曲阜師範大學

硕士学位论文

基于SOA的统一权限控制机制研究与应用

研究生姓名：付更丽
学科专业：计算机应用技术
研究方向：企业信息化与系统集成
导师 姓名：曹宝香
导师 职称：教授
论文完成时间：2010年4月

曲阜师范大学博士/硕士学位论文原创性说明

（在□划“√”）

本人郑重声明：此处所提交的博士□ 硕士☑论文《基于 SOA 的统一权限控制机制研究与应用》，是本人在导师指导下，在曲阜师范大学攻读博士□ 硕士☑学位期间独立进行研究工作所取得的成果。论文中除注明部分外不包含他人已经发表或撰写的研究成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中已明确的方式注明。本声明的法律结果将完全由本人承担。

作者签名：付更丽 日期：2010.6.2

曲阜师范大学博士/硕士学位论文使用授权书

（在□划“√”）

《基于 SOA 的统一权限控制机制研究与应用》系本人在曲阜师范大学攻读博士□ 硕士☑学位期间，在导师指导下完成的博士□ 硕士☑学位论文。本论文的研究成果归曲阜师范大学所有，本论文的研究内容不得以其他单位的名义发表。本人完全了解曲阜师范大学关于保存、使用学位论文的规定，同意学校保留并向有关部门送交论文的复印件和电子版本，允许论文被查阅和借阅。本人授权曲阜师范大学，可以采用影印或其他复制手段保存论文，可以公开发表论文的全部或部分内容。

作者签名：付更丽 日期：2010.6.2.

导师签名：[illegible] 日期：2010.6.3

摘　要

企业构建了一系列 Web 应用系统，但每个系统的权限管理是定制研发，与应用系统紧密耦合，很难实现权限管理复用，也给企业统一用户和授权管理带来不便。基于角色的访问控制模型 RBAC 是目前应用最广的访问控制模型。现代企业的用户权限经常变化，仅调整角色来改变用户授权，难以满足复杂情况变化的需要；同时传统 RBAC 模型的授权冲突解决策略有效性差。针对以上问题，本文对 RBAC 模型进行进一步的扩展提出了继承和优先约束驱动的用户和角色混合访问控制模型 IPC_URBAC，并基于 SOA 的 Web Service 技术设计实现了平台无关的、松耦合的、易扩展的统一权限控制机制。

文中首先分析了自主访问控制 DAC、强制访问控制 MAC 和基于角色访问控制 RBAC 的优缺点，验证了 RBAC 模型更适合 Web 应用系统的安全管理需要。然后，通过分析 RBAC 模型的不足，扩展 RBAC 模型提出 IPC_URBAC 模型，给出继承和优先约束的定义，提出个体和优先冲突解决策略，并给出求解用户权限的算法。其次，基于 SOA 思想和改进的 IPC_URBAC 模型设计实现了统一权限控制机制，分析了统一权限控制机制的特点，给出与 Web 应用系统集成的方法。最后，结合作者参与开发的 CRM 项目介绍了统一权限控制机制的应用实现。

IPC_URBAC 模型加入继承约束来灵活控制用户授权的有效性，增强了模型授权灵活性；设计继承和优先约束以及个体和优先解决策略很好的解决了用户和角色授权冲突问题，增强了模型安全性。本文在约束和授权方面的扩展，对 RBAC 模型的研究具有一定的借鉴意义。基于 SOA 架构思想将权限控制变为 Web 服务来实现，实现了安全模型最大粒度的可重用性，将权限控制从 Web 应用系统中最大限度的独立出来，使权限控制升级与原有系统分开，通过调用权限服务实现与不同 Web 应用系统集成，也为 SOA 的成功实践提供了一些新想法。

关键词： 访问控制；角色；继承和优先约束；Web Service；统一权限控制机制

Abstract

Enterprises build a series of Web application systems, but each system's right management is customized development, and tightly couples with application systems, it is difficult to achieve reuse of right management, and it brings inconvenience to the enterprises' unified user and authorization management. Role-based access control model RBAC is the most widely used access control model. Because user rights of modern enterprise always change, it is difficult to meet the changing needs of complex situations by adjusting roles to change users' authorization; At the same time, conflict resolution strategy of the traditional RBAC model is bad availability. In order to solve the above problems, this paper further extends RBAC model and proposes access control model based on user and role driven by inheritance and priority constraints IPC_URBAC, designs and implements a platform-independent, loosely coupled and easily extended unified right control mechanism based on SOA and Web service technology.

First, this paper analyzes the advantages and disadvantages of the Discretionary Access Control DAC, Mandatory Access Control MAC and role-based access control RBAC, verifies the RBAC model is more suitable for Web applications management. Second, analyzes the lack of RBAC model, extends RBAC model to IPC_URBAC model, gives the definition of inheritance and priority constraints, proposes individual and priority conflict resolution strategies, and gives calculation algorithm for calculating user rights. Third, designs and implements unified right control mechanism based on SOA and IPC_URBAC model, analyzes the characteristics of unified access control mechanism, put forward a method to achieve integration with Web applications. Last, introduces the realization of unified access control mechanism in a CRM project the author involved in developing.

IPC_URBAC model adds user inheritance constraint to control availability of user authorization, and enhances flexibility of authorization; Designs inheritance and priority constraints, as well as individual and priority conflict-solving strategies to solve the user and role authorization conflict, which enhances security of model. Expansion of constraint and authorization can give research on RBAC model some reference value. Making use of Web Service to implement the security model based on the thought of SOA realizes the largest grained reuse of security model, makes access control independent of Web application system, realizes integration of different Web application systems by calling right service, and provides some new ideas for successful implementation of SOA.

Keywords: Access Control; Role; Inheritance and Priority Constraint; Web Service; Unified Right Control Mechanism

目　录

第一章 绪论

1.1 研究背景

随着企业的信息化水平不断提升和 Internet 技术的不断发展，越来越多的企业建设了众多的 Web 应用系统并投入使用，这些应用系统已经成为企业管理的重要组成部分。企业中现存的 Web 应用系统可能由不同的开发商在不同的时期采用不同的技术开发的。这些 Web 应用系统，大多数具各自的授权及认证系统，不仅为用户的使用带来许多不便，更重要的是降低了企业信息系统的可管理性和安全性。因此，在推进和发展企业信息化建设的进程中，需要将每个应用系统的权限管理抽取出来，建设一套统一的授权控制机制，协调企业信息化建设进程，规范企业信息化建设。

RBAC 模型具有管理简单、易于实现和灵活性好等优点，已被广泛应用于各个领域。基于角色的访问控制(RBAC，Role-Based Access Control)通过引入角色实现用户和权限的逻辑分离，通过控制角色权限来间接地控制用户对系统资源的访问，方便了权限管理。随着 Web 应用的普及，权限管理模型和管理对象越来越复杂；随着企业业务的频繁变化，系统功能也在不断改变，要求权限管理系统具有较强的自适应性；目前的权限管理系统在设计没有考虑到各种可变因素，很难实现权限系统的复用。考虑权限管理的复杂和多变性，因此需要实现一个自适应和通用的、与应用系统无关、易于维护和升级的柔性权限控制机制。

面向服务的体系结构（SOA）作为一种软件架构理念，基于 SOA 的分布式松散耦合应用发展成为软件复用技术的主流。基于 SOA 的 Web Service 技术解决了软件重用的三个原则问题，将复杂的业务逻辑封装为可编程的网络组件，有效完成了异构系统的互操作和松耦合问题，通过集成可以快速实现粗粒度软件重用和快速系统集成。

从软件重用角度来说，如果能够将存在不同系统但功能相似的身份认证或权限控制模块独立出来，实现对应用系统的统一权限管理，那么对每个企业的不同应用系统开发具有重要意义。因此论文研究一种统一的权限控制机制，将设计者从繁琐的权限管理中解放出来，专心致力于业务系统的主要功能模块设计，减少系统的开发工作量，加快开发进程。

本文从 Web 应用系统的安全问题展开讨论，对原有的角色访问控制模型进行改进，设计了 IPC_URBAC 模型，并基于 SOA 架构思想将权限控制变为 Web 服务来实现，提高安全模型的可重用性。设计和实现一个基于 SOA 和 IPC_RBAC 的统一权限控制机制。通过友好的图形化界面实现对企业中的应用系统的统一权限管理，将权限控制从传统的应用系统中最大限度的独立出来，使得权限控制技术的改进和升级与原有系统分开，通过调用权限服务实现与不同网络应用系统集成。实现权限管理系统跨平台、跨语言的互操作，使权

限系统可以被任意的应用系统，在任何平台下、基于任意语言进行访问。此方法不仅实现权限系统最大粒度的重用，而且提高了企业管理水平，同时为 SOA 的成功实践提供了一些新的想法。

1.2 研究现状

1.2.1 访问控制研究现状

近些年，国内外认证和访问控制方法的理论研究很多[1,2]，研究主要集中于授权策略以及访问控制模型的建立和应用等方面。首先了自主访问控制 DAC 和强制访问控制 MAC，但自主访问控制的权限传递性给系统带来了安全隐患，而强制访问控制对访问控制的要求又过于严格。随着企业用户和资源的增加，系统的授权工作变的更加复杂，上述模型已经不能很好的工作。从 Web 信息系统发展的前景看，基于角色的访问控制模型 RBAC 比其他模型更加简单，更加容易扩展，更加适合分布式多用户和多资源的系统安全管理。

1996 年 Ravi Sandhu 提出了基于角色的访问控制模型 RBAC(Role Based Access Control)，通过引入角色实现了用户和权限的逻辑分离，简化了授权操作[3]。2001 美国国家技术标准局(NIST)发布了 RBAC 的全面统一的规范化标准[4]。后来，Ravi.Sandhu 增加管理角色的定义，对 RBAC96 进行扩展，建立了 RBAC 管理模型 ARBAC[5]，使得管理角色可以在规定范围进行管理操作。传统的 RBAC 的不足是针对主体进行权限划分，没有考虑客体和环境等要素，无法满足灵活授权控制的要求，Miehael J. Covington 等人改进了 RBAC，提出 GRBAC 模型[6]，扩展角色定义增加了客体角色和环境角色。国内也先后出现了大量的 RBAC 扩展模型，包括 New RBAC 模型（NRBAC）[7]，基于角色和任务的访问控制模型（TRBAC）[8]，以及基于角色和用户组的扩展访问控制模型(E-RBAC)[9]等。随着 Web Service 的应用普及，出现了一系列保证服务安全的角色访问控制模型[10-12]，文献[10][11]设计了基于角色的 Web 服务访问控制模型，实现 Web 服务的安全管理。

RBAC96 模型并没有给出约束机制得详细说明，无法解决现实权限管理中的一些授权问题，如授权冲突问题。近年来，出现了一系列的 RBAC 模型的约束机制[13-16]。Joshi 等人提出的带时间特性的角色访问控制模型 TRBAC[14]，韩伟力等人提出权限约束支持的基于角色的约束访问控制模型[16]，增加了权限约束来实现 RBAC 的约束管理。所有这些基于角色的约束访问控制模型，针对具体应用增加约束机制来增强 RBAC 模型的授权安全性，但随着企业需求的不断变化，需要提出更加灵活的约束机制满足授权要求。

目前信息安全领域的研究人员对 Web 环境下的 RBAC 模型展开了大量的研究，但扩展的 RBAC 模型大多停留在理论研究上，应用范围受到限制。尤其随着 Web 服务的出现，目前的 RBAC 应用存在大量的不足，许多问题只是理论的讨论而没有对应的解决方案，因此对 RBAC 的应用仍然需要进行深入研究。

1.2.2 权限管理系统发展现状

目前的权限控制实现一般是基于代码复用和数据库结构复用的方式将权限管理模块集成到业务系统中。传统权限管理架构如图 1.1 所示。

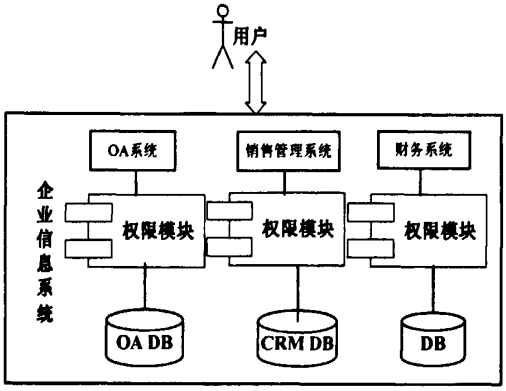


图 1.1 传统的企业权限管理架构

传统权限管理架构存在如下缺点：

(1) 权限管理与应用系统紧密耦合，一旦权限系统作了修改，就需要更新应用系统代码或数据库，并需要保证原有数据一致性，给开发和管理人员都带来了很大的负担。

(2) 缺乏有效的信息共享，每个系统都有自己的权限数据库，导致数据库保存相同的用户信息，造成大量的数据冗余，增加了数据的安全隐患。

(3) 管理员不方便管理权限数据，需要分别进入各个业务系统的权限模块进行授权操作，因此授权管理重复且复杂。

基于传统权限系统架构存在的不足，需要设计一种更加柔性化和通用的权限管理架构，能够做到以下两点：

(1) 改变基于代码和数据库复用的模式，使权限系统和业务系统解耦，权限系统的升级和修改尽量不影响应用系统。

(2) 方便企业统一管理所有应用系统的用户和权限，提高企业信息管理系统的开发和管理效率。

鉴于 SOA 的服务复用思想，需要将权限控制从应用系统解耦出来，封装成独立的模块，设计实现一种通用的、易扩展和跨平台的、与应用系统解耦的权限控制方法。

1.3 本文创新点

(1) 针对目前权限管理系统和角色授权存在的缺陷，设计了 IPC_URBAC 模型，在 RBAC 模型的基础上增加继承约束的用户直接授权机制和优先约束的用户角色分配机制，提出基于个体和优先的授权冲突解决策略，并给出了用户权限和角色权

限的求解算法。

(2) 运用IPC_URBAC模型，设计二进制授权掩码机制解决复杂权限设置问题。

(3) 基于SOA架构的服务复用思想和改进的IPC_URBAC模型，提出统一权限控制机制。给出统一权限控制机制的设计和实现方法，基于Web服务的软件重用开发方法，抽取权限控制服务供不同应用系统调用，实现权限管理与应用系统的解耦。

(4) 采用软件工程思想和J2EE开发平台，使用J2EE-SOA分层架构，将统一权限控制机制应用到开发CRM系统的权限管理中。

1.4 本文主要工作

本文主要目的是设计一种更加通用的权限控制机制，并构造一个基于SOA的统一权限管理系统达到对企业不同业务系统的统一授权管理。

主要研究工作包括理论和实践两个方面。理论研究方面：对比分析DAC、MAC和RBAC访问控制模型，指出传统访问控制技术的不足之处和RBAC的优势所在；提出继承和优先约束驱动的用户和角色访问控制模型IPC_URBAC，扩展模型使得访问控制的粒度更细，更能适应企业应用的需求。具体实践方面：设计了二进制权限掩码，实现复杂权限设置问题。提出基于SOA和IPC_URBAC的统一权限控制机制实现对企业不同业务系统有效的统一权限管理，提高权限管理的重用性，并给出统一权限控制机制的设计和实现方法。用J2EE分层体系结构和SOA思想来搭建系统平台，将统一权限控制机制应用到为中泰阳光电气科技有限公司开发的CRM系统中。

1.5 论文的组织结构

论文的章节组织如下：

第一章　绪论　介绍课题的研究背景、国内外研究现状、研究的创新点以及主要工作，强调研究基于SOA的统一权限控制机制的必要性。

第二章　访问控制理论和课题所需技术研究　首先，分析传统的访问控制技术DAC、MAC和RBAC，详细介绍了RBAC模型思想，并分析了RBAC模型存在的问题，为扩展RBAC模型奠定基础。然后，阐述课题研究所需的技术和知识：阐述SOA的定义和特点，着重讨论了实现SOA所需的技术Web Service，突出“Web服务复用是SOA的核心”，设计J2EE-SOA开发架构，以及在J2EE环境使用XFire开发Web服务。

第三章　IPC_URBAC模型的设计　改进RBAC模型提出了继承和优先约束驱动的用户和角色的柔性访问控制模型IPC_URBAC，给出了模型的形式化定义，并分析其可行性。

第四章　基于SOA的统一权限控制机制设计与实现　基于IPC_URBAC模型和SOA思想提出统一权限控制机制，并给出统一权限控制机制的设计和实现方法。

第五章　统一权限控制机制的应用　将统一权限控制机制应用到CRM平台的权限管

理中，验证机制的可行性。

第六章　结束语　对全文进行了总结，提出下一步要做的工作。

第二章 访问控制理论和课题所需技术研究

2.1 访问控制技术

访问控制是在访问主体和客体介入的一种安全机制，目的是来验证访问主体的权限，并管理受保护的客体对象。换句话说，访问控制即："通过某种策略明确规定哪些用户可以使用哪些资源"。如图2.1，访问控制包括3个主要部分：主体、客体和访问控制决策。

