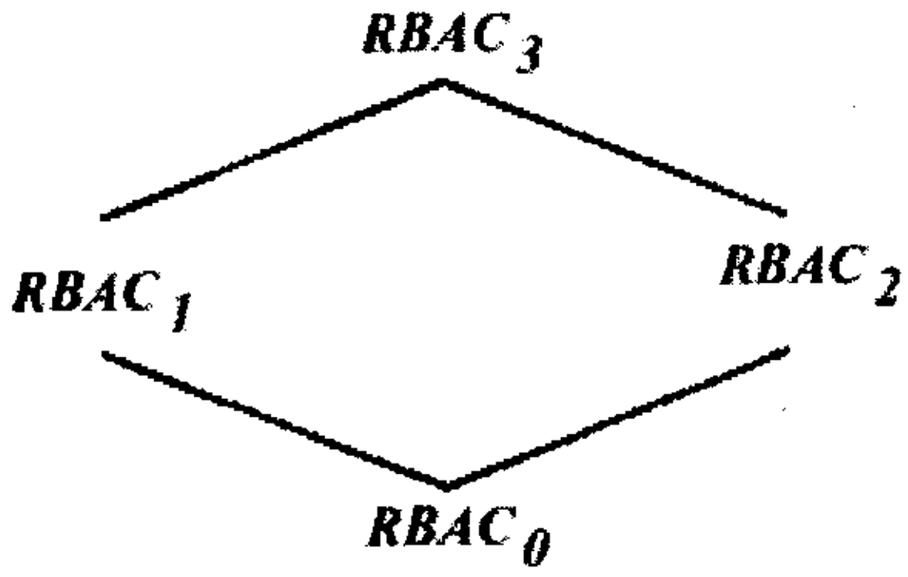

图 3.2 RBAC96 模型关系示意图

3.3.1 $RBAC_0$ 模型

$RBAC_0$ 模型处于 RBAC96 模型的最底层。它定义了基于角色访问控制机制所需的最基本的概念。这些基本元素包含三个集合 Users（用户集 U）、Roles（角色集 R）、Permissions（权限集 P）。另外 $RBAC_0$ 还定义了 Sessions（会话集 S）。

广义的用户可以代表人员（通常是企业内的某个成员）、程序模块、自动智能体（intelligent autonomous agents）、外部服务等，但通常只是代表企业人员。角色代表企业内某个职位或某个特定用户组及其相关的权限。权限代表系统内某个客体（object）的特定访问方式（比如读、写、修改）的许可。这里的权限等价于传统安全文献里的授权（authorization）、访问权限（access right）、特权（privilege）等术语。客体指的是系统内的数据对象（data object）或资源对象（resource object）。

$RBAC_0$ 定义了 UA（User Assignment，用户赋予）关系和 PA（Permission Assignment，权限赋予）关系[7]。它们分别是 Users 到 Roles 以及 Permissions 到 Roles 上的多对多的映射关系。一个用户可以具有多个角色，而每个角色也可包含多个用户。同样的，每个

角色都可以具有若干权限，而每个权限也可以被若干角色拥有。这两个关系是 RBAC 模型的基础，体现了设计者在访问控制系统中引入角色的核心思想。角色作为用户和权限联系的枢纽而存在，它大大简化了访问控制的配置和审计工作。

用户登录到系统之后就自动具有一个会话，这个会话记录了用户的当前属性，比如用户身份、用户所具有的当前活跃角色、权限等。从形式化模型上来看，会话也是 Users 到 Roles 的映射关系，但是这种映射关系与 UA 关系不同的是，Sessions 是因用户登录而动态出现的，而 UA 关系却是静态地存在于系统之中的。一个 session 只能对应一个用户，而一个角色却可以对应多个用户。

$RBAC_0$ 模型的形式化定义如下。

- U、R、P、S 分别代表用户集、角色集、权限集和会话集；
- $UA \subseteq U \times R$ 是用户集与角色集之间的多对多映射关系；
- $PA \subseteq P \times R$ 是权限集与角色集之间的多对多映射关系；
- $user: S \to U$，某个会话 S_i 与某个用户 $user(S_i)$ 的函数映射，该函数值在会话 S_i 的有效期间是固定的；
- $roles: S \to 2^R$，某个会话 S_i 到 Roles 的一对多函数映射，$roles(S_i) \subseteq \{r | (user(S_i), r) \in UA\}$，会话 S_i 所具有的权限为 $\bigcup_{r \in roles(S_i)} \{p | (p, r) \in PA\}$。

现实中，每个用户至少要具有一个角色，每个角色至少要具有一个权限，否则该用户或角色就失去了存在的意义。所以，尽管模型中没有上述约束，但是在实现时应该考虑它。

3.3.2 $RBAC_1$ 模型

$RBAC_1$ 模型是在 $RBAC_0$ 模型的基础上引入了角色层次（Role Hierarchies，RH）的概念而形成的模型。

角色层次通常用于反映企业内各部门之间或职位之间的上下级关系，是角色集 Roles 上的一个偏序关系，记为≤。对于角色层次有如下关系：

- ✓ $r_i \leq r_i$；
- ✓ 如果 $r_i \leq r_j$ 且 $r_j \leq r_k$，则 $r_i \leq r_k$；
- ✓ 如果 $r_i \leq r_j$，则 $r_j \leq r_i$ 不成立。

如果有关系：$r_i \leq r_j$，则称 r_j 为 r_i 的上级角色，r_i 为 r_j 的下级角色。引入角色层次这个概念之后，用户不仅具有通过 UA、PA 关系所指定的权限，而且还拥有 UA 指定角色的所有下级角色的权限，用户也可以用这些下级角色来登录系统、创建会话。用户使用这些下级角色与使用直接的 UA 关系所指定的角色没有任何不同之处。

$RBAC_1$ 模型的形式化定义如下：

- U、R、P、S、UA、PA 和 $user$ 的定义与 $RBAC_0$ 模型中的定义一致；

- $RH \subseteq R \times R$ 为角色集上的偏序关系；
- $roles: S \to 2^R$，对任一会话 $S_i \in S$，$roles(S_i) \subseteq \{r | (\exists r' \geq r)[(user(S_i), r') \in UA]\}$，会话 S_i 所具有的权限为 $\bigcup_{r \in roles(S_i)} \{p | (\exists r'' \leq r)[(p, r'') \in PA]\}$。

3.3.3 $RBAC_2$ 模型

$RBAC_2$ 模型是在 $RBAC_0$ 模型的基础上引入了约束（Constraint）的概念而形成的模型。它与 $RBAC_1$ 模型没有直接的联系。

RBAC 机制中的约束是一种判断模型中已有组件（UA、PA 等）的指派是否有效的机制。

如果两个访问权限被赋予同一个人会产生安全问题，那么这两个访问权限就叫互斥权限。如果一个角色中的某个权限与另一个角色中的某个权限为互斥权限，那么这两个角色就叫互斥角色。静态职责分离（Static Separation of Duties，SSD）原则指的是任意两个互斥角色都不能同时指派给同一个用户。动态职责分离（Dynamic Separation of Duties，DSD）原则指的是任意用户不能同时激活自己的任意两个互斥角色。静态职责分离原则实际上是对 $RBAC_0$ 模型中的 UA 关系和 PA 关系的约束，而动态职责分离原则就是对 $RBAC_0$ 模型中的 $roles$ 函数的约束。静态职责分离原则和动态职责分离原则统称为职责分离原则（Separation of Duties，SoD）。

$RBAC_2$ 模型中作用于 UA 关系的另一个约束就是基数约束。基数约束是对角色所能容纳的用户个数的约束。这在现实中也是很有意义的，比如，企业的董事长这个角色通常就只有一个人。

还有一种约束称为首备角色（prerequisite roles）约束。如果要将某角色指派给某用户，则必须先指派其他一些角色给该用户。比如，用户要当出纳，首先他就必须是财务部门的成员。

3.3.4 $RBAC_3$ 模型

$RBAC_3$ 模型是 $RBAC_1$ 和 $RBAC_2$ 的有机结合。因此，在 $RBAC_3$ 模型中，不仅有角色层次，而且还有约束。

RBAC3 模型示意图如图 3.3 所示。该图中既包含了 RBAC0 模型的基本组件，又包含了 RBAC1 模型的角色层次和 RBAC2 模型的约束。

如果增加约束：任何下级角色的权限都自动被其上级角色所享有，那么，就可以把角色层次看做一种约束来处理。

在 RBAC96 模型中还提出将系统的管理工作与日常业务工作分离开来，管理权限只能分配给管理角色（administrative roles），而普通操作权限只能分配给普通角色（regular roles）。这样，一方面，管理员只具有管理权限（比如指派某普通角色具有哪些业务操作权限）而不具有业务操作权限；而另一方面，普通的业务人员只具有管理员分配给他

的普通角色，从而也就只具有相应的普通业务操作权限。

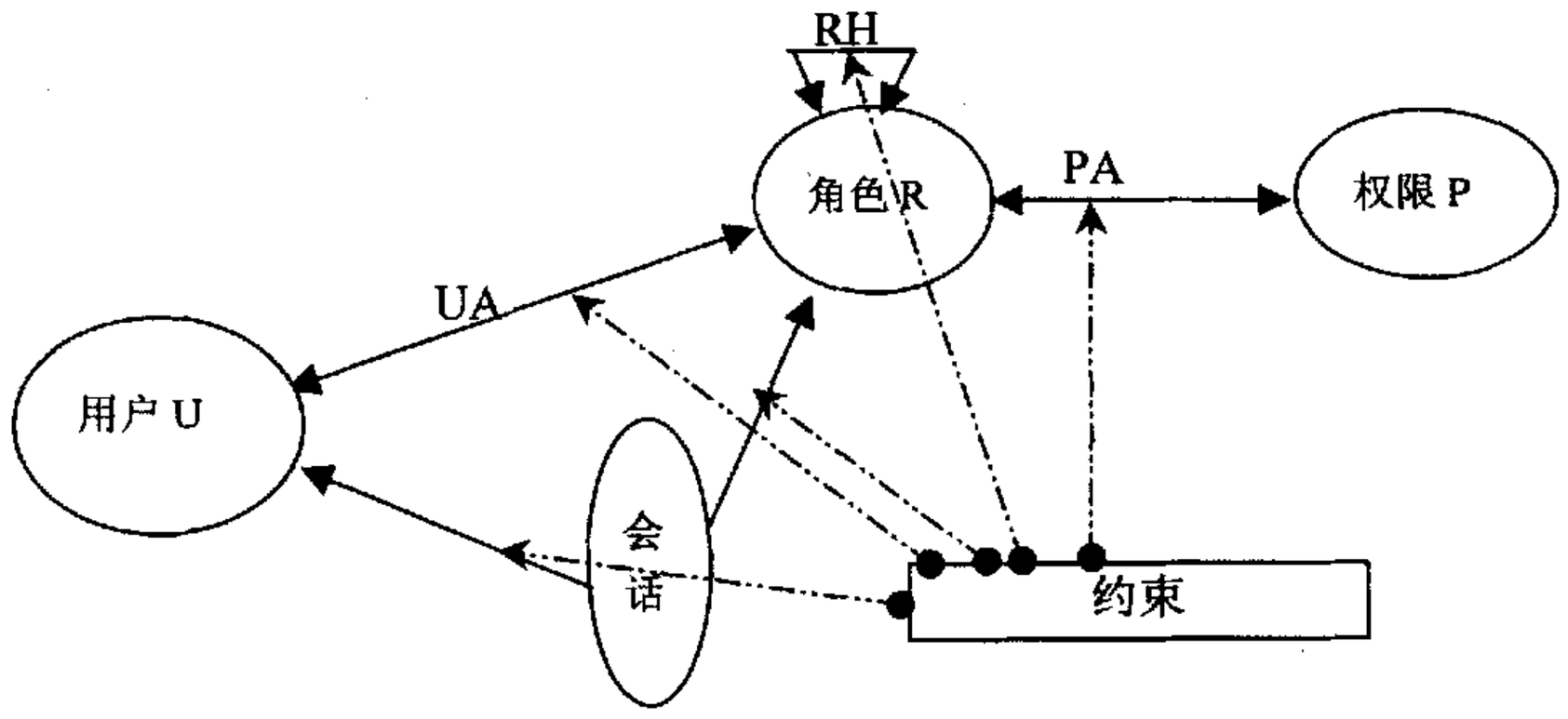

图 3.3　$RBAC_3$ 模型示意图

第四章 基于任务和角色的访问控制

目前的工作流安全模型主要是基于 RBAC 的安全模型。在工作流系统中，任务这个概念非常突出，所以也有人提出在工作流系统中使用基于任务的访问控制（Task Based Access Control，TBAC）安全模型。下面首先介绍这两种模型并分析它们的缺陷，然后在此基础上提出一种基于任务和角色的访问控制模型，并给出其形式化定义。

§4.1 传统工作流安全模型分析

传统工作流安全模型主要包括基于任务的工作流安全模型和基于角色的工作流安全模型。这两种模型都各有优缺点，下面简要介绍这两种模型。

4.1.1 基于任务的工作流安全模型

随着数据库、网络和分布式计算的发展，企业任务进一步自动化，与服务相关的信息进一步计算机化，这促使我们将安全问题方面的注意力从独立的计算机系统中静态的主体和客体保护转移到随着任务的执行而进行动态授权的保护上。由此发展出了基于任务的访问控制。基于任务的访问控制以任务为核心，强调随着任务的开始和完成而对相应用户进行动态授权。

TBAC 是一种新的安全模型[12]，从应用和企业层角度来解决安全问题（而非已往从系统的角度）。它采用“面向任务”的观点，从任务的角度来建立安全模型和实现安全机制，在任务处理的过程中提供动态的安全管理。在 TBAC 中，客体的访问权限控制并不是静止不变的，而是随着执行任务的上下文环境发生变化。具体说来，TBAC 有三点含义。首先，它是在工作流的环境考虑对信息的保护问题，在多数步骤中对信息的处理通常与以前的处理有关，因而 TBAC 是一种上下文相关的访问控制模型。其次，它不仅能对不同工作流实行不同的访问控制策略，而且还能对同一工作流的不同任务实例实行不同的访问控制策略。所以 TBAC 又是一种基于任务实例的访问控制模型。最后，因为任务都有时效性，所以在基于任务的访问控制中，用户对于授予他的权限的使用也是有时效性的。

4.1.1.1 典型模型介绍

授权步(authorization step)表示一个原子授权处理步，是指在一个工作流程中对处理对象(如办公流程中的文档)的一次处理。它是访问控制所能控制的最小单元。授权步由受托人集(trustee-set)和许可集(permissions set)组成，其中，受托人集是可被授予授权步的用户的集合，许可集则是受托人集的成员被授予授权步之后拥有的访问许可。当授权

步初始化以后，一个来自受托人集中的成员将被授予授权步，我们称这个受托人为授权步的执行者，该受托人执行授权步过程中所需许可的集合称为执行者许可集。在 TBAC 中，允许一个授权步的处理决定后续授权步对处理对象的操作许可，我们将这样的许可称为激活许可[12]。

授权结构体(authorization unit)是由一个或多个授权步组成的结构体，它们在逻辑上是联系在一起的。任务(task)是工作流程中的一个逻辑单元。它是一个可区分的动作，可能与多个用户相关，也可能包括几个子任务。

总之,一个工作流的业务流程由多个任务构成。而一个任务对应于一个授权结构体，每个授权结构体由特定的授权步组成。

作为一种新的主动访问控制模型,TBAC 的授权与传统访问控制模型的授权有很大的不同。传统访问控制模型的授权一般用三元组(S,O,P)表示，其中 S 表示主体，O 表示客体，P 表示许可。如果存在元组(S,O,P)，则表明 S 可在 O 上执行 P 许可。这些三元组都预先定义好并静态地存放在系统中，且无论何时都是有效的。

在 TBAC 中，授权需用五元组(S,O,P,L,AS)来表示。其中，S、O 的意义同前，P 是与授权步 AS 相关的权限，而 L 则是授权步 AS 的存活期限。L 和 AS 是 TBAC 不同于其他访问控制模型的显著特点。

TBAC 模型的示意图如图 4.1 所示。

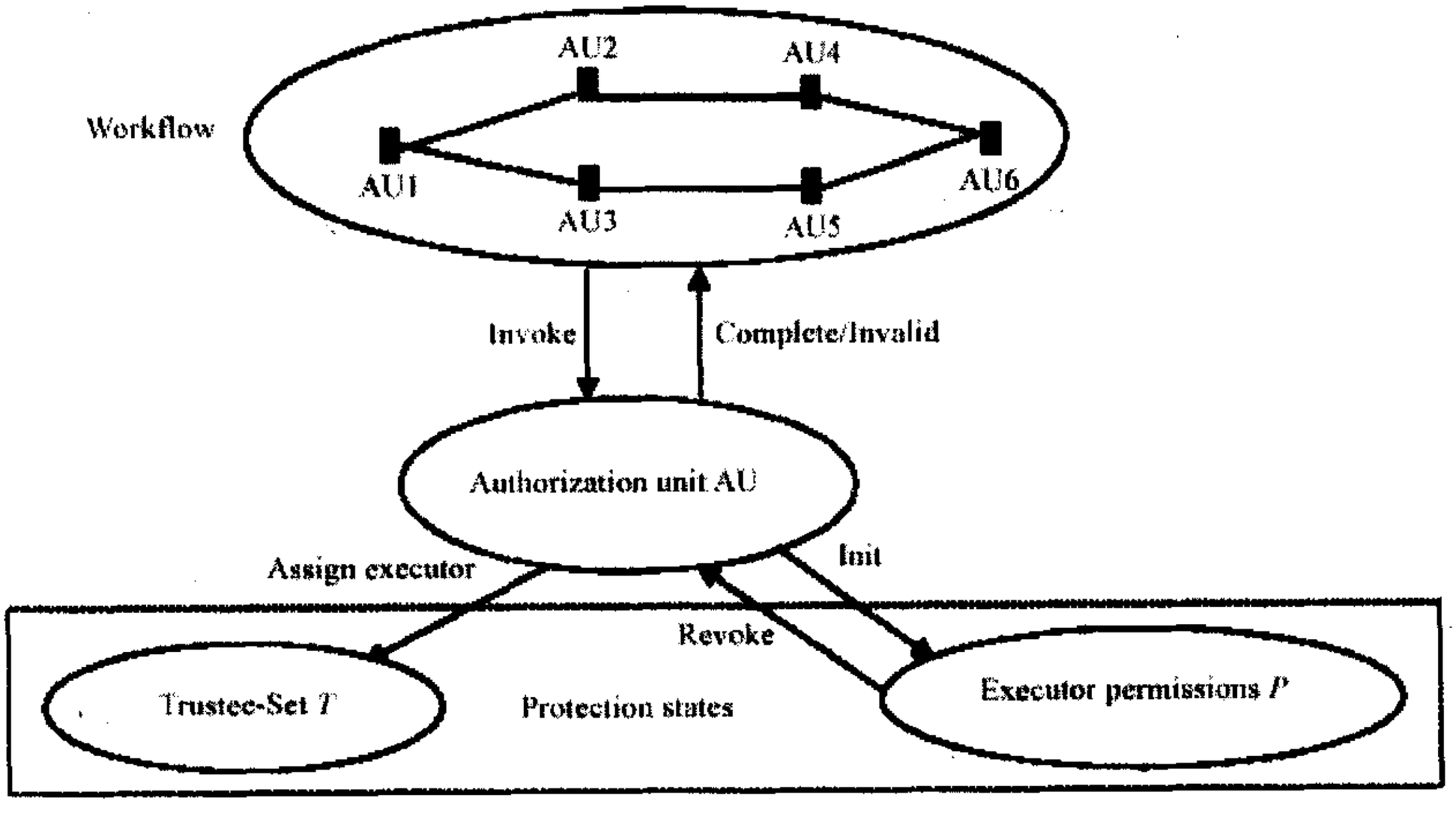

图 4.1　TBAC 模型示意图

4.1.1.2 缺陷分析

基于任务的工作流安全模型的主要缺陷在于它不适合大型企业的应用，因为工作流

管理系统主要是应用于大型企业的流程自动化管理，那么该系统的访问控制就会不可避免的牵涉到许多任务以及用户的权限分配问题，而 TBAC 只是简单的引入受托人集合来表示任务的执行者，而没有论及怎样在一个企业环境中确定这样的受托人集。这样的系统虽然可以运作起来，也达到了基于任务授权、提高安全性的目的，但是这种情况就象 RBAC 出现之前应用两层访问控制结构（这种模型直接指定主体对客体访问操作）的情况一样，都能运行，却存在配置过于繁琐的缺点。所以，有必要对这种机制进行改进，比如在模型中引入角色一类的概念简化其安全控管工作。

