东华大学学位论文原创性声明

本人郑重声明：我恪守学术道德，崇尚严谨学风。所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已明确注明和引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品及成果的内容。论文为本人亲自撰写，我对所写的内容负责，并完全意识到本声明的法律结果由本人承担。

学位论文作者签名：郭杨杨

日期：2009年3月10日

东华大学学位论文版权使用授权书

学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅或借阅。本人授权东华大学可以将本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和汇编本学位论文。

保密 □，在 ___ 年解密后适用本版权书。

本学位论文属于

不保密 ☑。

学位论文作者签名：[signature]　　指导教师签名：[signature]

日期：2009年3月10日　　日期：09年3月10日

基于多 Agent 交互协作的工作流管理系统的研究与应用

摘 要

随着计算机互联网技术的飞速发展与管理信息系统在办公、生产等领域的高速普及，工作流技术逐渐被引入，工作流管理系统（Workflow Management System）应运而生并掀起了一股热潮。随着现代企业自动化程度的不断提高，传统工作流管理系统的缺点不断暴露，比如耦合程度高、灵活性低、智能性弱等，因此如何有效的解决以上问题是工作流管理系统进一步深入发展与应用的关键所在。

Agent 的概念出现于 20 世纪 70 年代，80 年代后期不断成长起来并成为计算机科学技术领域十分活跃的前沿研究方向之一。Agent 一般具有自主性、交互性、反应性和主动性的特征，并且各 Agent 之间是一种松散耦合的关系。因此，Agent 与工作流管理系统的结合必将为工作流管理系统的发展注入新的活力。

本论文首先讨论了工作流管理系统与 Agent 技术的基本概念及相关理论，并对工作流管理系统参考模型、基于 Agent 的系统、多 Agent 系统的合作模型进行了分析，其中重点分析了工作流管理系统当前存在的不足、未来的发展方向和多 Agent 系统的研究与开发。根据以上相关理论，本文提出了一个基于多 Agent 交互协作的工作流管理系统的模型（MAWFMS），通过设计具有各种功能的 Agent，将相应的业务逻辑从系统中分离，有效降低了系统的耦合度，提升了系统的可扩展性和易维护性。文章阐述了 MAWFMS 模型的设计思想，结合工作流管理联盟给出的工作流管理系统体系结构，提出了 MAWFMS 模型的整体体系结构并进行详细设计以及分析了 MAWFMS 模型的性能。

本文在模型中设计了任务分配的智能决策，将管理学中的层次分析法思想引入模型的任务分配过程中，拓展传统的工作流任务分配模式——“推”模式和“拉”模式，实现了一个“智能选推”模式，通过引入具有智能的 Agent，进行决策判断，选择最合适的用户完成相应的任务，真正加强工作流管理系统，这也是本文的创新点之一。本文创新点之二，在 Agent 通信模块的设计中，通过使用 XML 技术包装 KQML 语言，将 XML 嵌入 KQML 中的内容层，用 XML 来表达内容

的语义信息，丰富 KQML 语义表达，方便地实现表达复杂语义，从而简化了 Agent 之间通信的复杂度。本文创新点之三，将面向切面（AOP）的设计思想引入到安全认证模块的设计中，通过将系统的权限验证模块封装成单独的模块，与核心业务模块解耦合，在调用核心业务模块时借助动态 AOP 框架将权限验证模块在运行时动态地织入，从而实现一种通用的、可维护的、易扩展的权限认证模块。

最后，本文通过一个基于 MAWFMS 模型的应用实例——员工绩效考核系统的设计与实现，验证了 MAWFMS 模型的可用性。

关键词：工作流；工作流管理系统；Agent；多 Agent；智能决策

RESEARCH AND APPLICATION OF WORKFLOW MANAGEMENT SYSTEM BASED ON MUTI-AGENT INTERACTIVE COOPERATION

Abstract

With the rapid development of computer internet technology and the boost popularization of using the management information systems for fields of business and production and etc., workflow technology is introduced gradually, workflow management system is produced with application and the hot tidal wave is brought. Some shortcomings of the traditional workflow management system are exposed with the automation degree of modern enterprises has been constantly enhanced such as the high degree of coupling, low flexibility and weak artificial intelligence. Hence, how to solve the problem mentioned above in effect is the key problem of workflow management system for further development and application.

The concept of "Agent" starts from 1970s and growing up in late 1980s which becomes a active research direction in the field of computer science. Generally, "Agents" have autonomous, interactive, reactive and proactive features. There has a loosely coupled relationship between different Agents. Therefore, the combination use of "Agents" and "workflow management systems" will definitely inject new vitality for the development of workflow management systems.

This paper discussed the basic concepts of workflow management systems and Agent technology. It also analyzed the reference model of workflow management system, the Agent-based systems, and the multi-Agent co-operation system model. It focused on an analysis of current workflow management system's deficiencies、the future directions and the Research & development of multi-Agent System. Based on the above theory, this paper puts forward a model based on cooperation of multi-Agent workflow management system (MAWFMS), the business logic will be separated from the system by designing agents with different features, and leading to improve scalability and easy maintenance of the system. The paper also described

designing ideas of MAWFMS model and combined with the workflow management system structure, which given by the League of Workflow Management System, the design and performance of a MAWFMS Model with the overall architecture have been detailed analyzed in this paper as well.

In this design, the new MAWFMS model adds a new function of intelligent task allocation decision-making. Leading AHP into the distribution of tasks, expanding the traditional allocation of workflow task model—"push" model and "pull" mode and achieve an "Intelligent Decision" mode. By introducing the Intelligent Agent, making intelligent decision, the system will allow right persons to do right tasks, which strengthen the performance of the management system, and this is the first Creative point. The second point is, in the design of Agent communication module, by using of XML technology to package KQML language, XML will be embedded in the content of KQML layer, using XML to express the semantic content of the information, enrich the semantic of KQML, achieve the complexity of the semantic easily expression simplify the Agent Communication's complexity. The Third creative point is, leading an aspect-oriented programming design concept into the system authorization module, by uncoupling the system authorization module from core business modules to an encapsulation, and dynamically embedding the authorization module under AOP framework when core business modules are called, we realize a universal、maintainable and extendable authorization module.

At last, a real application of MAWFMS—the staff performance appraisal system, has been referenced in this paper to verify the model's availability.

Key Word: Workflow; Workflow Management; Agent; Multi-Agent; Intelligent Decision

目 录

第一章 绪论

随着计算机互联网技术的飞速发展与管理信息系统在办公、生产等领域的高速普及，工作流技术逐渐被引入，工作流管理系统（Workflow Management System）应运而生并掀起了一股热潮。随着现代企业自动化程度的不断提高，传统工作流管理系统的缺点不断暴露，比如耦合程度高、灵活性低、智能性弱等，因此如何有效解决以上问题是工作流管理系统进一步发展与应用的关键所在。

1.1 研究背景

目前，在全球范围内，对工作流的技术研究以及相关产品的开发进入了更为繁荣的阶段，更多更新的技术被集成进来，文件管理系统、数据库、电子邮件、移动式计算、Internet 服务器等都已被纳入到工作流管理系统之中[1]。工作流产品的市场每年以两位数的速度迅猛增长，市场上的工作流产品发展迅速，工作流产品在相互竞争中得到不断改善[2]。

但是随着现代企业自动化程度的不断提高，传统的工作流管理系统的缺点不断暴露，比如缺乏支持松散耦合的体系结构、缺乏自主能力、灵活性低、智能性弱等。因此，如何有效的解决以上问题是工作流管理系统进一步深入发展与应用的关键所在。在这种情况下，引入一个新技术来继续发展和完善工作流管理系统无疑是一个行之有效的方法。

Agent 的概念出现于 20 世纪 70 年代，80 年代后期不断成长起来并成为计算机科学技术领域、信息工程领域和网络与通信领域十分活跃的前沿研究方向之一。Agent 的原意是“代理”，即一个人代表另一个人或组织去完成某些事情。在计算机领域，Agent 可认为是被授权的“个人软件助理”，是一种分布式系统或协作系统中能够持续自主地发挥作用的计算实体[3]。Agent 一般具有自主性、交互性、反应性和主动性的特征，并且各 Agent 之间是一种松散耦合的关系。因此，Agent 能够解决目前工作流管理系统中存在的普遍问题，它们的结合必将为工作流管理系统的发展注入新的活力。

本论文的研究对象是一个员工绩效考核系统的任务分派子系统。绩效考核系

统是一个基于 B/S 架构的软件系统，并包含了一系列辅助工具以支撑应用系统的运行。它集成了各异构数据库的访问并提供了统一的数据访问接口；它集成了各种软件架构和技术体系的应用软件并向上提供通用的应用程序访问接口；它提供系统集成工具与工作流程管理工具。本课题的重点是基于多 Agent 交互协作的工作流管理系统模型的设计。

1.2 国内外研究现状分析

1.2.1 工作流技术

工作流技术是当今计算机领域的研究热点之一，并具有非常广阔的发展前景。在国内各级政府主推信息化建设过程中，基于工作流技术的电子政务平台的建设如雨后春笋般不断涌现，工作流技术呈现出一片欣欣向荣的景象。但事实上，工作流技术还处于技术发展曲线上的初级阶段。