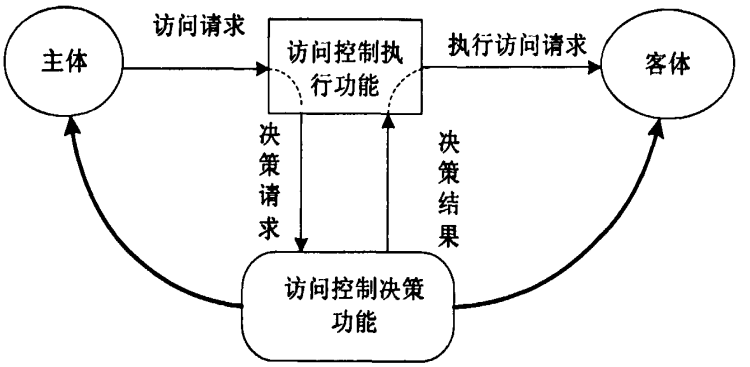


图2.1 访问控制模型

主体(Subject)：对其他实体施加动作的主动实体，称为用户(User)或访问者。包括用户、终端、主机或一个应用，主体可以访问客体。

客体(Object)：是能够以某种方式对其进行操作的任何资源对象，是需要保护的对象。包括信息、文件、记录等的集合体，或者网络设施等。

访问控制决策：是主体对客体进行访问的约束条件集，通过制定一组相关规则来管理主体对客体的访问。它决定访问主体能够做什么和做到什么程度，体现了一种授权行为。

访问控制流程：主体提出对客体的访问请求，首先被访问控制执行功能（实现访问控制的一段代码或监听器）截获，然后将请求的主体和客体对象都提交给访问控制决策功能，决策功能根据制定的访问控制规则返回决策结果（"允许"或"拒绝"），访问控制执行功能根据返回结果决定是否执行对客体的访问。

2.2 访问控制模型介绍

2.2.1 自主访问控制（DAC）

自主访问控制（Discretionary Access Control）[17]，又称任意访问控制，允许对象的拥有者制定针对该对象的保护策略。自主是指允许拥有访问权限的主体任意转让访问权限。DAC中，主体的拥有者可以自己设置访问权限。

自主访问控制的特点是授权主体自己负责赋予和回收其他主体对客体资源的访问权

限。DAC 模型一般采用访问控制矩阵和访问控制列表存放不同主体的访问控制信息，实现对主体访问权限的限制目的。

DAC 的优点是为用户提供灵活和易行的数据访问方式，表述直观、易于理解，并且比较容易查出对一特定资源拥有访问权限的所有用户，容易实施有效的授权管理，能够适用于许多的系统环境。但是，DAC 的最大缺陷是主体的权限太大，资源的拥有者可以自行决定资源的访问权限，提供的安全性还相对较低，不能够对信息资源提供充分的保护，容易造成信息的泄露。

2.2.2 强制访问控制（MAC）

强制访问控制（Mandatory Access Control）[18]是为了实现比 DAC 更严格的访问控制，根据客体和主体的安全级别来限制主体对客体的访问。系统根据主体和客体的安全等级分配一个安全级别，访问控制执行时对主体和客体的安全级别进行比较。MAC 独立于用户行为来强制执行访问控制，用户不能改变他们的安全级别或对象的安全属性。

MAC 通过安全标识实现信息流只能从低安全级流向高安全级，可以防止出现自主访问控制方法中出现的访问传递问题。MAC 具有层次性，级别高的权限可以访问级别低的权限。

MAC 的优点是管理集中，具有很强的等级划分，根据定义的安全级别实现严格的权限管理，经常用于军事管理中。美国政府和军方一直使用此种访问控制模型。但 MAC 太严格，并且实现起来工作量太大，管理不便，不适用于主体或客体经常更新的应用环境。MAC 对于防止因用户无意造成的信息泄露以及在防止木马盗窃信息上有所帮助，但 MAC 的控制机制缺少完整性，过于强调系统保密性，对授权的可管理性也考虑不足。

2.2.3 基于角色的访问控制模型 RBAC

DAC 是用户自主赋予或撤销访问权限，不是管理员确定哪些用户对资源有访问权限，不利于全局授权管理。MAC 过于强调保密性，对授权的可管理性考虑不足。20 世纪 90 年代NIST 提出的基于角色的访问控制RBAC(Role Based Access Control)被认为有效地克服了传统访问控制的不足。

基于角色访问控制(RBAC)模型是目前世界上使用最广泛的访问控制方法[3-6,19]。RBAC 的核心思想是通过分配和取消角色来完成用户权限的授予和取消，并且提供角色分配规则。图 2.2 描述了 RBAC 的基本思想，系统管理员根据需要定义各种角色，并分配合适的访问权限，而用户根据其工作职责被分派不同的角色。因此，整个过程可分为两部分：首先将访问权限与角色相关联；然后根据工作职责将角色与用户关联，实现用户与访问权限的逻辑分离。RBAC 支持最小特权、责任分离以及数据抽象三个基本的安全原则。

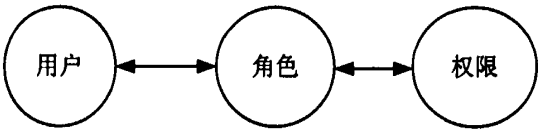


图 2.2 RBAC 模型的基本思想

1、 RBAC 中的相关定义：

- 主体（Subject）：能够向应用系统发出请求的任何实体，包括用户、其它与本系统有接口的应用程序、非法访问者。
- 用户（User）：用户是可以独立访问系统中的数据的访问主体，用户通常指人。
- 资源（Resource）：在访问控制中称为客体或受控对象，是系统中受到保护的对象。
- 操作（Operation）：对资源的各种执行动作，如增加、删除等。
- 角色（Roles）：角色是指一个组织或任务中的工作或位置，它代表了一种资格、权利或者责任。
- 权限（Permission）：是对系统中的数据或者受控资源进行访问的许可。 权限是资源和操作的一个二元关系。
- 角色权限分配（Permission assignment PA）：又称角色授权，是角色与权限之间的一个二元关系，我们用（r, p）来表示角色 r 拥有一个权限 p。
- 用户角色分配（Role assignment RA）：是用户和角色之间的一个二元关系，我们用(u, r)表示用户 u 拥有一个角色 r。
- 会话（Session）：是用户到角色之间的一种映射关系，用户必须通过会话来激活角色来获取相应的权限。
- 约束(Constraint)：是一组强制性的规则集合，是 RBAC 的重要组成部分。模型中定义了职责分离关系（separation of duty）来控制权限冲突问题，包含静态职责分离和动态职责分离。

2 、RBAC 模型

如图 2.3，RBAC 包含用户 (U)、角色 (R)、客体(OBS)、操作 (OPS)、权限 (P)五个基本数据元素。权限被赋予角色，而不是用户，当一个角色被指定给一个用户时，此用户就拥有了该角色所包含的权限。会话 S 是用户与激活的角色集合之间的映射。RBAC 与传统访问控制的差别在于增加一个角色中间层来增加授权灵活性。

RBAC 的形式化定义：

1. U、R、P 和 S 分别表示用户、角色、权限和会话的集合
2. PA⊆P×R，PA 是多对多的权限和角色指派关系。
3. UA⊆U×R，UA 是多对多的用户和角色指派关系。
4. 角色继承：如果角色 x 继承了角色 y，则称 x 是 y 的父角色，y 是 x 的子角色，记为 $x \geq y$
5. $RH \subseteq R \times R$，RH 是关于 R 的偏序集，称为角色继承或角色支配关系。记作≥。

6. Roles：$S \rightarrow 2R$，Roles(s)是一个映射函数，每个会话 s 对应一个角色集。
7. Roles(s) $\subseteq \{r|\{\exists r \geq r'[(u,r') \in UA]\}$，会话 s 拥有权限集 $\subseteq \bigcup_{r \in roles(s)} \{p|\exists r'' \leq r[(p,r'') \in PA]\}$
8. users：$S \rightarrow U$，users(s)是一个映射函数，将每个 s 映射到唯一的用户。

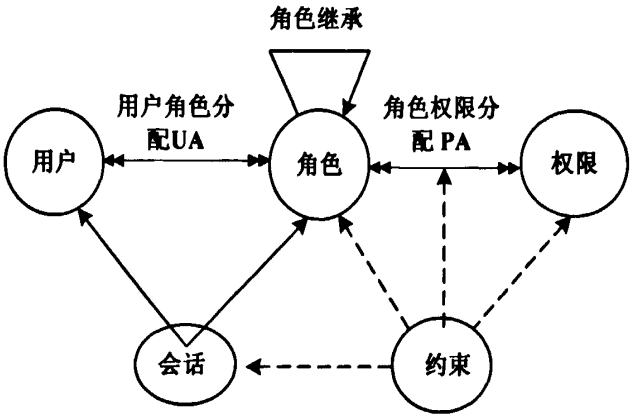


图 2.3 RBAC 模型

在 RBAC 中，为了避免相同权限的重复设置和减少定义角色的数目来方便授权，增加了角色的继承关系，即进行了角色分层，一个角色可以通过继承一个或多个角色来定义[19,20]。通过继承，角色不仅拥有自身的权限，同时拥有继承角色的权限。角色继承是 RBAC 的重要组成部分，角色分层能够很自然的体现现实世界中组织结构的分层关系，还可以避免权限的重复设置。

为了安全，现实通常是用户不能同时独立完成某些工作，每个操作要由不同用户完成，这种现象称职责分离(Separation of Duty)。RBAC 中定义的责任分离包括静态责任分离和动态责任分离。静态职责分离是指在对用户进行角色分配时，增强了用户角色分配的限制，使一个用户不能分配给两个互斥的角色。动态职责分离是用户可以分配多个互斥的角色，但在同一会话中互斥角色不能被同时激活[21, 22]。由于 RBAC 需要处理权限冲突问题，由角色-权限和用户-角色的关系，定义了冲突权限、冲突角色。

冲突权限：

$\forall p1, p2 \in P$,如果$\forall u \in U$,u不能同时拥有 $p1$和$p2$,那么$p1$和$p2$相互冲突。
则（$p1, p2) \in CP$

其中，CP 为冲突权限的集合，$CP = \{cp_1, cp_2, cp_3, ..\}$其中$cp_i = \{p_1, p_2, p_3, ...\} \subseteq P$

冲突角色：

$\forall r1, r2 \in R, \exists (p1, p2) \in cp$,如果$r1$拥有权限$p1$,同时$r2$拥有权限$p2$,那么$r1$和$r2$相互冲突。则$(r1, r2) \in CR$

其中，CR 为冲突角色集，$CR = \{cr_1, cr_2, cr_3, ..\}$其中$cr_i = \{r_1, r_2, r_3, ...\} \subseteq R$

在授权时需要处理用户的冲突权限和冲突角色问题。静态约束通过在对用户指定角色过程中制定授权约束规则来防止一个用户得到过多的权限。动态约束是在系统运行过程中执行。与静态约束不同的是，动态约束允许一个用户拥有冲突角色，但在会话过程中，只能激活冲突角色中的一个角色。动态约束需要通过代码实现冲突的解除,而静态约束只需要在授权时手工设定。

在实际应用中，RBAC中的职责分离主要通过定义互斥角色和角色基数等约束机制，在有些文献中还提及了基数约束、角色容量、先决条件[22-25]。第三章设计IPC_URBAC模型增加了新的约束机制：继承约束和优先级约束，结合动态和静态约束方法解决模型授权冲突问题。

2.2.4 传统访问控制特性分析

通过上面的分析可以看出在资源量巨大的情况下，授权工作量巨大，利用DAC来处理大规模的授权工作，开销巨大，用户工作量大，因此DAC不在适合现代资源量巨大的Web信息系统；MAC模型通过对用户和资源定义安全级别，在认证时通过判断安全级别来实现权限判断，这就决定了它不够灵活，况且安全级别也无法完全满足企业的需求，同时MAC无法满足动态业务的需求，而企业用户和资源都在不断变动，无法事先将用户和资源定义好安全级别。所以，MAC模型也不适合Web信息系统。RBAC通过角色来联系用户和权限简化了授权机制，并且支持多种访问控制策略，使它能适应Web系统所需要面对的数量巨大的动态的用户和资源的要求。

基于角色的访问控制方法（RBAC）显著的两个优点是：

(1) 角色-权限变化比角色-用户变化慢的多，基于角色授权降低了授权管理的复杂性，减少企业管理开销。例如，如果一个用户的职责变动了，只要将用户拥有的角色去掉，加入代表新职责的角色即可。

(2) 灵活实现企业的各种安全策略，随着企业的业务变动模型具有伸缩性。

尽管RBAC模型相对于MAC模型和DAC模型更适合Web系统的特性，可它并不是万能的，不能完全满足现代企业的安全管理需求。通过对比分析可以看出，RBAC模型的特点使它成为网络环境下企业信息系统的访问控制问题的基础解决方案，我们可以在RBAC模型的基础上进行扩展，以使其更适应现代Web应用系统的访问控制需要。

2.3 面向服务体系架构（SOA）

2.3.1 SOA的定义

面向服务的架构的概念最早有Garter公司在1996年提出。在2002年12月，Garter又提出面向服务架构是“现代应用开发领域最重要的课题”。从总体上来讲，面向服务架

构 SOA(Service Oriented Architecture)是一种将信息系统模块化为服务的软件架构风格，定义了通过使用服务来满足软件用户的需求。面向服务的架构目前还没有统一的定义，现在普遍认可的一种 SOA 定义如下："面向服务的体系结构是一个组件模型。它将应用程序的不同功能单元(称为服务)通过服务之间定义良好的接口和契约联系起来，接口采用中立、基于标准的方式进行定义，独立于实现服务的硬件平台、操作系统和编程语言，使得构建在各种系统中的服务可以用一种统一和通用的方式进行交互。"[26-28] 在 SOA 中，最核心的技术是服务，将业务组件化为一系列粗粒度的业务服务或业务流程。

2.3.2 SOA 的基本特征

如何区分 SOA 与现有编程思想？下面给出满足 SOA 编程的三个基本特征[26]：

(1) 松散耦合。松散耦合是指相互之间没有依赖，它是针对目前紧密耦合的应用系统提出的一个概念。

服务之间的松散耦合。不同服务的功能不能互相依赖，一个服务能够实现自己所提供的接口功能，不依赖其它服务。Web Service 已经实现了这一点，对于用 WSDL 定义的 Web Service 服务接口，既可以用 J2EE 实现，也可以用.Net 实现。

(2) 粗粒度服务。SOA 中服务是一般是粗粒度的。在 SOA 中，服务与业务对齐，服务的接口比面向对象的 API 更大，需要更接近用户的实际操作。客户通过一次请求就可以获取大量相关的业务数据，减少了请求的次数。

(3) 位置和传输协议透明。位置(传输协议)透明是指不论服务组件的实际 URL(传输协议)如何变化，客户端的调用程序的 URL（传输协议）都不需要改变。使得服务的请求者和提供者之间高度解耦，好处有两点：一点是它适应变化的灵活性；另一点是当某服务的内部结构和实现逐渐发生改变时，不影响其他服务。

2.3.3 SOA 的目标

(1) 从软件复用角度看，服务重用是 SOA 很重要的一个特点。但是相对于传统的重用方法，SOA 的重用粒度更粗。SOA 的重用是业务级的应用，与软件的发展规律是相一致的。

(2) 实现敏捷的、不受限制的应用集成，从而使 IT 能够随招业务的需求的变化而自由调整，达到"随需而变"的效果。解决传统集成方式"你中有我，我中有你"紧密耦合方式，避免牵一发而动全身的悲剧。

2.4 SOA 的主要支撑技术—Web Service

SOA 是一种架构理论概念，不是具体的实现技术。SOA 具有多种可能的实现技术，其中 Web Service 具有开放性和标准性，称为目前最好的一种实现 SOA 的技术。利用 Web

Service 实现 SOA 可以实现一个中立平台来获得服务，并且随着更多的软件商支持 Web 服务规范，将会取得更好的通用性。

2.4.1 Web Service 技术

Web Service 即“Web 服务”，通常被定义为一组模块化的 API，对外提供统一的调用接口，通过网络进行调用来执行调用者的服务请求[29,30]。Web 服务是为了使得原来各孤立的站点之间的能够相互通信、信息共享而提出的一种接口。换句话说，Web 服务就是将应用系统的函数暴露到网络上，让其它用户或应用系统进行访问或调用。Web 服务的实现目标是简单和易扩展，跨平台和跨语言的的互操作，独立于平台和软件供应商，且基于 XML 等标准技术，可以实现应用系统集成、代码和数据重用。

Web Service 是一种新型的分布式计算技术，它使用 Web 作为平台，具有以往其它分布式计算技术比如 COBRA、DCOM 和 RMI/IIOP 等不可比拟的优势。Web Service 的主要优点如下：

(1) Web 服务使用面向服务的体系结构，使用了标准的、规范的 XML 概念描述一些操作的接口，使其可以和任何基于不同平台和语言开发的组件进行通信，可以完成各种异构平台的数据或应用集成。

(2) 由于使用 HTTP 作为底层的传输协议，从而避免了防火墙的问题。同时 Web Service 也不拘泥于 HTTP，可使用如 FTP 等传输协议，非常灵活。

(3) Web 服务具有封装性、松散耦合、高集成性高等特点，使用 Web 服务可以实现代码或数据复用。

2.4.2 Web 服务体系结构

Web 服务采用了 SOA 的体系架构，通过观察 Web 服务涉及的角色及其各自功能来划分 Web 服务体系结构。Web 服务涉及三个角色，即 Web 服务的提供者、Web 服务的请求者和 Web 服务的注册中心。涉及的三个操作分别为服务的发布、服务的查找和服务的绑定。如图 2.4 所示。