4.1.2 基于角色的工作流安全模型

由于工作流管理系统多用于大型企业中，所以它的安全访问控制的管理工作也非常繁琐。在这种情况下，将基于角色的访问控制应用于工作流管理系统的优势就很明显了。

4.1.2.1 典型模型介绍

Savith Kandala 和 Ravi Sandhu 在 2002 年提出了一个工作流安全模型[15]，该模型采用基于角色的访问控制机制，沿用 RBAC96 中的符号，并引入工作流中的任务、任务实例的概念，定义了一个任务与任务实例之间的函数映射关系，并约定：如果将一个任务的执行权限赋予一个角色，则意味着该角色拥有执行该任务的所有实例的权限。

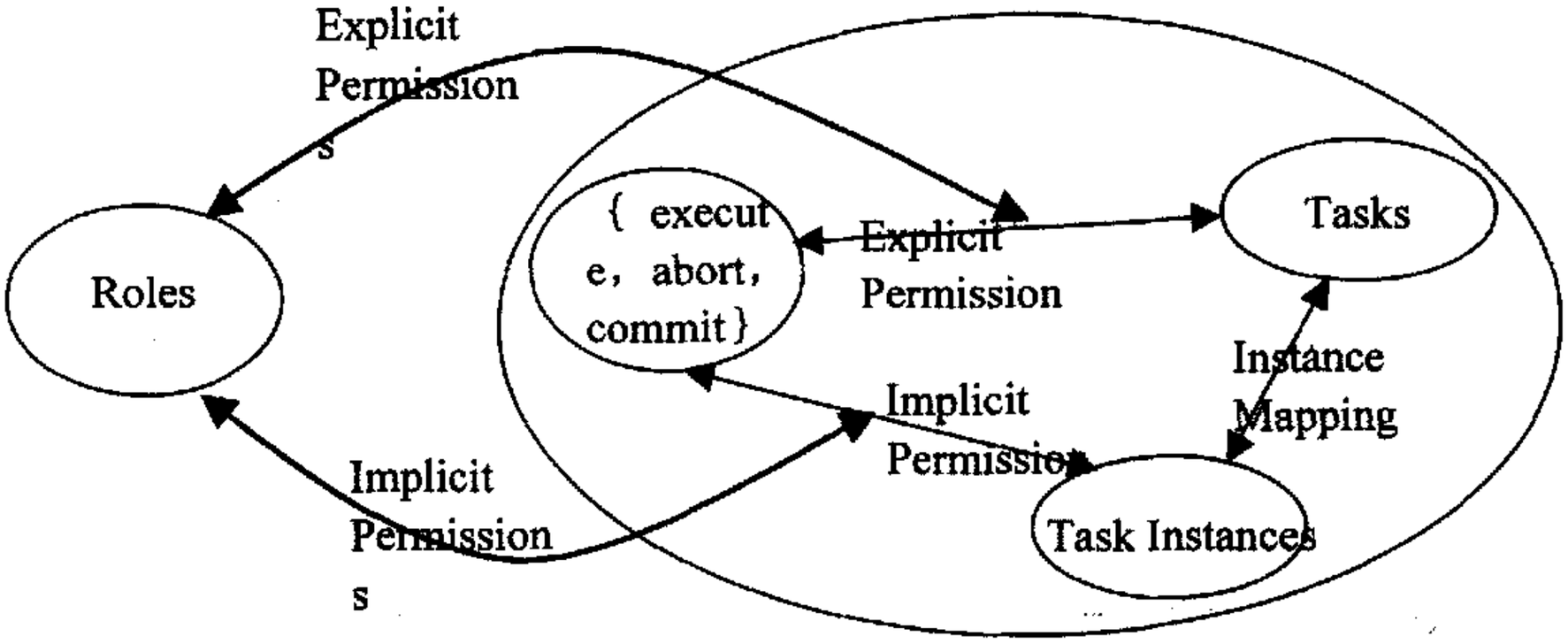

图 4.2　基于角色的工作流安全模型

该模型利用工作流系统的特点（任务与任务实例之间的关系）对工作流的访问控制重新建模。其简化模型可表述如下：

- TT 表示任务集；TI 表示任务实例集；S 表示状态集。
- 操作集：OP={execute, commit, abort}，分别表示执行任务、提交任务和中止任务。
- 状态集：S = {Initial, Executing, Committed, Aborted}，分别表示初始态、执行态、提交态、中止态。

- 可能的操作：PossibleOperations ={(Initial, execute), (Executing, commit), (Executing, abort)}，分别表示：在初始态时可以执行任务，在执行态时可以提交任务以及在执行态时可以中止任务。
- 状态转换集：T = {(Initial, execute, Executing), (Executing, commit, Committed), (Executing, abort, Aborted)}，分别表示：当任务处于初始态时执行任务，则任务状态变为执行态；当任务处于执行态时提交任务，则任务状态变为提交态；当任务处于执行态时中止任务，则任务状态变为中止态。
- 任务与其实例间的映射 $\Im$：$TT \to 2^{TI}$，对任意的 $a,b \in TT$，$\Im(a) \cap \Im(b) = \phi$ 当且仅当 $a \neq b$。
- 明权限关系：$EP = OP \times TT$；暗权限关系：$IP = OP \times TI$；权限集：$P = EP \cup IP$。
- 角色的明权限赋予：$EPA \subseteq R \times EP = R \times OP \times TT$。
- 角色的暗权限赋予：$IPA \subseteq R \times IP = R \times OP \times TI$，

$$IPA = \{(r,op,ti) | [\exists (r,op,t) \in EPA] \wedge [ti \in \Im(t)] \wedge [(CurrentState(ti),op) \in PossibleOperations] \wedge [StartCondition(ti) = True]\}$$

其中，$CurrentState(ti)$ 表示任务实例 ti 的当前状态，$StartCondition(ti) = True$ 表示任务实例 ti 的启动条件为真。

- 角色的权限赋予 PA：$PA = EPA \cup IPA$。

4.1.2.2 缺陷分析

上述模型就是典型的基于角色的工作流安全模型。它的缺点在于它只是简单地将RBAC模型用于工作流系统，并没有对RBAC模型本身做什么改进，存在最小权限约束假象、数据冗余、动态性差等安全问题。另外，该模型没有论及对执行任务时所需要的资源访问权限的控制，从而使得用户可以绕过任务获得访问权限。

以往的RBAC模型中存在一个特点，就是权限直接与角色相关，其简化示意图如图4.3所示。

图 4.3 RBAC 示意图

基于角色的工作流安全模型指出，能执行某任务的角色就具有相应的任务实例的执行权限。该模型为任务定义了3个操作权限：execute、commit、abort，且规定：1）对某一任务而言，任意一个角色要么拥有这个任务的所有的三种权限，要么不拥有这个任务的任何权限；2）如果用户执行了某任务的“execute”操作，那么该用户必须执行这个任务的“commit”或“abort”操作。该模型还定义了初始态、执行态等若干任务状态，且规定在初始态下只能执行“execute”操作，在执行态下只能执行“commit”操作或“abort”

操作。除了能够控制用户在特定的任务状态下只能执行某种特定的操作外，这样定义的作用不大。通过适当的界面编程（比如，将操作所对应的命令按钮放置在特定的页面）就可以达到这个目的，而不用让管理员去指定任务的某个操作允许哪些角色执行。比如，由于在初始态的时候，用户还没进入任务操作界面，只是在一个类似于work list一样的表单中能看到自己所能做的任务，用户也看不到“commit”或“abort”按钮，所以这个时候，用户必然只能执行“execute”操作，而不能执行“commit”和“abort”操作。相应地，只有当用户真正开始做这个任务实例之后，任务才处于执行态。任务实例进入执行态后要让用户能执行“commit”和“abort”操作，可以在相应的页面上显示“commit”或“abort”按钮。由此可见，在进行任务授权的时候，最自然的控制粒度就是让操作员指定任务允许哪些角色执行，而不是细化到任务操作一级，否则就增加了访问控制的复杂性和数据的冗余度。

该模型没有论及在任务执行过程中对客体（数据、资源等）的访问权限。在工作流管理系统中，访问控制的重点有两个：对任务的访问控制和对任务执行过程中所需资源的访问控制。如果只是单纯的看到工作流系统中任务的突出性，只对任务做了访问控制，而忽略了在任务执行过程中用户对资源的访问控制，那么设计出来的系统也是不安全的。

在模型中没有明确指定任务执行过程中采用什么资源访问控制策略的情况下，对资源的访问控制存在三种情况：

1) 无访问控制策略；
2) 对4.1.2.1节中所述的任务操作权限进行扩展。任务的操作不仅包括execute、commit、abort等任务一级的操作，而且还包括任务执行过程中对数据资源的访问操作；
3) 将任务执行过程中的数据资源访问权限直接赋给相应的角色。

在第一种情况下，由于对任务执行过程中的资源访问控制不采取任何措施，听任用户任意使用，所以系统中的资源将在任何情况下都可以被任何人访问，这显然是不安全的。

在第二种情况下，会产生数据冗余和动态适应性差的缺陷。在上述基于角色的工作流安全模型中，虽然在传统RBAC机制的基础上根据工作流系统中以任务为中心的特点引入了任务的概念，但是作者却把操作与任务捆绑在一起作为权限集（见4.1.2.1节中EP、IP的定义）来处理。由于该模型要求对每个任务的每个操作都要生成一个类似于二元组（TT，OP）形式的权限，所以将任务操作集合扩展到包括任务执行过程中对数据资源的操作访问权限之后，就会使得授权数据显得非常冗余。比如，如果一个任务TT_i对应了100个允许的操作，那么对该任务来说就要定义100个类似于(TT_i, OP_j)的二元组（其中，j为0到99中的任意一个整数）。这种机制的缺陷在于，当一个角色允许执行很多任务的时候，该角色所关联的权限数目大，难于管理。另外，当要添加、撤销、改变某个任务的访问权时，还涉及到许多权限（二元组（TT，OP））的更新，需要进行很多操作。

这种方式的效率低且动态适应性较差。而在分布式工作流管理系统中，动态权限的合法生成和及时撤销却恰恰是最重要的。

在第二种情况下，作者引入任务概念的唯一好处就是产生了一种将任务与操作捆绑在一起的新型权限，这样就能将传统的基于角色的访问控制模型应用到工作流系统中来。这虽然是一种将基于角色的访问控制模型应用到工作流系统中的方法，但是如前所述，这种方法将任务操作集合扩展到包括任务执行过程中对数据资源的访问操作之后，授权数据就显得非常冗余，而且这种冗余是不必要的。

通常，我们应该将执行任务所必需的资源访问权限分配给任务，这些资源访问权限对执行这个任务来说是必不可少的。对同一个任务的任务实例来讲，不管哪个用户执行，他所需要的资源访问权限都一样。这样做是自然的，这也是最小权限约束的意义所在。所以，不存在将任务所对应的资源访问权限中的一部分授予用户，一部分却不授予用户的情况，否则，用户就无法完成这个任务。如果有一部分用户比另一部分用户需要更多一些的资源访问权限才能完成任务，那么就应该把它分成两个不同的任务。正是因为在工作流系统中，不管由哪个用户来执行，同一个任务所需要的访问权限都是一样的，所以任务捆绑操作之后的细粒度不会带来好处。

在第三种情况下会产生最小权限约束假象和类似于第一种情况的安全问题。众所周知，最小权限约束是访问控制领域里判断某个安全模型好坏的一个重要标准。表面上看，传统的RBAC机制好像没什么问题，但在这种情况下，最小权限约束却只是个假象。这个假象是由两方面的原因造成的：一方面，在RBAC96模型中明确规定每个用户可以同时拥有多个会话，而且只要满足UA关系，每个会话可同时激活多个不互斥的角色，这就使得用户具有自己所有的会话中激活了的所有角色的权限；另一方面，传统的RBAC系统中，权限是直接与角色相关的。这就决定了权限的粒度最多只能细化到角色一级，最小权限约束也只是角色的最小权限约束，而现实世界中一个角色往往可以执行多项任务。这样一来，不管用户是否执行任务，只要用户激活某一角色就拥有该角色的全部权限。实际上，用户在执行一个任务实例时并不需要其它任务实例的访问权限，但是由于上述两方面的原因，用户在实际操作时可能拥有多于单独执行一项任务的权限，所以该系统中的最小权限约束只是个假象，并没有实现真正的最小权限约束。因此，第三种情况会产生类似于第一种情况的安全问题，即，不管用户做不做任务都拥有对某些不需要的数据资源的访问权限。

§4.2 基于任务和角色的访问控制模型

在工作流系统中应用访问控制时，传统的访问控制技术显得力不从心。一方面，当数据在工作流系统中流动时，执行操作的用户在改变；另一方面，当用户执行的任务实例改变时，用户所具有的操作权限也在改变。在这种以任务为中心的系统中，单独采用任何一种传统的访问控制技术（如DAC、MAC、RBAC等）都难以满足系统的安全需求，因此，有必要建立一种新的访问控制模型。下面介绍的基于任务和角色的访问控制模型

能很好地适用于工作流系统。

4.2.1 新模型的提出

究其根源，产生RBAC模型缺陷的原因就是RBAC模型的三层访问控制结构（用户—角色—权限）。这种结构决定了RBAC模型直接应用于工作流系统时会产生4.1所述的一些缺陷。另外，如果在工作流管理系统中采用基于任务的访问控制，那么管理员的管理工作势必非常繁琐。这种方案不适合于大型企业。

目前的工作流安全模型研究多数关注于4.1所述的两个方案中的一种，都是将RBAC或TBAC稍加改进而应用于工作流管理系统。由于这种改进没有改变各自的访问控制结构，从而不能克服已有访问控制模型的缺陷。为了解决上述问题，必须在保留RBAC和TBAC各自优点的前提下，从根本上改变RBAC和TBAC的模型结构。

本文将任务集和任务实例集的概念明确引入到RBAC模型中，形成基于任务和角色的访问控制模型（Task & Role Based Access Control，TRBAC）。该模型将任务独立出来作为一个单独的概念，并置于角色和权限之间，将传统RBAC模型的3层访问控制结构改为4层访问控制结构，其示意图如图4.4所示。

图 4.4　简化的 TRBAC 模型

整个工作流系统中的用户都赋给特定的角色，再规定每个角色可以执行哪些任务以及每个任务的最小访问权限。该模型中，权限不再直接与角色相关，而必须通过任务才能与角色关联起来，而这时的权限集合也不再表示为操作集合与任务集合的迪卡尔乘积，而只单纯的表示为数据资源的访问操作集合。如此一来，权限可以作为任务的属性来实现，而不必建立类似于传统RBAC模型中的二元组权限，没有了数据冗余，而且方便了管理员的安全控管工作。

不管用户采用其拥有的何种角色登录，都必须在执行系统分配的任务实例的过程中才拥有该授权任务实例的资源访问权限，这样就自然地实现了权限的动态分配和撤销，提高了工作流安全系统的动态适应性。从而也就克服了传统的基于角色的工作流安全模型中的数据冗余、动态适应性差的缺陷。

在TRBAC模型中，由于权限直接与任务相关，用户只有在执行某任务实例的情况下才拥有该任务所对应的访问权限，所以，最小权限约束自然就细化到了任务一级。由于执行任务所需要的资源访问权限是按最小权限约束原则来指定的，所以，只要让用户做这个任务，那么就应该允许用户拥有执行该任务所需的所有资源访问权限，否则用户就无法完成这个任务。TRBAC模型的这两个“自然”的特性就使得它不会存在最小权限约束假象，且较好地克服了它的数据冗余、动态性差的缺点。

到此为止，传统RBAC模型中的上述几类缺陷都得到了克服。下面详细介绍本文所

提出的基于任务和角色的访问控制模型。除了沿用经典的RBAC96模型的一些基本概念外，本文也采用包含四个子模型的模型族来描述TRBAC：一个基本模型，两个中间模型和一个高级模型。它们的关系如图4.5所示。

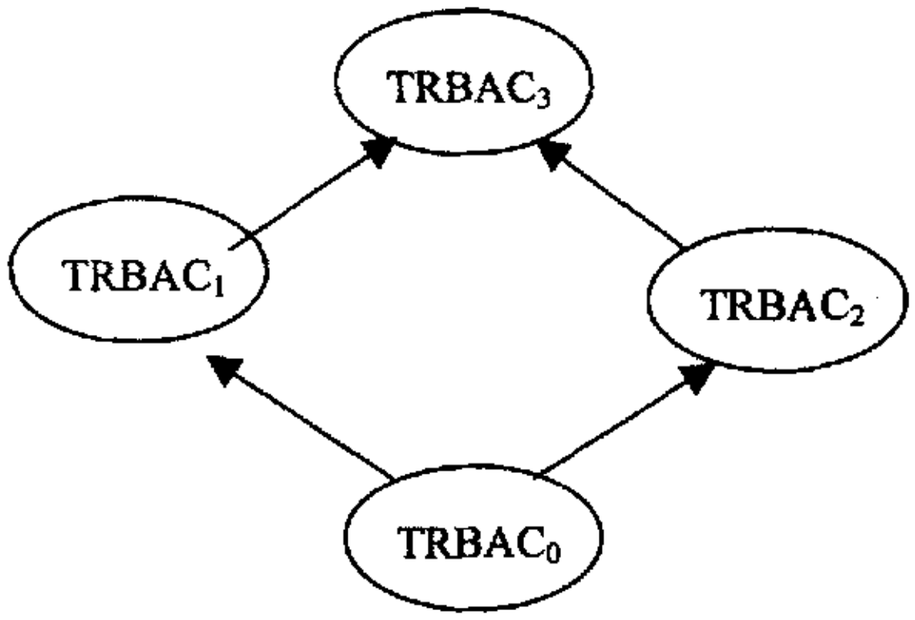

图 4.5　TRBAC 模型族

4.2.2 基本模型

这一节描述基于任务和角色的访问控制模型族中的第一个模型：$TRBAC_0$ 模型，这个模型处于 TRBAC 模型族的最底端，描述了 TRBAC 模型所需要的基本概念。

4.2.2.1 基本概念

任务定义为包含若干人机交互操作的一个动作序列。系统中所有任务的集合用 TT 表示，单个任务用 tt 表示。任务并不是现实世界中的一个任务，而是现实世界中的一类任务的抽象表示，所以又可以称为任务模板。每一个任务对应若干执行该任务所必需的资源访问权限。

任务实例是现实世界中的某个具体任务，它与系统中已存在的某一个任务（任务模板）对应，是该任务的实例化结果。系统中所有任务实例的集合用 TI 表示，单个任务实例用 ti 表示。如果管理员将某一任务执行权授予某用户，那么系统就可以将该任务所对应的所有任务实例的执行权分配给该用户。任务实例具有原子性，即，如果将某个任务实例授予某用户执行，则该用户不仅具有该任务实例的启动、执行、挂起、保存、提交、重启、放弃等常规权限，而且还具有执行这个任务实例所需的所有资源访问权限。任务实例的常规权限可以被自动授予该任务实例的执行者，系统不用再额外指定。

为了模型讨论的方便，在这里定义一个任务实例映射关系 h 如下：

$$h: TT \rightarrow 2^{TI},$$

并且满足条件：对任意 $TT_i, TT_j \in TT$，$h(TT_i) \cap h(TT_j) = \phi$ 当且仅当 $TT_i \neq TT_j$。

$TRBAC_0$ 模型将角色定义为任务的集合。权限定义为对系统中的各种资源的访问权限。

用户角色赋予关系（User Role Assignment，URA）是用户集和角色集上的多对多映

射关系。

任务权限赋予关系（Task Permission Assignment，TPA）是任务集与权限集上的多对多的映射关系。

实例权限赋予关系（Instance Permission Assignment，IPA）是为了清晰地描述模型而定义的，它是任务实例集与权限集上的多对多的映射关系。系统运行时可以通过这个关系检查任务实例所具有的权限。

任务角色赋予关系（Task Role Assignment，TRA）是任务集与角色集上的多对多的映射关系。

实例角色赋予关系（Instance Role Assignment，IRA）这个关系和 IPA 一样也是为了清晰地描述模型而定义的，它是角色集与任务实例集上的多对多的映射关系。系统运行时，可以通过这个关系检查某角色可以执行哪些任务实例，或者某任务实例可以由哪些角色来执行。