在国外工作流技术研究中，比较著名的有 IBM 公司 Almaden 研究中心的 Exotica，佐治亚大学计算机系的 Meteor、WIDE 以及 Mentor 等研究项目。Exotica 是基于持久消息队列的分布式工作流管理系统，它的工作流管理系统由许多具有自治能力的节点组成[4]。Meteor 是具有自适应能力的工作流管理系统，它的研究目的是开发出一个能够支持大规模复杂的工作流应用的系统，并保证这些应用能在企业间异构的环境中正常运行[5]。WIDE 是基于分布式主动数据库技术的工作流管理系统，它是由西班牙、意大利和荷兰等国的五个合作单位协同开发的工作流管理系统，主要目的是利用分布式数据库和主动数据库技术来实现工作流管理，并提供先进的、面向应用的软件产品[6]。Mentor 是基于状态与活动图的工作流管理系统，它的研究是为工作流模型的定义、执行和控制提供一个中间件平台，可将所建的模型自动转化为状态活动图[7]。

在国内工作流技术研究中，比较著名的公司有杭州信雅达，北京世纪金政，四川金科成，北京有生博大，西安协同数码等。杭州信雅达工作流管理系统的最大的特色是采用基于域的联邦系统架构，对分布式管理、运行支持较好。而且也是目前国内为数不多的可以支持“仿真”的工作流系统。北京世纪金政主要定位于 OA 和电子政务平台。四川金科成从早期的工作流产品转移向“业务基础软件平台”，但是整个产品平台目前还只能算是一个 OA 开发平台。北京有生博大的整个工作流产品基本上为“OA 审批流程”量身定做。其表单处里、权限控制以及审

批历程的处理很有特色。西安协同数码的工作流管理系统基本上非常严格遵循了 wfmc 的规范，完全实现了 interface1、interface2、interface3、interface5[8]。

1.2.2 Agent 的研究与应用

起源于 20 世纪 70 年代，于 80 年代后期逐步成长起来的 Agent，不仅成为 AI 和计算机领域最活跃的研究内容之一，而且引起了科技界、教育界甚至娱乐界的广泛关注，其应用也越来越广泛。

在国外，目前已经有许多机构研究 Agent 技术，例如成立于 1996 年的 FIPA 组织致力于制定基于 Agent 系统的规范[9]，日本东芝 TOSHIBA 公司研究的 bee-gent，美国 comet way 公司的 Comet Way Agent，IBM 公司的 Aglets 平台。目前国内一些高等院校和科研院所也在开展有关研究，例如国防科大在开展基于 Agent 的分布集成环境、多 Agent 合作模型的相关理论和方法的研究；南京大学在开展面向 Agent 的软件工程及安全性研究；中科院计算所在开展基于 Agent 的信息过程建模方法及面向 Agent 的软件开发方法研究；中国科技大学在开展基于多 Agent 的智能仿真系统研究等。

近几年来，将 Agent 技术应用于工作流管理系统展现很好的发展前景，很多领域的工作流管理系统中均引入了 Agent 技术。这些 Agent 按照功能可以分为：用户 Agent、任务 Agent、资源 Agent、管理 Agent。

在国外，Shen and Norrie 将 Agent 技术运用到制造企业的工作流程管理，提出了一种以协调 Agent 为中心的混合分布式结构，将核心企业以及它的合作伙伴、供应商、客户通过它们各自的 Agent 集成在一起。在这个多 Agent 系统中，Agent 可以用来代表资源、工件、对已有的软件系统进行封装，充当系统/子系统之间的协调者[10]。Jeff，YC.P 等人提出采用多智能 Agent 技术建立企业工作流管理的计算机基础结构，将复杂的企业活动划分成多组元任务，每一个组元任务由一个智能 Agent 执行。将人的行为看作一类智能 Agent，采用多媒体界面，通过大量智能 Agent 的交互，实现企业中地域分散的各个生产部门知识的共享与协调，利用人—机、机—机的交互协调进行复杂问题的求解，完成企业各项功能的集成[11]。

在国内，许青松等人在分析了多联盟环境下企业对管理系统需求的基础上，将多 Agent 技术引进到虚拟企业中，提出了支持动态联盟的多 Agent 企业管理模

型—AMIS。构成动态联盟的各个单元通过多 Agent 系统相互连接，实现企业参与多个联盟的构想[12]。韦文斌等人采用多 Agent 技术解决车间调度问题，根据车间中的人物类型提出了相应的各种任务集合的定义，并在此基础上给出了任务 Agent 和资源 Agent 交互协调的流程以及它们各自的运作流程[13]。

综上，国内外基于 Agent 的工作流技术仍然处于研究探索阶段，其中既有理论方面的研究，也有系统设计、应用方面的研究，虽然不断涌现新的成果，但并没有一个很成熟的产品和系统。

1.3 论文的研究内容及组织结构

1.3.1 研究内容

本文从阐述工作流技术的相关理论开始，对工作流技术进行深入分析和研究，总结出现有工作流技术的不足并指明今后的发展方向，阐述 Agent 技术的相关理论，总结出 Agent 的特性以及多 Agent 系统的合作模型；根据以上理论基础，提出一个基于多 Agent 交互协作的工作流管理系统的模型，并以一个员工考勤系统的任务分派子系统为例，从功能需求、设计目标、系统结构和功能模块等几个方面进行了详细的分析与设计，并特别讨论了系统中各个 Agent 的功能及他们之间的协作关系；通过实现一个完成员工任务分派的基于多 Agent 交互协作的工作流管理系统，验证以上理论和设计的正确性与可行性；论文最后对该模型系统的优缺点和今后的工作进行了分析与总结。

在研究过程中，本论文的关键内容是将 Agent 技术应用于工作流管理系统中，解决下列问题：

1 任务处理质量低效率差：基于 Agent 的智能计算的特点，使原本将数据集中计算处理转为数据分布于各 Agent 中计算处理再汇集，分担了系统的压力，加速了系统的运转，从而使任务处理又好又快。

2 系统中资源冲突问题：工作流管理系统在企业业务系统中的资源是独立的，因此当两个工作流实例同时需要某一资源时即发生竞争，利用 Agent 交互协商调用相关资源可有效解决此类问题。

3 任务分配方式僵化问题：在工作流执行过程中，工作流引擎被动的解释执行过程定义，在任务分配过程中，目前常用的两种方式是“拉”模式和“推”模

式，方式单调缺乏一定的灵活性，不能满足一些复杂的需求。

4 用户在工作流管理系统中的被动性：在协同工作中，强调用户参与协作过程的主动性。不同企业的员工日程安排自由度不同，Agent 可以协助用户自主的安排日程，提高用户在系统中的主动性。

5 系统高度耦合的问题：各个 Agent 将其独自承担一些业务从系统中分离出来，实现业务分离，从而降低了系统的耦合性。

1.3.2 组织结构

全文共分为六章，按如下结构组织：

第一章：绪论。本章讨论了本文的研究背景，国内外研究现状并介绍了文章的主要研究内容及组织结构。

第二章：工作流技术。本章深入研究分析工作流技术，包括工作流的定义、工作流参考模型、工作流管理系统等。

第三章：Agent 与多 Agent 技术。本章介绍 Agent 的特性、基于 Agent 的系统、多 Agent 系统的合作模型等。

第四章：MAWFMS 模型的设计：本章提出一个基于多 Agent 交互协作的工作流管理系统的模型并进行了相关设计。

第五章：基于 MAWFMS 模型的绩效考核系统的设计：在上一章提出 MAWFMS 模型设计的基础上，本章详细介绍了一个基于 MAWFMS 模型的应用实例——员工绩效考核系统的设计。

第六章：基于 MAWFMS 模型的绩效考核系统的实现：本章主要介绍员工绩效考核系统的实现，通过系统的运行验证 MAWFMS 模型的可用性。

第七章：总结与展望：对全文进行总结并就下一步的工作进行讨论。

1.4 本章小结

本章从本论文的课题背景开篇，阐述了当今工作流管理系统发展的瓶颈问题以及解决问题的思路——将 Agent 技术引入到工作流管理系统之中，介绍了本论文的课题来源及简介。然后通过对国内外研究现状的分析，给出了本论文的主要研究内容及组织架构。

第二章 工作流管理技术

要想解决目前工作流管理系统中存在的问题，首先必须深入了解工作流管理系统的相关概念，本章将从工作流问题的起源开始，介绍工作流的基本概念、工作流参考模型、工作流建模，对工作流管理系统领域进行了重点分析，说明什么是工作流管理系统，主要功能是什么，体系结构是什么样，如何实施以及实施的意义。

2.1 工作流

工作流是由 work 和 flow 组成的单词 workflow 翻译而来。单词 work 表示工作或者任务，单词 flow 表示流动或者流程。Work 反映了一个要完成的工作，是一个既定的目标。Flow 反映了一种变化以及变化的过程，它的概念相对比较抽象，但是当它与具体的过程相结合的时候，也就具有了实际的含义，比如气流、水流等。在企业经营管理与生产组织中，flow 也有重要意义，比如表示资金流动的资金流，表示信息传递与处理的信息流等。综上，我们可以将工作流描述为用活动及活动之间变化的过程表示的业务流程。

工作流的概念起源于生产组织和办公自动化领域，是根据日常工作中具有一定规则的活动提出的一个概念。它的目的是通过将工作分解成定义良好的任务、角色，按照一定的规则和过程来执行这些任务并对它们进行监控，从而提高生产效率、降低生产成本、提高生产经营水平和企业竞争力。在企业的实际应用中，虽然工作流的概念相对于物料流、资金流、信息流等概念要相对抽象，但是工作流从更高的层次上提供了实现物料流、资金流、信息流及其涉及的相关过程与应用的集成机制，从而使得企业能够实现业务过程集成、业务过程自动化与业务过程管理。在工作流的概念下实现业务过程集成与业务过程自动化的集成机制是通过定义不同任务之间相互关系的工作流模型来实现的[1]。