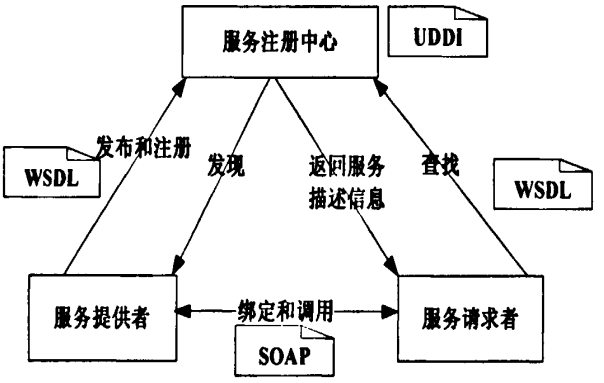


图 2.4 Web Services 体系架构

Web 服务的提供者提供 Web 服务。它主要有三个功能：第一是实现具体逻辑功能。第二是遵循一定的接口规范、并将 Web 服务发布到 Web 服务注册中心，以方便服务请求者找到该 Web 服务。第三是与服务请求者进行交互，为服务请求者提供服务。

Web 服务请求者请求 Web 服务。它的主要功能有两个：第一是在 Web 服务注册中心查找，以发现合适的服务。第二是利用查找到的信息，和服务的提供者进行交互，以消费该服务。

Web 服务注册中心是服务请求者和服务提供者的中介。主要功能有两个：第一是允许服务的提供者发布服务。第二是允许服务的请求者查找服务以发现合适的服务。服务的注册中心提供服务的目录，使得服务请求者和服务提供者之间构成了松耦合关系。

图 2.4 中给出了 Web 服务对应的相关协议，即 WSDL、SOAP 和 UDDI，这三个协议是 Web 服务的核心协议。SOAP 支持服务之间的通信。WSDL 协议描述了服务的接口。UDDI 协议支持 Web 服务的查找。

2.4.3 Web Services 的技术支持

Web Services 在 SOA 架构的具体实现技术主要由 XML，WSDL，SOAP 和 UDDI 等组成，由这些协议共同完成 Web 服务的互操作。下面依次介绍这些协议。

(1) XML（eXtensible Markup Language，可扩展标记语言）[31]：XML 是一套定义语义标记的规则，这些标记将文档分成许多部件并对这些部件加以标识，是 Web 服务平台中表示数据的基本格式。和 HTML 语言不同，它是一种简单和轻便的元标记语言，可根据用户需求，定义与特定领域有关的、语义化、结构化的标签和元素的句法语言。XML 的目的是传输数据，不仅易于建立和理解，而且可扩展、内容与表示的分离、与平台无关和厂商无关等特点。Web Services 使用基于 XML 的通信协议，WSDL、SOAP 和 UDDI 作为 Web Services 使用的主要协议，它们都基于 XML 格式。可以说，XML 是 Web 服务强大的功能的实现基础

(2) SOAP（Simple Object Access Protocol，简单对象访问协议）[32]：是一种标准化的传输消息的 XML 轻量级通讯协议。它属于 Web 服务的消息层协议，定义了消息交互的具体格式。它可自定义，易于扩展。一条 SOAP 消息就是一个普通的 XML 文档，主要包含下列元素：Envelope 元素，标识 XML 文档一条 SOAP 消息；Header 元素，包含头部信息的 XML 标签；Body 元素，包含所有的调用和响应的主体信息的标签 ； Fault 元素，错误信息标签。SOAP 的优点是它完全和厂商无关，相对于平台、操作系统、目标模块和编程语言可以独立实现。

(3) WSDL[33]（Web Services Description Language，网络服务描述语言）：是一门基于 XML 的语言，用于描述 Web Service 以及如何对它们进行访问。它可规定服务的位置，以及此服务提供的方法。WSDL 的主要目的在于 Web Service 的提供者将自己的 Web 服务

所有相关内容，如服务的传输方式、服务的方法接口、接口参数、服务路径等，生成相应的完全文档，发布给使用者。换言之，WSDL 的主要目的是告诉外界自己能够提供什么样的服务。根据 WSDL 的输入和输出变量就可以知道 SOAP 请求消息和相应消息。

WSDL 主要是利用<portType>、<message>、<types>和<binding>元素来描述 Web Service。<types>元素定义 Web service 使用的数据类型。为了最大程度的平台中立性，WSDL 使用 XML Schema 语法来定义数据类型。<message>元素定义一个操作的数据元素。每个消息均由一个或多个部件组成。可以把这些部件比作传统编程语言中一个函数调用的参数。<binding>元素为每个端口定义消息格式和协议细节。<portType>元素是最重要的 WSDL 元素。它可描述一个 Web 服
务、可被执行的操作，以及相关的消息。可以把 <portType> 元素比作传统编程语言中的一个函数库。

(4) UDDI[34]（Universal Description，Discovery，and Integration，统一描述、发现和集成协议）：它是一种创建注册表的规范。通过它，企业可注册并搜索 Web Service。UDDI 是基于 XML 和 SOAP 标准，为诸如 WSDL 之类的服务描述语言提供了统一的 XML 词汇，供描述 Web 服务及其接口使用。

但是，UDDI 是一个可选的 Web 服务组件。因为 WSDL 文件中已经给出了 Web Service 的地址 URI，外部可以直接通过 WSDL 提供的 URI 进行 Web 服务的调用，服务方完全可以不进行 UDDI 注册。

2.4.4 基于 Web 服务的软件重用的优点

软件重用一直是软件开发追求的目标。软件复用是在开发新系统过程中，对已经存在的软件成品的使用技术，这里指出了开发与复用的关系。软件复用技术经历从面向过程、面向组建，以及目前关注的面向服务。软件重用经历了一个逐步降低耦合性的过程，软件抽象的程度一直在提高，直至和企业业务对齐，显然，服务提供了一个更高层次的重用。当然软件重用也成为实现软件工业化、标准化的重要手段，SOA 更是强调粗粒度的复用。

基于 Web 服务的软件重用具有与开发平台和开发语言无关、高度松耦合性、分布式和易扩展性、可移植、跨平台分布式重用能力强等优点。Web 服务同时满足了目前软件重用提出的三个要求：(1)存在性，即必须存在可以重用资源。(2)可发现性，即可重用资源可以被发现。(3)标准化，满足软件重用的标准，与开发平台、开发工具和语言等无关[35]。

2.5 J2EE 与 SOA

2.5.1 J2EE 分层体系结构

J2EE（Java2 Enterprise Edition）[36]是 Sun 公司推出的一项中间件技术，是在 Java 2 平

台上的企业级应用的解决方案，使用 Java 语言开发企业应用系统。J2EE 包含很多组件，利用 Java2 平台可以简化和规范分布式多层应用系统的开发与部署，提高系统可移植性和安全性。如图 2.5，J2EE 的分布式多层应用主要定义了四层，分别为客户层、Web 层、业务层和企业信息系统层(EIS)。组件在对应的层上部署，且在对应容器中运行。每层的容器提供了标准服务套件，使得组件能够访问适合本层的 API。

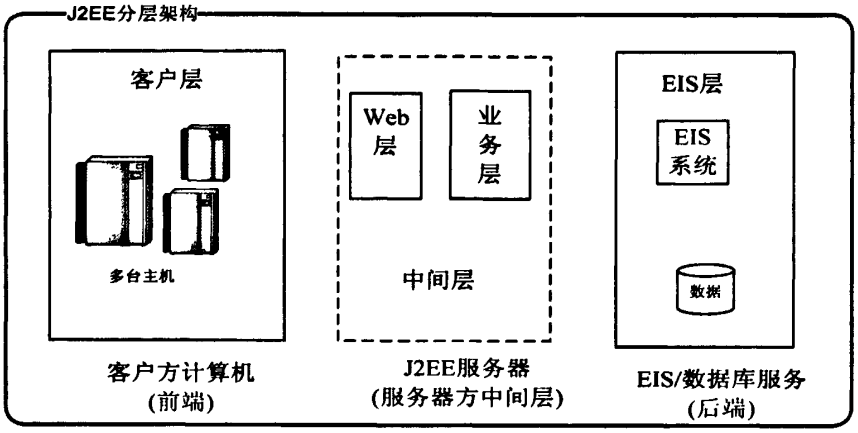


图 2.5　J2EE 分层体系结构

J2EE的分层应用架构一方面能够创建结构良好、高度可重用性、模块化和高度可伸缩性的应用系统；另一方面能够节省人力资源，实现面向团队的开发。

2.5.2 基于 J2EE 和 SOA 的分层架构

结合上面 J2EE 的分层架构和 SOA 思想，设计了 J2EE-SOA 分层架构。这个架构是基于 SOA 的，也是基于 J2EE 的，所以同时含有了二者的特征[37]。从图中可以看出，缺少 SOA 中服务注册中心的功能，这是由于服务的发布地址事先已知，服务的请求者(服务调用层)可以直接调用 Web 服务。J2EE-SOA 基于 SOA 思想，将经常使用的业务功能包装成服务，通过服务调用层调用存在的服务，实现了业务重用。SOA-SSH 分层架构如图 2.6。

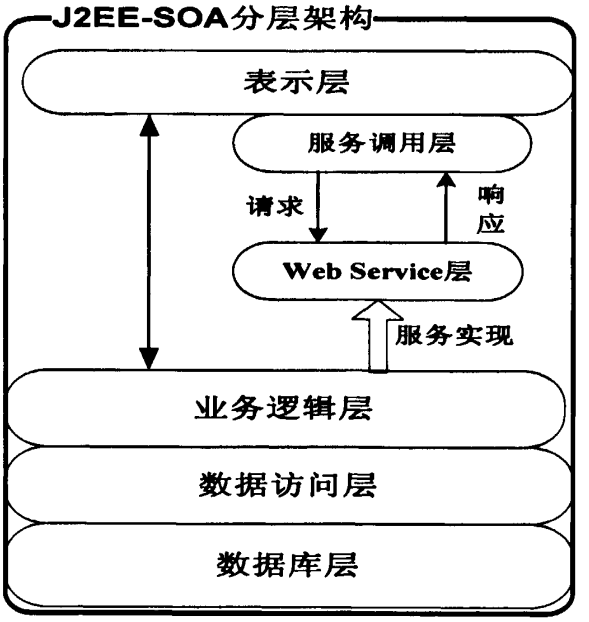


图 2.6 J2EE-SOA 分层架构

架构中有关部分的说明：

- 表示层： JSP 和 Html 是表示层的实现技术，用来生成用户与系统交互的 Web 页面，用户通过 Web 浏览器来访问 Web 页面。
- 服务调用层：Web 服务的请求处理程序，用来与服务交互。编写服务调用代码时需要获得服务的 WSDL 描述文件，然后根据此文件进行编写。
- 服务层：Web Service 是服务的提供者，包装重用的业务功能，是粗粒度业务重用。
- 业务逻辑层：负责实现应用系统的业务逻辑功能。通常由 Spring 或 EJB 来完成业务组件的开发。
- 数据访问层：封装数据操作，并提供数据访问接口。可以同通过 DAO 技术或 ORM（Object Relation Mapping）工具来实现。
- 数据库层：使用关系型数据库，来存储企业的数据。

2.5.3 J2EE Web Service 开发——XFire

Web 服务一般的开发过程是：

1. 开发普通的功能类；
2. 将其包装成 Web 服务，发布于服务器并生成 WSDL 描述；
3. 调用者根据 WSDL 生成客户端代码。

基于 Web Service 实现的开发包也很多，笔者采用的是 codehaus 的一个开源 Web Service 项目开发包 Xfire[26,29,39]。主流 J2EE 开发工具 MyEclipse 提供的 MyEclipse web service 工具基于开源的 XFire Java SOAP 引擎构建。XFire 是一款比较少见的而且非常流行的厂商独立的 SOAP 引擎。XFire 是面向服务的开发更加简便，提供容易使用的 API 并支持各

种标准。

1 Xfire 的基本原理

XFire 本质上是一个 Servlet，所以要运行于 Servlet 容器上。XFire 首先提供一个 XFire lib，它能够解析 SOAP 请求消息，执行相应的结果，并将结果生成 SOAP 返回。XFire 通过 web.xml 定义 Servlet，通过 services.xml 定义服务的 Java 类，实现 XFire 的 Servlet 和 Java 实现类之间的相应绑定。同时 XFire 可以使用 Spring 进行服务的发布。XFire 的工作原理如图 2.7 所示。

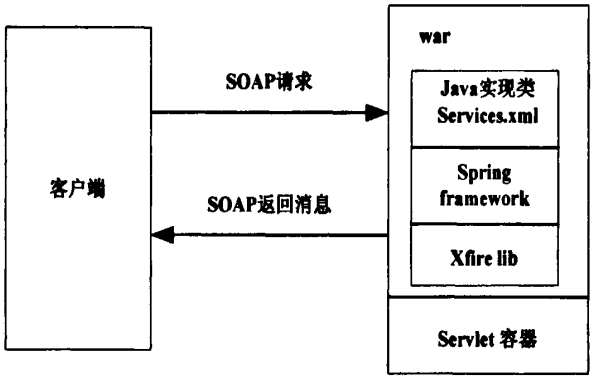


图 2.7 Xfire 的工作原理

2 创建 XFire 的 Web Service

XFire 的创建步骤如下：

1. 创建服务器端的 Java 程序
2. 修改 web.xml
3. 创建 services.xml

3 MyEclipse Web Service 项目

MyEclipse6.0 引入的新的项目名为 Web Services Project，它对 MyEclipse Web 项目进行了扩展，来支持更多的概念包括 web services 配置，开发和发布。为了方便 Web 服务开发，笔者使用 MyEclipse Web Services Project 向导来创建和配置新的 Web Service 项目。操作步骤如下：

1. 创建 Web Services Project
2. 在项目的 web.xml 文件中配置 XFire Servlet
3. 创建 XFire 配置文件 services.xml
4. 将 MyEclipse-XFire 类库添加到项目的构造路径中
5. 在项目文件.project 中添加特殊的 MyEclipse Web 项目构造器，这样能够发布项目时自动将 services.xml 发布到正确的位置，例如：<webroot>/WEB-INF/classes/META-INF/xfire/
6. 创建 Web Service 客户端

2.6 本章小结

本章作为论文的理论基础，首先介绍了访问控制的相关知识：首先给出访问控制的基本概念，然后通过深入分析 DAC、MAC 和 RBAC 的优缺点，证明了 RBAC 模型更适合于 Web 系统的访问控制，并指出了 RBAC 模型在现代应用领域的不足。

然后介绍课题所需的技术知识：阐述了 SOA 定义和特点，及主流实现技术 Web Services 的定义、体系架构和主要实现技术，强调了基于 SOA 的 Web Service 技术在推动了软件重用的发展。简单介绍了 J2EE 技术，并基于 J2EE 分层架构和 SOA 设计了 J2EE-SOA 分层架构，最后介绍了 J2EE 的 Web Service 开发技术 XFire。

第三章 IPC_URBAC 模型设计

同传统访问控制模型相比，基于角色的访问控制（RBAC）模型具有管理简单、容易实现且灵活性好等优点，然而 RBAC 模型的适用领域还存在着一定局限和不足。RBAC 模型将资源的访问权限附加到角色上，为用户分配相应的角色，通过控制角色权限来间接地控制用户对系统资源的访问，这种方法实现了用户授权的简单性。但 RBAC 模型是通过角色进行粗粒度的管理和控制用户权限，无法获得某个角色的部分权限，缺少灵活性。结合实际 Web 系统开发中的经验，发现单独的角色授权在实际应用中柔性不够，仅调整角色来改变用户授权，对角色的权限管理略显生硬，难以满足企业复杂情况变化的需要。比如由于业务需要某用户需要临时拥有某一权限，如果给该用户拥有的角色增加此权限必然会影响其他用户，但若重新建立一个角色，由于是临时授权并且权限比较少，使用完之后还需要删除角色，显然会比较繁琐。同时 Web 系统存在大量的页面和表单等动态变化元素，RBAC 技术不能很好的满足需要，因此需要设计更加有效的权限控制方法。

本文为了增强 RBAC 模型的柔性授权管理，设计了继承和优先约束驱动的用户和角色授权模型 IPC_URBAC(User and Role Based Access Control Model Driven by Inheritance and Priority Constraints)。 RBAC 模型可被看成是扩展模型的一种特例，扩展模型不仅具有 RBAC 模型的所有优点，而且比 RBAC 模型具有更好的可用性。该模型在 RBAC 模型基于角色授权的基础上增加了用户直接权限分配，用继承约束用户直接权限分配的有效性，用户的最终权限通过继承约束来确定。模型增加优先级约束用来灵活处理用户-角色授权和角色-角色授权冲突问题。

3.1 扩展 RBAC 模型(IPC_URBAC)

IPC_URBAC 模型如图 3.1，图中黑框内部是传统 RBAC 模型，扩展的 RBAC 模型支持用户和角色两种授权方式，并且分离操作集和资源集来灵活适应系统资源的动态变化。扩展模型主要解决授权灵活性和授权粒度问题。在授权粒度方面，模型增加了基于用户的直接授权机制，解决单独角色授权存在的问题。传统的 RBAC 模型中，将系统资源和相应的操作进行绑定作为系统的权限集合，操作不灵活。模型将权限化分为操作集合和资源集合，通过规则运算生成权限集合。通过分离操作和资源，实现权限系统动态适应业务系统资源动态变化的问题。为了解决授权冲突问题，模型为用户直接授权增加了继承约束机制，为用户-角色分配增加了优先约束机制。