另外，$TRBAC_0$ 模型也定义了一个会话集（Sessions），每个会话记录用户登录系统之后的一些状态信息，如用户登录所用的角色、登录时间、地点等。

用户（user）、用户集（U）、权限（permission）、权限集（P）的概念与 3.3 节所描述的 RBAC96 模型中的定义一致，这里不再赘述。

$TRBAC_0$ 模型中，系统管理员平常一般只是对 URA、TRA、TPA 进行管理。实际操作中，角色只有通过 IRA 和 IPA 才能真正获得权限。

4.2.2.2 形式化模型描述

$TRBAC_0$ 模型的形式化描述可以表述如下：

- 用户集（U）、会话集（S）、角色集（R）、任务集（TT）、任务实例集（TI）、权限集（P）。
- 用户角色赋予关系 URA：$URA \subseteq U \times R$。
- 任务与任务实例之间的映射关系 $\hbar$：$TT \to 2^{TI}$，
  并且满足条件：对任意 $TT_i, TT_j \in TT$，$\hbar(TT_i) \cap \hbar(TT_j) = \phi$ 当且仅当 $TT_i \neq TT_j$。
- 任务角色赋予关系 TRA：$TRA \subseteq R \times TT$。
- 实例角色赋予关系 IRA：$IRA \subseteq R \times TI$，由 $\hbar$ 和 TRA 决定：
  $IRA = \{(r, ti) | (r \in R) \wedge (\exists tt \in TT)[((r, tt) \in TRA) \wedge (ti \in \hbar(tt))]\}$。
- 任务权限赋予关系 TPA：$TPA \subseteq TT \times P$，每一项任务对应一个最小权限集。
- 实例权限赋予关系 IPA：$IPA \subseteq TI \times P$，由 $\hbar$ 和 TPA 决定：
  $IPA = \{(ti, p) | (p \in P) \wedge (\exists tt \in TT)[((tt, p) \in TPA) \wedge (ti \in \hbar(tt))]\}$。
- $user : S \to U$，某个会话 S_i 与某个用户 $user(S_i)$ 相关联。
- $roles : S \to 2^R$，对任意的 $S_i \in S$，$roles(S_i) \subseteq \{r \in R | (user(S_i), r) \in URA\}$，用户在

每个会话中可以同时激活多个角色。

- $tasks: R \to 2^{TT}$，对任意 $r \in R$，$tasks(r) = \{tt \in TT | (r, tt) \in TRA\}$，$tasks(r)$ 表示可以分配给角色 r 的任务实例所对应的任务模板。
- 可以分配给用户 u 执行的任务实例：

$$taskIns\tan ces(u) = \bigcup_{S_i \in S} \bigcup_{r \in roles(S_i)} \{ti | u = user(S_i) \wedge (\exists tt \in TT)[(tt \in tasks(r)) \wedge (ti \in \hbar(tt))]\}。$$

- 对任意任务实例 $ti \in TI$，ti 所能分配的用户集：

$$executors(ti) = \{u \in U | (\exists r \in R)[((u, r) \in URA) \wedge ((r, ti) \in IRA)]\}。$$

基本模型的示意图如图 4.6 所示。

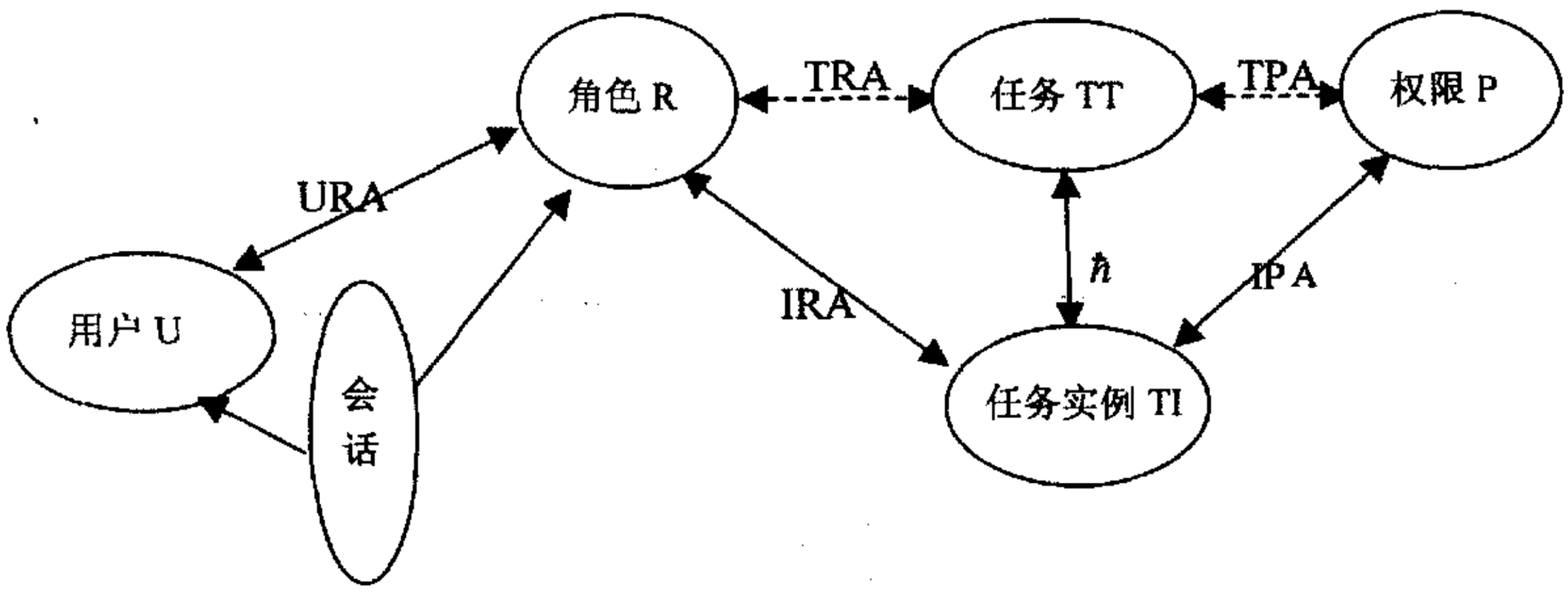

图 4.6 $TRBAC_0$ 模型示意图

4.2.3 层次模型

层次模型是基于任务和角色的访问控制模型族中的第二个模型（$TRBAC_1$），主要是对企业中人员组织层次和任务层次进行建模以达到简化管理员工作的目的。

层次模型中不仅具有传统 RBAC 模型的角色层次（Role Hierarchies，RH）、私有角色（Private Role，PR）等概念，而且还引入了任务层次（Task Hierarchies，TH）、私有任务（Private Task，PT）等概念。

4.2.3.1 角色层次

用角色层次来对企业中用户的权利等级、职位高低关系进行建模是很自然的。比如，对于一个只有研发部和市场部的一个软件公司来说，其简化的角色层次模型可以建模如图 4.7 所示。

这里的角色层次概念沿用 RBAC96 模型中的角色层次的概念，定义为角色集上的偏序关系，记为≤。如果有关系：$r_i \le r_j$，则称 r_j 为 r_i 的上级角色，或者称 r_i 为 r_j 的下级角色。偏序关系在数学上是一个具有自反性、传递性、反对称性的关系：

- ✓ $r_i \leq r_i$；
- ✓ 如果$r_i \leq r_j$且$r_j \leq r_k$，则$r_i \leq r_k$；
- ✓ 如果$r_i \leq r_j$，则$r_j \leq r_i$不成立。

上级角色自动继承得到下级角色的所有任务执行权，也就是说，上级角色可以做系统分配给下级角色的所有任务而不需要管理员再额外指定。这样就简化了管理员的管理工作。

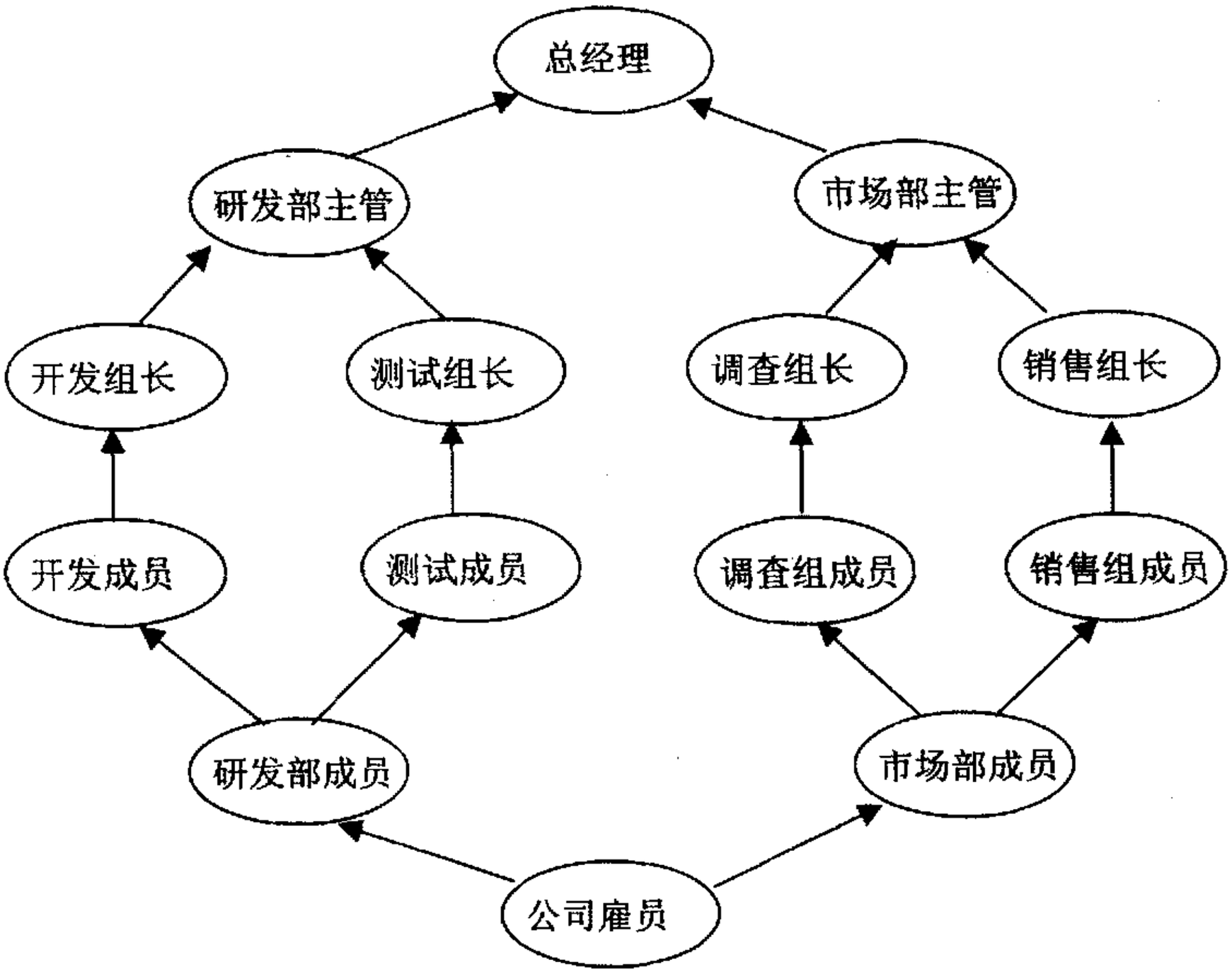

图 4.7　软件公司角色层次建模示例

4.2.3.2 私有角色

有时候系统不希望上级角色继承到下级角色某些任务的执行权。比如，有时候领导只具有某些文档的审批权，而没有起草权。这时候，下级的起草文档这个任务就不能被其上级继承到。在 RBAC96 模型中引入私有角色的概念来解决这个问题。将下级角色中不允许被其上级角色继承的任务分离出来，新增加一个角色来容纳原下级角色的任务中不允许上级角色继承的那部分任务，而原下级角色就只剩下允许上级角色继承的那些任务。这种新增角色就叫私有角色。

在如图 4.8 所示的私有角色模型[7]中，顶级角色 S 因继承而得到角色 T1、T2、S3、

T3、T4、P3、P 的任务执行权。但是它得不到私有角色 T1'、S3'、T3'、T4'、P3'的任务执行权。T1 的私有角色 T1'也继承得到 T1 中的任务执行权。私有角色 S3'、T3'、T4'、P3'组成了一个私有的子角色层次。这样的私有的子角色层次允许私有角色中的任务在一定范围内被其上级角色继承。

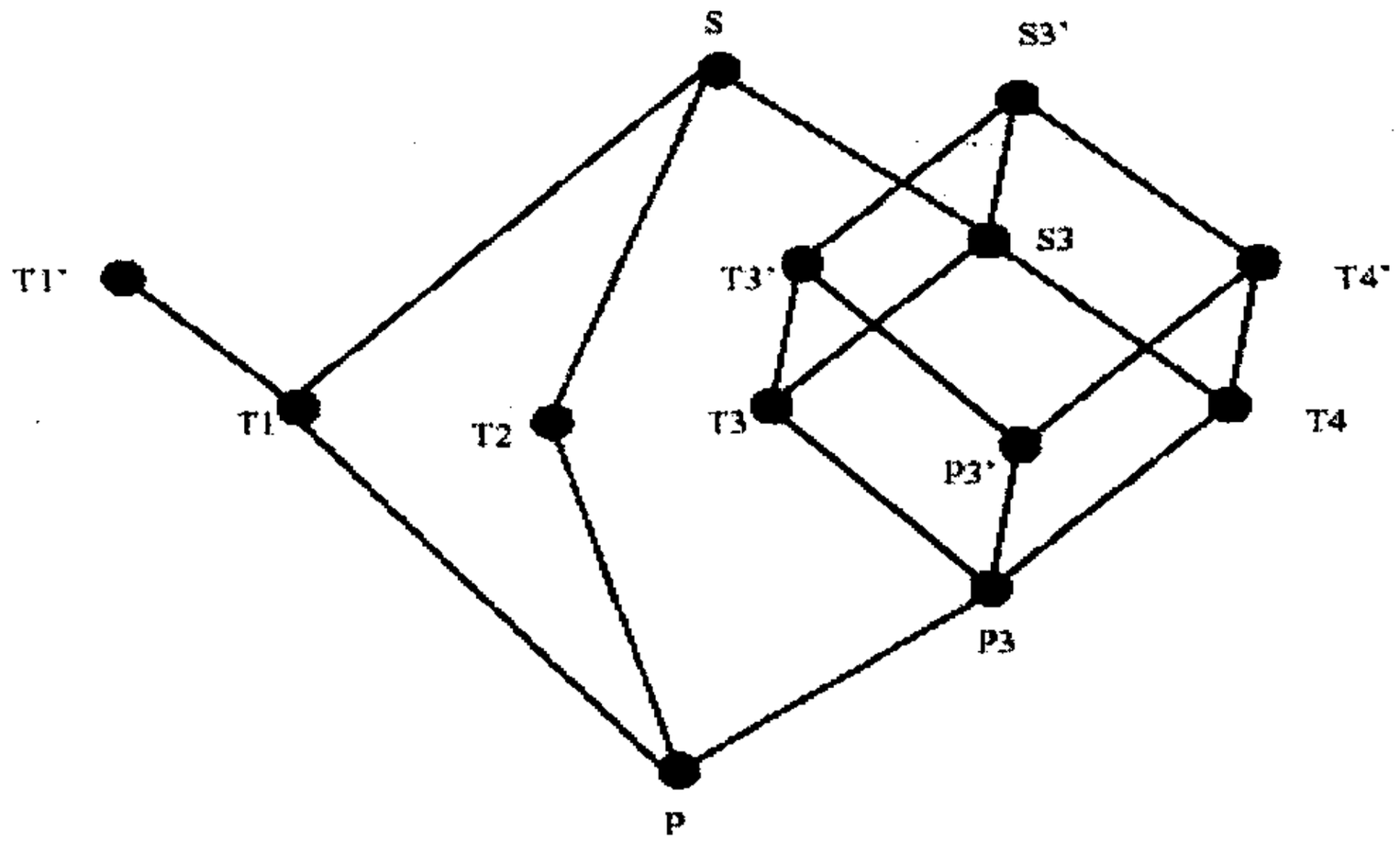

图 4.8　私有角色示意图

4.2.3.3 任务层次

任务包括复合任务和原子任务。复合任务不一定会有实例。如果复合任务可以实例化，那么该复合任务不仅仅代表一个任务集合，而且该集合中的任务之间应该是强相关的（通常是有序的）。相应的复合任务实例只代表复合任务中的任务序列已经启动了，但是，复合任务实例本身并不分配给任何人做。如果管理员将某复合任务授予某角色，则表示该角色中的用户可以执行这个复合任务中的任何原子任务所生成的任务实例。原子任务包含若干人机交互操作的一个动作序列，原子任务实例是原子任务的实例化结果。原子任务和复合任务统称为任务。

任务层次定义为任务集上的偏序关系，同样具有与角色层次类似的性质，这里不再赘述。上级任务只能是复合任务，下级任务可以是复合任务也可以是原子任务。下级任务又可称为上级任务的子任务。

与角色层次不同的是，角色层次中的人员包含关系是下级角色的成员包含上级角色的成员，上级角色成员的权限大于下级角色成员的权限；而在任务层次中，上级任务包含下级任务，赋予下级任务的操作权限大于赋予上级任务的操作权限。

在系统运行过程中，分配给用户执行的是原子任务所生成的实例。如果由于某种原因（比如人手不够）需要将一个上级任务下属的所有原子任务的实例指派给某用户执行，

那么对该用户权限的指派也应该细化到原子任务一级。具体来说就是当用户执行一个任务的时候，系统不是将该任务中所有的原子任务的实例所需的所有资源访问权限都一次性的授予给这个用户，而是当用户启动一个原子任务实例的时候，系统将该任务实例所对应的权限授予该用户，当该任务实例做完之后系统就动态地将这些权限收回。当用户启动下一个原子任务实例时再把相应的权限授予该用户。工程实现的时候，任务的大小应该视具体需求而定。有了这样的原则，就能够方便灵活的根据企业需要定制其所需要的访问控制模型。

4.2.3.4 私有任务

为了进一步方便任务管理，这里引入私有任务的概念。首先定义任务继承的概念，任务继承是指，如果任务 A 继承任务 B，则任务 A 包含任务 B。和私有角色类似，将下级任务中不允许上级任务继承到的子任务独立出来作为私有任务。私有任务概念的引入是很自然的。在企业中，任务的组合往往根据具体业务而定。企业中的新业务往往包含已存在的任务中的部分子任务，当定义此种新业务的时候，必须控制别的子任务不被新业务继承到。

任务层次模型中的每个非叶节点都代表复合任务，可以包含若干原子任务。如图 4.9 所示的任务层次模型表示一个顶级任务 T_{old} 下面包含三个子任务 T1、T2、T3。T1 又包含子任务 T4、T5，T5 包含 T6 和 T7。图 4.9 中所有的任务都是复合任务，该模型中的表示原子任务的叶子节点全部被省略。

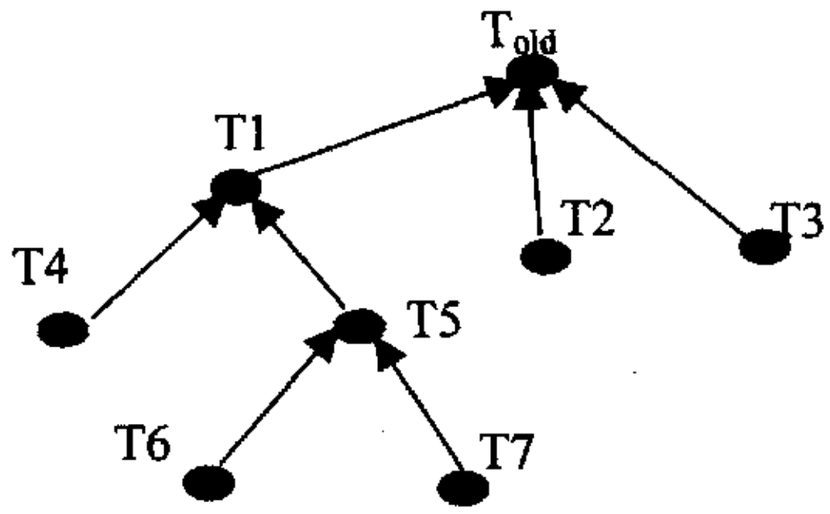

图 4.9　已有任务层次模型

假如图 4.9 所示任务层次为企业中已有的一个任务层次模型，现在企业中要新建一个任务 T_{new}，包含新的复合任务 T8（T8 由两个任务 T9、T10 组成）和 T1 的部分任务组成，而 T1 的这部分允许 T_{new} 继承的任务中的部分任务又分别来自 T4 和 T5。这时，由于 T1、T4、T5 都只是允许其中的部分任务被新任务继承，所以可以用私有任务进行建模。首先将 T1、T4、T5 中的不允许被继承的那部分任务分离出来，形成三个私有任务 T1'、T4'、T5'，而其剩下的允许继承的那部分任务则继续留在任务 T1、T4、T5 中，这样，新任务 T_{new} 就可以继承到它所需要的那部分任务了。所建立的新的任务层次模型如图 4.10 所示。