1993 年工作流管理联盟（Workflow Management Coalition，WfMC）的成立标志着工作流进行开始进入一个相对成熟的阶段。通过在工作流管理系统的相关术语、体系结构及应用编程接口（WAPI）等方面制定一系列标准，工作流管理

联盟希望实现不同工作流产品之间的互操作。工作流管理联盟给出的工作流定义是[22]：工作流是一类能够完全或者部分自动执行的经营过程，它根据一系列过程规则、文档、信息或任务能够在不同的执行者之间进行传递与执行。图 2.1 给出了 WfMC 给出的工作流参考模型[25]，包括涉及到的几种数据、系统中的各个组成部分以及五类接口。

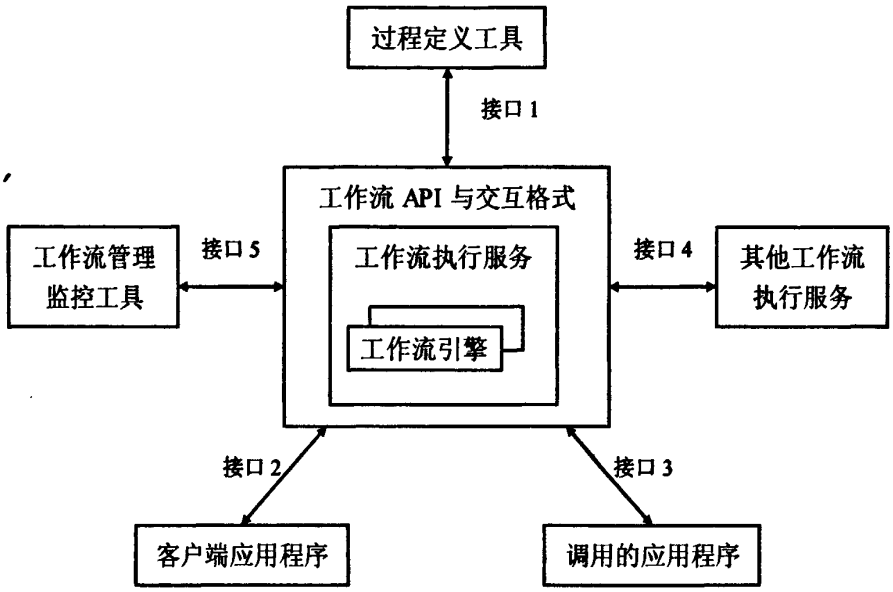

图 2.1 工作流参考模型

工作流参考模型中涉及到的数据有：

1 工作流控制数据（Workflow Control Data）：工作流执行服务/工作流机通过内部的工作流控制数据来辨别每个过程或活动实例的状态。这些数据由工作流执行服务/工作流机进行控制。用户、应用程序或其它工作流执行服务/工作流机不能对其进行直接读写操作，它们可以通过向工作流执行服务/工作流机发送消息请求来获得工作流控制数据的内容。

2 工作流相关数据（Workflow Relevant Data）：工作流管理系统通过工作流相关数据来确定过程实例状态转换的条件，并选择下一个将执行的活动。这些数据可以被工作流应用程序访问并修改。因此，工作流管理软件需要在活动实例之间传递工作流相关数据。

3 工作流应用数据（Workflow Application Data）：由应用程序操作的数据，它们是针对应用程序的，是完成企业具体的业务功能所需要的数据，工作流管理系

统无需对他们进行访问。

WfMC 在工作流参考模型中定义了五类接口，以下将对各个接口的功能进行简要的介绍。

接口 1：工作流服务和工作流建模工具间接口，包括工作流模型的解释和读写操作，在功能上是一个过程定义输入输出接口。

接口 2：工作流服务和客户应用之间的接口，它约定所有客户方应用与工作流服务之间的功能操作方式。在功能上是一个客户端函数接口。

接口 3：工作流机和直接调用的应用程序之间的直接接口。在功能上是一个激活应用程序接口。

接口 4：工作流管理系统之间的互操作接口。在功能上是一个工作流执行服务之间的互操作接口。

接口 5：工作流服务和工作流管理工具之间的接口。在功能上是一个系统管理与监控接口。

由于工作流是一个比较抽象的概念，不同的角度有不同的具体含义。因此，不同的研究机构和工作流产品开发商给出的工作流定义也不尽相同。Georgakopoulos 给出的工作流定义是：工作流是将一组任务组织起来完成某个经营过程。IBM Almaden 研究中心给出的工作流定义是：工作流是经营过程的一种计算机化的表示模型，定义了完成整个过程所需要的各种参数。这些参数包括对过程中每一个步骤的定义、步骤间的执行顺序、条件以及数据流的建立、每一步骤由谁负责以及每个活动所需要的应用程序。

此外，在工作流技术的大框架下，还有一些其他相关概念，它们在工作流管理系统中是相互联系的，图 2.2 给出了它们之间的联系[24]：

1 工作流管理系统（Workflow Management System）：运行在一个或多个工作流引擎上的计算机软件系统，它定义、创建工作流，管理工作流的执行

2 业务过程（Business Process）：为实现企业特定业务目标的一个过程，在部分或者全部组织机构和人员的参与下，利用企业资源，按照预定规则，在参与者之间进行信息传递与处理，从而实现预定目标。

3 过程定义（Process Definition）：将一个业务过程或流程规格化描述为形式化的、计算机可处理的定义。

4 过程实例（Process Instance）：过程定义的运行实例。

5 活动（Activity）：对应于企业经营过程中的任务，主要反映完成企业经营过程需要执行的功能操作，。

6 活动实例（Activity Instance）：过程中活动的执行。

7 变迁（Transition）：活动之间通过变迁相联系，基本属性有：前继活动、后续活动、变迁条件。从一个活动到另一个活动之间的变迁可以是有条件的，也可以是无条件的。

8 角色（Role）：主要描述企业经营过程中参与操作的人员和组织机构。

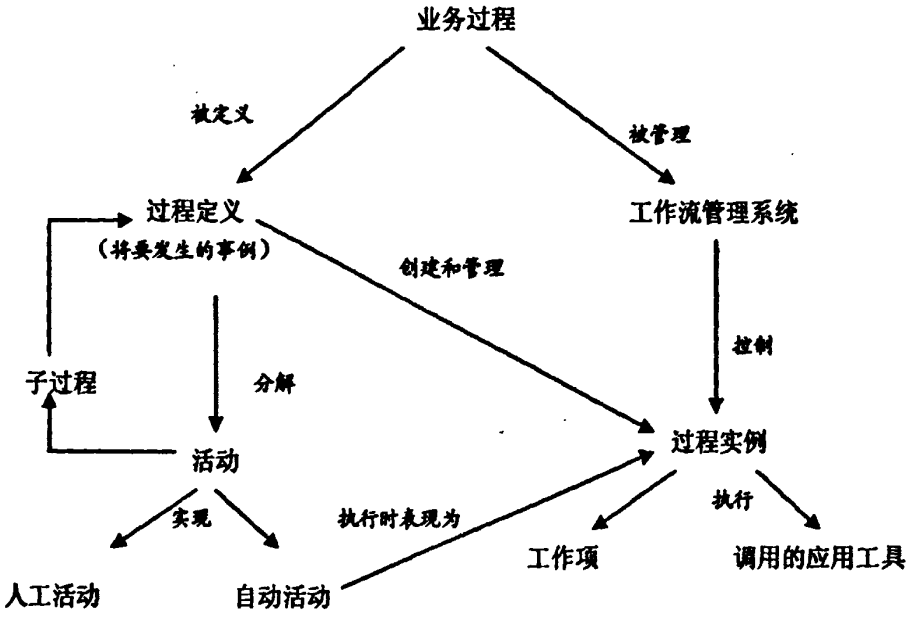

图 2.2 工作流相关概念之间的联系

在工作流建模上，工作流管理联盟主要提出两方面的内容[26]：

1 定义了一个元模型，所谓元模型即描述模型的模型。这里的元模型是用来描述工作流模型内在联系的模型。它用来描述工作流模型内部包含的各个对象、对象之间的关系以及对象的属性。

2 定义了一个可以在工作流管理系统之间、管理系统与建模工具之间交互的过程模型定义的 API。

目前，主要的工作流建模方法有：基于活动网络的工作流模型、基于形式化表示的建模方法、基于对话模型的建模方法、基于状态与活动图的建模方法、基于事务模型的建模方法、图形化建模方法等。

2.2 工作流管理系统

2.2.1 工作流管理系统及其体系结构

WfMC 关于工作流管理系统的定义是：工作流管理系统是一个软件系统，它完成工作流的定义和管理，并按照在计算机中预先定义好的工作流逻辑推进工作流实例的执行。

在实际应用中，工作流管理系统通常是指运行在一个或多个称之为工作流机的软件上的用于定义、实现和管理工作流运行的一套软件系统，它和工作流执行者（人或者应用程序）交互，推进工作流实例的执行，并监控工作流的运行状态。工作流不是实质意义上的企业业务系统，在很大程度上来说，它为企业的业务系统运行提供了一个软件支撑环境，非常类似于单个计算机上的操作系统，只不过工作流管理系统支撑的范围大、集成面广、环境复杂，所以工作流管理系统也被称为业务操作系统（Business Operating System，BOS）。在工作流管理系统的支撑下，通过集成具体的业务应用软件和操作人员的界面操作，才能够良好地完成对企业经营过程运行的支持。所以，工作流管理系统在一个企业或部门的经营过程中的应用过程是一个业务应用软件系统的集成与实施过程。