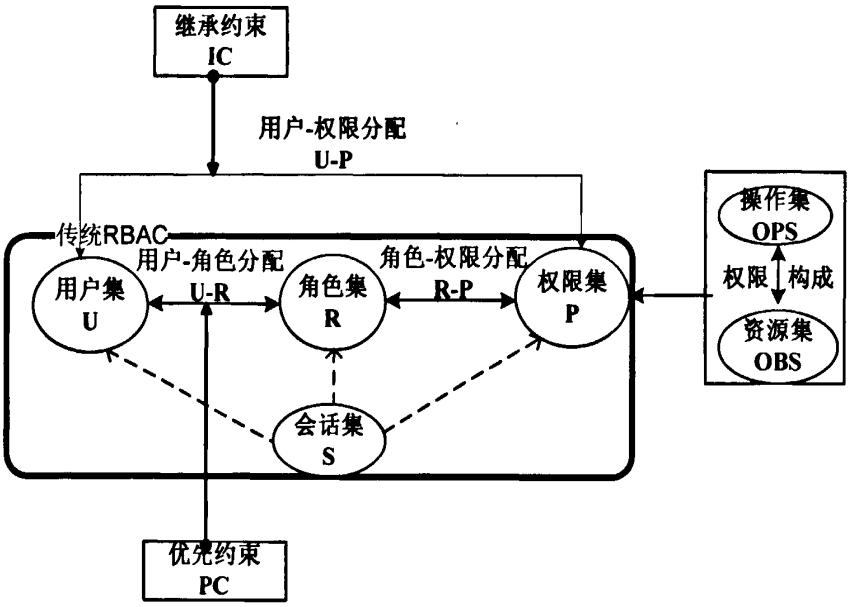


图 3.1 IPC_URBAC 模型

下面对 IPC_URBAC 模型进行分析。

3.2 IPC_URBAC 模型的形式化定义

用户集：$U=\{u|\bigcup_{i=1}^{n}u_i\}$。U 是访问系统资源的用户集合。

角色集：$R=\{r|\bigcup_{i=1}^{n}r_i\}$。角色是一定数量的权限集合。

会话集：$S=\{s|\bigcup_{i=1}^{n}s_i\}$。S 是所有会话的集合。

权限集：$P=\{p|\bigcup_{i=1}^{n}p_i\}$。权限集是系统权限的集合，规定系统中每个资源的所有可用权限，是授权的直接对象。权限集 $P=2^{OBSxOPS}$，任意权限 p_i 可表示为二元关系<ob,op>，其中 ob 为资源，op 为操作。权限集 P 通过资源集 OBS 与操作集 OPS 关联运算生成，解决了传统的资源权限固定的不足。

为了实现对资源的动态性，模型分解权限集，定义了资源集 OBS 和操作集 OPS。

资源集：$OBS=\{ob|\bigcup_{i=1}^{n}ob_i\}$，OBS 定义为系统中资源或功能模块的集合，在 Web 系统中一般以按钮、页面或功能模块的形式存在。

操作集：$OPS=\{op|\bigcup_{i=1}^{n}op_i\}$，OPS 定义为系统操作类别的集合，例如增加、修改、审核、读取等。

主体集 S=U∪R：用户或角色的集合，是实现参与授权的实体。

R-P⊆P×R： 权限角色分配，是权限到角色的多对多的关系。

U-R⊆U×R：用户角色分配，是用户到角色的多对多的关系。

role(s_i) ⊆{r|(user(s_i),r) ∈U-R}：其中，user：S→U，会话与用户的函数映射，每个会话 s_i，对应一个 user(s_i)。Roles:S→2^R，会话与角色集合的映射关系。

U-P⊆U×P：用户权限分配，是用户到权限的多对多的关系，是为了弥补单独角色授权的不足而增加的授权方式。

3.3 扩展模型的约束机制

传统 RBAC 中定义约束主要针对权限分配中存在的不安全问题，即前面介绍的职责分离机制 SOD。但在 RBAC 中，除了考虑权限的正确分配外，还应该考虑权限分配的灵活性和权限的潜在授权冲突问题。授权模型中的约束(**Constraint**)指有关谓词应满足的条件[40]。本文针对用户和角色混合授权过程中可能存在的安全问题和如何提高模型的权限分配灵活性，提出了基于用户的继承约束 IC 和基于角色的优先约束 PC。

继承约束 IC(Inheritant Constraint)：继承约束是 U-P 的补充，是控制 U-P 有效性的开关。IC={true，false}。IC 为 true 时，规定 U-P 失效，用户最终的权限通过 U-R 获得。IC 为 false 时，继承约束无效，用户权限是通过某种运算规则得到的 U-R 和 U-P 的权限集合。

优先约束 PC(Priority Constraint)：指在用户角色分配 U-R 过程中给角色增加优先级。优先级，这里是一个数字，表示角色的权限的优先级别。数字越小，级别越高。为用户分配多个角色时，为了体现角色在用户中的重要程度，在 U-R 中增加“优先级”约束，PC 越小，表明角色越重要。本文增加优先约束的主要目的是为了解决角色授权冲突问题，当用户的多个角色存在权限冲突时通过保留优先级别高的角色授权来消除冲突权限。

继承约束增强了用户和角色授权的灵活性，IC 可以确定是启用或禁用用户直接授权。而优先约束可以保证用户同时拥有多个权限，并且保证用户不会同时启用冲突的权限，增强了授权的安全性。

3.4 扩展模型的权限冲突解决策略

任何访问控制策略在实施时，授权冲突是一定要解决的。解决冲突的思想是通过定义一些冲突解决策略(CRS Conflict Resolution Strategy)来化解授权冲突。在第二章我们介绍了冲突权限、冲突角色等关于授权冲突的相关概念。下面我们具体给出冲突解决策略和权限冲突的定义。

冲突解决策略 CRS 可以定义为一组规则集合的形式：

$Condition \rightarrow Permit(s,op,ob)$ 或

$Condition \rightarrow \Pr ohibit(s,op,ob)$

其中 Permit (s,op,ob)和 Prohibir(s,op,ob)分别代表授权管理中的允许和禁止。

权限冲突：$\exists s,op,ob \quad Permit(s,op,ob) \wedge Prohibit(s,op,ob)$

其中 Permit (s,op,ob)允许主体 s 对客体 ob 执行 op 的操作；而 Prohibir(s,op,ob)表示禁止。在一个授权模型中，冲突权限即对象相同而权限相反的权限。例如 u 同时拥有权限 1 和权限 2，权限 1 规定对象 a 可以打印账单，权限 2 规定对象 a 不可以打印账单，则权限 1 和 2 为一对冲突权限。

扩展的 IPC_URBAC 基于用户和角色混合授权，因此权限冲突可能存在用户与角色之间和角色与角色之间。两种冲突定义如下：

用户与角色授权冲突：Permit(u,op,ob)∧Prohibit (r(u),op,ob)

角色与角色间授权冲突：Permit(r (u),op,ob)∧Prohibit (r (u),op,ob)

其中，r(u)表示用户 u 拥有的角色集合。

实际授权中需要使用冲突解决策略来消除冲突，本文针对冲突的两种可能情况，并根据模型增加优先约束 PC，定义了基于个体的冲突解决策略 CRS1 和基于优先级的冲突解决策略 CRS2。

CRS1(个体优先策略)：个体优先策略使用的前提是继承约束 IC=false，即当用户直接授权有效的时候才会出现用户与角色之间的授权冲突问题。为保证用户直接授权和角色授权中互斥权限不会被同时激活，个体优先策略只激活冲突权限中的用户直接拥有的权限。

CRS1：Permit(u,op,ob)∧Prohibit(r(u),op,ob) ∧{IC(u, op,ob)=false}

→Permit(u,op,ob)

其中，IC(u, op)=false 表示用户直接授权有效。

CRS2(优先级策略)：优先级策略使用的前提用户存在多个冲突的授权角色。此策略是存在多个授权冲突的角色中只激活高优先级角色的权限，即高优先级角色授权覆盖低优先级角色授权，保证两个角色中的互斥权限不被同时激活。

CRS2：Permit(r_1(u), op,ob)∧Prohibit(r_2(u), op,ob) ∧{$pri_u(r_1)$>$pri_u(r_2)$ }

→Permit(r_1(u),op,ob)。

其中 r_1(u)和 r_2(u)表示含有冲突权限的用户 u 的两个角色。pri_u(r)表示用户 u 拥有的角色 r 的优先级。

下面通过一个例子来说明如何利用提出的冲突解决策略解决模型授权冲突问题。

对客户信息 client 的管理用户 Mary 拥有的权限描述如下：

Mary 直接授权 U-P(Mary)：prohibit(r1,delete,client) ，

permit(Mary,delete,client)，IC(Mary,delete,client)=false

Roles(Mary)={R1，R2}， $prio_{r1}$(Mary)> $prio_{r2}$(Mary)

角色 R1：permit(r1,read,client)，permit(r1,add ,client)

角色 R2：prohibit(r2,add,client)，permit(r2,delete,client)

根据前面“冲突角色”的定义我们可以看出角色 R1 和角色 R2 是一对冲突角色，因为存

在冲突权限 permit(r1,add ,client)∧prohibit(r1,add ,client)。利用优先级策略 CRS2：

permit(r1(Mary),add ,client)∧prohibit(r2(Mary),add,client) ∧{ $prio_{r1}$(Mary)> $prio_{r2}$(Mary) }→ permit(r1(Mary),add ,client)

Mary 通过角色获取的权限 U-R(Mary)={ permit(r1,read,client), permit(r1 ,add ,client), permit(r1,delete,client)}

考虑 Mary 存在直接授权并且继承约束 IC 是有效的，但观察 U-P(Mary)和 U-R（Mary）可以发现存在冲突权限 permit(r1(Mary),delete,client) ∧prohibit(Mary, delete,client)。利用个体优先策略 CRS1：

permit(r1(Mary),delete,client) ∧prohibit(Mary,delete,client) ∧IC(Mary, del ete,client) = false→ prohibit(Mary,delete,client)

利用冲突解决策略之后 Mary 最终权限 P(Mary)={<read,client>，<add,client>}

3.5 基于角色的权限生成过程

根据图 3.1 的模型定义，下面给出用户权限的生成过程：

1. 生成系统权限集合。首先创建资源集 OBS，明确系统中的哪些模块和页面需要参与权限控制，为资源指定唯一编码和序号。为了清晰表示资源的级别，资源序号采用分级编号，下级资源的序号以直接上级资源的序号作为前缀。根据 Web 系统的操作需要生成操作集合 OPS，利用 OBS 和 OPS 进行关系运算，得到系统的权限集合 P。

2. 定义角色，并为角色分配权限。根据系统要求，按照部门或工作职责需要定义系统角色集合 R，然后针对 R 中的每个元素 r，在权限集 P 为其分配权限子集 p，组成角色-权限分配表(r-p)。

3. 给用户分配角色。新建用户集合 U，然后根据用户的工作需要从角色集 S 中为其分配相应的角色子集 R'(r1,r2,..,rn)，组成用户-角色分配表（u-r)。

到此，基于角色的授权过程已经完成。当某个用户登陆系统的时候，系统建立一个会话 s，通过查询用户表，验证用户身份合法后，保存用户信息 u；通过查询用户-角色对照表(u-r)得到用户的角色集 r，以此类推，得到用户 u 在系统中的权限 p，通过函数 Fr(p)就可以确定用户 u 可以访问的资源和操作。

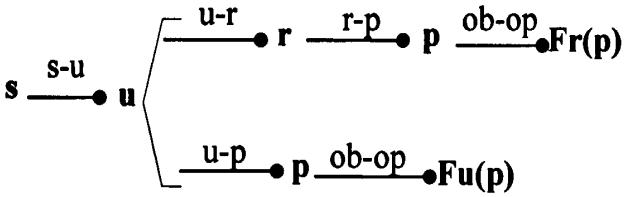
上面的权限生成过程的公式表示如下：

$$s \xrightarrow{s\text{-}u} u \xrightarrow{u\text{-}r} r \xrightarrow{r\text{-}p} p \xrightarrow{ob\text{-}op} Fr(p)$$

3.6 直接用户授权的模型扩展

IPC_URBAC 模型增加用户直接权限分配通道 U-P，增加了模型的授权灵活性，解决用户的临时少量授权和撤销问题。从图 3.1 中可以看出，扩展模型除了角色授权以外，直

接用户授权是对角色授权的补充。加入用户直接授权后的权限生成公式如下：



其中 Fr（p）是用户 u 通过角色得到对资源 OBS 进行 OPS 操作的权限，Fu(p)是用户通过直接授权得到的对资源 OBS 进行 OPS 操作的权限。用户的最终权限是 Fr 与 Fu 的并集。即：

F（p）=Fr(p)∪Fu(p)

这种方法解决了临时授权问题，通过 RBAC 中角色授权的粗粒度和管理复杂性问题，提高了用户授权的灵活性。在权限验证时，只需要同时判断用户拥有的角色和用户对资源的直接访问权限，来确定用户的最终权限。

3.7 扩展模型的权限计算算法

通过以上分析可知在 IPC_URBAC 中，用户所拥有的权限来源于两部分：一部分是用户直接授权，另一部分是用户-角色分配中角色授权。根据授权管理中用户实际权限是经过冲突解决策略过滤后的用户授权。为了得到正确的用户权限，在权限计算过程中使用上面提出的冲突解决策略对用户授权限进行过滤，消除权限冲突，得到用户的最终实际权限集合。

已知 R-P、U-R、U-P，定义算法中用到的函数和变量如下：

SortR(u)表示用户 u 拥有的按照优先级 pri 从低到高排序的角色集。SortR(u)={(<r_1, pri_1>,<r_2, pri_1>,...<r_n, pri_n>)|pri_i<pri_{i+1},i=1,2...}。

obj(p)表示权限 p 的应用资源对象。obj (p)={ob|<ob,op>∈p}

PS_r(r)表示角色 r 的拥有的权限集，PS_r(r)={p|<r,p>∈R-P}。PS_ur(u)表示按优先级策略得到的用户 u 拥有的角色权限集合，PS_ur(u)={p|<u,r>∈U-R∧<r, p>∈R-P∧不存在 p 的冲突权限}。

PS_u (u)表示用户 u 拥有继承约束 IC 无效的直接权限集合。PS_u (u)= {p|<u,p>∈U-P∧IC=false}。

用 PS(u)表示按照个体优先级策略得到的用户 u 拥有的全部权限集合。PS(u)= PS_u(u)∨{PS_ur(u)-{p|p∈PS_ur(u)∧p 与 PS_u(u)存在权限冲突}}。

算法 1：PermissionOfUR(u) 求用户 u 的所有角色授权

PermissionOfUR(u)

输入：u：用户集合中的一个用户

输出：PS_ur:：用户 u 拥有的角色权限集合

```
{
PS_ur=ϕ //初始化 PS_ur 为空集
SortR=SortR(u)//SortR 为 u 的优先级从低到高的角色集
For each <r,pri >∈SortR//依次遍历角色集
  PS_r=PS_r(r)//角色 r 的权限集
If PS_ur=ϕ  then
  PS_ur= PS_r
Else
For each p∈PS_r//使用 CRS2 解决角色-角色授权冲突
    For each p'∈PS_ur
      If obj(p)=obj(p') then
      PS_r=PS_r-{p'}//删除冲突权限
      End if
    End for
  End for
  PS_ur=PS_ur∪PS_r
End for
End If
Return PS_ur
}
```

算法 2：PermissionOfUser(u) 求用户 u 拥有的直接授权集合

PermissionOfUser(u)

输入：u：用户集合中的一个用户

输出：PS_u：用户 u 拥有的继承约束无效的直接授权权限

```
{
PS_u=ϕ /初始化 PS_u 为空集
For each <u', p >∈U-P
If u'=u ∧IC=false then
PS_u=PS_u∪{p}//保留继承约束无效的用户授权
End if
End for
Return PS_u
}
```

算法 3：AllPermissionOfUser(u) 求用户 u 的拥有的最终权限集合

AllPermissionOfUser(u)

```
输入：u：用户集合中的一个用户
输出：PS：用户 u 拥有的所有权限集合
{
PS =ϕ //初始化 PS 为空集
PS_ur=PermissionOfUR(u)
PS_u=PermissionOfUser(u)
For each p∈PS_u//使用 CRS1 消除用户和角色授权冲突
    For each p'∈PS_ur
      If obj(p)=obj(p') then
      PS_ur=PS_ur-{p'}//按照个体优先策略保留用户的直接授权。
      End if
    End for
End for
PS =PS_ur∪PS_u
End for
Return PS
}
```

3.8 IPC_URBAC 模型的优势分析

1. 模型具有详细的形式化描述，与以往模型相比，IPC_URBAC 是有特色的，它虽然是基于角色的，但并没有仅局限于角色，引入了用户直接授权的概念；考虑到企业中人员职能的频繁变动，模型引入了继承约束的用户直接授权，来满足企业为用户动态改变权限的要求。IPC_RBAC 是细粒度的模型，克服了传统 RBAC 的角色粗粒度授权。
2. 模型增加了继承约束和优先约束来解决权限分配过程中的职责分离问题，实现模型的灵活授权。为了解决模型中存在的授权冲突问题，提出了两种冲突解决策略：个体优先策略和优先级策略。灵活的实现了 IPC_URBAC 的最小特权和职责分离原则。
3. 需要特别强调 IPC_RBAC 模型引入用户直接授权的好处。模型中引入了角色，几乎用户直接授权是多余的，事实不然。

目前在大型的企业系统中，用户的业务需求灵活多变，系统的功能会不断的增加或更改，加上用户工作的需要，用户权限也可能灵活多变，角色的定义就会随着时间和业务的变化在不断的调整，因此仅仅通过调整角色来改变用户授权，权限管理会显得柔性不够，满足不了复杂变化的需要。例如下面的情况：

情形一、如果用户 a 由于工作需要暂时拥有权限 p1，但目前角色都不合适。如果给某个角色 r 指派此权限，那么会影响到拥有角色 r 的所有用户，因此行不通。如果新建一个