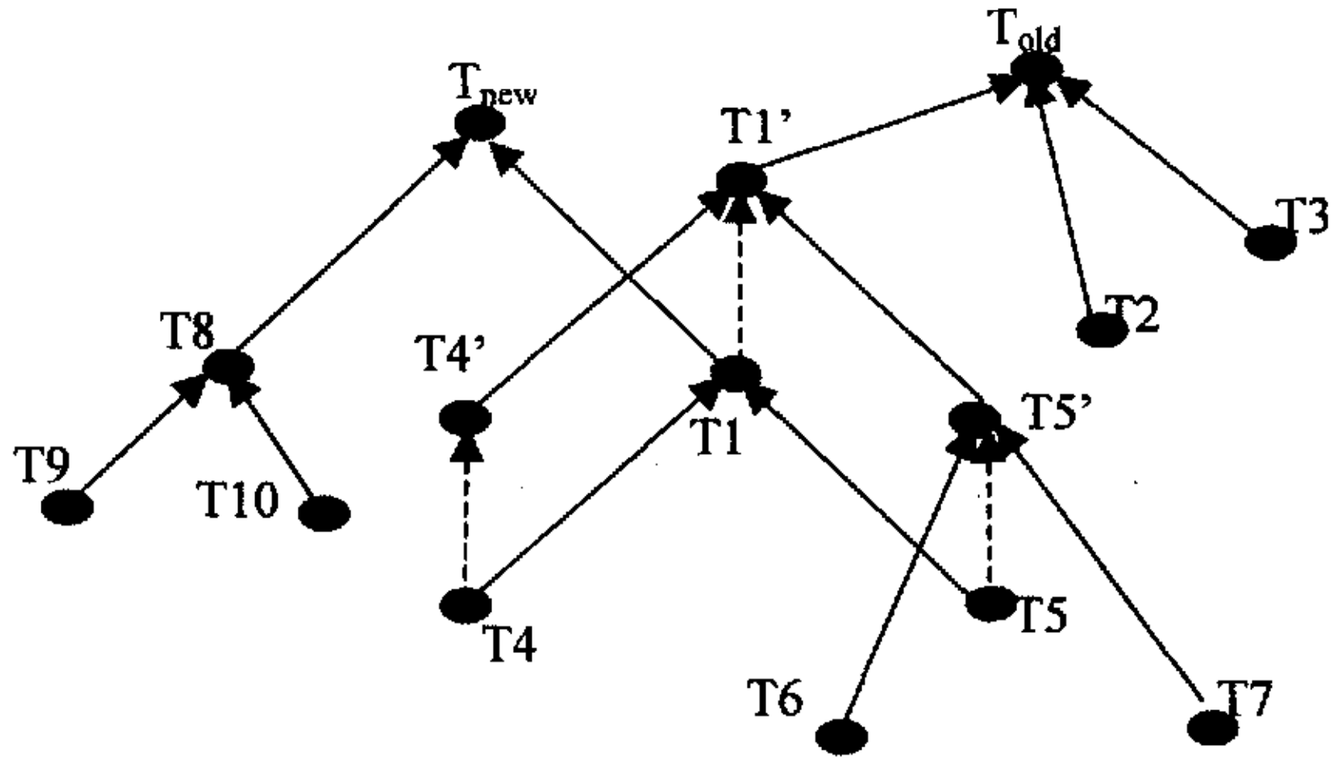

图 4.10　带私有任务的任务层次模型

4.2.3.5 形式化模型描述

层次模型可以形式化表述如下：

- 用户集（U）、会话集（S）、角色集（R）、任务集（TT）、任务实例集（TI）、权限集（P）、用户角色赋予关系（URA）、任务实例映射关系h、任务角色赋予关系（TRA）、实例角色赋予关系（IRA）、任务权限赋予关系（TPA）、实例权限赋予关系（IPA）以及 user 函数定义与基本模型中的定义保持一致。
- 角色层次关系 RH：$RH \subseteq R \times R$，是角色集 R 上的一个偏序关系。
- 任务层次关系 TH：$TH \subseteq TT \times TT$，是任务集 TT 上的一个偏序关系。
- $roles^h : S \to 2^R$，对任意的$S_i \in S$，S_i对应的角色为$roles^h(S_i)$，

$$roles^h(S_i) \subseteq \{r \in R | (\exists r' \in R)[(r \le r') \wedge ((user(S_i), r') \in URA)]\}。$$

- $tasks^h : R \to 2^{TT}$，对任意$r \in R$，

$$tasks^h(r) = \{tt \in TT | (\exists r' \in R)(\exists tt' \in TT)[(tt \le tt') \wedge (r' \le r) \wedge (r', tt') \in TRA]\},$$

$tasks^h(r)$表示可以分配给角色 r 的任务实例所对应的任务模板。

- 可以分配给用户执行的任务实例：

$$taskIns\tan ces^h(u) = \bigcup_{S_i \in S} \bigcup_{r \in roles(S_i)} \{ti | u = user(S_i) \wedge (\exists tt \in TT)[(tt \in tasks^h(r)) \wedge (ti \in \hbar(tt))]\}。$$

- 对任意任务实例$ti \in TI$，ti 所能分配的用户集：

$$executors^h(ti) = \{u \in U | (\exists r' \in R)(\exists r \in R)(\exists tt \in TT)(\exists tt' \in TT)[(r' \le r) \wedge (tt \le tt') \wedge [(ti \in \hbar(tt)) \wedge ((u, r) \in URA) \wedge ((r', tt') \in TRA)]\}。$$

层次模型的示意图如图 4.11 所示。

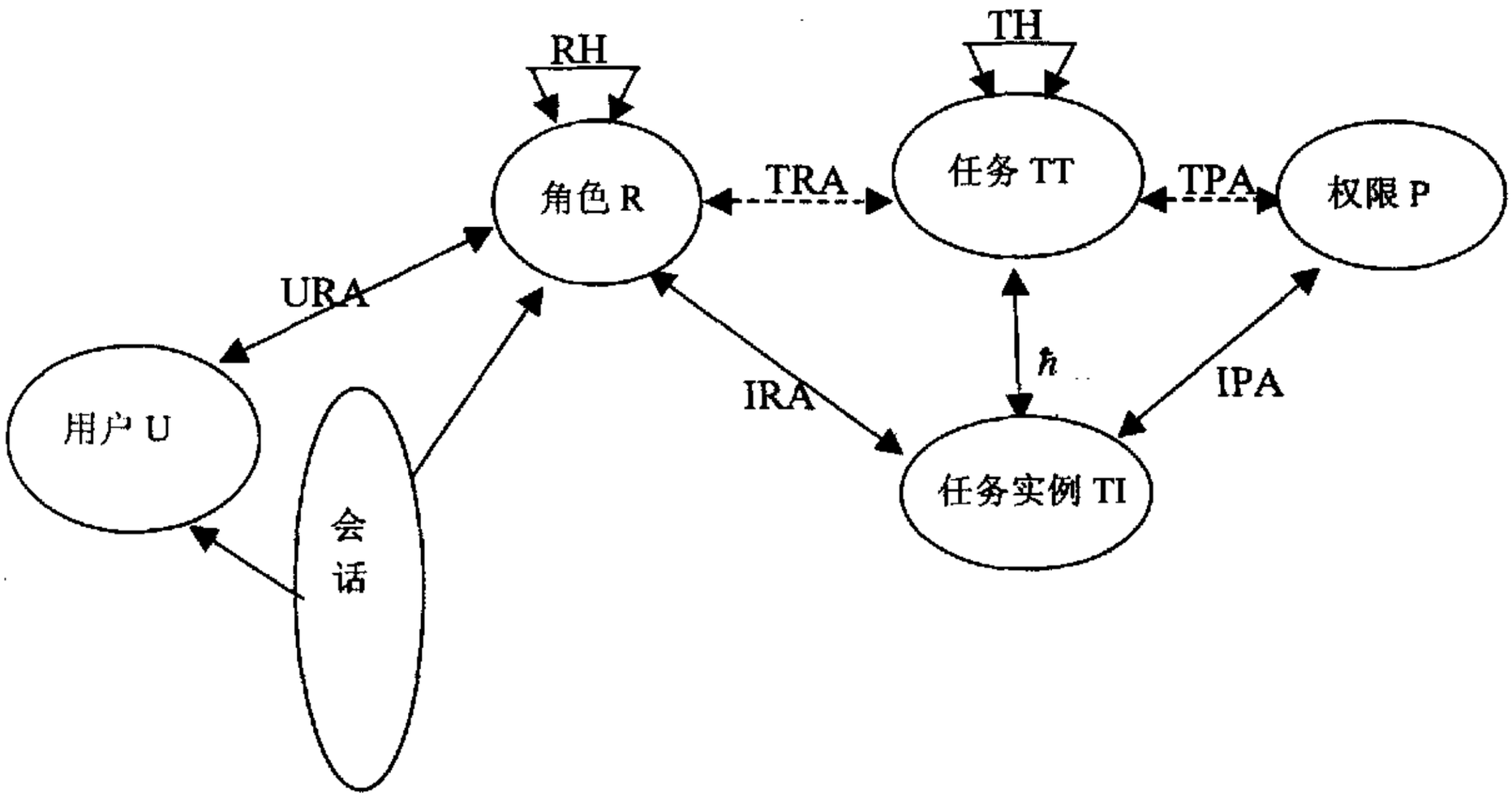

图 4.11　$TRBAC_1$ 模型示意图

4.2.4 约束模型

约束模型是基于任务和角色的访问控制模型族中的第三个模型（$TRBAC_2$），这个模型引入了约束的概念。虽然 $TRBAC_2$ 是该模型族中的第三个模型，但是它是在第一个模型 $TRBAC_0$ 的基础上建立起来的，与第二个模型 $TRBAC_1$ 没有直接联系。

4.2.4.1 职责分离约束

如果两个不同的访问权限 p_i 和 p_j 不允许同一个人拥有，则称这两个权限为互斥权限（mutual exclusive permission），或称权限 p_i 和 p_j 为互斥的，用二元组 (p_i, p_j) 表示，如果有 n 个互不相同的权限是冲突的，则用相应的 n 元组表示。互斥权限集（MP）是一个 n 元组集合（$n \geq 2$），其中的每一个元素都表示互斥权限的一个 n 元组。

如果两个不同的原子任务 tt_i 和 tt_j，tt_i 中的某权限 p_x 和 tt_j 中包含某权限 p_y 为互斥的，或者任务 tt_i 和 tt_j 中所包含的信息不能被同一用户所知晓（数据具有不可流通性），那么就称这两个任务为互斥任务（mutual exclusive task），或称任务 tt_i 和 tt_j 为互斥的，用二元组 (tt_i, tt_j) 表示。互斥任务集（MT）是一个 n 元组集合（$n \geq 2$），其中的每一个元素表示互斥任务的一个 n 元组。

最后再定义互斥角色和互斥角色集的概念。如果两个不同角色 r_i 和 r_j，r_i 所能执行的任务中的某一个任务 tt_x 和 r_j 所能执行的任务中的某一个任务 tt_y 为互斥的，那么就称

r_i和r_j为互斥角色（mutual exclusive role），或称r_i和r_j为互斥的，用二元组(r_i,r_j)表示。同样，互斥角色集（MR）是一个n元组集合（$n\geq 2$），其中的每一个元素表示互斥角色的一个n元组。

根据定义，互斥任务集和互斥角色集可由互斥权限集生成，所以，通常情况下，安全管理员只需要指定系统中的互斥权限集即可。

访问控制中最引人注目的约束是职责分离（Separation of Duty，SoD）约束。从执行任务所涉及的数据（通常是应用数据）来看，任务之间的关系有三种：数据可相互流通型、数据可单向流通型和数据不可流通型。约束模型中的职责分离约束定义为，如果两个任务之间的信息具有不可流通性，则这两个任务不可由同一个用户执行；如果两个任务之间的信息可单向流通，比如，任务A的信息可以流向任务B，而任务B的信息不能流向任务A，则执行了任务A的用户可以继续执行任务B，而执行了任务B的用户不能执行任务A。

职责分离约束分为静态职责分离（Static Separation of Duty，SSD）和动态职责分离（Dynamic Separation of Duty，DSD）两种。如果用户在建模时（主要是在建立URA关系的时候）就保证没有任何互斥角色被分配给同一个用户，那么这就是静态职责分离；如果用户在建模的时候允许互斥角色被分配给同一用户，而在用户登录系统时限制其不能同时激活任何两个（或以上）的互斥角色，那么这就是动态职责分离。不管系统采用哪种约束方式最终都能达到安全控管的目的，然而，静态职责分离不如动态职责分离灵活，而动态职责分离则增加了系统在其它方面的压力（如实现复杂度、运行负载等）。模型实现时采用哪种约束由具体业务需求来定。

对任意任务而言都有一个任务划分约束。该约束定义为，不允许同一个原子任务当中具有两个互斥权限，否则无论如何都会产生安全问题。

4.2.4.2 上下文约束

现实生活中，不同用户执行同一任务时所对应操作对象的范围不一定相同。比如，在一个软件公司中，对同一个“修改软件需求文档”这一操作而言，项目组i中的成员只能修改项目组i所拥有的需求文档，而公司总负责人却可以修改所有项目组的需求文档。

另外，大多数企业对某些任务的执行有一定限制，最常见的是时间、地点限制等。比如，在银行中，正常的业务操作不允许在下班时间进行。对于某些高度机密的任务，企业通常会限制员工在一定场合下执行[29,30]。

怎样才能解决这样的安全问题呢？由于本文所提出的模型中多了一个任务集，其安全机制的关键也在于对任务的执行权限的控制上，所以在约束模型中引入了任务上下文（task context）和用户上下文（user context）这两个概念，它们分别包含了与任务相关的安全信息和与用户相关的安全信息。

任务上下文是对任务是否能被执行所加的各种约束条件，包括与该任务互斥的任

务、执行时间、地点以及管理员自定义的与该任务有关的约束。

用户上下文是用户的当前状态集，包括了当前的用户 ID 和角色 ID、时间、地点、待执行任务列表和已执行任务列表等。

任务上下文包含了执行某一特定任务的实例时所需要的系统环境，而用户上下文提供了用户执行任务实例时的环境。在用户要启动任务实例的时候，系统检查这两个上下文是否满足约束。有了任务上下文和用户上下文，就能够保证某一权限只能在某特定的时间段内、特定的地点、在输入约束条件满足的情况下由具有特定角色的用户所拥有。

需要说明的时，用户只能执行系统分配给他的任务实例。即使用户具有执行某任务 tt 的权限，而且任务 tt 也实例化产生了任务实例 ti，但是，如果系统没有将这个任务实例 ti 分配给该用户，那么他也无法执行这个任务实例，当然也没有这个任务实例所对应的访问权限。

另外，本模型同样具有 RBAC96 模型中定义的基数约束和首备角色约束。

4.2.4.3 形式化模型描述

$TRBAC_2$ 的形式化模型可以表述如下。

- 用户集（U）、会话集（S）、角色集（R）、任务集（TT）、任务实例集（TI）、权限集（P）、用户角色赋予关系（URA）、任务实例映射关系 h、任务角色赋予关系（TRA）、实例角色赋予关系（IRA）、任务权限赋予关系（TPA）、实例权限赋予关系（IPA）以及 user 函数定义与基本模型中的定义保持一致。
- 互斥权限集（MP）、互斥任务集（MT）、互斥角色集（MR）。
- 判断是否互斥的函数：
  - ✓ $isMutexP : P \times P \to \{True, False\}$，$isMutexP(p_i, p_j) = True$，

    当且仅当 $(\exists p_1 \in P)\cdots(\exists p_n \in P)((p_1,\cdots,p_i,\cdots,p_j,\cdots,p_n) \in MP)$；
  - ✓ $isMutexT : TT \times TT \to \{True, False\}$，$isMutexT(tt_i, tt_j) = True$，

    当且仅当 $(\exists tt_1 \in TT)\cdots(\exists tt_n \in TT)((tt_1,\cdots,tt_i,\cdots,tt_j,\cdots,tt_n) \in MT)$；
  - ✓ $isMutexR : R \times R \to \{True, False\}$，$isMutexR(r_i, r_j) = True$，

    当且仅当 $(\exists r_1 \in R)\cdots(\exists r_n \in R)((r_1,\cdots,r_i,\cdots,r_j,\cdots,r_n) \in MR)$。
- $tp : TT \to 2^P$，对于任意 $tt \in TT$，

  $tp(tt) = \{p \in P \mid (tt, p) \in TPA$ 且不存在 p' 使得 $(tt, p') \in TPA$ 且 $isMutexP(p, p') = True\}$，

  $tp(tt)$ 表示单个任务 tt 的最小权限集。
- $roles^c : S \to 2^R$，对任意的 $S_i \in S$，会话 S_i 的当前活跃角色为 $roles^c(S_i)$，

  $roles^c(S_i) \subseteq \{r \in R \mid (user(S_i), r) \in URA$

  且不存在 r' 使得 $(user(S_i), r') \in URA$ 且 $isMutexR(r, r') = True\}$。

- $tasks^c : R \to 2^{TT}$，对任意 $r \in R$，

  $tasks^c(r) = \{tt \in TT | (r, tt) \in TRA$

  且不存在 tt' 使得 $(r, tt') \in TRA$ 且 $isMutexT(tt, tt') = True\}$，

  $tasks^c(r)$ 表示可以分配给角色 r 的任务实例所对应的任务模板。

- $executor^c : TI \to U$，某个任务实例 ti 与某个用户 $executor^c(ti)$ 相关联，表示系统将任务实例 ti 分配给用户 $executor^c(ti)$。

- 用户可以执行的任务实例：

$$taskIns\tan ces(u) = \bigcup_{r \in roles(S_i)} \{ti \in TI | (S_i \in S)[u = user(S_i)] \wedge (r \in R) \wedge$$

$$taskIns\tan ces^h(u) = \bigcup_{S_i \in S} \bigcup_{r \in roles(S_i)} \{ti | u = user(S_i) \wedge$$

$$(\exists tt \in TT)[(tt \in tasks^c(r)) \wedge (ti \in \hbar(tt)) \wedge (executor(ti) = u)]\};$$

约束模型的示意图如图 4.12 所示。

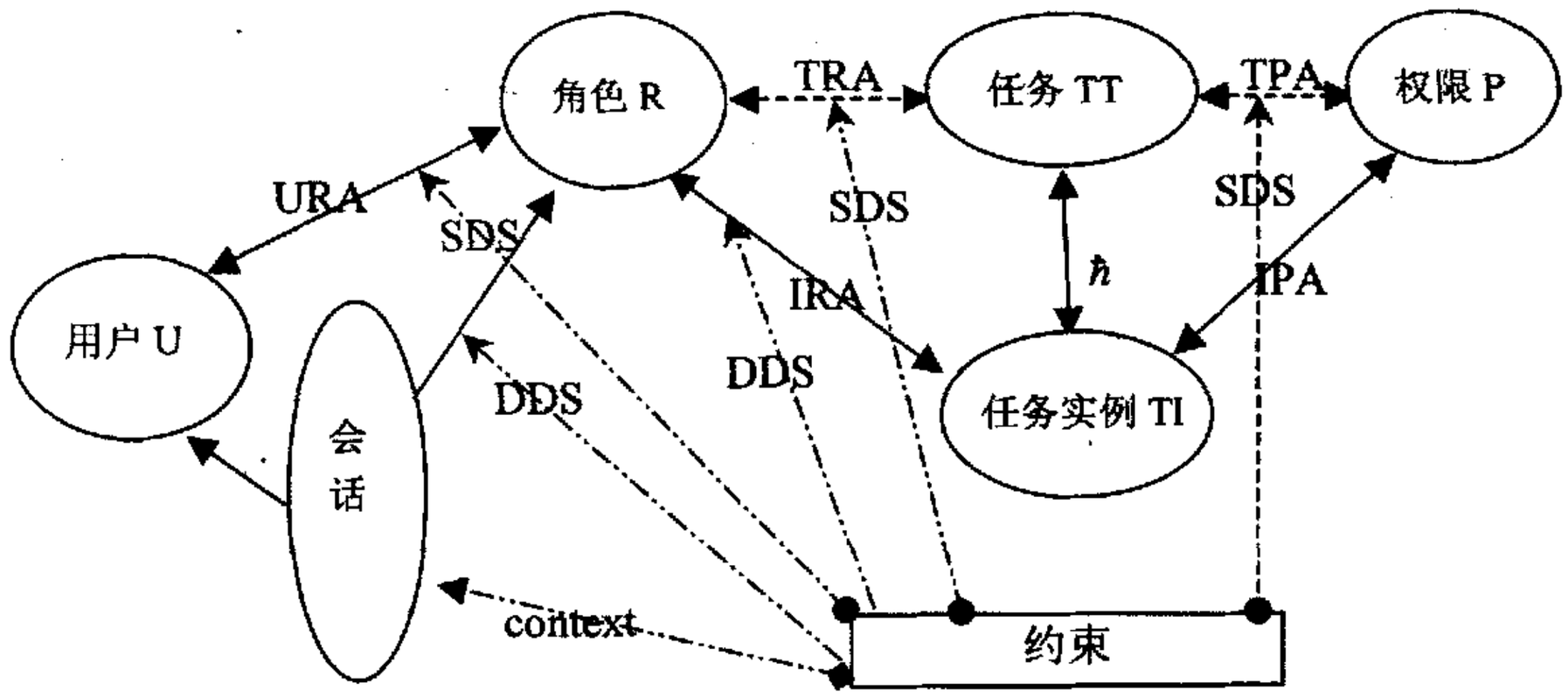

图 4.12　$TRBAC_2$ 模型示意图

4.2.5 高级模型

高级模型是基于任务和角色的访问控制模型族中的第四个模型（$TRBAC_3$），也是其顶级模型。它将前述三个低级模型有机地融合在一起。该模型是 TRBAC 模型族中最完善的模型。