工作流管理系统可以用来定义与执行不同覆盖范围、不同时间跨度的经营过程，这完全取决于实际应用的要求。按照经营过程以及组成活动的复杂程度的不同，工作流管理系统可以采取许多实施方式。在不同的实施方式中，工作流管理系统所使用的信息技术、通信技术和支撑系统结构会有很大的差别。

虽然不同的工作流管理系统具有不同的应用范围和不同的实施方式，但究其本质，各种工作流管理系统具有许多相同的特性。从比较高的层次上抽象地观察工作流管理系统，可以发现所有的工作流管理系统都提供了 3 种功能。

1 建立阶段功能：主要考虑工作流过程和相关活动的定义和建模功能。

2 运行阶段的控制功能：在一定的运行环境下，执行工作流过程，并完成每个过程中活动的排序和调度功能。

3 运行阶段的人机交互功能：实现各种活动执行过程中用户与 IT 应用工具之间的交互。

图 2.3 形象的展示了工作流管理系统的三个主要特性以及他们之间的关系。

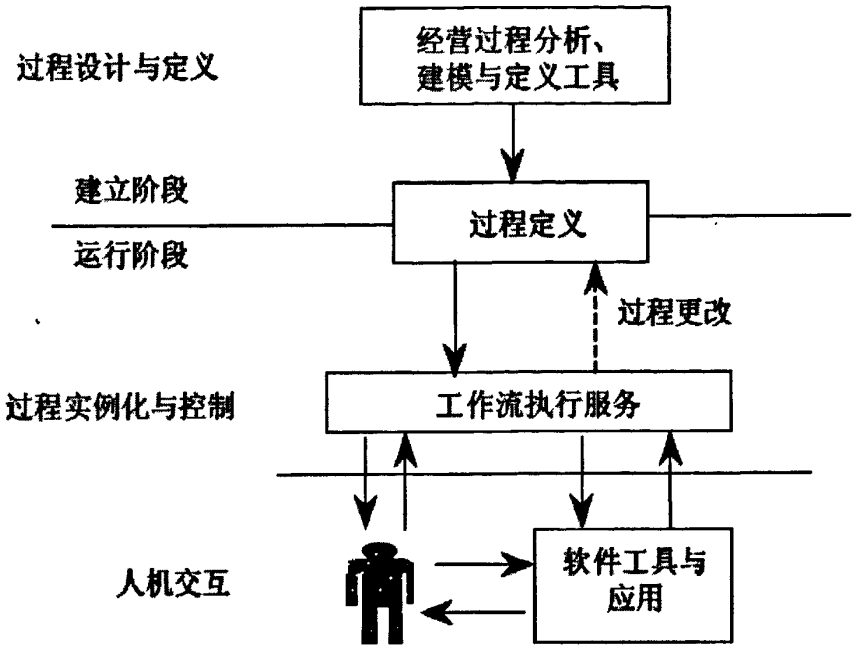

图 2.3 工作流管理系统的特性

根据上文所述，工作流管理系统主要有三个主要功能，即过程建模、工作流管理中的运行控制和工作流管理中的人机交互。下面我们将对这三个主要功能一一进行阐述。

过程建模是经营过程中分析与经营过程重组的重要基础。过程建模所处的阶段是工作流管理系统的建立阶段，这个阶段主要完成经营过程的计算机化定义也即过程建模，它利用一种或者多种建模技术与工具，完成企业经营过程从客观实际到可被计算机处理的形式化定义的转化。在工作流管理联盟对工作流管理系统的定义中，将过程建模的结果统称为过程定义。过程建模主要解决的问题是如何根据过程目标和系统约束条件，将系统内的活动组织为适当的业务过程。

工作流管理中的运行控制是指在完成了上述的过程建模后，所生成的工作流模型将由工作流管理中的运行控制接管，进行实例化创建并控制其执行过程。工作流管理中的运行控制对使用工作流模型描述的过程进行初始化、调度和监控过程中每个活动的执行，在需要人工介入的场合完成计算机应用软件与操作人员的交互。这样，工作流管理中的运行控制实现了在模型中定义的业务过程与现实世界中实际过程之间的连接。这个连接通过工作流执行服务与应用软件、操作人员的交互来完成。实现这个连接的核心功能是工作流管理软件，工作流管理软件又

称为工作流引擎。

工作流管理中的人机交互是指在工作流的运行阶段，通过主体参与者与计算机的交互推进过程实例的进行。因为在工作流管理系统的运行过程中，人或者应用程序是完成整个业务过程的主体参与者。工作流定义工具、工作流执行服务和任务表管理器都是为了完成业务过程和支持人员工作提供的运行环境和工具。具体来说，在整个工作流的执行过程中，不同的操作人员需要完成的工作大约有模型定义、人机交互、系统运行监控三种。

由于工作流管理技术和产品缺乏统一的标准，不同工作流产品在术语的定义和使用、系统结构的设计到与应用程序之间的接口规范上都存在较大的差异，所以它们之间互操作能力差，集成相当困难。因此，一个称为“工作流管理联盟”（简称 WfMC）的国际组织成立了，它提出了有关工作流管理系统的一些规范和标准，主要目的就是为了实现工作流技术的标准化和开放性，从而实现工作流管理系统与产品之间的互操作，提高工作流管理系统与产品的集成能力。图 2.4 为工作流管理联盟提出的工作流管理系统的体系结构图。

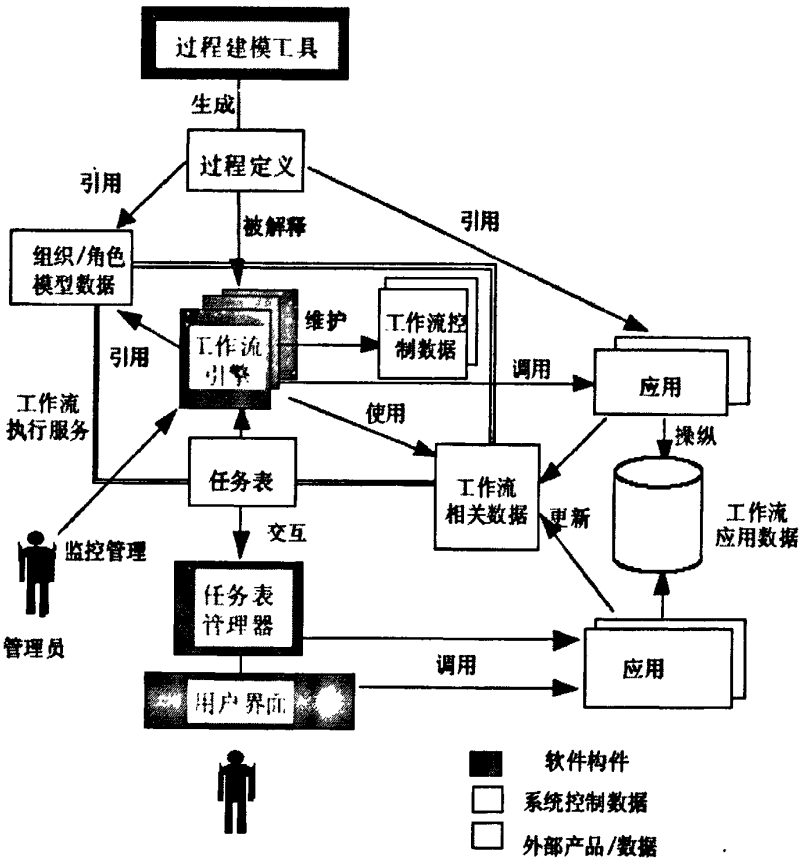

图 2.4 工作流管理系统的体系结构图

这个工作流管理系统的体系结构图给出了抽象的工作流管理系统的功能组成部件和接口，它能够满足工作流管理系统应该具有的主要功能特征，可为实现工作流管理系统之间的互操作提供公共的基础。当然，组成工作流管理系统的每个功能部件可以在不同的软硬件平台上采用不同的实现技术来实现，同样接口也可以在不同的软硬件平台上采用不同的设计技术和编程语言来实现。一般情况下，工作流产品供应商基于商业机密不会将这些部件之间的所有接口完全对外公开，但是，如果为了实现不同工作流管理系统之间的集成，它们会按照互操作和协作的具体要求，在一定层次上开放相应的接口。

此外，这个工作流管理系统的体系结构图还展示了工作流管理系统的三个组成构件：

1 软件构件：完成工作流管理系统中各个不同组成部分功能的实现。

2 系统控制数据：工作流管理系统中的一个或者多个软件构件所需要使用的数据。

3 应用与应用数据：对于工作流管理系统来说，它们不是工作流管理系统的组成部分，而是属于外部参与的系统和数据，它们被工作流管理系统调用来完成相应的工作流管理的功能。

2.2.2 工作流管理系统的实施

工作流管理系统不同于普通的企业信息管理系统如 ERP、CRM。普通的企业信息管理系统是企业的事务处理系统，它的主要目标是完成企业业务操作的功能，提高企业事务办事效率和水平，以及提高办事准确性。从企业整体的业务流程和企业经营目标上看，普通的企业信息系统一般局限于解决某个或某些特定领域的问题，并且它一般局限于解决企业内部的具体操作问题，面向企业内部而不是面向市场、面向客户。而工作流管理系统从一开始就着眼于面向市场、面向客户，它的根本目标是在整个企业的业务过程中提高企业的业务完成效率、增强企业的业务处理水平、强化企业的市场意识。

由于工作流管理系统与普通的企业信息管理系统存在一定的差别，前者注重完成企业业务而后者主要关注的是企业的事务。所以，工作流管理系统的实施虽然在软件工程领域的方法一致，但具体的开发方法却不同于一般的企业信息管理