角色 r1，然后分派角色 r1 给用户 a，由于权限 p1 是临时的，用完后还需要撤销角色 r1，此种做法显然比较繁琐。

情形二、用户 a 拥有角色 r1，r1 的权限有(read，write，delete)，由于工作需要用户 a 需要短时期取消权限 delete，也就是说 a 需要拥有 r1 的绝大部分权限。这时如果新建一个角色 r2=(read，write)，然后指派给用户 a，会显得授权繁琐，因为授权只是暂时的，当 a 退出后还需要删除角色 r2。此时如果给 a 指派角色 r1，然后通过给用户直接进行负向授权，即不允许 a 拥有 delete 权限来对角色 r1 授权进行裁剪，裁剪掉 delete 就能满足用户权限需求。

扩展 IPC_URBAC 模型通过在 RBAC 基础上增加用户直接授权，通过继承约束控制用户直接授权实现角色权限的裁剪和增加，解决了以上授权问题，增强了模型的授权灵活性：

(1) 当用户的工作职责发生变化时，通过简单的用户直接授权，直接给用户赋予或撤销相应的权限，不需要额外的操作就可以实现权限的分配。

(2) 由于工作的需要，要求某些用户不能拥有整个角色的所有权限时，同过调整用户直接授权，采用继承约束来满足授权需求，而不需要建立新的角色，防止了角色泛滥。

通过对 IPC_URBAC 模型分析，扩展模型使得权限管理变的简单快捷、安全有效，适合大型企业系统的权限管理需要，IPC_URBAC 模型在实际的应用中占有很大优势。因此，下文将基于 IPC_URBAC 模型进行统一权限控制机制的设计开发，增强权限系统的通用性。

3.9 本章小结

本章首先分析了 RBAC 模型存在的不足，扩展 RBAC 模型提出了一种改进的授权模型即 IPC_URBAC 模型，该模型使得授权管理更加简便和灵活。接着，介绍了 IPC_URBAC 模型的设计原则、思想和形式化定义；给出模型的约束和冲突解决策略；描述了权限生成过程和基于冲突解决策略给出了模型的权限计算算法；详细分析了模型的竞争优势，为后文设计统一权限控制机制奠定基础。

第四章 基于 SOA 的统一权限控制机制设计与实现

本章主要介绍统一权限控制机制定义、体系架构、功能模型和设计实现。

4.1 统一的权限控制机制介绍

4.1.1 统一权限控制机制定义

统一权限控制机制：是基于服务复用思想和改进的 IPC_URBAC 模型设计，采用 B/S 架构和 Web Service 技术开发，能够为企业各种 Web 应用系统提供统一完整的权限管理服务，将权限管理从传统 Web 应用系统中最大限度的独立出来。实现用户信息的集中管理，实现权限管理技术的升级与应用系统分开，并通过调用 Web Service 定义的权限控制服务接口实现与不同 Web 应用系统柔性连接的权限管理系统。

统一权限控制机制的架构如图 4.1 所示。

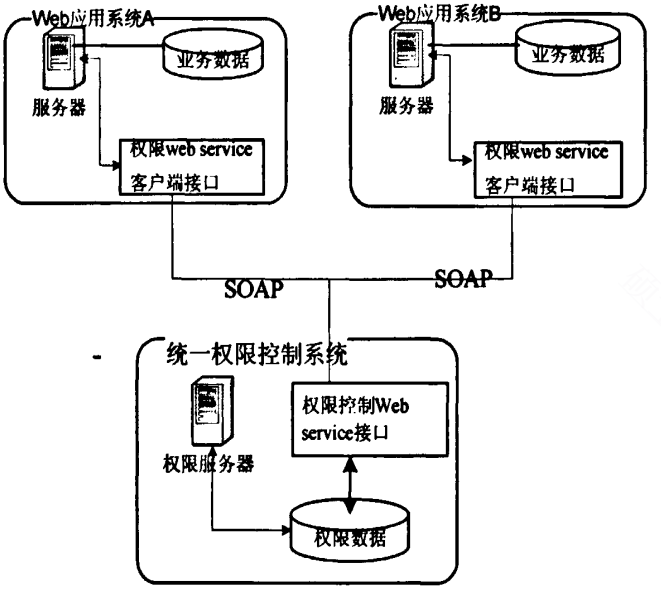


图 4.1 统一权限控制机制的架构

4.1.2 统一权限控制机制的组成结构

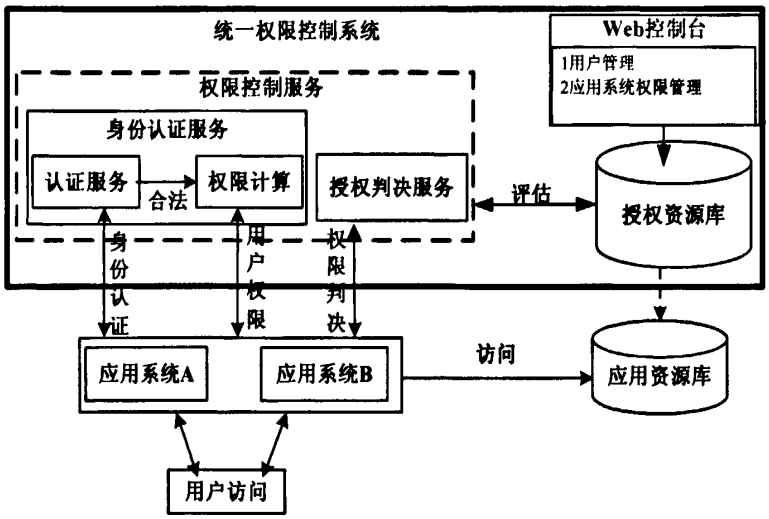


图 4.2 统一权限控制机制的组成结构

统一权限控制机制基于扩展的 IPC_URBAC 模型设计，为企业的各 Web 应用系统提供统一的用户授权和权限管理机制。并基于 Web 技术实现权限平台友好的用户界面，提供对整个权限系统中的用户、角色、权限、资源和操作等进行配置与管理。

如图 4.2 所示，统一权限控制平台主要包含 3 部分：

授权资源库：用于保存所有的用户信息和授权信息。它是权限系统和应用系统联系的纽带，权限系统通过访问或更新库中的权限信息来调整授权策略，权限系统通过库中的安全信息来控制用户对应用系统的访问。

Web 控制台：基于 B/S 的 web 页面，系统管理员用来统一定义、修改、管理授权和用户信息。 Web 控制台的功能包含：用户管理，挂载应用系统资源，为应用系统提供角色管理、权限管理、角色权限分配等功能，平台自身的权限管理。

权限控制服务：是权限控制平台与每个 Web 应用系统交互的 Web Service，接受每个应用系统的权限管理请求，评估相应策略，返回授权结果。由于身份认证和授权判决需要提供给不同的应用系统调用，为了方便集成不同的应用系统，本文将权限控制服务发布为 Web Service。 通过调用 Web Service 实现权限控制服务在应用系统的身份认证和权限判断功能，利用 Web Service 实现了权限系统与应用系统的逻辑划分，开发人员不用考虑权限控制的具体实现细节，直接调用服务即可，可以全心投入到业务系统的实现中，实现权限系统与不同业务系统解耦。

1．身份认证服务

身份认证服务包括认证和权限计算功能。认证是用户在访问应用系统验证用户身份的合法性，在身份认证合法后，用户获得相应的访问应用系统资格，同时系统将登录用户的角色、权限信息等存入 Session 中。以后用户在访问各个应用系统时，只需根据这些信息就能得到相应的权限控制，用户在各个应用系统之间切换时，无需重新的登录。

身份验证对任何应用系统都是必不可少的，传统的开发方法要求每个子系统都需要建立自己的身份认证模块，降低了开发效率。因此统一权限控制机制以 Web Service 实现了身份验证，所有应用系统都可以使用身份验证 Web 服务实现身份认证功能，提高了开发效率。

权限计算是在用户通过身份认证后，通过用户标识获取用户所有的资源权限。“权限计算”是粗粒度的权限控制，主要用来生成系统展现界面。应用系统的资源具有层次性，在数据库中采用树形结构存储，应用系统展现界面将用户权限内的资源以树形菜单形式展现给用户。

2. 授权判决服务

授权判决服务是细粒度权限控制，主要提供权限判断功能。授权判断就是计算用户对业务数据是否有查询、添加、修改和删除等权限。细粒度权限控制是控制应用系统 Web 窗口中的可视对象，如菜单项、按钮等，当用户访问具体资源时触发对象权限检查。

4.1.3 统一权限控制机制的意义

基于 SOA 思想的 Web Service 技术是一种新型的分布式计算方法，能够构建一种简单的、通用的、与平台和语言无关的松耦合系统，轻松完成不同应用系统的集成，成为下一代网络应用系统开发的优选技术。目前权限控制逐渐网络化、普遍化，因此基于 Web Service 构建的统一权限控制机制具有重要意义：

1. 统一性

统一性指权限控制的设计和部署与具体的业务系统无关[40-43]。统一权限控制机制适用于基于 Web 的任何系统。实现统一性具有以下优点：

(1) 可以快速构建新业务系统，不用进行权限管理的重复设计和开发，降低开发成本和缩短开发周期；另外，重用已有的方法，能够提高系统的使用性能，减少维护。

(2) 基于 Web Services 技术构建统一权限系统使得权限管理脱离了传统权限控制模式，构建的统一权限控制系统能够集中管理企业的每个应用系统的用户和授权信息。

(3) 统一访问控制便于实现企业多 Web 系统的权限控制集成，解决了信息孤岛问题，为企业开发企业信息化平台奠定了基础。

2. 授权管理更加灵活

IPC_URBAC 模型中引入的用户和角色混合授权，很好的解决了企业临时授权问题，模型中的角色优先约束，给管理员进行用户授权带来了方便，不用设置大量的冲突角色和冲突权限信息。

3. 具有专门的 Web 控制台

本文设计的统一权限控制机制基于 J2EE 技术开发，具有专门权限管理 Web 页面，来管理应用系统的模块、用户、权限等信息，在开发应用系统时无需开发新的权限控制页面。

4. 集成简单，方便升级和扩展权限管理系统

统一权限控制机制提供了与 Web 系统集成的 Web 接口，通过接口调用可以快速实现应用系统的权限控制。将权限控制发布成权限控制服务，使得应用系统开发者脱离了开发权限管理的花销，Web Service 具有的简单性、平台无关性和松耦合性使得统一权限系统适用于多种异构领域，并且可以降低权限系统和易用系统之间的耦合性。应用系统中使用 Web Service 的客户端接口来获取权限信息，具体实现在统一权限控制服务中实现，减少访问控制策略的改变对应用系统的影响。只要不改变权限控制提供的服务接口，内部权限管理可以任意升级和扩展，不会影响到应用系统的使用。

4.2 系统的权限控制设计

基于上面提出的统一权限控制机制，可以将权限控制从应用系统中抽取出来，独立开发为一个统一权限控制子系统，供不同的应用系统使用。下面介绍统一权限控制系统中各关键技术的设计和实现。

4.2.1 二进制权限掩码

权限掩码是一个数字，根据特定的含义拆分，每一个子项代表特定的权限[40]。二进制权限掩码是一个多位的二进制数，掩码的每一位与相应的权限对应，没有该权限时，对应位为 0，否则对应位为 1。权限掩码是根据用户的角色权限信息由系统动态生成的，掩码是权限判断的控制阀。为了系统权限设置具有可扩展性，本文权限掩码通过二进制计算来完成，首先建立一个功能枚举类：

```
public class Permission {
    public static final int CREATE=0;//添加操作
    public static final int READ=1;//读取操作
    public static final int UPDATE=2;//更新操作
    public static final int DELETE=3;//删除操作 ...
    }
```

为每一个操作设定一个唯一的整数值。理论上可以有 N 个操作。实现时取决于权限值的数据类型。本文采用 SQL Server 中的 Int 类型，则 N=32。当授权给用户时，按照这些数的“2 的权的和”来赋予权限。假设用户拥有权限：添加 A=0；读取=1；修改 A=2；删除 A=3。那么用户权限计算原理如表 1：

表 1 权限掩码计算

功能	2 的权计算	二进制表示
添加 A=0	2^0 = 1	00001
删除 A=3	2^3=8	01000
修改 A=2	2^2= 4	00100
用户权限 p	2^0+2^3+2^2＝13 == 01101	

表中，每一个操作权限值都是只有一个“1”和 N 个“0”组成的唯一排列，几个权限相加时就不会出现进位的问题。保证了每个操作权限都是独立的，不会出现权限的包容问题。权限值只是一个代号，与大小无关。如果要验证用户是否有删除 A 的权限，可通过“位与”运算来实现。在 java 中，“位与”运算符号为&，p&(2^3)=(01101)&(01000)=01000，删除 A=3 位置对应权限值为 1，因此有删除权限。

本文的权限判决函数为：

```
public static boolean getPermission(int permission，int aclState) {
    int tmp= (int)Math.pow(2, permission);
    return (aclState & tmp) = =permission;
}//aclState 是用户具有的总权限值，从数据库中获取；permission 是一个操作权限，为一个整数。
```

4.2.2 系统数据库设计

依照改进 IPC_URBAC 模型，建立了用户表(user)、角色表(role)、用户和角色关系表(userrole)，授权表 acl，资源表 module 和操作表(operator)六个表来实现模型中的实体和关系。数据实体关系图如图 4.3。

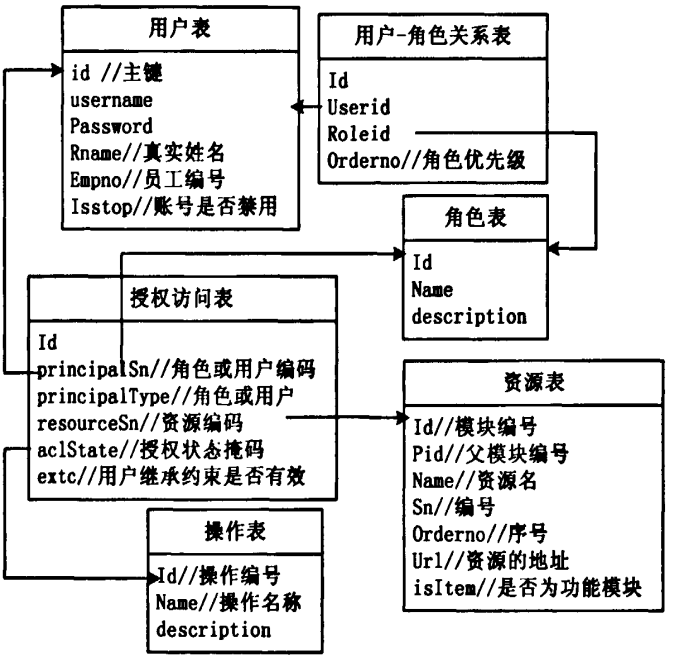


图 4.3 数据库实体关系图

(1) 用户表 user，用来保存所有需要使用系统资源的用户信息。Isstop 字段用来记录用户账号的可用状态。

(2) 角色表 role，用来保存角色信息。

(3) 用户和角色关系表 userrole，用来记录用户和角色的分配关系。Orderno 用来记录用户角色的优先级，解决角色授权冲突问题。

(4) 资源表 module，用来保存应用系统的所有模块和窗口信息。模块表设计成树形结构。Sn 表示模块编号，用来唯一标识一个模块，orderNo 是模块的序号，采用分级方式，序号的每两位表示一个级别。isItem 区分资源是数据对象或业务模块。这两个字段用来控制用户与角色拥有的系统权限查询时，以业务功能模块列表的树状结构显示出来。

(5) 授权访问表 acl，用来保存用户和角色的授权信息。用 principalType 来区分是用户或角色授权。extc 控制用户直接授权是否有效，实现 IPC_URBAC 中的继承约束。aclState 是权限掩码，记录用户或角色对资源的权限。

(6) 操作表 operator，保存应用系统定义的操作信息。

4.2.3 系统的功能模型

统一权限控制系统的具体功能模块包括用户管理、角色管理、模块管理、机构管理、功能管理和日志管理六部分,图 4.4 所示系统的整体用例包图。

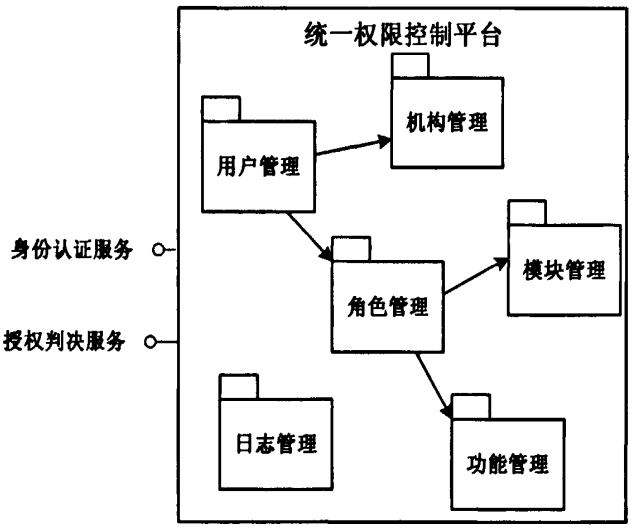


图 4.4 统一权限控制的用例包图

权限管理：为管理员和用户提供了使用平台的图形化界面，包含用户管理、机构管理、模块管理、权限管理、角色管理和功能管理。管理员更新、维护权限控制信息，平台能根据用户的身份返回用户权限信息。