高级模型中既有层次的概念又有约束的概念。$TRBAC_3$ 模型对角色层次、任务层次增加一些约束。如上级角色允许有多少个下级角色，哪些角色不能具有公共的下级角色或上级角色，这些都可以由管理员做出约定。

层次模型和约束模型融合在一起之后会产生一些新问题。比如，在软件公司中，软件开发人员不能测试自己开发的程序，而软件测试人员不能参与开发，这样才能更彻底的达到软件测试的目的。即，开发任务和测试任务是互斥的。一方面，按照层次模型定

义，项目主管作为开发人员和测试人员的上级角色而继承了下级的任务，所以既能做开发又能作测试；另一方面，按照约束模型中职责分离约束的定义，项目主管不能具有这两种互斥任务。这样就可能产生矛盾。如果项目主管只做任何一种任务，那么职责分离约束的问题就可以得到解决。再如基数约束规定某角色中所含的用户数最少或最多或恰好为多少，在没有层次这个概念的时候，要算用户数或任务数当然无任何争议，但是，有了层次这个概念之后，到底要不要把下级的下级角色的个数算进去呢？在高级模型中，没有明确规定该采用哪种方案而是建议根据应用需求做出决定。

高级模型引入监管任务集的概念、按级别授权策略和双重验证策略，增强了系统的安全性。另外，高级模型还允许用户自定义任务分配策略、任务指派方式和委托授权这三种约束。这样就使得以任务为中心的 TRBAC 模型更加灵活。

4.2.5.1 监管任务集

为了保证某些任务实例在任务上下文中只能被用户执行一次，高级模型增加一个监管任务集。监管任务集中的任何一个任务的任务实例都只允许执行一次。系统对这样的任务实例进行额外的“监管”。任务内部定义若干有序的提交点，当提交点的前一个操作完成时系统自动提交，任务不允许回退到前一个提交点之前。每一个提交点即为一个任务状态，且这些状态之间的转换具有不可逆转性，执行失败的任务实例将被删除，如果要再次执行则必须由系统重新实例化并进行任务实例分配。

4.2.5.2 按级别授权

在金融、军事等领域，安全性要求非常高。这时可借鉴强制访问控制的思想，按级别授权。为每个用户、角色、任务、任务组设定一个安全级别。用户是否能执行某任务由它们之间的级别关系决定。按级别授权遵循三个约束条件：

- 基本约束条件：只有当用户的安全级别大于或等于任务的安全级别时，才允许用户执行任务；
- 角色级别约束条件：用户只能拥有小于或等于自身安全级别的角色；
- 任务组级别约束条件：任务只能属于大于或等于自身安全级别的上级任务。

后面的两个约束条件能保证在任务指派时满足基本约束条件。否则，当指定某个角色能执行某任务组的时候，即使角色权限大于或等于任务组级别，也有可能使得角色中某用户的级别低于任务组中某任务的级别，从而造成安全隐患。

4.2.5.3 双重验证

高级模型还允许管理员对具有高安全性要求的资源定制双重验证策略。在通常情况下，用户必须通过一定的角色在执行某任务实例的时候获得该任务实例的资源访问权，用户在发起对某资源的访问请求时，系统验证用户是否具有对该资源的访问权限，这种验证是由于主体的主动操作请求引起的，所以可以称之为主动验证。另一方面，从资源

的角度来讲，当有用户对该资源提出访问请求时，系统也检查该请求是否被允许，这种检查是由于用户的访问造成的，资源是被访问者，所以又可以称之为被动验证。既进行主动验证又进行被动验证的验证策略称为双重验证。

企业中可能会存在一些安全性要求很高的资源，对于这些资源，仅仅主动验证是不够的，必须采用双重验证策略。访问具有高安全性要求的资源时系统进行两次验证，增强了系统的安全性。引入双重验证策略是很有现实意义的，因为，尽管有专门的安全控管人员，但是也难保他们不会因一时疏忽而将一个高安全性资源的访问权限授予某一个本不应该享有该权限的用户或角色，这样就会使得某角色中的一个低级用户有权通过某一任务来获得这个高安全性资源的访问权。

为了实现双重验证中的被动验证，定义权限角色赋予关系，这个关系是权限集到角色集的二元关系，表示允许哪些用户和角色享有特定权限。当用户提出某资源访问请求时，一方面，系统根据其角色、任务、权限之间的映射关系、任务分配策略和任务上下文来判断用户是否能执行这个操作，另一方面，系统根据被动验证策略，也即根据权限角色赋予关系检查是否允许该用户访问该资源。有了这样的被动验证策略之后，系统就更安全了。

4.2.5.4 任务分配策略

为了增强任务指派的灵活性，高级模型引入任务分配策略这个概念。任务分配策略是在静态关系 URA、TRA 的基础上定制的任务分配约束。在进行任务分配时，它可以针对同一个任务模板的不同任务实例，动态地将有权执行该任务实例的用户（角色）集中的部分用户（角色）剔出去，也可以将新用户（角色）加进来。

每个任务可以定制多个任务分配策略。策略分两大类：肯定授权策略和否定授权策略，分别表示允许或禁止相应的用户或角色执行这个任务。每个策略中包含一个约束条件，用布尔表达式表示。将任务实例的属性值代入这个布尔表达式并进行运算，如果该表达式的最终运算结果为 true，那么就表示允许或禁止策略指定对用户或角色做这个任务。如果运算结果为 false，则表示约束条件不成立，系统不考虑该策略对任务指派的影响。

用五元组来表示一个策略：（taskID，policyID，constraint，isPermitted，subjectIDs）。taskID 表示这个策略是针对哪个任务而定制的；policyID 表示该策略是 taskID 所指任务的第几个策略；constraint 是表示策略约束条件的布尔表达式；isPermitted 是个布尔值，表示这个策略是肯定授权策略还是否定授权策略；subjectID 表示该策略针对哪些主体有效，它可能是用户 ID，也可能是角色 ID。

当查找 taskID 所对应的任务的实例的执行者时，taskID 的所有策略都将被考虑。首先将任务实例的属性值代入布尔表达式并进行运算，根据运算结果判断约束条件是否被满足。如果任务实例的属性值不满足约束条件，则表示这个策略无效，系统不考虑这个策略对任务指派的影响；如果任务实例的属性值满足约束条件，则表示这个策略有效。

这时再判断这个策略是肯定授权策略还是否定授权策略，如果是肯定授权，则表示允许 subjectIDs 所对应的用户或角色执行该任务实例；如果是否定授权策略，则表示禁止 subjectIDs 所对应的用户或角色执行任务实例。

4.2.5.5 任务指派方式

任务指派方式是指系统根据 URA 关系、TRA 关系、任务分配策略从整个企业的用户中筛选出可以执行某个任务实例的用户集合之后，将该任务实例分配给这个用户集合中的具体某个用户时所采取的方式。这个方式包括：

1) All：分配给应用所有约束之后仍有权执行这个任务的所有用户。当任务实例需要多人同时执行的时候可以采用这种方式。
2) Least WorkLoad：根据负载平衡原则来选择任务实例的执行者。
3) First Applying First Executing：谁先申请谁先做。这种情况下需要系统先通知有权做该任务实例的所有用户，然后等待用户请求做这个任务实例。
4) High Level：谁的级别最高谁做。
5) Low Level：谁的级别最低谁做。
6) Round Robin：依次让有权做该任务实例的所有用户轮流做。

4.2.5.6 委托授权

在企业中，由于用户生病、出差等种种原因而引起的缺勤是很常见的。这时，就需要将缺勤人员的任务委托给别人去做。高级模型中定义有如下委托概念。

委托者（delegator）：将任务委托给别的用户的用户。

被委托者（delegatee）：接受别的用户的委托任务的用户。

委托撤销(revocation)：一种是委托者撤销，一种是管理员撤销。

委托分为如下几类[31]：

1. 永久/临时委托（permanent/temporary delegation）：永久委托是指委托者将自己的任务永远委托给被委托者。临时委托则指委托只在指定时间段内有效，一旦过期，任务执行权将自动收回。
2. 单一/非单一委托（monotonic/non-monotonic delegation）：单一委托指的是 delegator 在将自己的任务委托给别人后，自己仍然享有执行该任务的权限。非单一委托则指委托者在任务委托期间不再享有执行该任务的权限。
3. 全部/部分委托（total/partial delegation）：指委托者将自己的所有/部分任务委托给被委托者。
4. 单步/多步委托（single step/multi step delegation）：若一个任务只被委托一次则称单步委托，若一个任务被多次委托（如，从 A 委托给 B，又从 B 委托给 C）则称为多步委托。
5. 单人/多人委托（single/multi）：若被委托者只有一个，则称之为单人委托，否

则称为多人委托。

6. 单向/双向委托(bilateral/unilateral)：若只是委托者单方面决定将任务委托给被委托者，则称为单向委托；若委托需得到双方同意，则称为双向委托。
7. 自管理/代理委托(self-acted/agent-acted)：表明该委托由自己来控制还是由他人来控制。

委托过程中需考虑以下问题：

1. 委托不允许发生在同一角色中，这是无意义的。
2. 要保证约束在委托发生之后仍然有效，委托任务的执行条件（除了 user 和 role 条件外）不变。
3. 根据具体业务需求考虑委托过程的控制（如：防止主体任务负载过重或权限过大）和被委托者是否能拒绝等问题。
4. 系统必须对委托进行日志记录。

4.2.5.7 管理策略

TRBAC 模型也能对自身进行管理，它将管理员用户（Administrator Users）、管理员角色（Administrative Roles）、管理任务（Administrative Tasks）、管理权限（Administrative Permissions）与普通的用户、角色、任务、权限完全分离开来。一方面，普通用户不允许做管理员任务，另一方面，它也限制管理员不能做除了管理任务之外的其他任务。这一点与传统的基于角色的访问控制[7]中的做法是一致的。

管理员执行管理任务的情况与普通用户执行任务的情况完全一样，也需要实例化任务，只不过该任务是比较特殊的管理任务（比如角色管理，任务管理等）而已。

4.2.5.8 形式化模型描述

$TRBAC_0$、$TRBAC_1$、$TRBAC_2$ 中的各定义均有效，除此之外，$TRBAC_3$ 还有如下定义。

- 级别集合 L：该集合中的元素之间可以相互比较大小。
- 监管任务集 ST：该集合中的每一个任务的实例都只允许执行一次。
- 高安全性操作权限集 SP：该集合中的每一个操作权限都将进行双重验证。
- 权限角色赋予关系 PRA：$PRA \subseteq P \times R$，权限集与角色集的映射关系。
- 任务指派方式 TAM：

$$TAM = \{all, leastLoad, FAFE, highLevel, lowLevel, roundRobin\}$$。

- $levelU : U \to L$，对任意 $u \in U$，$levelU(u) \in L$，$levelU(u)$ 表示用户 u 的级别。
- $levelT : TT \to L$，对任意 $tt \in TT$，$levelT(tt) \in L$，$levelT(tt)$ 表示任务 tt 的级别。
- 按级别授权满足：$canExecute(u,tt) = true$，当且仅当用户 u 的级别大于或等于任务 tt 的级别。

- 任务分配策略 policy：

$$policy = (taskID, policyID, constraint, isPermitted, subjectIDs)。$$

- 任务分配策略中的约束表达式解析：$parse(ti, constraint) = true$，当且仅当任务实例中的属性值满足约束表达式。
- 任务分配策略有效时与策略相关的用户：

$$getUserOfPolicy(ti, policy) = \{u \mid (pase(ti, policy.constraint) = true) \wedge [(u \in policy.U) \vee (\exists r \in policy.R)[(u, r) \in URA]]\}$$

其中，ti 表示一个任务实例，policy 表示 ti 的一个策略，policy.constraint 表示策略中的约束表达式，policy.U 表示策略指定的用户集合， policy.R 表示策略指定的角色集合。

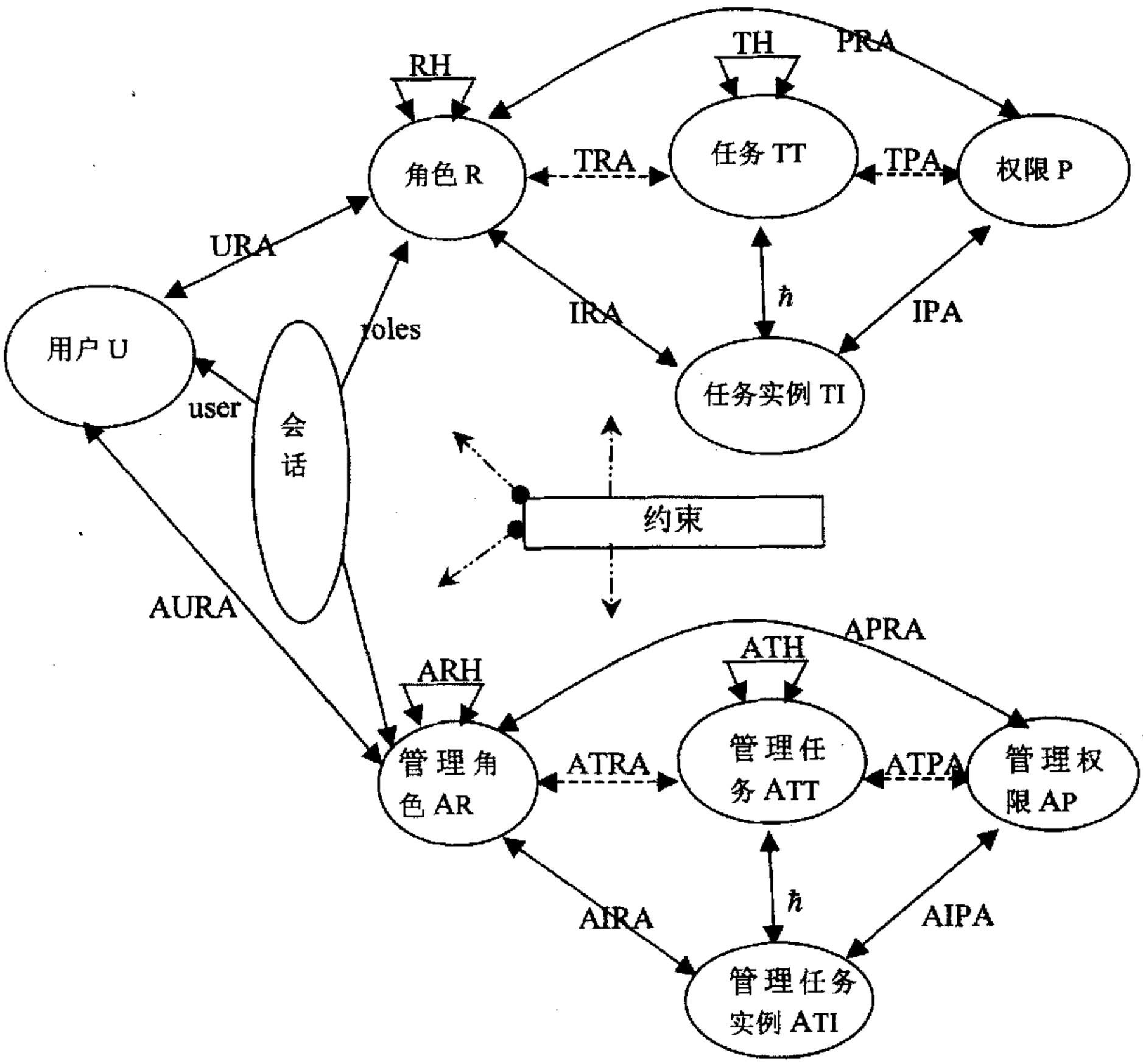

图 4.13　$TRBAC_3$ 模型示意图

- 被动验证：$isPermitted(u, p) = true$，

当且仅当，$(\exists r \in R)[((u, r) \in URA) \wedge ((p, r) \in PRA)]$ 成立。

- AU、AS、AR、ATT、ATI、AP、AURA、ATRA、AIRA、ATPA、AIPA、ARH、ATH、ARPA 等的定义及其相关的定义与模型中已有的相应定义类似，如，ATT 表示管理任务，ATRA 表示管理任务与管理角色的赋予关系，等等。

$TRBAC_3$ 模型的示意图，如图 4.13 所示。

4.2.6 小结

在基于任务和角色的访问控制模型中，用户和任务被分别赋予角色，而权限则被赋予任务。用户在会话过程中，以某一角色执行一项任务，从而获得该任务所具有的权限。

TRBAC 模型构建的 4 层访问控制结构将访问控制的粒度控制在任务一级。在工作流系统中，将任务实例授权给某用户，就意味着将执行这个任务实例所需要的全部资源访问权限授予该用户。可见 TRBAC 的访问控制粒度与工作流系统的这种访问控制粒度要求是一致的。

在高级模型中，定义了许多可选安全组件，比如任务分配策略、安全级别等，有了这些组件，系统会更安全、更灵活的运作。

TRBAC 模型图在传统的 RBAC 模型图的基础上增加了任务、任务实例等组件。图中的虚线表示在实际的工作流系统运行过程中，角色只能执行任务实例，也只有任务实例才真正拥有任务所对应的权限，这就保证了角色所拥有权限的动态授予和撤销。

虽然 TRBAC 模型因增加了几个组件看起来比传统的访问控制模型复杂，但是它的建模能力以及安全性能却提高了。在强调任务特性的工作流系统中，其优越性尤其明显。TRBAC 模型还简化了管理员的安全维护工作。

第五章 工作流系统安全模型实现

§5.1 工作流基础件系统

5.1.1 工作流基础件系统介绍

在攻读硕士学位期间，作者所在课题组在863项目（面向金融领域的分布式工作流关键技术及应用框架）的资助下一直致力于构建一个基于J2EE的工作流基础件系统平台。以下将这个平台简称为工作流基础件系统（Workflow Infrastructure System，WFIS）。该平台在金融业务流程处理领域具有通用价值，能够为多种金融业务应用系统提供二次开发工具集和运行环境。

该平台应具有以下功能：

- 以可视化方式支持金融业务处理流程的直观定义，最大限度地实现业务流程的自动化，支持对业务流程的实时监控；
- 内嵌通用功能，尽量简化二次开发工作；
- 提供便捷的系统配置、管理设施。

WFIS的目标应用领域为金融行业（包括银行、证券、信托、财务、保险等），系统的使用者包括客户、机构职员、机构主管及系统管理员。要求该系统具有先进性、可靠性、开放性、安全性、可扩充性和可维护性。

WFIS的客户端环境主要是主流的PC机上安装的Windows操作系统。使用者可采用浏览器（如IE或Netscape Navigator）或瘦客户端登录系统进行功能操作。未来，使用者可能在Internet上通过多种通道，如浏览器（Browser）、手机（Mobile）、PDA、交互式语音处理系统（IVRS）等访问本系统提供的功能。