系统。工作流管理系统的实施是一个不断循环、不断改进的过程，它首先要解决的问题是在战略层次上对企业的业务目标进行分析，确定企业的战略目标和组织要求。

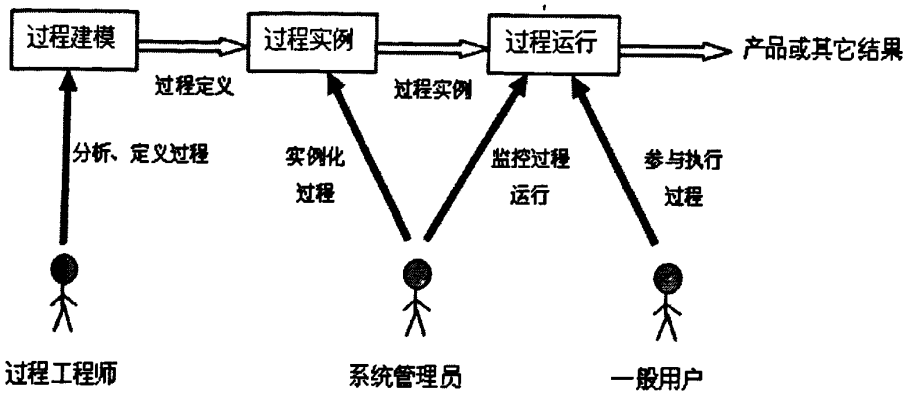

图 2.5 工作流管理系统实施的三个阶段

如上图 2.5 所示，在工作流管理系统的实际实施阶段，一般可以分为以下三个阶段：

1 模型建立阶段：利用工作流建模工具完成企业经营过程的建立，将企业的实际经营过程转变为可被计算机处理的工作流模型。

2 模型的实例化阶段：为系统中的每个过程设定运行所需要的参数，并为她们分配执行所需要的资源，包括设备、参与者、其他应用程序等。

3 模型执行阶段：完成经营过程的执行，主要是完成人机交互和应用的执行，并对整个过程和活动的执行情况进行监控与跟踪。

由于信息技术的发展和日趋激烈的商业竞争，人们不再满足于独立、零散的办公自动化和计算机应用，而是需要综合的、集成化的解决方案。作为一种对常规性事务进行管理、集成的技术，工作流管理系统的出现是必然的，它将使企业传统的按照功能来配置人员的组织结构变成按照企业实现的主要业务流程来配置组织结构，这样大大缩短主要业务过程的处理时间，提高了对市场的响应速度。总体来讲，工作流管理系统的应用可以为企业带来以下收益：

1 改进和优化业务流程，提高业务工作效率；

2 提高企业管理的规范化程度；

3 实现更好的业务过程控制，提高顾客服务质量；

4 提高业务流程的柔性，避免不必要的重复工作；

5 加快业务过程的处理，减少业务过程中不必要的中间状态；

6 在工作人员之间更好的均衡负荷；

7 通过在工作流模型中加入可预计的故障处理策略来强化系统的整体安全性；

8 通过对工作流实例的分析，找出其中存在的不足，进而不断优化整个业务处理流程；

9 使业务工作得以简化，从而提高工作人员的业务能力。

此外，采用工作流管理系统对于提高企业信息的现代化程度具有显著的作用，它可以在最大程度上集成企业的现有信息资源，实现资源的充分利用。由于工作流管理系统具有较好的柔性和开放性，因此，可以保证企业的信息系统可以满足不断变化的市场环境所要求进行的扩展变化。

2.3 工作流管理技术发展方向的研究分析

2.3.1 当前工作流产品存在的不足

尽管工作流技术已经经过十几年的研究发展，工作流产品供应商也在不断努力，工作流技术已经从最初的试验阶段逐步发展起来，并取得了一系列优秀的成果。但是，从工作流产品的实际应用情况来看，仍然存在诸多问题，远未达到人们期望的水平。缺乏统一的标准以及软件支撑技术的水平，导致各个工作流产品供应商在工作流管理系统的功能上呈现出非常大的差别，这是目前工作流管理系统存在问题的主要原因之一。

此外，工作流技术自身的不成熟也是目前工作流产品不达标的主要原因。在工作流模型描述方面，缺乏一种能够支持过程定义、过程演进以及过程分析的形式化数学模型。在工作流执行方面，缺乏一个标准化的集成框架来支持对企业常用的分布式应用的集成。在工作流仿真方面，缺乏仿真方法与仿真工具，该领域的研究成果几乎空白。

以下，我们将结合目前企业的实际应用情况，说明工作流管理系统存在的主要不足，它们分别是：

1 在运行阶段缺乏底层通信基础结构的支持。

2 工作流标准化程度低。

3 工作流的生成需要人机交互人工生成。

4 工作流管理系统对异常情况处理能力不足。

5 工作流管理系统的动态修改能力差。

6 工作流管理系统中各部门业务间优化调度和协调能力差。

7 工作流管理系统对运行中出现的并发访问缺乏正确和可靠的支持。

8 工作流管理系统在性能上无法满足当今企业的密集任务处理。

9 工作流管理系统之间几乎彼此不兼容。

2.3.2 未来工作流管理技术的发展方向

在工作流技术日益受到重视的今天，对工作流技术的研究也在不断深入。针对前面所述的工作流管理系统中存在的问题，今后一段时间内，工作流技术的研究主要有两个方面。一是理论问题，为工作流技术的发展解决理论上存在的问题，探讨工作流模型和语义的形式化表示方法等。二是实现问题，从工作流技术实现的角度探讨利用先进的技术提高工作流管理系统的性能和可靠性。

以下，我们将结合目前企业的实际应用情况，从技术和应用的角度出发，说明工作流管理技术以及工作流管理产品在今后的发展方向。

1 基于 Web 的工作流：随着在全球范围内 Internet 技术的不断发展并迅速普及，许多工作流产品供应商把工作流管理产品扩展到互联网上，各个产品对 Web 的支持程度不尽相同，但总体来讲，它们都支持用户在 Web 浏览器中控制工作流实例的运行并管理任务项列表。因此，将工作流通过 Web 扩展到多个企业，工作流/Web 服务器的协同工作也将逐步发展起来。在 B/S 架构上，基于 Web 的工作流将会在今后相当长的时间内备受关注。

2 分布式工作流：经过近年来的长足发展，工作流管理系统的结构已经从原来只能支持单一的工作组环境发展到现在可以支持企业级的工作流环境。一个工作流实例可以通过局域网、城域网甚至广域网分布在全球任何地方的服务器或者客户端上，这样服务器端的故障所造成的影响将被降低到最小程度，也提高了工作流管理系统的可扩展性、实用性和系统的管理能力，目前在这一领域的研究工作还在进一步深入之中。

3 基于复杂应用的工作流：随着工作流技术的不断发展，人们对工作流的认

识也在不断深入，企业对工作流技术的需求也在日益加大。但目前工作流产品在支持复杂的企业应用、集成企业现有的软件系统方面，存在着一定的不足。因此，支持复杂企业应用和优化系统集成能力在最近几年会成为一些研究机构或者大学的主攻对象。

4 基于 Agent 的工作流：Agent 技术与工作流技术一样，都是目前广受关注并获得长足发展的计算机领域的软件技术。在现有知识结构的基础上，充分发挥工作流技术在灵活性上面的优势和 Agent 在自主性、主动性、智能性、社会性等方面的特长，将二者有效结合在一起开发出来的工作流管理系统将有效的解决目前工作流管理系统中存在的普遍问题。该研究方向是当下国际工作流研究领域的热点，也是本文所要研究的内容。

2.4 本章小结

本章首先对工作流进行了概述，从工作流问题的起源开始，介绍了工作流的基本概念、工作流参考模型以及工作流建模，使读者对工作流技术有个初步了解。然后在工作流管理系统领域进行了重点分析，说明了什么是工作流管理系统，主要功能是什么，体系结构是什么样，如何实施以及实施的意义。最后，通过现有工作流技术不足的分析，阐述了今后工作流技术发展的方向。

第三章 Agent 与多 Agent 技术

Agent 一般具有自主性、交互性、反应性和主动性的特征，并且各 Agent 之间是一种松散耦合的关系。通过将 Agent 技术引入工作流管理系统中，是解决当前工作流管理系统发展瓶颈问题的行之有效的方法。本章将介绍 Agent 与多 Agent 技术的相关概念及开发方法。

3.1 Agent 概述

Agent 可以被看作是在线的伪人类(Peseudo-People)[31]。Agent 可以是一个人、一台机器或者一个软件。Agent 与对象既有相同之处，又有很大的不同。Agent 和对象一样具有标识、状态、行为和接口。但 Agent 和对象相比主要有以下差异：

（1）Agent 具有智能，通常拥有自己的知识库和推理机。而对象则一般不具备智能性。

（2）Agent 能够自主地决定是否对来自其它 Agent 的信息做出响应，而对象却必须按照外界的要求行动。也就是说 Agent 系统能封装行为，而对象只能封装状态，不能封装行为，对象的行为取决于外部的方法调用。

（3）Agent 之间的通信通常采用支持知识传递的通信语言。Agent 可以被看作是一类特殊的对象，即具有心智状态和智能的对象。Agent 本身可以通过对象技术构造，而且目前大多数 Agent 都采用了对象技术。从分布式人工智能的角度来看，一个多 Agent 系统是一个由问题求解实体构成的松散网络，多 Agent 系统用于由多个自治构件组成的系统，该系统一般具有以下特征：每个 Agent 都不具备解决问题的完整知识；没有全局系统控制；数据是分散的；计算是异步的。