用户管理：实现管理员对用户信息的注册、角色分配、用户直接授权和用户账号使用状态等管理。

日志管理：主要记录用户、管理员等的使用行为，确保系统安全。

4.3 统一访问控制系统的实现

4.3.1 基于 IPC_URBAC 的权限管理实现

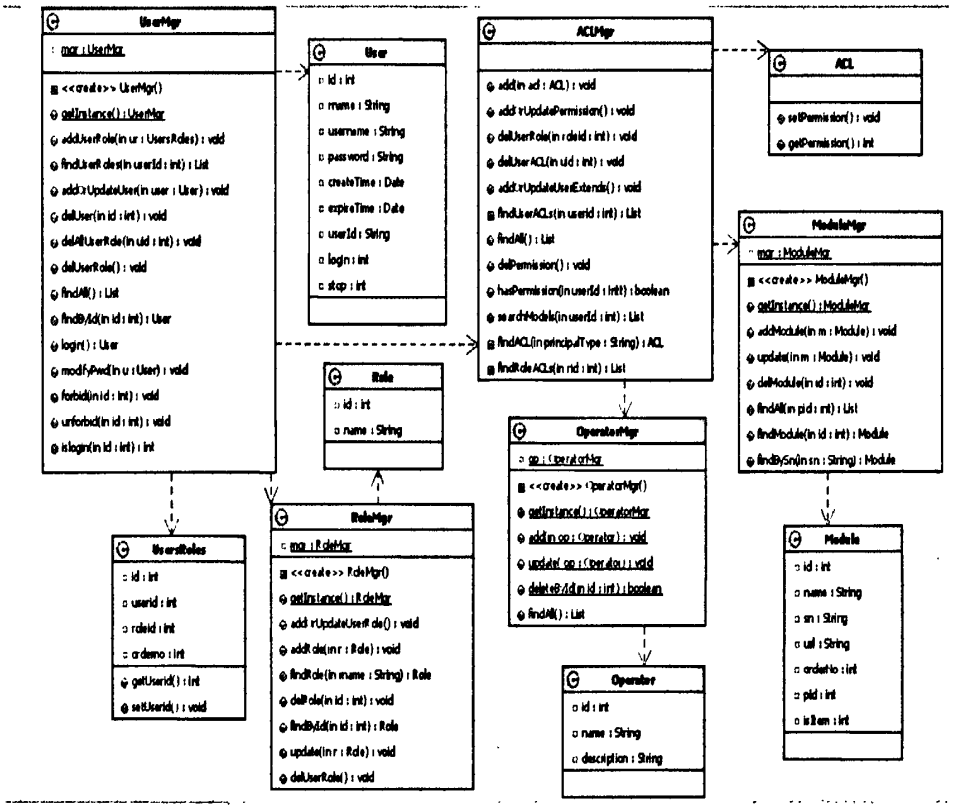


图 4.5　权限管理类图

图 4.5 所示权限管理核心类图，User，Role，Module，Operator 和 ACL 分别对应 user，role，module，operator 和 acl 数据表的实体类。

UserMgr 是用户管理的核心类，封装对用户信息各种操作，包含用户-角色分配管理、用户账户的冻结、解冻等功能，核心方法说明如下：

(1) searchUserRoles(int userId) 查找用户拥有的角色

(2) addOrUpdateUserRole(int userId,int roleId,int orderNo) 添加或更新用户拥有的角色，如果用户[userId]已经拥有角色[roleId]，则更新其优先级[orderNo]，否则给用户分配相应的角色，并设置优先级

(3) delUserRole(int userId,int roleId) 删除分配用户的角色（关联）

(4) login(String username,String password) 用户登陆操作

(5) forbid(int userid) 冻结用户账号

(6) unforbid(int userid) 解除用户账号冻结状态

RoleMgr 是角色管理的核心类，封装对每个 Web 应用系统的角色信息的管理操作。

ModuleMgr 是模块管理的核心类，封装对挂载的所有 Web 信息系统的模块信息的管理操作。

ACLMgr 是权限管理的核心类，封装权限管理的各种操作方法。主要方法描述如下：

(1) addOrUpdatePermission(String principalType, int principalSn,intresourceSn, int permission, boolean yes) 授权函数，给主体类型[principalType]为 Role 或 User 的主

体[principalSn]授予是否具有[yes]权限功能[permission]

(2) addOrUpdateUserExtends(int userId, int resourceSn, boolean yes) 设置用户某个资源授权的继承特性

(3) delPermission(String principalType, int principalSn,int resourceSn) 删除用户或角色的授权

(4) hasPermission(int userId, int resourceSn, int permission) 授权判决函数，验证用户[userid]是否对资源[resourceSn]具有功能权限[permission]

(5) hasPermissionByResourceSn(int userId, String resourceSn, int permission) 根据资源名称判断用户的权限，内部调用方法 hasPermission

(6) searchModels(int userId) 搜索某个用户拥有读取权限的模块列表（用于登录，形成导航菜单的时候）

(7) findACL(String principalType, int principalSn,int resourceSn) 根据主体类型、主体标识和资源标识查找授权 ACL 实例

(8) findRoleACLs(int roleId) 根据角色编号查找 ACL 授权实例

(9) findUserACLs(int userId) 根据用户查找直接授予用户的授权列表（注意：如果直接授予用户的授权是继承的话，则不应该包含在这个列表中），返回的列表元素是：ACL 实例

(10)searchAclRecord(String principalType, int principalSn) 返回主体 principalSn 的授权 ACL 列表。

OperatorMgr 是应用系统操作管理的核心类，封装对应用系统的各种业务操作的管理。

4.3.2 基于个体和优先的 CRS 授权策略实现

授权策略的实现是权限控制的核心。本文将权限控制服务分为决策查询策略和决策判决策略。决策判决策略是判断某用户能不能访问某资源，返回判决结果“允许”或“拒绝”；决策查询策略是查询某登录用户能够访问的资源，返回结果是由资源定义的业务数据集合。

在授权策略的实现中必然要解决授权冲突问题，实现时使用 CRS1 和 CRS2 来解决授权冲突和授权重复问题。

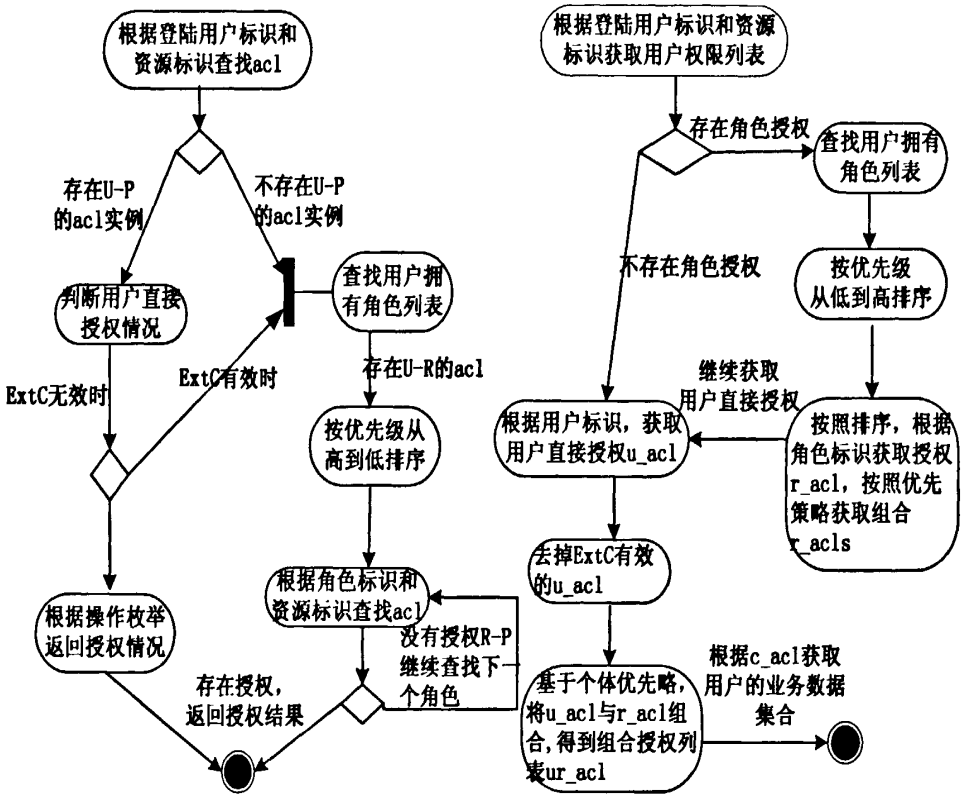


图 4.6 决策判决和查询流程

决策判决实现流程如图 4.6 左部所示，通过调用系统提供的决策判决服务来完成应用系统的权限验证，决策判决函数代码实现如下：

```
public boolean hasPermission(int userId, int resourceSn, int permission) {
//根据 CRS1，查找直接授予用户的授权列表 acl
ACL acl = findACL(ACL.TYPE_USER, userId, resourceSn);
if(acl != null){//存在用户直接授权 acl
        boolean extc= acl.getExtc(permission);//获取继承约束标识
         if(!extc){ //如果 extc=false，返回授权结果。
                  return getPermission(permission,acl.getAclState()); }
}
//不存在用户直接授权时，获取用户的有序角色列表
//rs 存储按优先级从高到低的用户角色集，依照 CRS2 从高到底依次判断每个角色的授权 acl
 while (rs.next()){
    acl = findACL(ACL.TYPE_ROLE, rs.getInt("rid"), resourceSn);
    if (acl != null)//找到正确的 acl
   return acl.getPermission(permission)；//使用权限掩码判决函数返回判决结果
                }
```

```
//省略部分代码
return false;//没有找到用户或角色授权时，返回 false。
}
```

查询授权策略采用上一章的算法 PermissionOfUR (u)计算用户的所有权限。查询授权流程如图 4.6 右部。本文采用 JAVA 的 Map 接口实现基于个体和优先的冲突策略，Map 存储<键,值>(<key, value>)对，由于 Map 的 key 具有不可重复性。分别存储 acl 表中的资源编号和权限掩码。实现代码从略。

4.3.3 基于 Web Service 的权限控制服务设计

以上论述了权限控制功能设计与实现，为了实现一种通用的、与应用系统解耦的统一权限控制，方便集成不同的 Web 应用系统，需要提供公共的接口让开发者调用。本文将身份认证和权限判断设计为 Web Service，对外部提供统一调用接口。本系统采用开源 Web Service 开发包 Xfire 来实现 Web 服务开发。下面只给出授权判决服务 PermissionService 的实现过程

首先设计系统的授权判决 Java 服务层接口和实现类：

接口类：

```
package com.permission.service;
public interface PermissionService {
 //省略其它代码
    public boolean hasPermission(int userId, int resourceSn, int permission) ;
}
```

实现类：

```
package com.permission.service;
public class PermissionServiceImpl implements PermissionService {
        public boolean hasPermission(int userId, int resourceSn, int permission) {
          // 实现代码参照上文
          return false;
      }
}
```

在 web.xml 文件中配置 Xfire，如下：

```
<servlet>
    <servlet-name>XFireServlet</servlet-name>
 <servlet-class>
```

```
    org.codehaus.xfire.transport.http.XFireConfigurableServlet
  </servlet-class>
      <load-on-startup>0</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>XFireServlet</servlet-name>
    <url-pattern>/services/*</url-pattern>
  </servlet-mapping>
```

创建服务 services.xml 进行服务接口定义如下：

```
<beans xmlns="http://xfire.codehaus.org/config/1.0">
  <service>
      <name>PermissionService</name>
      <serviceClass>
          com.permission.service.PermissionService
      </serviceClass>
      <implementationClass>
          com.permission.service.PermissionServiceImpl
      </implementationClass>
      <style>wrapped</style>
      <use>literal</use>
      <scope>application</scope>
  </service>
</beans>
```

通过上面设置完成了授权判决服务的配置，通过服务发布得到 PermissionService.wsdl 文件，URI=http://localhost:8888/pm/services/PermissionSevice?wsdl，外界所有 Web 应用系统都可以调用此 Web 服务接口，实现和平台的无缝连接。应用系统在开发时通过 MyEclipse 的 Xfire Client Library 调用发布的服务描述文件 wsdl 生成 Service 客户端调用代码。如图 4.7 所示。

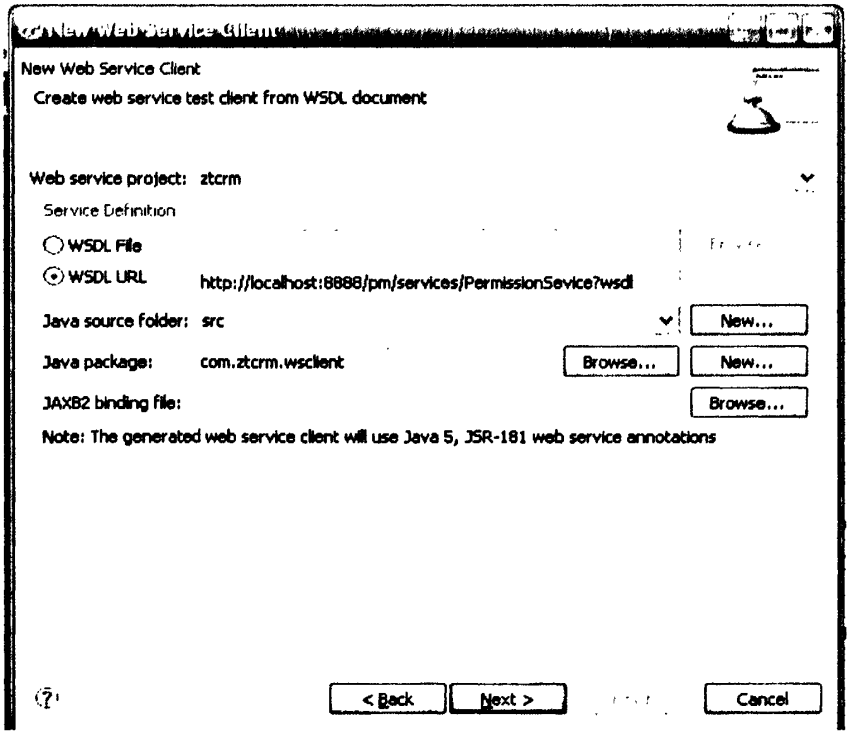


图 4.7 Xfire Web 客户端生成

4.4 统一权限控制机制与 Web 应用系统

4.4.1 二者的关系

统一权限控制机制与 Web 应用系统的关系如图 4.8 所示。

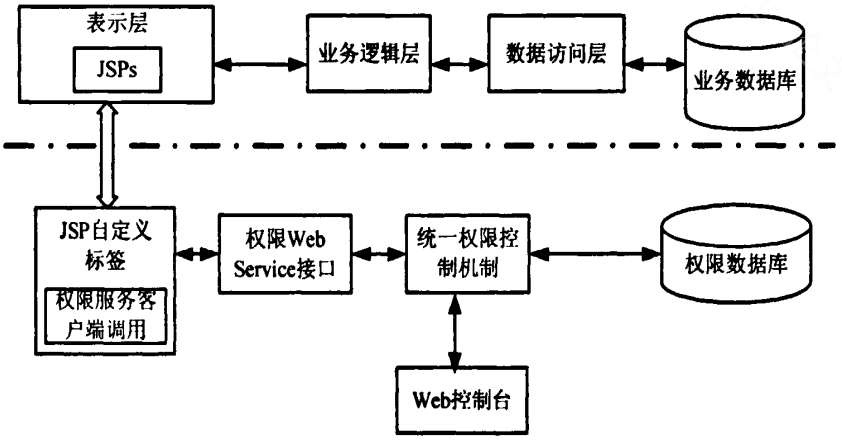


图 4.8 统一权限控制机制和 Web 信息系统的关系

图 4.8 中虚线上半部分是 Web 应用系统，采用第二章设计的 J2EE-SOA 分层架构设计。虚线下部分是统一权限控制机制，它作为独立于应用系统的权限独立子系统，对 Web 应用系统的业务资源进行访问控制。Web 应用系统通过使用统一权限控制机制提供的 Web 服务接口，实现对系统资源的权限控制管理。在运行中，权限 Web Service 接口会将访问结果提供给 Web 应用系统，实现对应用系统资源的安全访问。

4.4.2 统一权限控制机制的应用集成

第一步，根据系统对外提供的 Web Service 接口规范，开发应用系统的访问控制客户端，内嵌到新开发的应用系统。本文使用 MyEclipse 提供的 XFire 的客户端类库编写 Web Service 客户端。

第二步，使用系统的 Web 控制台在数据库中添加该应用系统的实例。

第三步，通过 Web 控制台设置应用系统的权限、角色、模块资源等信息。

第四步，在新应用系统开发完成后，新应用系统运行时会自动调用 Web Service 接口从授权资源库中获取安全信息进行权限控制。

4.4.3 基于 Web Service 的统一权限控制流程

用户通过登录认证之后，获取用户的所有权限信息并存入授权证书 Session 中，当用户访问具体应用系统时，应用系统根据 Session 中的权限数据来判断用户的访问权限，如果存在，则显示系统资源，然后根据用户操作来决定是否执行。登录认证流程如图 4.9 所示。