WFIS与其它系统的关系如图5.1所示。

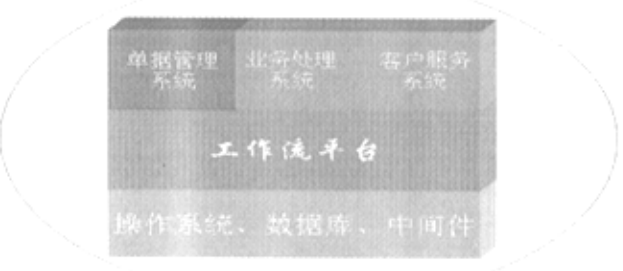

图 5.1 WFIS与其它系统的关系示意图

5.1.2 工作流基础件系统框架

本文作者所在课题组成员根据需求分析结果、安全设计指导思想和设计原则，设计了工作流基础件系统。其框架结构示意图如图 5.2 所示。

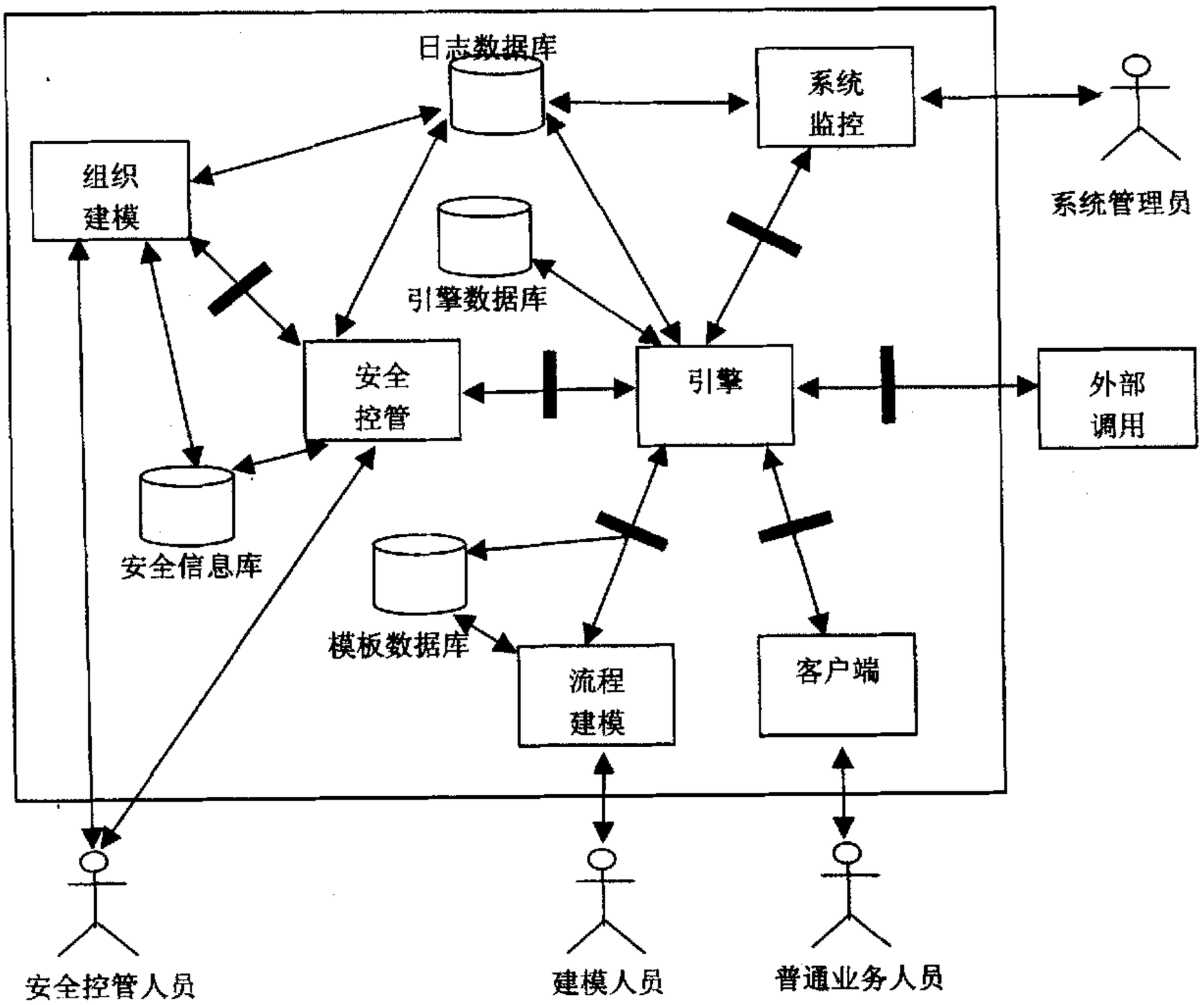

图 5.2 WFIS 框架结构示意图

流程建模模块为企业流程建模人员提供一个图形化的工作流建模工具。引擎负责对该模块所提供的流程模板进行解析并执行。组织建模模块为安全控管人员提供组织建模的功能，安全控管模块根据已有的组织模型和工作流模板设置相应的访问控制策略和约束。系统监控端则主要负责对整个工作流系统的运行状态以及日志进行监控。普通业务员可以通过客户端在进行一定的认证程序登录系统之后执行系统分配的任务实例。系统还可以调用外部应用。

所有模块都对外提供专用接口，各模块之间只能通过各自提供的外部接口相连。

当用户要执行某敏感操作或某敏感操作被调用的时候，引擎就调用安全控管接口中的相应方法进行验证，以决定是否准许执行或调用相应操作。除了专门的安全控管人员外，任何人都不允许对安全信息库进行操作。

安全控管模块主要实现基于任务和角色的工作流管理系统中的访问控制。下面对这种实现方法进行详细讨论。

§5.2 模型实现

在 WFIS 中，TRBAC 模型主要由组织模型模块和安全控管模块来实现。这两个模块是工作流系统中必不可少的两个重要模块，同时，它们又相对独立于工作流引擎。

组织模型模块主要负责整个系统的组织模型的建立以及维护，并为安全控管模块提供必要的检索功能。

安全控管模块主要负责工作流系统中的安全控制和管理，并为引擎提供必要的任务指派、权限判断等功能。

由于维护企业组织模型的工作量比较大且相对独立，为了减轻安全控管人员的负担，更合理的职责划分，将组织建模从安全控管中分离出来。组织模型建模人员只需要清楚本企业的人员组成情况就可以了，而安全控管人员在已有组织模型的基础上，对这些人员进行任务指派。否则，就必须要求安全控管人员不仅对本企业人员的组成情况了若指掌，而且还对本企业的任务划分情况非常熟悉。另外，WFIS 中的组织模型为通用模型，相对来说比较复杂，组织模型独立出来之后便于部分中小型企业根据自身特点设计更简单的组织模型。

TRBAC 模型的实现可简单描述为：定制 UserInfo 表、UserGroup 表、UserToGroup 表、RoleHierarchy 表存储企业内的组织模型信息；定制 TaskInfo 表、TaskGroup 表、TaskToGroup 表、TaskHierarchy 表以及建模客户端的包表、模板表、任务表存储系统中的任务及其层次信息；定制 TaskAssign 表存储任务指派关系；定制 Policy 表存储任务分配策略。管理员利用系统的组织模型和任务层次模型进行静态任务指派（结果存于 TaskAssign 表中），然后为特殊的任务定制任务分配策略（结果存于 Policy 表中）。在运行期间，系统进行任务实例指派时先检索静态任务指派关系，再检索任务分配策略，如果该实例所对应的任务存在任务分配策略，则考虑策略的影响，得到一个可以执行实例的执行者集合。最后根据任务指派方式选择一个或多个任务执行者来执行任务。

5.2.1 组织模型

在工作流模型中建立组织模型的目的是在计算机中刻画和表示组织结构的现实情况，并便于查找和维护。

各企业的组织模型一般比较复杂，关注的重点各不相同。组织建模模块的主要功能有：

- ✧ 管理组织模型中的各个实体，如添加、删除、修改用户及用户组。
- ✧ 建立并维护组织模型中各实体之间的关系，如：用户—用户组关系、用户组内部的关系等。

- ✧ 关于用户、用户组的检索功能，如：检索用户信息、检索用户所属用户组、检索用户组内的所有用户等。

5.2.1.1 角色扩展

由于 WFIS 不是只用于某一个企业而是面向一类企业，所以设计的组织模型应该能对所有企业的人员组织关系进行建模。为此，定义如下概念：

- ✓ 机构：地理上分散在各地的一个单位，如总行、分行、支行等，可划分为多个子机构。
- ✓ 部门：隶属于机构的下级单位，如信贷部，研发部，市场部等，可划分为多个子部门。
- ✓ 工作组：可能隶属于某个机构或某个部门，因某项临时（或特殊）业务而存在，可划分为多个子工作组。
- ✓ 角色：是指组织内具有一定技能、可以执行某些任务的人员集合。通过给员工赋予不同的角色，对员工的多职能、多权限进行表达。
- ✓ 用户：即为本系统的使用者，包括企业的内部员工和外部客户。

从本质上讲，机构、部门、工作组、角色都是用户的集合，可以将它们统称为用户组。组织模型中各实体的关系如图 5.3 所示。

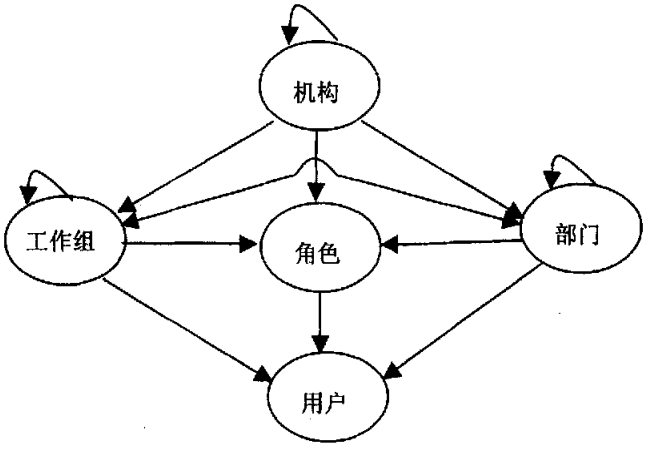

图 5.3 组织逻辑模型

图 5.3 刻画了组织模型，每个箭头代表一对多的关系。机构由若干个部门和工作组构成，部门通常表示纵向的行政隶属关系，其存在较为稳定。工作组通常表示横向的合作关系，可以随业务需要而变动。不仅每一个部门或工作组都可细分为多个下级部门或工作组，而且工作组可以包含部门，部门也可以包含工作组，这样就形成了组织的层次结构。一个简单的银行组织模型如图 5.4 所示。

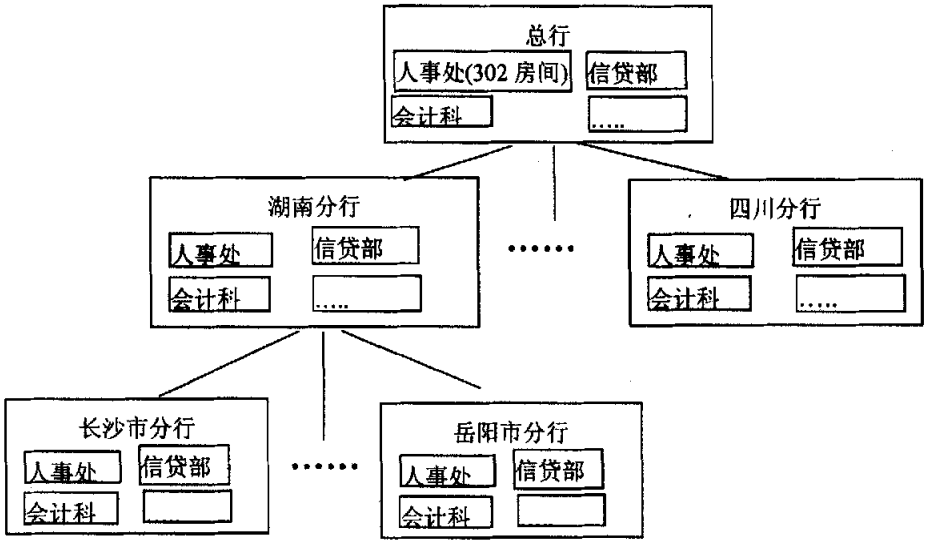

图 5.4 组织模型建模示例

5.2.1.2 用户状态

组织模型定义的用户状态主要包括以下几种：

- USER_STATUS_ENABLE 有效状态；
- USER_STATUS_DISABLE 无效状态：意味着该用户可能已被删除或处于长时间不用状态；
- USER_STATUS_LOCK 临时锁定状态：表示暂时不允许该用户登录到系统中来；
- USER_STATUS_DELEGATOR 委托状态：将任务委托给别人时的状态；
- USER_STATUS_DELEGATEE 被委托状态：接受别人委托的状态；
- USER_STATUS_SIGNIN 签入状态：表示用户正处于在线状态；
- USER_STATUS_DELEIN 委托签入状态：表示用户即处于委托状态又处于在线状态。
- USER_STATUS_SIGNOUT 签出状态：表示用户处于离线状态；
- USER_STATUS_DELEOUT 被委托签出状态：表示用户既处于被委托状态又处于离线状态。

委托状态隐含签出状态，被委托状态隐含签入状态。当用户处于委托状态时，也可能需要登录系统察看一些信息，所以设置了委托签入状态。

当用户处于无效状态或临时锁定状态时，不能分配任务给该用户；当用户处于委托状态时，要进一步查找该用户的被委托者，如果这个被委托者确实处于被委托状态或被委托签出状态，则将本该分配给委托者的任务分配给被委托者，如果查出来的被委托者不处于被委托状态或被委托签出状态，则抛出异常。

5.2.1.3 数据库表描述

这个模型将会在数据库中维护 4 张表。

- USERINFO 用户信息表：这个表负责记录用户的负载、级别及状态等信息。
- USERGROUP 用户组信息表：这个表记录系统中所有用户组（机构、部门、角色、工作组）的信息（用“groupType”字段表示用户组类型）。
- USERTOGROUP 用户—用户组映射关系表：这个表记录用户组下有哪些用户。
- ROLEHIERARCHY 角色层次关系表：该表包含两个字段：superiorID（上级用户组 ID）、inferiorID（下级用户组 ID），记录了用户组的上下级关系，反映了企业的整个组织模型。

角色层次关系表记录了所有的用户组上下级关系。也可以在用户组表中增加一个字段来记录其上级 ID。但是，在现实世界中，用户组可能会有多个上级用户组。面对这种情况时，在用户组表中增加上级 ID 字段的方法将难以处理。上级用户组的用户自然成为下级用户组的用户，这是通过继承关系确定的，而用户组之间的所能做的任务集合是自上向下继承的。越到下层，用户组所拥有的用户数就越大，越到上层，用户组所能执行的任务就越多。

5.2.1.4 外部接口描述

组织模型模块向外界提供的主要接口如图 5.5 所示。

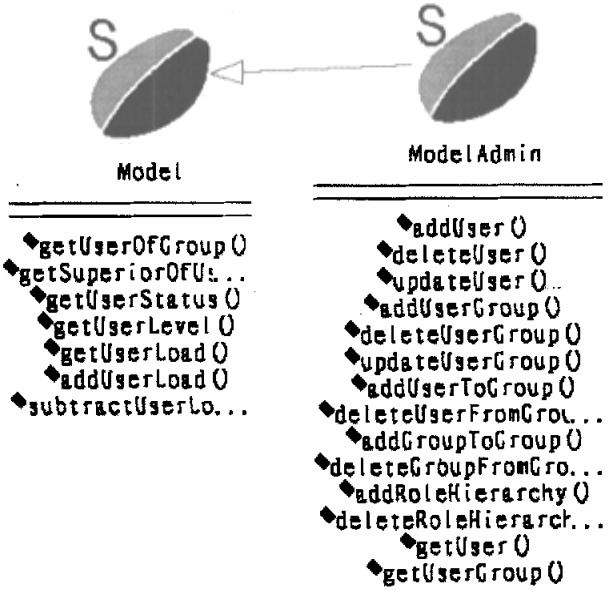

图 5.5 组织模型模块外部接口

组织模型模块对外提供的接口放在 Model 中，它主要实现本模型的检索功能；对内提供的接口放在 ModelAdmin 中，它主要实现对模型自身的维护功能。

- Model 接口
  - 向安全控管模块提供 5 个方法：
    - ✓ getUserOfGroup()：获取某个用户组内的所有用户；
    - ✓ getSuperiorOfGroup()：获取某个用户组的所有上级用户组的 ID；
    - ✓ getUserStatus()：获取某个用户的状态，在任务指派时，判断用户是否处于有效状态或委托状态，如果用户处于委托状态，则进一步查找用户的被委托人；
    - ✓ getUserLevel()：获取某个用户的级别，当系统按级别授权（任务指派方式为 High Level 或 Low Level）时调用该方法；
    - ✓ getUserLoad()：获取某个用户的负载情况，当系统考虑用户负载平衡（任务指派方式为 Least WorkLoad）时调用该方法；
  - 向引擎提供 2 个方法：
    - ✓ addUserLoad()：当将某个任务实例分配给用户时，增加该用户的负载；
    - ✓ subtractUserLoad()：当用户提交某个任务实例时，减少该用户的负载。
- ModelAdmin 接口

  该接口提供了构建及维护组织模型的所有方法，部分方法如下：
  - ✓ addUser()：添加一个用户；
  - ✓ deleteUser()：删除一个用户；
  - ✓ updateUser()：更新一个用户的详细信息；
  - ✓ getUser()：获取一个用户的详细信息；
  - ✓ addUserGroup()：添加一个用户组；
  - ✓ deleteUerGroup()：删除一个用户组；
  - ✓ updateUserGroup()：更新一个用户组的详细信息；
  - ✓ getUserGroup()：获取一个用户组的详细信息；
  - ✓ addUserToGroup()：将用户加入某个用户组；
  - ✓ deleteUserFromGroup()：将用户从用户组中删除；
  - ✓ addRoleHierarchy()：在角色层次上增加一个层次关系（用户组的上下级关系）；
  - ✓ deleteRoleHierarchy()：在角色层次中，删除一个层次关系。

5.2.2 安全控管

组织模型模块为企业组织模型建模，为访问控制提供了企业内人员的组织结构信息。安全控管模块在已有组织模型的基础上实现访问控制。设计安全控管模块的目的可归纳为：

- ✧ 对人员的任务指派能够简捷、无错误、直观地完成。

- ✧ 便于工作流引擎在运行时进行任务动态分配，使系统具有灵活性和适应性。
- ✧ 为引擎提供获取任务执行者以及操作是否允许的接口。

安全控管模块的主要功能有：

- ✧ 建立并维护任务组信息。
- ✧ 建立并维护静态任务指派关系，如任务角色赋予关系等。
- ✧ 建立并维护动态任务分配关系，如任务分配策略、任务代理信息等。
- ✧ 关于以上所维护安全控管信息的检索功能，如检索某一任务实例的实际执行者等。

5.2.2.1 任务层次

管理员必须针对企业组织模型和工作流程定制相应的角色任务赋予关系和任务权限赋予关系。为此，管理员必须先建立企业的任务层次，确定系统中的互斥权限，进而确定互斥任务。

在 WFIS 系统中，将每个工作流程看作是这个流程内任务的上级任务，将包看成是包内流程的上级任务。流程建模完成之后，就有了相应的企业任务层次模型。除此之外，该模块还允许管理员对系统已有的任务自由分组并建立额外的任务层次，从而使任务指派更方便。

任务层次是自上向下包含的，而执行任务所需的资源访问权限是自下向上包含的。也即是，上级任务包含了下级任务，而下级任务所具有的资源访问权限大于或等于上级任务所具有的资源访问权限。需要说明的是，上级任务并不真正“具有”这些访问权限，允许管理员将权限赋予他们只是为了减少 TPA 关系的指定次数。比如，管理员可以将包或模板中所有任务都具有的公共的资源访问权限赋予相应的包或模板，在管理员真正为包或模板内任务指定资源访问权限的时候，就不必再指定这些公共权限，系统在运行时可以根据任务层次之间的继承关系来找到执行任务所需要的所有权限。

5.2.2.2 制定约束

该模块具有 4.2.4 节定义的所有约束。允许管理员对任务定制任务上下文约束。比如，管理员可以指定执行某一个特定任务的时间、地点约束等。

WFIS 中的任务权限赋予关系分以下三种：

1. 无权限：在这种情况下，任务执行者不需要访问系统的任何数据资源就可以完成任务。
2. 允许权限：这种方式明确规定执行某一任务的实例时所拥有的操作访问权限。
3. 禁止权限：这种方式规定执行某一任务的实例时所禁止拥有的操作访问权限。