那么 Agent 的定义到底是什么呢?在目前 Agent 的研究中，最为经典且最为广泛接受的是 1995 年 Wooldridge 和 Jennings 等人给出的有关 Agent 的两种定义[32]：

1.弱定义：Agent 具有这样的特性：自主能力、社交能力、反应能力和预动力。

2.强定义：Agent 具有在 Agent 弱定义的基础上，还具有人类才含有的一些特性，如知识、信念和义务等；同时，还强调 Agent 的一些情绪化的特性：适应

性、真实性、合理性等。

综上所述，Agent 是一个具有一定智能性的软件实体，能模拟人类行为和观念的软件实体。Agent 的基本特性包括自治性、反应性、能动性、学习性、通信性、移动性等[43]。

(1) 自治性

自治性是 agent 最基本的特性，指行动上的独立性。agent 一旦被初始化后，无需人直接干预而独立执行。agent 控制着自己的外部行为和内部状态，它可以被授权去做某种决定，完成一些重要的事情，例如代替用户操作、处理日常工作。这是 agent 区别于普通软件程序的基本属性。

(2) 反应性

反应性指 agent 能够清醒地对待所处的环境，感知和作用其所处的环境(环境可能是物理的世界、使用图形接口的客户、其它 agent 集合或者所有这些的组合)，能够对环境发生的改变及时做出响应。当 agent 遇到例外情况时，可以及时采取措施。

(3) 能动性

能动性指为达到目标，agent 不是等待着接受指令要求做什么，而是事先有计划，并做一些初始化。agent 能探测到适合客户目标的有利场景，通知客户这个场景出现的时机。也就是说，agent 不仅能对所处环境做出响应，也能主动地展现面向目标的行为。这种特性称为能动性。

(4) 学习性

基于历史活动的执行情况(经验)指导未来的行为，agent 这种对时间上的适应性称为学习性。例如，agent 学习客户的技能水平，从而提高支持客户的水平。又如在供应链中，agent 从大量顾客数据中发现顾客的需求和偏好，然后逐步调整生产以适应市场需要。

(5) 通信性

通信性指 agent 有能力敏捷地与其它 agent 交互。agent 之间的接口和联系不是固定不变的，而是随任务驱动者的改变而改变。为了协作完成一件复杂任务，一些 agent 可以形成 agent 群。agent 之间的接口、项和条件可以在运行中协商，这样就减少了 agent 之间的耦合性，意味着 agent 可以以最小的代价和较小的冲

突加入系统或从系统中移除。

(6) 移动性

移动性指 agent 有能力在一个网络上随时、随地、自主地从一台主机迁移到另一台。正在运行中的 agent 状态可以被存储且传送到新主机上，在那里 agent 程序被恢复且继续从暂停的地方开始执行。agent 将代码和数据封装在执行的一个线程中，每个 agent 独立于其它 agent 之外。从编程角度讲，agent 是自包含的，当 agent 从一个网络节点移动到另一个节点时，agent 保留它的所有状态信息。

除上述基本特性外，根据 agent 具体的应用情况，它又可以拥有其他的特性，这些特性表现在不同的应用场合。根据最近几年的研究和应用，重现了移动 Agent 的概念，那么在分布式的应用场合，移动 Agent 将表现出移动性、自适应性、通信能力(包括协商协作等能力)、理性等。在其他的一些应用场合，Agent 也常表现出持续性、自启动、自利等特性。

Nwana 定义了 Agent 的三层概念结构，见图 3.1：定义层、组织层和合作层。这个概念结构提供了一种描述 Agent 应用特征的框架[34]。

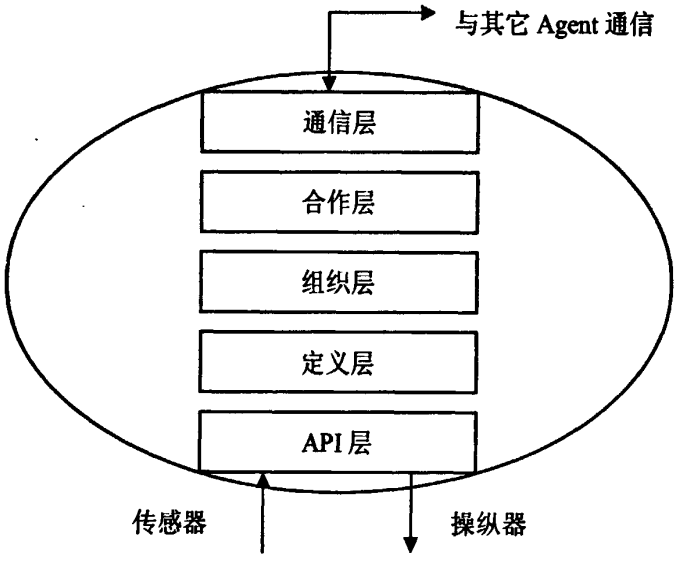

图 3.1 Agent 的三层概念结构

1 定义层：在这一层中，Agent 被描述为一个自治的理性实体，包括 Agent 的推理学习机制、目标、资源、技能等。

2 组织层：在这一层中，定义了 Agent 与其他 Agent 的关系，包括 Agent 在 Agent 团体中所扮演的角色，以及 Agent 之间的相互感知等。

3 合作层：在这一层中指明了 Agent 的社会能力，例如它的合作和协商技术。

图 3.1 中的另外两层：通信层定义了 Agent 之间通信的更低一级的细节；API 层将 Agent 与它的资源、技能的物理实现联系起来。

3.2 多 Agent 系统的研究与开发

3.2.1 多 Agent 系统的研究分析

多 Agent 系统(Multi-Agent System，MAS)是多个不同的 Agent 为了完成某个特定的任务而组成的集合。这些 Agent 相互协作相互交互以完成系统共同的目标，每个 Agent 都处在多 Agent 系统的环境中。多个 Agent 构成的系统是复杂动态的、不确定的，Agent 要对熟悉的环境做出迅速的响应，并且能够同时处理与其他 Agent 的冲突或协调其他 Agent 解决其他冲突，规划其行为，并做出决策。

MAS 含有一组自治型 Agent，它们各自可以完成局部的问题求解，同时还能协作地求解单个目标问题或多个目标问题。一般来说，每个 Agent 拥有推理机、通讯接口、冲突检测与消解三个执行部件，推理所用长期知识分别来自事实、规划、启发式知识和解标准四个知识模块中，黑板存放协商过程中的中间信息。另外，通过外部接口可调整外部知识库，进而协调事实与规划两个知识模块。

在 MAS 中，单个 Agent 的能力是有限的，但它们通过相互间的通讯、合作和协调，能改善各个体的基本能力，提高整个系统的性能，并通过单个 Agent 完成各自的子目标来达到一个相对整个系统的总体目标，而这个总体目标对 Multi.Agent 中的单个 Agent 来说，都会因其能力有限而不可能独立完成。另一方面，MAS 中模块化方法的采用使得系统能很好地适应环境和任务的变化，具有较强的自适应和鲁棒性。在 MAS 中，Agent 不仅能作用于自身，而且能接受环境的反馈信息，重新评估自己的行为，同时，当它无法独立完成目标，需要其它 Agent 的帮助时，它能与其它 Agent 通信、合作完成任务，反复此过程，即构成了 MAS 中 Agent 的行为[36]，如图 3.2 所示。

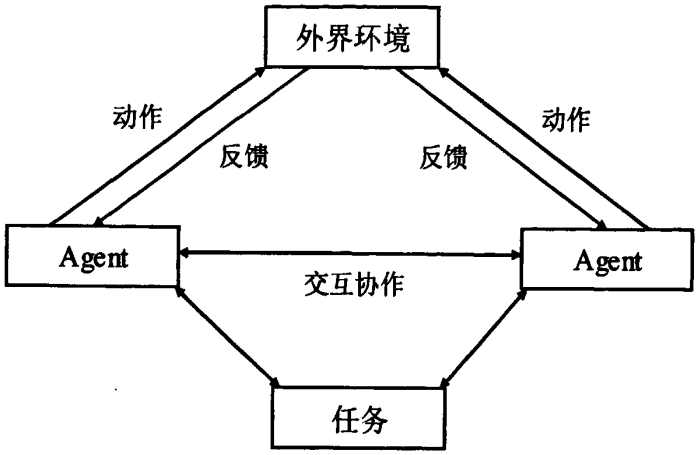

图 3.2 MAS 中的 Agent 行为图

MAS 的体系结构主要是指多 Agent 系统中各个 Agent 之间的信息关系和控制关系，以及问题求解能力的分布模式，它是结构和控制的有机结合，是提供 Agent 活动和交互的框架。也即各个 Agent 以什么样的形式组织起来，以及每个 Agent 具有什么样的结构来共同完成系统任务的求解，通过定义 Agent 之间的权威关系，为 Agent 提供一种交互框架。在多 Agent 技术研究的早期已提出了许多典型的分布式人工智能系统(DAI)组织结构，如 Davis 和 Smith 提出的合同网、Kornfeld 研究的多元团体组织结构等[37]。

总之，用多 Agent 技术进行控制，系统的体系结构不仅要具备一般分布式系统所具有的资源共享、宜于扩展、可靠性强、实时性好的特点[11]，更重要的是要建立一种体系结构能使各 Agent 通过相互的协调、相互协作来解决大规模的复杂问题，同时克服 Agent 之间的行为冲突以及建立一个庞大知识库所造成的知识管理和扩展的困难，使系统对环境和外部干扰具有很强的鲁棒性、可靠性和自适应、自组织能力。这也是多 Agent 技术中体系结构研究的主要方向。