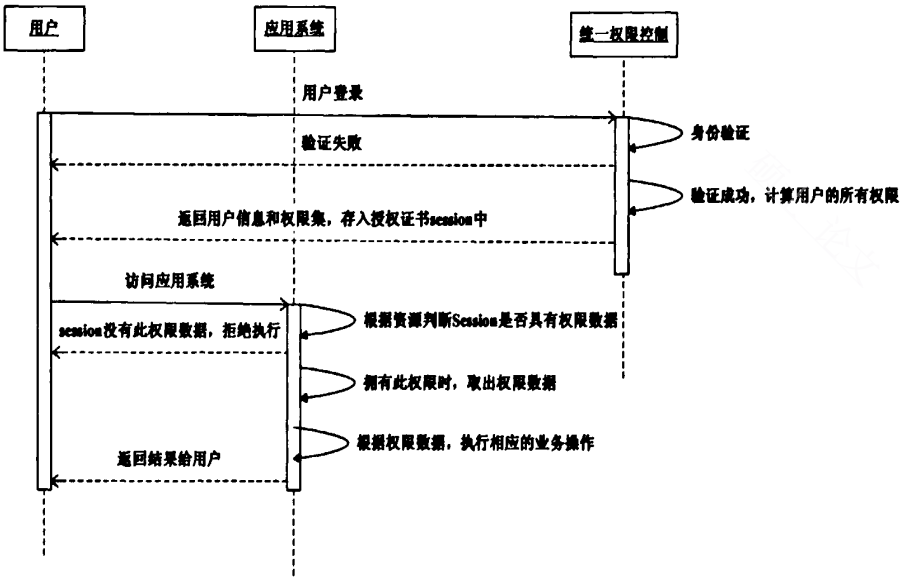


图 4.9 登录认证序列图

基于 Web Service 的权限控制的工作流程描述如下：

(1) 用户通过 Web 浏览器登录应用系统，通过身份认证服务的认证后，获取用户的授权证书 Session，用户要对某个应用系统资源进行访问时，通过浏览器发送访问资源的请求。

(2) Web 服务器中的受控资源接到用户访问请求后，应用系统与权限平台的权限控制

服务器建立连接，权限控制服务器接收到用户权限验证请求后，调用权限系统提供的授权判决 PermissionService 提供的权限验证函数 hasPermission（int userId, int resourceSn, int permission），验证用户是否对待访问的应用资源具有访问权。

(3) 权限控制服务通过与授权策略库连接处理访问请求，最后输出验证结果返回给应用系统。在权限控制服务器端，hasPermission 通过 userid 获取用户的直接授权列表、角色列表，依照冲突解决策略 CRS1 和 CRS2 判断是否拥有对应用系统访问资源的访问权限。

(4) 应用系统根据返回的验证结果，如果通过，则应用系统将资源访问权限授予访问者。

4.5 本章小结

本章提出了一种新型的基于 Web Service 和 IPC_URBAC 模型的统一权限控制机制。首先给出统一权限控制机制的思想、组成结构和实现意义；然后详细介绍了统一权限控制系统中 IPC_URBAC 权限控制策略的设计，设计二进制权限掩码实现灵活授权，给出平台数据库设计和系统功能模型；接着给出了系统的实现过程，包括权限控制核心类图、基于个体和优先 CRS 授权策略实现，并设计了 Web Service 接口实现权限系统与应用系统的解耦；最后给出了统一权限控制机制与业务系统的集成方法和权限控制流程。

第五章 统一权限控制机制的应用

统一权限控制机制是一种新式的权限管理模式，理论研究只有获取很好的实际效果才能证明研究意义。本章将统一权限控制机制用于我们开发的 CRM 系统中。基于改进 IPC_URBAC 模型的授权的灵活性和可用性，统一权限控制系统在具体实现时采用 J2EE 技术和 SOA 思想，同时为了给用户更好的体验，增强系统的可视化和交互性，客户端采用了具有强大异步交互能力的 AJAX 技术 DWR 进行客户端开发。

5.1 CRM 系统

5.1.1 CRM 介绍

CRM，即客户关系管理，是指用计算机对有关分析流程自动化的软件系统，其中涉及销售、市场营销、客户服务以及支持应用等软件。该系统的开发主要目的是满足山东中泰阳光电气科技有限公司客户关系管理需求，并对其公司的各项业务进行梳理与整合，使其更好的满足市场的需求。同时该软件在适用中泰阳光业务流程的基础上，在客户关系管理上还具有一定的通用性。

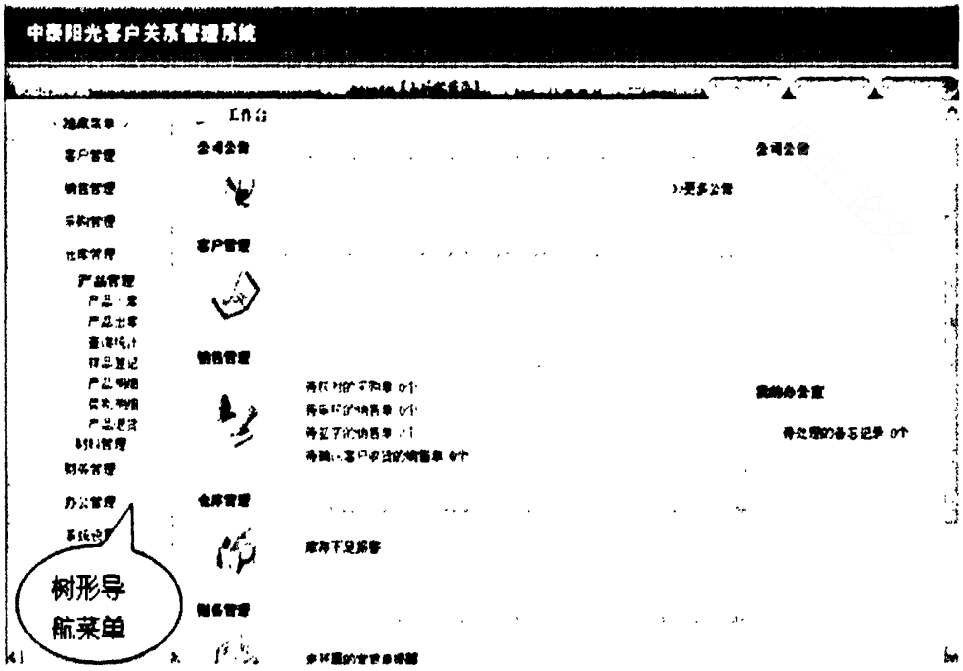


图 5.1 中泰阳光 CRM 系统主界面

CRM 系统逐渐发展为企业办公自动化平台，如图 5.1 所示，主页面左侧显示 QQ 下拉导航菜单，包括了 CRM 系统所有的功能模块。CRM 集成了客户关系管理、销售管理、采购管理、仓库管理、财务管理、办公管理等六个子系统，为了方便企业管理，会不断升级

系统加入新的子系统。CRM 系统应用到公司的每个部门，用户数量繁多，并且每个用户的权限各不相同，因此系统的权限管理工作繁重。因此本文尝试将权限控制部分最大限度的解耦出来，利用基于 IPC_URBAC 的统一权限控制机制来设计实现 CRM 系统的权限控制，同时增强了 CRM 的可扩展性，将来加入新的子系统时，只需要通过统一权限控制机制进行授权设置就可完成对新系统的权限控制。如图 5.2 所示，基于统一权限控制机制的 CRM 系统的实施。

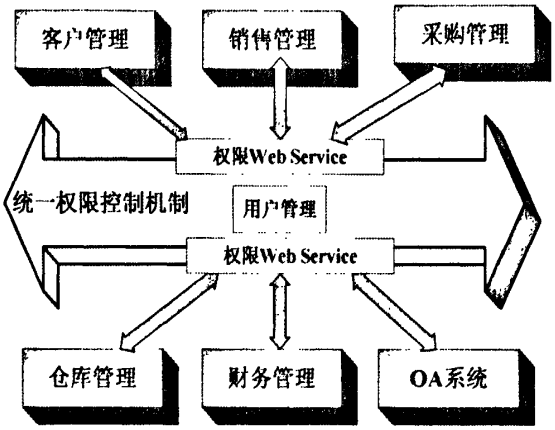


图 5.2　基于统一权限控制机制的 CRM 系统实施

5.1.2 CRM 系统的开发架构

CRM 系统开发基于 J2EE 的 MyEclipse 开发平台，在集成权限控制系统时需要调用统一权限控制提供的身份认证和授权判决服务，为了在系统中集成权限控制 Web 服务，CRM 使用第二章设计的 J2EE-SOA 分层架构进行开发。CRM 系统体系结构如图 5.3，左侧表示 CRM 业务系统的开发架构，右侧表示与权限管理集成时的开发架构。

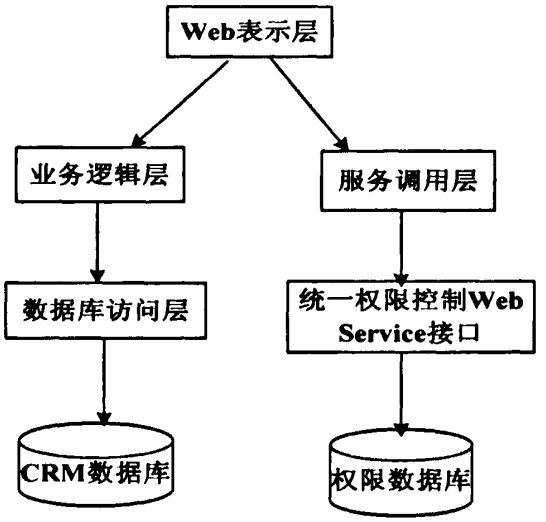


图 5.3　CRM 体系结构

服务调用层：是 Web Service Client，可以直接被 Web 表示层调用。CRM 系统在调用统一权限控制提供的 Web 服务时，需要通过 WSDL 文件首先生成 Web 服务客户端程序。

5.2 CRM 系统中权限管理实现

权限管理包含角色管理、用户管理、模块和功能管理和授权管理，各个功能的内部管理流程如图 5.4。用户授权是在角色、用户和系统资源模块存在的条件下进行。

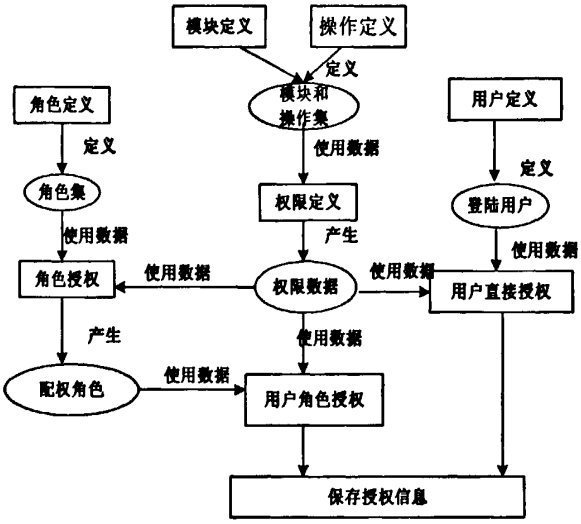


图 5.4 授权管理内部功能图

5.2.1 资源定义

由图 5.5 可知，CRM 系统资源包含客户管理、销售管理、采购管理、仓库管理、财务管理、办公管理六个子系统模块，每个子系统模块包含下级业务模块信息，仓库管理包含产品管理和材料管理两个二级业务模块，每个业务模块中又包含许多三级业务模块，如产品入库、出库和样品登记等。可见 CRM 系统资源分为三级层次关系。

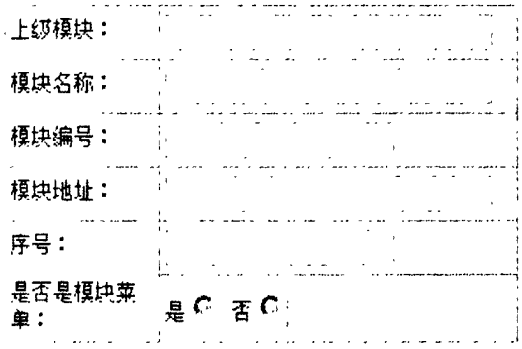


图 5.5 资源添加页面

如图 5.5，系统管理员在权限控制平台的模块管理定义界面添加应用系统的资源模块信息，首先登记客户管理、销售管理等一级子系统信息，并指定上级模块为 CRM 系统，然后分别登记各个子系统的下级模块，如产品管理、材料管理，并指定上级模块是仓库管理，最后登记底层业务模块，如产品入库等，指定上级模块是产品管理。为了方便实现资源分级显示，资源序号使用分级编码方式，从左到右依次代表从高到低的资源分级关系，

如仓库管理模块下的产品入库对应的序号是 01040201，编号对应的资源分级关系如图 5.6。

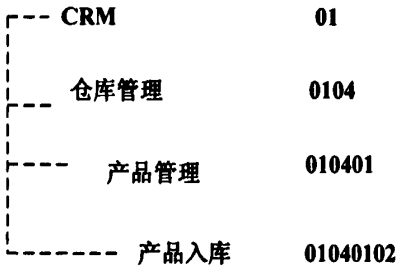


图 5.6 资源分级

模块地址对应业务模块对应的操作页面地址，底层业务模块对应具体的 Web 操作页面。根据资源的分级存储，平台实现树形菜单显示，如图 5.1 左侧所示。

5.2.2 权限定义

为了实现业务操作的重用，本文将权限划分为资源和操作两部分，将操作与业务模块分离，CRM 系统主要定义了增加、删除、修改、查询等操作。在操作管理页面增加操作信息，并选择需要此操作的业务模块。如表 2 的仓库管理中产品入库、查询统计中权限定义。

表 2 权限定义实例

业务模块	定义的权限
产品入库	增加产品
查询统计	查询产品 删除产品

5.2.3 机构和角色定义

在权限平台的机构管理中增加企业的机构信息，企业机构信息同样使用分级方式加入。然后在角色管理页面中增加机构对应的角色信息。实现过程从略。

5.2.4 用户直接授权实现

用户直接授权时，通过 IPC_URBAC 定义的用户继承约束 IC 来说明用户授权是否有效。用户授权如图 5.7 所示。系统模块分级显示，授权下级模块时同时需要授权上级模块。“不继承”复选框用来控制用户直接授权的有效性，选中时用户直接授权有效。“启用”复选框来灵活取消用户的授权。

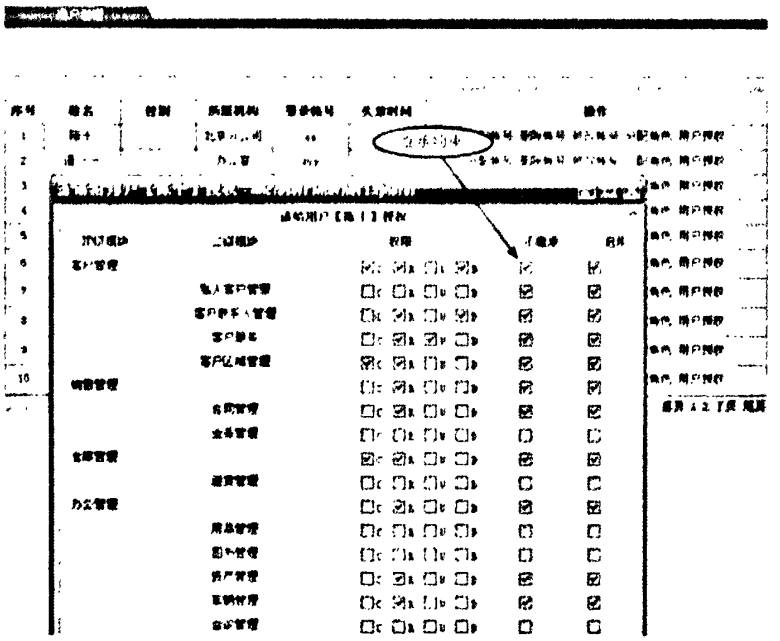


图 5.7 用户直接授权页面

角色授权页面和用户授权类似，区别是不存在用户继承约束。上面的授权方式，可以适用任何系统，只要知道系统的资源，每个资源拥有的访问权限，建立好枚举，就可实现权限的设置、分配和检查。

授权界面采用基于 AJAX 的 DWR 框架来实现用户授权，实现页面无刷新与服务器异步交互。DWR 是一个可以允许你去创建 AJAX WEB 站点的 JAVA 开源库。它可以让你在浏览器中的 Javascript 代码调用 Web 服务器上的 Java 代码，就像在 Java 代码就在浏览器中一样，页面核心代码如下：

```
<!--//省略部分代码 ,下面是引入DWR  -->
<script language="javascript" src="script/public.js"></script>
<script type="text/javascript" src="dwr/engine.js"></script>
<script type="text/javascript" src="dwr/util.js"></script>
<script type="text/javascript"src="dwr/interface/aclManager.js"/>
<script type="text/javascript">
//授权
function addOrUpdatePermission(field){
    //如果被选择上，则同时选择其"不继承"和"启用"checkbox
    if(field.checked){
        $(field.resourceSn+"_USE").checked = true;
//如果是用户直接授权，那么不继承复选框有效，否则无效。
        <c:if test="${aclForm.principalType eq 'User' }">
        $(field.resourceSn+"_EXT").checked = true;
        </c:if>
```

```
    }
    aclManager.addOrUpdatePermission(
        "${aclForm.principalType}",
        ${aclForm.principalSn},
        field.resourceSn,
        field.permission,
        field.checked
    );
}
//设置用户的继承约束
function addOrUpdateExtends(field){
    aclManager.addOrUpdateUserExtends(
        ${aclForm.principalSn},
        field.resourceSn,
        !field.checked
    );
}
//点击启用checkbox
function usePermission(field){
    //如果checkbox被选中，意味着需要更新ACL的状态
    //更新C/R/U/D以及Extends状态
    //设置为同步方式，以便DWR依次发出下列请求
    dwr.engine.setAsync(false);
        if(field.checked){
        addOrUpdatePermission($(field.resourceSn+"_C"));
        addOrUpdatePermission($(field.resourceSn+"_R"));
        addOrUpdatePermission($(field.resourceSn+"_U"));
        addOrUpdatePermission($(field.resourceSn+"_D"));
        <c:if test="${aclForm.principalType eq 'User' }">
        addOrUpdateExtends($(field.resourceSn+"_EXT"));
        </c:if>
    }else{
        aclManager.delPermission(
            "${aclForm.principalType}",
            ${aclForm.principalSn},
            field.resourceSn
```

```
            );
            $(field.resourceSn+"_C").checked = false;
            $(field.resourceSn+"_R").checked = false;
            $(field.resourceSn+"_U").checked = false;
            $(field.resourceSn+"_D").checked = false;
            <c:if test="${aclForm.principalType eq 'User' }">
            $(field.resourceSn+"_EXT").checked = false;
            </c:if>
        }
    }
    //省略部分代码
```