对于第 2、3 种情况而言，任务都对应一个权限列表。这个表中的数据都会保存到

数据库中的任务权限关系表。另外，要根据最小权限约束为任务分配权限。

允许管理员定制任务的任务分配策略以及指定任务指派方式。任务指派方式详见4.2.2.4 节。任务分配策略在数据库中用 6 个字段来表示，分别是 taskID，policyID，policyName，constraint，isPermitted，description，其中，policyName 表示这个策略的名称，description 表示这个策略的描述信息，其余字段与 4.2.5 节定义一致。另外再用一张表来保存与策略关联的用户或角色，该表有 4 个字段：taskID，policyID，objectID，objectType，其中，objectType 表示客体的类型，可以是用户或角色。定制任务分配策略的界面如图 5.6 所示。

图 5.6 任务分配策略界面

5.2.2.3 强制访问控制

由于金融领域安全性要求较高，因此，采用按级别授权的策略。每个用户以及每个任务分别分配一个级别。按级别授权包括两方面的含义：首先，只有当用户级别大于或等于任务级别时，才允许用户执行这个任务的任务实例。这是强制访问控制的一个简单实现。这个功能作为一个可配置选项提供，管理员可以指定系统是否按级别授权，默认情况下，系统把它作为进行任务实例指派时的一个基本约束条件加以考虑。其次，如果某任务的任务指派方式为 High Level 或 Low Level，这时系统在进行相应任务实例的指派时将选择最高级别或最低级别的用户。当按级别授权时本模块会调用组织模型模块的 Admin 接口所提供的 getUserLevel（）方法。

5.2.2.4 委托授权

在企业中，由于用户生病、出差等种种原因而引起的缺勤是很常见的。这时就需要将缺勤人员的任务委托给别人去做[31]，详细定义见 4.2.5.6 节。

WFIS 系统采用的是临时、非单一、全部、单步、单人、单向、代理委托。

在 WFIS 中被委托者不能拒绝委托，也没有永久委托。如果需要永久委托，那么说明这个用户从任务指派这个角度来讲，已经没有存在于系统中的价值，所以，可以由管理员重新定制相应的任务角色赋予关系来达到永久委托的目的。

这里的代理委托是指委托统一由管理员来管理。委托和委托撤销都必须通过管理员进行，职员之间不能私自进行委托。

5.2.2.5 数据库表描述

在该模块中使用了流程建模端维护的三张表：

- TASKINFO 任务信息表：包含了任务的级别、指派方式等信息；
- TASK_TEMPLATE 记录任务与模板的映射关系；
- TEMPLATE_PACK 记录模板与包的映射关系。

通过这三张表可以查到任务详细信息和任务所属模板和包。该模块还维护以下表格：

- TASKGROUP 任务组表：存储管理员为方便任务指派而建立的任务组；
- TASKTOGROUP 任务与任务组的映射关系表：存储任务组中所包含的任务；
- TASKHIERARCHY 任务层次关系表：存储任务组层次关系，该关系不包括流程建模端建立的层次关系（包－模板－任务）；
- TASKASSIGN 任务指派表；
- POLICY 任务分配策略表；
- POLICYSUBJECT 任务分配策略与用户（用户组）的映射关系表；
- PERMISSION 权限表；
- DELEGATION 委托表：记录系统中的委托授权信息；
- OBJECTINFO 资源信息表；
- TASKCONTEXT 任务上下文表；
- OBJECT_ROLE 资源访问权限与用户组的映射关系。

任务指派表中记录管理员将任务（任务组）指派给用户（用户组）的信息。有 4 个字段：subjectID、subjectType、objectID、objectType。其中，主体类型（subjectType）包括用户和用户组两种，客体类型（objectType）包括任务、任务组、模板、包四种。

任务分配策略表也根据任务分配策略的定义而定制了相应的字段，包括 taskID、policyID、constraint、isPermitted 等字段。其中 constraint 是一个字符串型数据，用于记录约束表达式。系统在处理策略的时候，首先用专门的布尔表达式解析器解析这个约束表达式，然后再进行一系列的处理。

委托表记录系统中的委托事件。该表主要包括 delegatorID、delegateeID、validFrom、validTo 等字段，其中，validFrom、validTo 为日期型数据，表示该委托事件的有效时间段。如果 validTo 字段为空，则表示该委托事件在管理员人为撤销前一直有效。

资源信息表记录系统中各个敏感资源的信息。该表主要有 ID、name、accessType、

path 等 4 个字段，其中访问类型（accessType）包括对数据库表的读、写、读写操作以及对某功能模块（函数）的调用等。当访问类型为数据库表的读、写、读写时，path 存储数据库表的数据库路径，当访问类型为功能调用时，path 为该函数的类路径和方法名构成的功能调用路径。

任务上下文表中记录了管理员为某特定的任务定制的上下文约束，如果用户当前执行环境不能满足这个上下文，那么不允许用户做该任务所对应的任务实例。该表包括 taskID、validFrom、validTo、IP 等字段，可以对任务执行的时间或地点进行约束。

权限表记录了系统中的权限信息，主要有 subjectID、subjectType、accessType、isPermitted、objectID、objectType 等字段。

客体类型（objectType）包括 6 种：日志、引擎、流程模板、任务、功能模块（函数）、数据库表。

访问类型（accessType）包括 10 种：任何访问、读、写、读写、启动、暂停、重启、停止、监视、调用。

主体类型（subjectType）包括 5 种：用户、用户组、事件（触发）、父流程（调用）、定时器（触发）。

允许标识（isPermitted）为布尔类型。若 isPermitted 值为 true，则表示允许权限；若 isPermitted 值为 false，则表示禁止权限。

当客体类型为“日志”时，访问类型只能为“读”，此时，客体 ID（日志 ID）只能为：

- 1 表示引擎日志；
- 2 表示工作流建模日志；
- 3 表示组织建模日志；
- 4 表示安全控管日志；
- 5 表示客户端日志；
- 6 表示系统管理日志。

当客体类型为“流程模板”且访问类型为“启动”时，主体类型为：

- “用户”或“用户组”对应流程模板的人工启动方式，此时主体 ID 为用户 ID 或用户组 ID；
- “事件”对应流程模板的事件触发启动方式，此时主体 ID 为能发送该事件的用户 ID；
- “父流程”对应流程模板的父流程调用启动方式，此时，此时主体 ID 为能该该流程模板的父流程模板的 ID；

✧ “定时器”对应流程模板的父流程调用启动方式，此时，此时主体 ID 为用整型表示的时间，如 20031206。

5.2.2.6 外部接口描述

安全控管模块向外界提供的主要接口如图 5.7 所示。

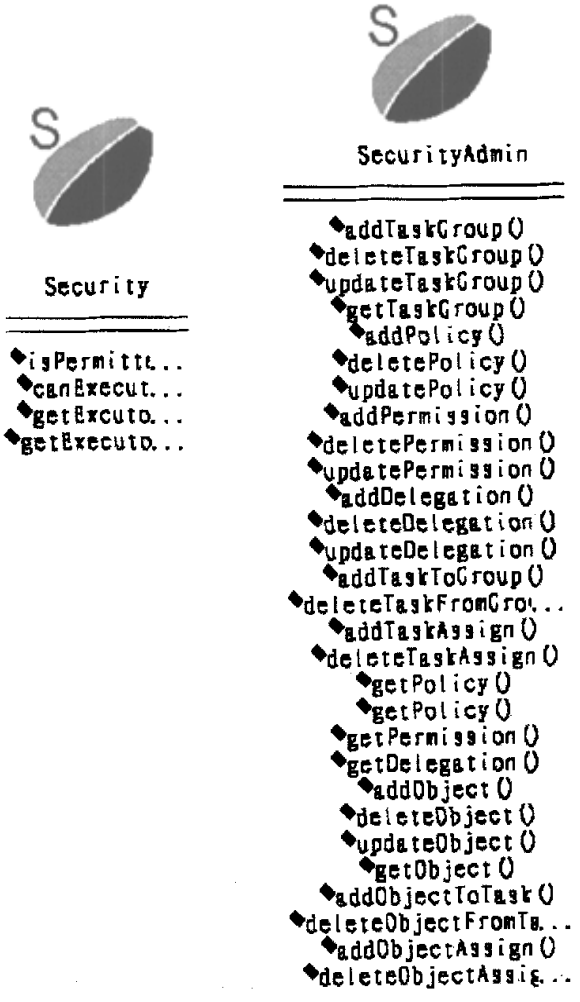

图 5.7 安全控管模块外部接口

安全控管模块对外提供的接口放在 Security 中，它主要实现任务指派、权限判断等功能；对内提供的接口放在 SecurityAdmin 中，它主要实现对模型自身的维护功能。

◆ Security 接口

面向引擎提供的接口主要有如下 4 个：

✓ isPermitted()：判断资源是否准许被访问，在资源被访问的时候调用；

✓ canExecute()：判断用户的访问请求是否被准许，在用户发起请求时调用；

- ✓ getExecutors()：在不考虑负载平衡的情况下，获取所有能做某一任务实例的用户；
- ✓ getExecutor()：在考虑负载平衡的情况下，获取能做某任务实例的用户。

◆ SecurityAdmin 接口

面向模块内部提供的安全控管模块管理功能的接口主要有：

- ✓ addTaskGroup()：添加一个任务组；
- ✓ deleteTaskGroup()：删除一个任务组；
- ✓ updateTaskGroup()：更新任务组信息；
- ✓ getTaskGroup()：获取某个用户组的详细信息；
- ✓ addTaskHierarchy()：添加任务组的层次关系；
- ✓ deleteTaskHierarchy()：删除任务组的层次关系；
- ✓ addTaskToGroup()：将任务添加到某个任务组；
- ✓ deleteTaskFromGroup()：将任务从任务组中剔除；
- ✓ addObject()：添加一个有安全要求的资源信息；
- ✓ deleteObject()：删除一个有安全要求的资源信息；
- ✓ updateObject()：修改一个有安全要求的资源信息；
- ✓ getObject()：获取一个有安全要求的资源信息；
- ✓ addObjectToTask()：将资源访问权授予某个任务；
- ✓ deleteObjectFromTask()：从某个任务的资源列表中将某资源访问权删除；
- ✓ addPolicy()：添加一个策略；
- ✓ deletePolicy()：删除一个策略；
- ✓ updatePolicy()：更新一个策略；
- ✓ getPolicy()：获取某个策略的详细信息；
- ✓ addPermission()：添加一个权限关系；
- ✓ deletePermission()：删除一个权限关系；
- ✓ updatePermission()：更新一个权限关系；
- ✓ getPermission()：获取权限；
- ✓ addDelegation()：添加一个委托授权；
- ✓ deleteDelegation()：删除一个委托授权；
- ✓ updateDelegation()：更新一个委托授权；
- ✓ getDelegation()：获取某个委托授权详细信息；
- ✓ addTaskAssign()：添加任务指派关系；
- ✓ deleteTaskAssign()：删除任务指派关系；
- ✓ addObjectAssign()：添加权限角色赋予关系；
- ✓ deleteObjectAssign()：删除权限角色赋予关系；

5.2.2.7 关键流程描述

用 UML 中的顺序图表示 WFIS 中的任务实例指派流程，结果如图 5.8 所示。

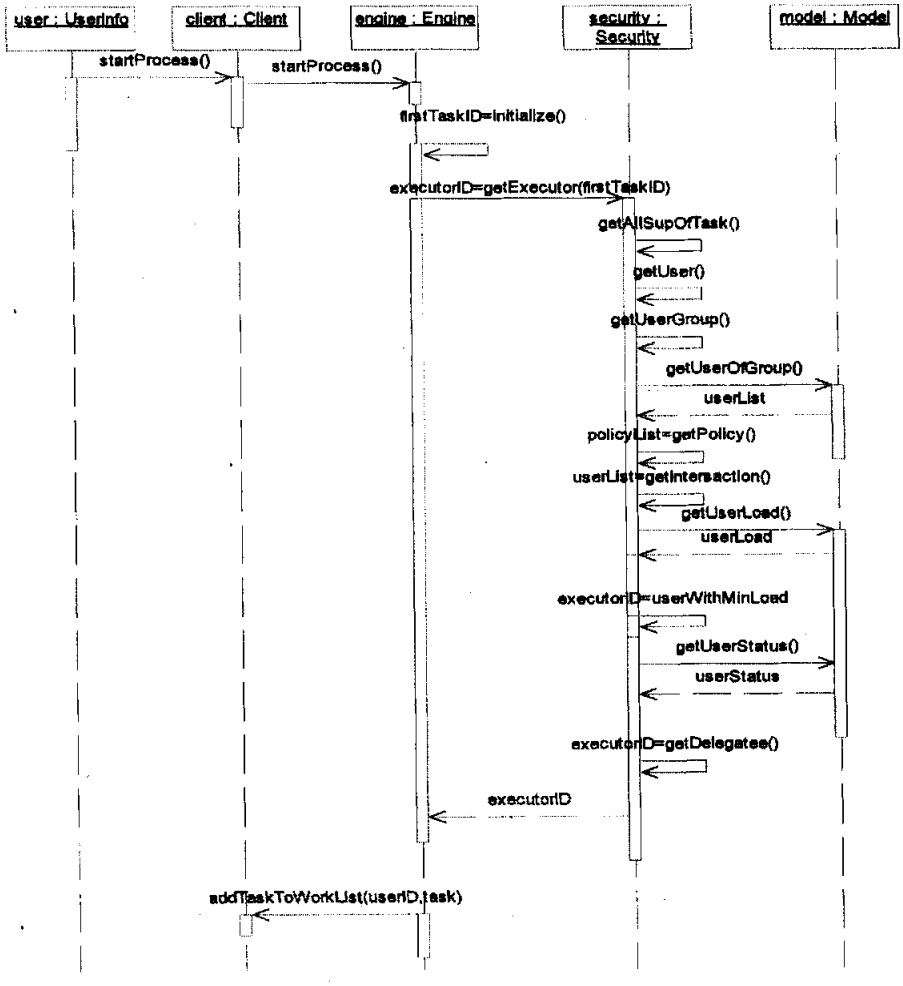

图 5.8 任务实例指派顺序图

系统进行任务实例指派时，首先查找实例所对应任务的所有上级任务（getAllSupOfTask()），然后检索能做任务和其上级任务的所有用户（getUser()）和用户组（getUserGroup()）。如果检索出来的用户组不为空，则调用组织模型 Model 接口方法（getUserOfGroup()）检索用户组的下级用户。所有这些用户就是任务角色赋予关系所指定的能执行该实例的用户。接下来检索任务的任务分配策略，若该任务存在任务分配策略且实例满足其约束表达式，则对策略指定的用户集合和先前查出来的用户集合取交集，将该交集作为新的能执行该实例的用户集合（userList）。最后根据任务指派方式从交集中选取适当的用户作为该任务实例的执行者。图 5.8 假定任务指派方式是 Least WorkLoad，如果是按级别授权则要调用组织模型 Model 接口的 getUserLevel()方法。

用 UML 中的顺序图表示检查操作权限的流程，如图 5.9 所示。

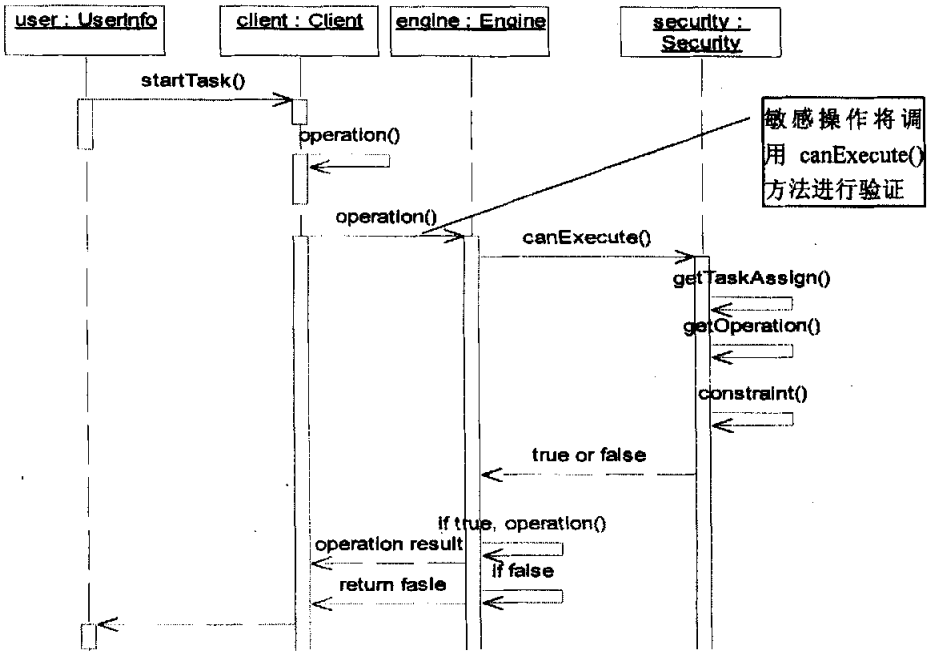

图 5.9 操作权限检查顺序图

用户执行任务实例的过程中，如果要进行敏感操作，则先检查系统是否将该任务实例分配给了这个用户，然后检查该任务实例所对应的任务是否具有相应的资源访问权限，最后检查用户环境是否满足任务上下文约束。

5.2.3 案例分析

这一节用一个实际案例（银行开证业务）来说明新工作流安全模型的应用。银行中比较常见的开证业务的工作流程模板如图 5.10 所示。

该流程的主要任务包括“开证登记”、“开证”、“保证金变更”、“改证”、“撤证”、“结清”等。在同一个流程实例中，根据具体的业务需求，可以做若干次“保证金变更”和“改证”任务。

TRBAC 在该流程中通过如下操作来实现：

1. 主要任务的任务角色赋予关系 TRA：“开证接待员”执行“开证登记”任务；“开证营业员”执行“开证”、“保证金变更”、“改证”、“撤证”、“结清”任务；
2. 职责分离约束：指定任何一个任务与其相应的复核任务为互斥任务，比如，“开证”与“开证复核”任务为互斥任务，在同一个开证流程实例中，如果一个营业员执行了“开证”任务实例就不能再执行“开证复核”任务实例；
3. 上下文约束：规定所有任务都只能在每天上午 8 点到下午 5 点之间执行（时间约束）；通过 IP 地址来限定“开证登记”任务只能营业厅的开证登记柜台的专

用 UML 中的顺序图表示检查操作权限的流程，如图 5.9 所示。

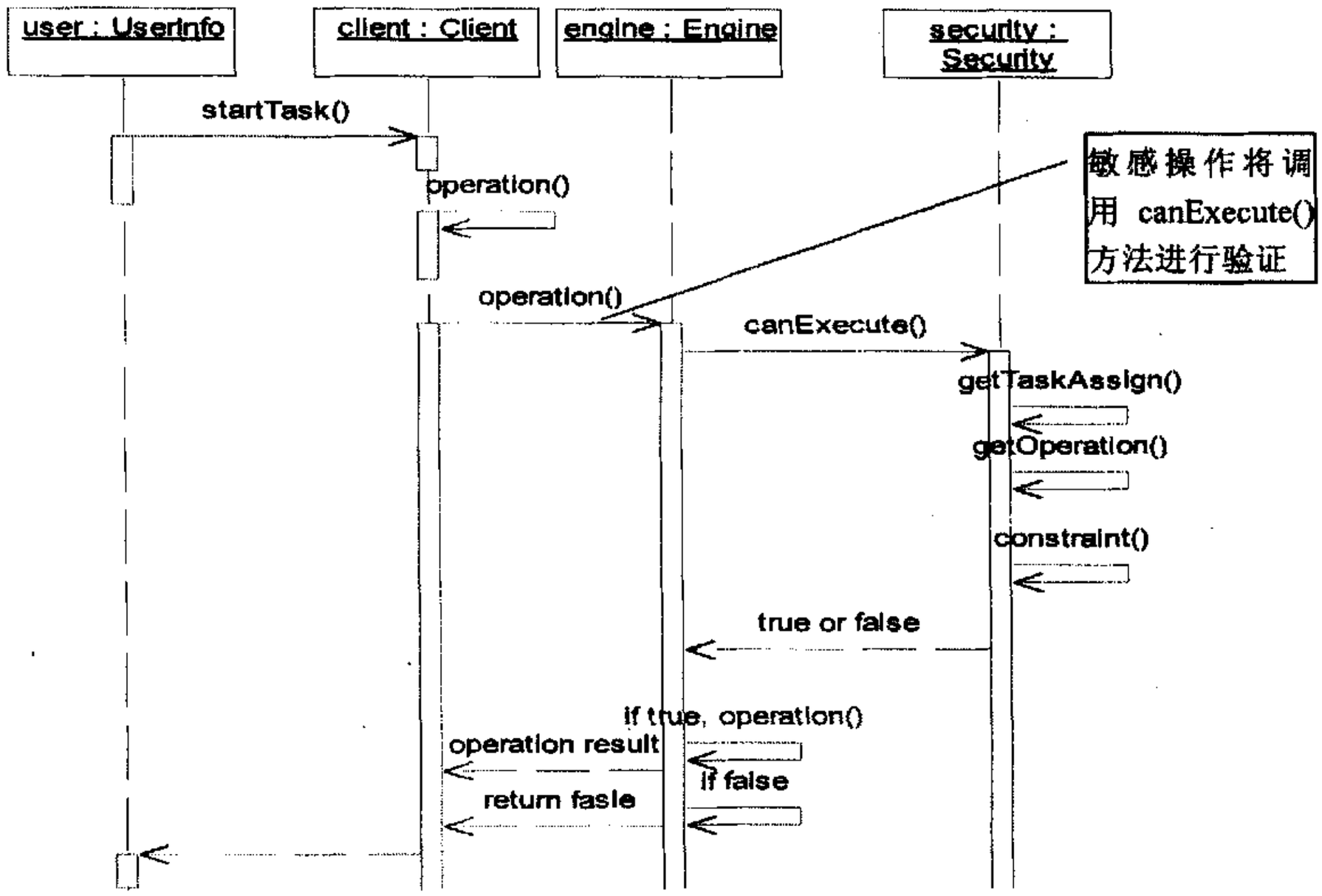

图 5.9　操作权限检查顺序图

用户执行任务实例的过程中，如果要进行敏感操作，则先检查系统是否将该任务实例分配给了这个用户，然后检查该任务实例所对应的任务是否具有相应的资源访问权限，最后检查用户环境是否满足任务上下文约束。