3.2.2 多 Agent 系统的特性及其与单个 Agent 的比较

多 Agent 系统主要研究的是一组自治 Agent 间行为的协调。包括这组 Agent 如何协调知识、目标、技巧和规划，使之能够联合起来进行问题的求解。其主要特性包括：

(1) MAS 中的每个 Agent 均具有解决局部问题或完成局部任务的能力，且独立性较强，即使系统中的某个 Agent 出了问题，其它具有相似功能的 Agent 可代

替它继续执行该任务，对系统造成的影响相对较小，有助于提高系统的可靠性。

(2) 系统中的各个 Agent 为异步的，即各 Agent 按照自己的步调执行任务，这也是多 Agent 系统适合于分布式应用的主要原因。

(3) MAS 具有较强的灵活性。当系统需要扩展或缩减系统功能时，只需通过加入或删除具有相应功能的一个或多个 Agent 即可实现。

(4) MAS 中一项任务的完成有赖于一组分布在不同节点的、功能各异的 Agent 间的通信与协作。系统中，任务通常被分解为多个子任务分配给这组 Agent 并行地执行。

(5) 实践证明，在复杂的系统中一致性的要求很难满足。因此 MAS 还必须考虑如何解决冲突、欺骗以及不同观点和目标集成等问题。

MAS 与单个 Agent 的主要区别为：

(1) 在单个 Agent 系统中，一个 Agent 独立面对一项任务，其内部结构相对复杂。多 Agent 系统中则将任务分解为多个相对简单、易完成的子任务执行，各 Agent 间的交互与协作才是关键，因而 Agent 的结构相对简化。

(2) 由于单个 Agent 功能单一且能力有限，其面对的问题领域和应用范围较窄。但在多 Agent 系统中，通过协作不但可提高单个 Agent 和整个系统解决问题的能力，还可以使系统具有更好的灵活性，拓宽其应用领域。

(3) 对单个 Agent 来说，它只关注自身的需求和目标，因而其设计和实现独立于其它的 Agent。但在多 Agent 系统中，各 Agent 的行为必须遵守其所在系统的社会规范，Agent 间内在的相互依赖关系使得对各 Agent 的设计和实现受到一定的制约。

(4) MAS 与单个 Agent 系统的本质区别在于 MAS 支持 Agent 间的通信与协作。

MAS 可解决一些传统方法所不能解决的问题，诸如开放性、动态性和复杂性等，尤其适合应用于高度开放、松散藕合的网络环境中。利用 Multi-Agent 技术开发的系统，不但易于维护，效率高，而且还具有较高的灵活性。

3.2.3 多 Agent 系统的开发方法

多 Agent 系统牵涉的面比较广，因为 Agent 的特殊性，使得 MAS 开发别具

一格，一般基于 Agent 技术的应用开发可以按照一下几个步骤来执行[42]：

1 分析系统的特点，选择最合适可行的技术。当系统具有跨平台。跨网络、跨地域、跨行业的互操作性以及较高的个人化、智能性等特点时可以考虑采用 Agent 技术来实现。

2 Agent 的功能设计。当决定采用 Agent 技术来实现系统时，接下来要做的就是确定系统采用 Agent 技术实现的数据及功能，考虑采用什么种类的 Agent。

3 Agent 的接口设计。确定了 Agent 的功能后，那么接下来的步骤也是最重要一步则是 Agen t 的接口设计。特别是在多 Agent 系统中，Agent 的接口设计非常关键，通常它关系到系统的效率，扩展性和安全等方面。Agent 的接口设计包括 Agent 间的交互、Agent 与非 Agent 部分的交互。

4 Agent 的详细设计和实例化。在前面的设计工作做好后，接下来要做的就是 Agent 的详细设计，包括 Agent 的内部设计、包括开发平台的搭建和编码工作。

5 Agent 的运行和维护。在所有的软件工程中，系统开发的最后一步都是系统的运行和维护。同样，在多 Agent 系统的开发中，Agent 的运行和维护也是极其关键的一步。

3.3 多 Agent 的协作机制

3.3.1 Agent 间的协作

Agent 之间的协作是保证多个 Agent 能在一起共同工作的关键。在开放、动态的多 Agent 环境下，具有不同目标的多个 Agent 必须对其目标、资源的使用进行协调。在出现资源冲突时，若没有很好的协调，就有可能出现死锁，使得多个 Agent 无法进行各自的下一步的工作。协调与协作是多 Agent 技术研究的核心问题之一。

多 Agent 系统必须找出一种使各 Agent 能够协调工作的适当方法，这种方法是建立在多 Agent 共享资源和各 Agent 自主性之上的。虽然独立的 Agent 有各自分散的目标、知识和推理过程，但是它们之间必须有一种方法能够互相协调、互相帮助，以找到整个系统的目标[43]。这种 Agent 之间在合作前或合作中的通信过程可称为 Agent 之间的协作。

从不同角度观察，对协作的定义也有多种。如 Sycara 把协作定义为：协作过

程包括通过通信或对系统中其他 Agent 目前的状态或意图的推理找出潜在的交互，并通过修改其他 Agent 的意图来避免有害交互或进行合作。Durfee 把协作定义为：通过结构化地交互相关信息，提高对共同视图或规划的一致认识(减少不一致性和不确定性)的过程。T.Matone 把协作定义为：协作是指对各个行为之间依赖关系的管理。

Agent 的交互机制是 Agent 社会性的重要表现，也是多 Agent 系统研究的核心问题。健全的交互机制是多 Agent 间进行协调和协作的基础。一般说来，Agent 与 Agent 之间的交互包括两方面的内容：交互协议和交互策略。

交互协议是对 Agent 有组织的信息交换过程的一种抽象和规定。它直接反映了 Agent 交互目的和交互规则，与 Agent 内部的推理机制也紧密相关，是多 Agent 系统研究的重点。

交互策略涉及到对所求问题的分析，对相关 Agent 情况的了解，以及对相关交互协议的分析。交互策略的制定是 Agent 智能特性的重要体现，也是 Agent 成功交互的关键。

在多 Agent 中，具有不同目标的各个 Agent 必须对其目标和资源使用进行协作。因此，多 Agent 研究的关键就是参考经济学和社会学的有关理论和模型来处理 Agent 之间的协作。通常协作有如下几类[44]：

资源共享：资源共享也许是最常见的一种协作方式。在经济学中，市场的价格和拍卖机制是被广泛研究的一种资源分配方式，在计算机系统中，资源分配也是很重要的。例如，操作系统中的调度程序就是用来分配处理器、内存以及外设等资源的。值得一提的是，任务分配是一种特殊的资源分配，是把个体有限的时间分配给个体需要执行的任务。

生产者/消费者关系：这是另一种非常普遍的协作方式。在这种方式中，个体 A 生产的产品(包括信息)为个体 B 所使用。在经济学中，生产者消费者的关系是研究市场的基本出发点，而在计算机科学中，数据流和 Petri 网分析也是研究生产者/消费者关系的例子。

任务/子任务关系：这种协作方式又可分为自顶向下的目标分解和自底向上的目标辨识两种。自顶向下的目标分解经常出现在人类组织的规划中，在计算机系统中则表现为程序设计中的模块化思想。自底向上的目标辨识是指一些个体认识

到它们正在做的事情，可以联合起来以达到一个新的目标。对策论中的结盟就是研究这种情况。

也有一些研究者就问题的求解方式把多 Agent 系统的协作形式分为任务共享和结果共享两种方式。任务共享是指单个 Agent 可以用最少的通信和全局同步信息完成子问题求解。任务共享要求对任务进行适当的分解。合同网协议正是基于任务共享的。基于任务共享的协作模式见图 3.3。结果共享是指 Agent 之间通过共享部分结果的形式互相协助。这种方式在 Agent 之间的部分结果互有交互时比较合适。基于结果共享的协作模式见图 3.4。

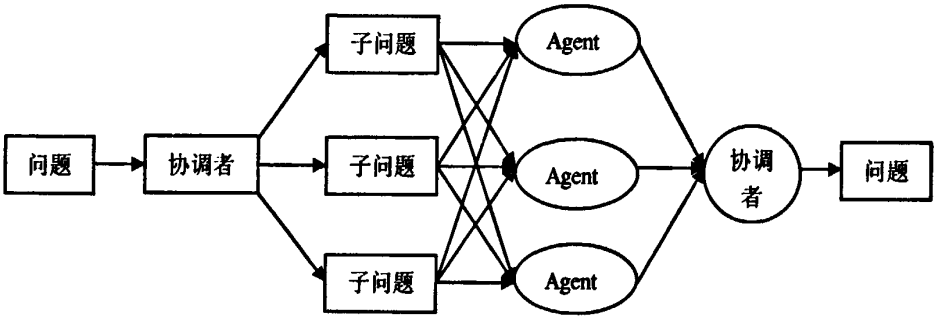

图 3.3 基于任务共享的协作模式

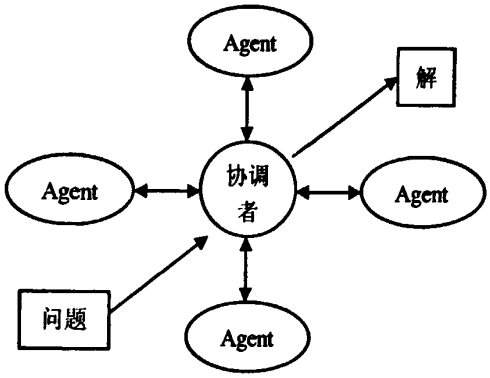

图 3.4 基于结果共享的协作模式

3.3.3 Agent 间协作的实现

目前，虽然多 Agent 间的协作机制的实现方法众多，但究其根本有如下三种实现方法：

无通信的协作：其方法与传统对策论相似。

有控制中心的协作：它类似于传统操作系统的方法，即为每个 Agent 群体增设一个可称为助理器的装置，所有的位于同一群体中的 Agent 通过向助理器声明一组描述信息的方式与此装置通信。助理器还可负责完成与其它群体中的助理器的交互，实现与其它助理器管理下的 Agent 的协作。