5.2.5 用户角色分配实现

角色分配就是给用户分配需要的功能角色。当用户同时充当多个角色并且权限重复时，重复的权限仅一次有效，用户拥有他充当的所有角色的权限的并集，为了防止多角色权限冲突问题，在角色分配时加入了优先级约束，实现权限判定时使用优先约束来消除角色授权冲突。角色分配实现如图 5.8。

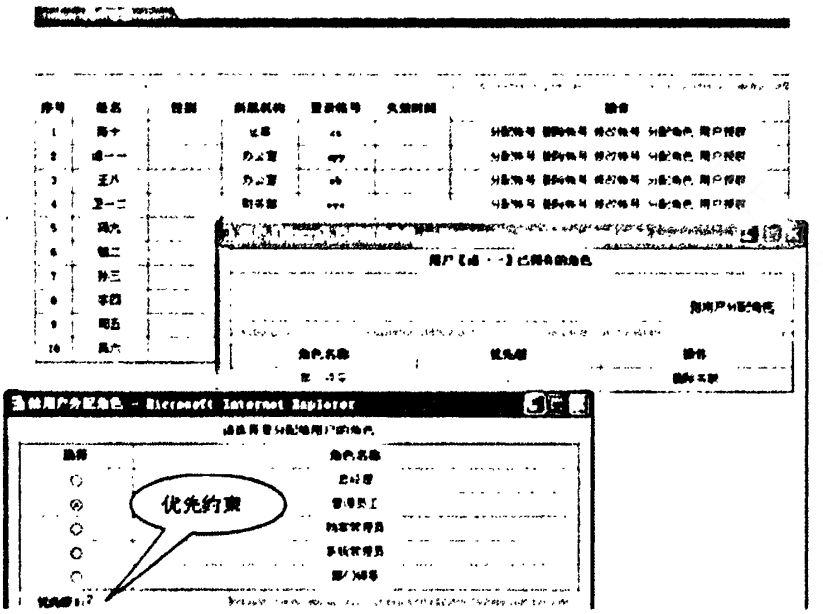


图 5.8 优先约束的角色分配界面

5.2.6 基于 Web Service 的权限控制实现

用户登录时菜单权限控制流程是：用户登录时，首先调用身份认证服务验证其使用系统的身份，通过相应用户名通过查询授权策略取得有权限操作的页面及在各页面的操作权限信息，然后返回给应用系统并展现给用户一个可以操作的模块资源导航树，树中包含用

户可以操作的所有页面的链接。图 5.1 是用户“manager”登录后的 CRM 主页面。

由于每个应用系统对授权访问的检查机制是相同的，在 CRM 开发中调用统一权限控制提供的一个 Web 服务 PermissionService 来实现对象权限的检查，达到应用系统与权限管理的解耦。在该 Web Service 中给出了一个方法 hasPermission()，其参数 userId 为登录用户，resourceSn 是用户要访问的数据，permission 为功能枚举常量。返回值“true”或“false”，相当于令牌。若为“true”，说明该用户具有该功能号所标识的功能。每个模块可以根据返回值决定用户可以访问的功能(页面、菜单、控件和数据元素)。

通过编写 Web 服务客户端代码调用 PermissionService，然后将客户端权限检查代码封装成 JSP 标签函数，为应用系统的每个页面引用，增强权限控制的灵活性。

每个页面引用 JSP 自定义标签<%@ taglib prefix="my" uri="http://www.fgl.com/ztcrm/functions" %> 来实现按钮等页面数据对象的权限控制。在每个需要控制的数据对象前加入 JSP 标签判定语句。例如 “删除”操作权限的判断代码为：

```
<c: if test="${my:hasPermission(login.id,'person',3) }">
<!--//下面是<c:if>的 body-->
<a href="#" onclick="del('person.do?method=del&id=${person.id }');">删除</a>
</c :if>
```

如果返回真，表示用户有权限访问，那么<c: if>的 body 部分的页面数据对象可视，否则<c:if>的 body 不被执行，页面数据对象不可见。

5.3 本章小结

本章具体介绍了统一权限控制机制在我们开发的山东中泰阳光公司的CRM系统中的应用，统一权限控制机制是一种新型的权限控制方式，不仅需要加强理论研究，更需要在实践中验证其有效性。

第六章 总结和展望

6.1 总结

访问控制是一种实现企业信息安全的有效措施，阻止用户的非授权访问。针对目前访问控制模型和权限管理系统本身存在的不足，本文采用理论与实践相结合的方法，扩展 RBAC 模型提出了一种更为完善和灵活的访问控制模型，并基于 SOA 提出一种新的权限控制机制。本文的工作总结和取得的成果如下：

（1） 深入研究了 MAC、DAC 和 RBAC 三种传统访问控制模型，指出其优缺点和在目前网络应用领域存在的问题。通过详细介绍角色访问控制模型突出了 RBAC 的优势和存在的主要问题。结合 J2EE 分层架构优点和 SOA 的特点，设计了 J2EE-SOA 分层架构。

（2） 根据 RBAC 模型是在角色级管理和控制用户的权限，难以满足复杂情况变化的需要；RBAC 模型中缺少完善的冲突解决策略，只能通过管理员在授权时消除冲突，授权复杂且存在安全隐患。改进传统 RBAC 模型，设计了继承和优先约束驱动的用户和角色混合的授权模型 IPC_URBAC 模型。模型加入了继承机制来控制用户直接授权，引入角色优先约束来消除角色权限冲突。同时设计了个体优先策略 CRS1 和优先级策略 CRS2 来解决用户和角色混合授权中存在的权限冲突问题。

（3） 在 IPC_URBAC 的设计中，给出了基于用户和角色混合的权限生成过程，并设计了扩展模型的权限计算算法，并分析了 IPC_URBAC 模型的应用优势。

（4） 基于 SOA 和 IPC_URBAC 模型提出了统一权限控制机制，通过 Web 服务实现权限控制与应用系统的解耦。统一权限控制机制可以同时管理企业的多个应用系统，不仅方便了企业管理，同时也能提高企业应用开发效率。它存在通用性、集中性、松耦合和跨平台等优点。

（5） 详细介绍了统一权限控机制的设计和实现过程，包括数据库设计、权限掩码设计、基于个体和优先的 CRS 授权策略设计和权限管理的功能模型等。并给出了基于 Web Service 的权限控制服务的实现过程，以及统一权限控制机制与应用系统的集成方法和访问控制流程。

（6） 结合作者参与的中泰阳光 CRM 系统平台的开发，阐述了统一访问控制机制在企业管理平台中的应用实现。实践表明统一权限控制机制使用 Web 服务将权限管理从应用系统中独立出来，实现了权限管理的重用；通过 IPC_URBAC 模型实现了更加柔性化的授权管理。统一权限控制机制为企业实现对所有 Web 应用系统的一站式管理提供了可能，为企业应用系统的权限管理集成提供了参考方案。

6.2 展望

随着Web服务的应用普及，Web服务本身存在很多安全隐患；企业信息化水平的提高，需要建立一种高效的权限管理方法实现企业多Web应用系统的权限管理集成。因此，今后还需要在以下方面努力：

首先，进一步完善IPC_URBAC模型，实现更加灵活和完善的授权管理，并能够应用到对Web服务的访问控制管理中，实现对Web服务的安全管理。

其次，统一权限控制机制只是一个模型，还存在许多问题，需要进一步完善，以期设计出一种权限服务平台来实现企业多Web系统的统一权限控制，达到企业多Web系统的权限控制集成，实现企业的统一用户管理、统一权限管理和统一身份认证机制。

最后，权限管理只是保证Web系统安全的一个方面，需要进一步研究与其它安全机制，如加密传输、审计等的融合。

参考文献

[1] Liu Hong-yue, Fan Jiu-lun, Ma Jian-feng.Research Advances on Access Control [J]. mini-micro systems, 2004, 25(1): 56-59.

[2] 林闯, 封富君, 李俊山. 新型网络环境下的访问控制技术[J]. 软件学报, 2007, 18(4): 955-966.

[3] Ravi Sandhu, Coyne E J, Feinstein H L et al.Role_Based Access Control Modals[J]. IEEE Computers, 1996, 29(2): 8-47.

[4] David F.Ferraiolo, Ravi S.Sandhu,Serban Gavrila, D.Richard Kuhn a Ramaswamy Chandr-Amouli. Proposed NIST standard for role-base access control [J]. ACTransanctions on Information and Systems Security, 2001-9, 3: 224-274.

[5] Sanhu R, Bhamidipati V, Munawer Q. The ARBAC97 model for role-based administration of roles [J]. ACM Transactions Information and System Security, 1999, 2(1): 105-135.

[6] Miehael J. Covington, MatthewJ. Moyeretal.Generalized Role-Based Access Control for Securing Future Applications[C]. Proe.2000 National Information Systems Seeurity Conference Baltimore, MD, 2000-10.

[7] 乔颖, 须德, 戴国忠. 一种基于角色访问控制(RBAC)的新模型及其实现机制[J]. 计算机研究与发展, 2000, 37(1): 37-44.

[8] 郭惠, 李阳明, 王丽芬. 基于角色和任务的访问控制模型[J]. 计算机工程, 2006, 32(16): 143-145.

[9] 邢汉发, 许礼林, 雷莹. 基于角色和用户组的扩展访问控制模型[J]. 计算机应用研究, 2009-4, 26(3): 1098-1100.

[10] 许峰, 林果园, 黄皓. Web Services 的访问控制研究综述[J]. 计算机科学, 2005, 32(2): 1-4

[11] Wonohoesodo, R.; Tari, Z. A role based access control for Web services[C]. Sevices Computing, 2004. (SCC2004). Proceedings. 2004 IEEE International Conefrence. 2004-9: 49-56.

[12] 许峰, 赖海光等. 面向服务的角色访问控制技术研究[J]. 计算机学报, 2005, 28(4): 686-693.

[13] Bertino Elisa, Bonatti Piero Andrea, FerrariElena. TRBAC: A Temporal Role-Based Access Control Model[J]. ACM Transaetionson Information and Systems Seeurity, 2000, 4(3): 21-30.

[14] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A generalized temporal role-based access control model[C]. IEEE Trans. Knowl. Data Eng. 2005, 17(1): 4–23.

[15] 黄建, 卿斯汉, 温红子. 带时间特性的角色访问控制[J]. 软件学报, 2003, 14(11): 1944-154.

[16] 韩伟力, 陈刚, 尹建伟等. 权限约束支持的基于角色的约束访问控制与实现[J]. 计算机

辅助设计与图形学学报, 2002, 14(4): 333-338.

[17] Thomas R, Sandhu R. Discretionary access control in object-oriented databases: issues and research directions[C]. Proceedings of the Sixteenth National Computer Security Conference, Baltimore, MD, 1993: 63-74.

[18] Sandhu R. Mandatory controls for database integrity [J]. Proc. of the IFIP WG11, Worksh op on Database Security, Monterey, Californian, 1989, 143-150.

[19] AviS Sandhu, David Ferraiolo, Richard Kuhn. The NIST Model for Role based Access Control: Towards an Unified Standard [J]. ACM, 2000, 47-63.

[20] 胡金柱, 陈娟娟. RBAC 模型中角色的继承与互斥问题的研究[J]. 计算机科学. 2003, 30(11): 160-163.

[21] 史永昌, 鲁书喜. 基于多父角色 RBAC 模型的研究与应用[J]. 计算机工程. 2008-9, 34(17): 183-185.

[22] 朱佃波. Web 信息系统中统一细粒度访问控制的研究[D]. 苏州大学硕士学位论文, 2008-4.

[23] 黄锐. 一种动态 RBAC 模型研究[D]. 四川大学硕士论文, 2006-4.

[24] 董光宇, 卿斯汉. 带时间特性的角色授权约束[J]. 软件学报,2002, 13(8): 1521-1527.

[25] 汪厚祥, 李卉. 基于角色的访问控制研究[J]. 计算机应用研究, 2005, 4: 125-127.

[26] 梁爱虎编著. SOA 思想、技术与系统集成应用详解[M]. 北京: 电子工业出版社, 2007-12.

[27] Raghu, Koadali. What is service-oriented architecture[EB/OL]. http://www.javaworld.com/javaworld/jw-06-2005/jw-0613-soa.html, 2006.

[28] BEA Inc. SOA-软件架构新超越[J]. dev2dev 专刊, 2004.

[29] 余号等. SOA 实践[M]. 北京: 电子工业出版社, 2009-1.

[30] Alonso G, Casati F. Web Services and service-oriented architectures[C]. Proceedings of the 21st International Conference on Data Engineering, Tokyo, Japan, 2005: 1147-1147.

[31] W3C. XML Schema Part2: Struetures[EB/OL]. http://www.w3c.org/TR/2001/ REC-xmlschema-2-20010502. 2001-5.

[32] Microsoft, IBM. Developmenter SOAP: Simple Object Access Protocol Specificationl[EB/OL]. http://www.w3.org/TR/SOAP, 2007.

[33] Web services description language(WSDL)1.1[EB/OL]. http://www.w3.org/TR/2001/NOTE-wsdl-20010315, 2007.

[34] UDDI Technical White Paper[EB/OL]. http://www.uddi.org/pubs/Iru-UDDI-Technical-White_Paper. Pdf, 2007.

[35] 门永奎. 基于 WebService 的软件分布式重用的研究与实现[J]. 微计算机信息, 2006, 22(9): 278-280.

[36] (美)Nadir Gulzar. 实用 J2EE 应用程序体系结构[M]. 清华大学出版社, 2003.

[37] 付更丽, 曹宝香. SOA-SSH 分层架构的设计与应用[J]. 计算机技术与发展, 2010-1, 20(1): 74-77.

[38] 陈启祥, 王凯. J2EE 中 SOA 架构的实现[J]. 湖北工业大学学报, 2005-12, 20(6): 25-27.

[39] XFire 入门[EB/OL]. http://www.ibm.com/developerworks/cn/java/j-lo-xfire/, 2007.

[40] 蔡昭权. 基于业务无关的权限管理的设计与实现[J]. 计算机工程. 2008-5. 34(9): 183-185.

[41] 朱细波. Web 信息系统中统一细粒度访问控制的研究[D]. 苏州大学硕士学位论文, 2008-4.

[42] 李福盛, 曹宝香. 基于 Web Services 的通用权限管理服务设计[J]. 沈阳师范大学学报, 2009-1, 27(1): 68-70.

[43] 张川波. Web 统一用户权限控制模型的研究与设计[D]. 上海交通大学硕士学位论文, 2007-12.

在校期间的研究成果及发表的学术论文

[1] 山东省信息专项“基于SOA的PLM系统构件库和开发平台建设”，（关于下达2007年信息产业专项的通知 鲁财建指2007（94）号）.

[2] 山东省日照市科技攻关项目“基于Web的CRM系统的研究与实现”，通过鉴定，鉴定证书编号：日科成鉴字【2010】第002号.

[3] 付更丽，曹宝香. SOA-SSH分层架构的设计与应用[J].计算机技术与发展. 2010-1，20(1):74-77.

[4] 付更丽, 曹宝香, 夏小娜. 继承和优先约束驱动的柔性授权机制研究[J]. 计算机工程. 2010年第24期.

[5] 付更丽, 曹宝香, 夏小娜. 属性驱动的多策略网格授权机制的研究与设计[J]. 通信技术, 2010年第9期.

[6] 付更丽.“基于Web的CRM系统研究与应用”学术报告. 获“曲阜师范大学2010年第四界研究生学术论坛”一等奖，2010年5月.

致　　谢

首先，我要衷心感谢我的导师曹宝香教受。在我攻读硕士期间，曹老师在学习、论文研究和生活等方面给予了我全方位的指导、关心和帮助，我为成为曹老师的学生而倍感自豪。曹老师在项目开发，论文选题、研究内容和研究方法，及论文写作等方面指导和鼓励，是我能够顺利完成论文研究的基础。另外，曹老师也教会了我为人处事、做人的道理，帮助我以后能更好的面对工作和生活。曹老师严谨的治学态度、渊博的学术知识和严于律己的工作态度都给我留下了深刻的印象，教导我在学业上和工作中不断求新务实，开拓进取，并必将使我终身受益。在此谨向曹老师表示深深的谢意!

衷心感谢夏小娜和李天盟等老师，感谢他们在论文研究、项目开发过程和生活中所给予的帮助。同时，感谢同在一个实验室的各位研友，在共同合作开发项目的过程中给予我无私帮助。在平时与他们的讨论中，我得到了许多有益的思想。

衷心感谢我的家人和朋友，感谢他们在生活和学习中给予我支持和鼓励，使我能够完成这份学业。

感谢抽出宝贵时间对本论文进行评审的专家和参加论文答辩的各位老师!