5.2.3 案例分析

这一节用一个实际案例（银行开证业务）来说明新工作流安全模型的应用。银行中比较常见的开证业务的工作流程模板如图 5.10 所示。

该流程的主要任务包括“开证登记”、“开证”、“保证金变更”、“改证”、“撤证”、“结清”等。在同一个流程实例中，根据具体的业务需求，可以做若干次“保证金变更”和“改证”任务。

TRBAC 在该流程中通过如下操作来实现：

1. 主要任务的任务角色赋予关系 TRA：“开证接待员”执行“开证登记”任务；“开证营业员”执行“开证”、“保证金变更”、“改证”、“撤证”、“结清”任务；
2. 职责分离约束：指定任何一个任务与其相应的复核任务为互斥任务，比如，“开证”与“开证复核”任务为互斥任务，在同一个开证流程实例中，如果一个营业员执行了“开证”任务实例就不能再执行“开证复核”任务实例；
3. 上下文约束：规定所有任务都只能在每天上午 8 点到下午 5 点之间执行（时间约束）；通过 IP 地址来限定“开证登记”任务只能营业厅的开证登记柜台的专

用计算机上执行，相应地可以限定“开证”、“撤证”、“改证”等业务都只能在某些专用计算机上执行；

4. 任务分配策略：当金额大于 100000 时由“开证营业员”中的“张某某”执行“开证”任务，由“部门经理”执行“开证复核”任务和“结清复核”任务；当开证币种为“欧元”时由“开证营业员”中的“李某某”执行“开证”任务；
5. 任务指派方式：默认的任务指派方式为“最小负载”（Least WorkLoad）方式；
6. 按级别授权：为每个任务指定级别，分配任务之前根据级别进一步筛选任务的执行者，比如，如果指定“撤证”任务的级别为 8，那么就要剔除所有能执行“撤证”任务实例的执行者中级别小于 8 的营业员；
7. 双重验证：在 OBJECT_ROLE 表中指定“经理”和“高级开证营业员”能够执行“撤销”任务中的删除开证信息表记录的操作，在执行该表记录的删除操作之前要检查 OBJECT_ROLE 表，如果执行删除动作人员的当前活跃角色为“经理”或“高级开证营业员”，则该动作允许执行，否则抛出异常。

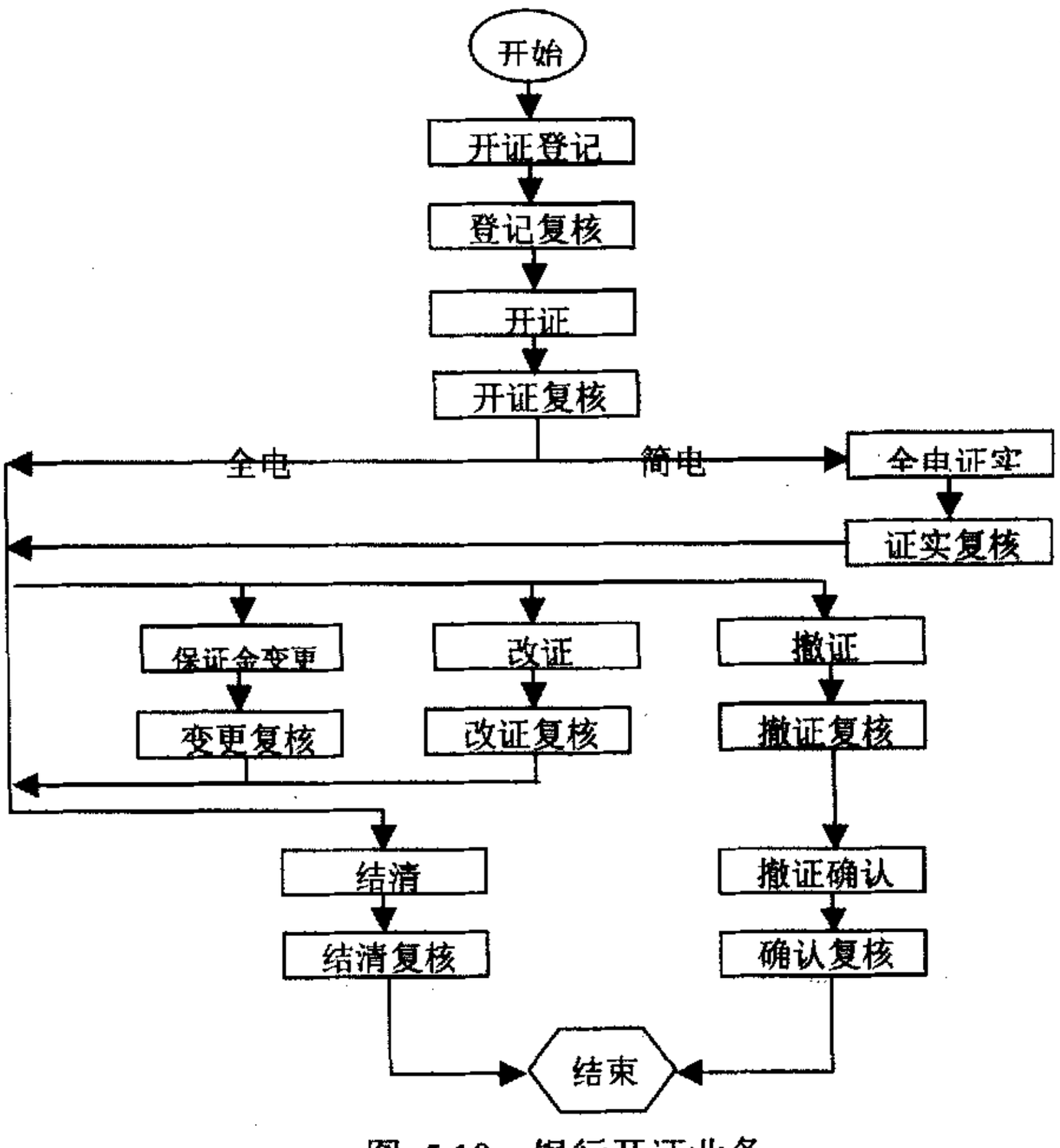

图 5.10　银行开证业务

5.2.4 小结

由以上讨论可见，将基于任务和角色的访问控制应用于工作流系统中的实现过程并不复杂，两个模块对外部提供的接口都很简单。新模型具有很强的实用性。

引入级别、上下文等概念增强了系统的安全性；引入任务分配策略和任务指派方式使得系统动态地进行任务指派时更加方便、灵活；引入任务层次、私有任务等概念大大简化了管理员的安全控管工作。

本文提出的新模型是在工作流系统已有组件（任务、组织模型、任务与任务执行者之间的映射关系）基础上构造的访问控制模型。安全强度介于自主访问控制和强制访问控制之间。与没有采用访问控制模型的工作流系统相比，本文的工作流系统增加了访问控制检查。对于有访问控制需求的工作流系统而言，这些检查是必不可少的。检查本身所带来的额外开销并不大，因此新模型对工作流系统的效率不会造成过大影响。

该模型是一个行之有效的访问控制模型。与传统的工作流安全模型相比，这个新模型的安全性、高效性、灵活性、实用性更好。

第六章 结束语

在工作流技术发展的早期，工作流系统主要应用于某些特定且相对独立的领域，采用的是“集中式”管理技术。由于规模较小，这种方式不会有太大的安全隐患。随着计算机网络技术的发展，企业业务向多元化、分布式、异构式等方向发展，业务流程所涉及的人员和资源的规模以及复杂性也成倍增长，从而使得现代工作流管理系统中的访问控制也变得复杂起来。

比较常见的工作流安全模型主要包括：基于任务的工作流安全模型和基于角色的工作流安全模型。基于任务的工作流安全模型能够解决工作流系统中任务的访问控制问题，但是它在关注工作流系统任务特性的同时忽视了安全控管的便利性，从而使得模型实施起来过于复杂。这种模型只适用于传统的集中式、小规模工作流系统。基于角色的访问控制模型的主要优势在于其安全控管的便利性，它解决了现代工作流系统由于规模庞大而带来的问题。然而，由于它采用的仍然是三层访问控制结构（用户—角色—权限），没有对任务及任务执行过程中所需的资源访问权限进行有效控制，所以它不能很好地适用于任务特征很强的工作流管理系统。传统的访问控制模型已不能满足现代工作流管理系统的需要。

本文针对现代工作流管理系统的安全特性，结合工作流管理联盟公布的工作流安全规范，在传统的访问控制模型基础上提出了一种新的访问控制模型：基于任务和角色的访问控制模型，并将其应用于面向金融领域的工作流基础件系统中。实践表明，新模型能更好地满足现代工作流管理系统的需要。

§6.1 本文主要贡献

本文主要贡献如下：

1）在传统的基于角色的访问控制模型中明确引入任务以及任务实例的概念，提出了一个基于任务和角色的访问控制模型。我们将传统的访问控制的 3 层结构（用户—角色—权限）扩展为新模型的 4 层结构（用户—角色—任务—权限）。该模型不仅解决了基于任务的工作流安全模型不适合大型企业应用的问题，而且解决了基于角色的工作流安全模型存在的最小权限约束假象、数据冗余、动态性差等问题。

传统的基于角色的工作流安全模型中也有任务、任务实例、任务的操作集等概念，但是它们与本模型中的相应概念在模型中的地位和作用都不相同。首先，在传统工作流安全模型中，任务（任务实例）与操作绑定在一起，它只作为基于角色的访问控制模型中的权限使用；新模型中的任务（任务实例）是 4 层访问控制结构中的一层，它作为模型的一个组件来使用。其次，传统的基于角色的工作流安全模型只定义了诸如 execute、commit、abort 一类的操作，而没有考虑到对执行任务过程中所需资源操作的访问控制。

虽然两者都是用户对任务的操作，但是它们本质大不一样。前者是所有任务都需要的一些“共同性”操作，而后者是对系统资源（外部应用程序、内部功能模块、数据库、文件系统等）的操作。

2） 对新模型进行了形式化描述。该描述使我们的模型更严谨、准确，并有助于机器表示和处理，其结果包含 4 个形式化模型：$TRBAC_0$、$TRBAC_1$、$TRBAC_2$、$TRBAC_3$。$TRBAC_0$ 是基于任务和角色的访问控制模型族的基本模型；$TRBAC_1$ 在 $TRBAC_0$ 基础上增加了角色层次和任务层次概念；$TRBAC_2$ 在 $TRBAC_0$ 基础上定义了约束的概念，描述了基于任务和角色的访问控制模型中的一些常用约束，如基数约束、静态职责分离约束、动态职责分离约束、任务执行上下文约束等；$TRBAC_3$ 是前几个模型的有机结合，该模型还引入了几种高级特性，如任务分配策略、任务指派方式、按级别授权等。

3） 模型实现与应用。作者在 863 项目（面向金融领域的分布式工作流关键技术及应用框架）中，将新模型成功应用到我们自主研发的基于 J2EE 的分布式工作流基础件系统中。结果表明，该模型是一个行之有效的访问控制模型。与传统工作流安全模型相比，本文模型在安全性、高效性、灵活性、实用性等方面都有所提高。

§6.2 新模型的创新点

本文提出的基于任务和角色的访问控制模型的主要创新点如下：

1）提出了“用户－角色－任务－权限”的四层访问控制结构。与传统的 RBAC 模型的三层访问控制结构（用户－角色－权限）相比，该模型突出了工作流系统中的任务概念，能更有效地适用于工作流系统。

2）将最小权限约束细化到任务一级。工作流系统中，管理员应该指定执行任务所必需的资源访问权限，这些权限对执行相应任务来说是必不可少的。所以，不存在将任务所对应的资源访问权限中的一部分授予用户，一部分却不授予该用户的情况，否则用户就无法完成这个任务。如果一部分用户比另一部分用户需要更多的资源访问权限才能完成某任务，那么就应该把该任务分解成两个不同的任务来处理，否则就增加了任务和访问控制的复杂度。正是因为在工作流系统中，不管由哪个用户来执行，同一任务所需访问权限都是一样的，所以任务一级的访问控制粒度对工作流系统来说是最恰当的。

3）在模型中首次引入任务层次、私有任务、任务分配策略、任务指派方式等概念。这些概念的引入使模型更灵活、实用、简便、高效。我们的模型还采用了按级别授权、上下文约束检查、委托授权、双重验证等机制，从而增强了系统的安全性。

在我们的模型中，既有对同一个任务的所有实例起作用的静态任务指派关系和任务指派方式，又有对同一个任务的不同实例起不同作用的动态任务分配策略，因此模型更加灵活、实用。

§6.3 工作展望

除了访问控制外，工作流管理系统中还有一些问题尚待进一步研究，比如事务管理。工作流系统中的事务管理相当复杂，当前的工作流系统事务管理方案的通用性和实用性都不太好。因此，本文的后续工作主要是研究工作流管理系统中的事务特性及事务安全问题，并构建一个适合当代工作流管理系统的高级事务模型。

另外，工作流系统中分布式多引擎的负载平衡和安全控制也是一个值得研究的课题。

致谢

在我的课题和硕士论文完成之际，谨向在攻读硕士学位过程中曾经指导过我的老师，关心过我的朋友，关怀过我的领导，和所有帮助过我的人们致以崇高的敬意和深深的感谢。

首先，要感谢我的导师谭庆平教授，谭教授在学术上的造诣使我受益匪浅，在课题遇到困难时，他总能给我有益的启示，使我得以顺利完成课题任务。除了学术研究上的指导外，谭教授的为人师表、高尚品质和敬业精神也都是我学习的楷模。

感谢王戟主任以及毛新军等 602 教研室的老师们，在研究生学习阶段，他们给了我极大的帮助，他们平易近人、治学严谨、强调实践，非常值得我学习。

感谢一起参与工作流基础件系统项目开发的所有同学，在过去的一年多时间里大家一起工作，共同探讨，既有技术上的支持，又能感受到大家庭的温暖。

感谢研究生队的领导，是他们在日常生活中的关心、教育使我顺利地完成了研究生阶段的学业。

感谢付伟、李宝峰、文健以及身边所有的同学和朋友们，正是你们使我来到国防科大之后的日子精彩无限。

最后，我要感谢的是我的父母和哥哥，他们不仅给予我精神上的大力支持，而且还对我的人生道路起了很大的引导作用，正是在他们的关心和帮助下，我才有了今天，也才使我能更专注于我的研究工作。谨以此文献给他们，以表示我对他们的无尽感激之情。

附录：攻读硕士学位期间发表的论文

[1] 付松龄，谭庆平. 基于任务和角色的分布式工作流安全模型. 长沙: 国防科技大学学报. （已录用）

[2] 付松龄，谭庆平. 基于J2EE的分布式工作流管理系统方案. 成都: 计算机应用, 2003, 23(8): 117-120.

参考文献

[1] 范玉顺. 工作流管理技术基础. 北京: 清华大学出版社. 2001.

[2] WfMC. TC00-1003:Worflow Management coalition Workflow Reference Model. WfMC. 1995.

[3] 曾月, 范玉顺. 基于 COM 和 ASP 的工作流管理系统的设计与实现. 北京: 计算机工程与应用. 2002, 38(1): 241-244.

[4] WfMC. TC00-1019:Workflow Management Coalition Workflow Security Considerations White Paper. WfMC. 1998.

[5] 付松龄，谭庆平. 基于任务和角色的分布式工作流安全模型. 长沙: 国防科技大学学报. （已录用）.

[6] 付松龄, 谭庆平. 基于J2EE的分布式工作流管理系统方案. 成都: 计算机应用. 2003, 23(8): 117-120.

[7] Ravi. S. Sandhu, et. Role Based Access Control Models. In IEEE Computer. 1996, 29(2): 38-47.

[8] Ravi Sandhu. Future Directions in Role-Based Access Control Models. MMM-ACNS. 2001.

[9] Sejong Oh, Ravi Sandhu. A Model for Administration Using Organization Struture. ACM. 2002: 155-162.

[10] David F. ferraiolo, Ravi Sandhu. Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and System Security. 2001, 14(3): 224-274.

[11] Gail-Jone Ahn, Michael. E. Shin. Role-based Authorization Constraints Specification Using Object constraint Language. ACM. 2002.

[12] 邓集波, 洪帆. 基于任务的访问控制模型. 北京: 软件学报. 2003,14(1): 76-82.

[13] J. A. Miller, et. Security in Web-Based Workflow Management Systems. ACM Transanctions on Information and System Security, 2002.

[14] Joon S. Park and Ravi Sandhu. Role-Based Access Control on the Web Using LDAP. Database Security XV: Status and Prospects. 2002, 4(1): 37-71.

[15] Savith Kandala and Ravi Sandhu. Secure Role-Based Workflow Models. In Proceedings of the 15th IFIP WG 11.3 Working Conference on Database Security. 2002: 45-58.

[16] ShengLi Wu, AmitSheth. Authorization and Access Control of Application Data in Workflow Systems. Journal of Intelligent Information Systems. 2002: 71-94.

[17] R. A. Botha, J.H.P.Eloff. Separation of Duties for Access Control enforcement in Workflow Environments. IBM Systems Journal. 2001, 40(3).

[18] Carlos Ribeiro, Paulo Guedes. Verifying Workflow Processes against Organization Security Policies. ACM. 2002.

[19] Myong H. Kang, Joon S. Park. Access Control Mechanisms for Inter-Organizational Workflow. ACM SACMT. 2001: 66-74.

[20]聂伯敏，熊桂喜. 分布式环境下基于角色访问控制的实现. 北京：计算机工程. 2002, 28(8): 181-183.

[21]尹存燕等. 公文流转系统的授权机制. 北京：计算机工程与应用. 2002, 09: 218-224.

[22]胡和平，汪传武. 一种办公自动化系统的安全机制. 长沙：计算机工程与科学. 2001, 23(6):95-97.

[23]胡和平，汪传武. 一种基于角色访问控制新模型. 长沙：计算机工程与科学. 2002, 24(4).

[24]洪帆，韩兰胜. 基于角色访问控制的办公自动化系统. 武汉：华中科技大学学报（自然科学版）. 2002, 30(6):67-69.

[25]董光宇，卿斯汉，刘克龙. 带时间特性的角色授权约束. 北京：软件学报. 2003,13(8).

[26]杜栓柱，谭建荣，陆国栋. 工作流模型中多粒度时间约束描述及其分析. 北京：软件学报. 2003, 14(11): 1834-1840.

[27]洪帆等. 多级安全工作流授权模型. 武汉：华中科技大学学报. 2002, 30(1): 20-22.

[28]Dr. David F.C.Brewer, Dr. Michael J. Nash. the Chinese Wall Security Policy. IEEE Symposium on research in Security and Privacy. 1989: 206-214.

[29]Arun Kumar, Neeran Kamik, Girish Chafle. Context Sensitivity in Role-based Access Control. ACM SIGOPS Operating Systems Review. 2002.

[30]Arun Kumar, et. Context Sensitivity in Role-based Access Control. India : ACM, IBM India Research Laboratory. 2002.

[31]Ezedin S. Barka. Framework for Role-Based Delegation Models. A Dissertation submitted to the Graduate Faculty of George Mason University for doctor degree. 2002.