协商：参与者通过交换相关的结构化信息，形成一致的观点和规划，达成意图上的一致。即协商是自治 Agent 通过协调它们的世界观及相互动作、解决矛盾和冲突来达到它们目的的过程。协作中的协商机制通常应考虑以下因素：

1 行为自治性：Agent 可以根据自己当前的状态决定采取合作或拒绝的态度。

2 协商快速性：协商过程应尽快完成，否则将可能失去协作的优势。

3 计算简便性：计算复杂度尽可能低，易于实现。

4 协作有效性：达成的合作局面稳定，协作完成任务的效果应好于独立完成。

5 协作历史性：Agent 协商时应考虑历史上的合作记录。

3.4 本章小结

本章介绍了 Agent 的定义及其基本特性以及阐述了软件 Agent 是什么。接着对基于 Agent 的系统进行深入介绍。在阐述多 Agent 系统概念，分析多 Agent 系统的特性的基础上比较了多 Agent 系统与单 Agent 系统的差别，进而对多 Agent 系统的体系结构进行研究并介绍了多 Agent 系统的开发方法。最后，结合以上理论，详细介绍了多 Agent 间的协作机制，包括多 Agent 间协作的定义、分类以及实现方法。

第四章 MAWFMS 模型的设计

针对传统工作流管理系统中存在的问题，本章结合 Agent 技术提出了一个基于多 Agent 交互协作的工作流管理系统的模型（WORKFLOW MANAGEMENT SYSTEM BASED ON MUTI-AGENT INTERACTIVE COOPERATION，MAWFMS），通过设计具有各种功能的 Agent，将相应的业务逻辑从系统中分离，有效降低了系统的耦合度，提升了系统的可扩展性和易维护性。本章将阐述 MAWFMS 模型的设计思想，结合工作流管理联盟给出的工作流管理系统体系结构，提出 MAWFMS 模型的整体体系结构并进行详细设计以及分析了 MAWFMS 模型的性能。

4.1 MAWFMS 模型的设计思想

随着现代企业自动化程度的不断提高，传统的工作流管理系统的缺点不断暴露，比如缺乏支持松散耦合的体系结构、缺乏自主能力、灵活性低、智能性弱等。因此如何有效的解决以上问题是工作流管理系统进一步深入发展与应用的关键所在。在这种情况下，引入一个新技术来继续发展和完善工作流管理系统无疑是一个行之有效的方法。Agent 一般具有自主性、交互性、反应性和主动性的特征，并且各 Agent 之间是一种松散耦合的关系。因此，Agent 能够解决目前工作流管理系统中存在的普遍问题，它们的结合必将为工作流管理系统的发展注入新的活力。

基于 MAWFMS 模型的设计经过了对大量工作流管理技术与 Agent 技术的深入学习，对当前工作流管理系统中存在的问题深入分析，并在此基础上参考工作流管理联盟（Workflow Management Coalition，WfMC）关于工作流管理系统的参考模型，将 Agent 技术引入模型架构并对各个成员 Agent 进行功能设置。MAWFMS 模型的设计思想来源于解决当前工作流管理系统中存在的问题，具体问题如下：

1 任务处理质量低效率差：基于 Agent 的智能计算的特点，使原本将数据集中计算处理转为数据分布于各 Agent 中计算处理再汇集，分担了系统的压力，加

速了系统的运转，从而使任务处理又好又快。

2 系统中资源冲突问题：工作流管理系统在企业业务系统中的资源是独立的，因此当两个工作流实例同时需要某一资源时即发生竞争，利用 Agent 交互协商调用相关资源可有效解决此类问题。

3 任务分配方式僵化问题：在工作流执行过程中，工作流引擎被动的解释执行过程定义，在任务分配过程中，目前常用的两种方式是“拉”模式和“推”模式，方式单调缺乏一定的灵活性，不能满足一些复杂的需求。

4 用户在工作流管理系统中的被动性：在协同工作中，强调用户参与协作过程的主动性。不同企业的员工日程安排自由度不同，Agent 可以协助用户自主的安排日程，提高用户在系统中的主动性。

5 系统高度耦合的问题：各个 Agent 将其独自承担一些业务从系统中分离出来，实现业务分离，从而降低了系统的耦合性。

4.2 MAWFMS 模型的体系结构

在深入学习工作流与 Agent 的相关技术以及认真分析当前工作流管理系统存在问题的基础上，本文提出一种基于多 Agent 交互协作的工作流管理系统的模型，简称 MAWFMS 模型。在设计过程中，本文以工作流管理系统的运行控制为主线，工作流定义与监控为辅助，设计出一种基于 MAWFMS 的模型，对工作流程的整个生命周期进行管理，包括流程的启动、任务分派、执行、控制、结束、监控等。MAWFMS 模型的提出主要是为了解决上文中提出的当前工作流管理系统中存在的诸多问题，因此在模型设计上，需要遵循如下原则：

1 在系统的整体架构上，采用先进的工作流管理办法，引进先进的管理思想，统一规划，分布实施，提高工作流管理的规范化程度，通过更好的规范化流程提高工作流管理系统的整体运作效率。

2 在技术层面上，采用先进的工作流技术与 Agent 技术相结合的方法融合角色管理，进而提高工作流管理系统的安全性、实用性，增强系统的可扩展性与灵活性。

本模型的核心功能为“运行控制”，注重管理过程中的流程控制环节，结合其他系统管理要素的综合性工作流管理系统模型。它的目的是为了帮助用户更加

科学的管理管理工作流实例，达到任务的高效分派与执行。目前工作流系统中的流程通常由多个环节组成，每个流程又需要相应的表单，整个流程将会涉及多个部门的不同级别的工作人员的信息交互。针对以上提出的问题，MAWFMS 模型在设计环节中应该注意以下五点：

1 开放性，采用符合国际标准的协议规范，选取具有很强的与其他主流产品互通能力的产品。

2 柔性，可以较少的改动来适应将来未知的变更。

3 可复用性，在设计过程中要尽可能的识别出一些具有共同特性的模块。

4 可靠性，采用先进的系统架构、技术措施、开发手段，确保模型的可靠性。

5 安全性，提供身份验证、角色授权与访问控制，实现系统的整体安全。

综上，本文参考工作流管理联盟给出的工作流管理系统体系结构，将多 Agent 集合加入到工作流执行服务模块中，给出了 MAWFMS 模型的整体体系结构图，如下图 4.1 所示。

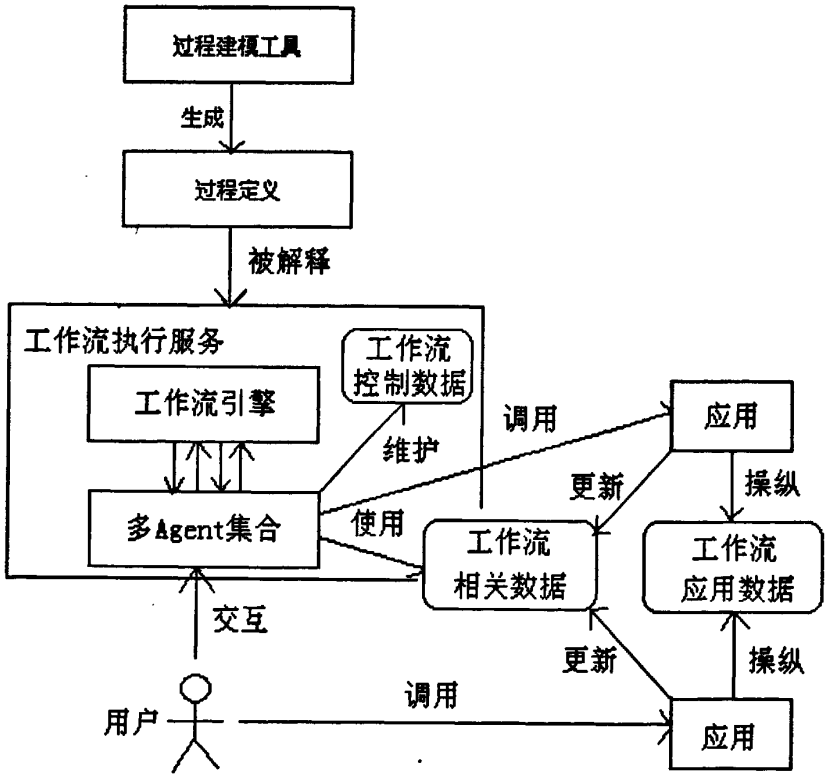

图 4.1 MAWFMS 模型的体系结构图

在 MAWFMS 模型所涉及的数据层面上，本文继承了工作流管理联盟提出的

工作流参考模型中的三种数据类型，但再加入多 Agent 集合后，对它们做出一定修改，如下：

1 工作流控制数据（Workflow Control Data）：工作流执行服务中的多 Agent 集合维护工作流控制数据，并携带这些数据与工作流引擎进行交互，使工作流引擎辨别每个过程或活动实例的状态。这些数据由工作流执行服务中的多 Agent 集合进行控制。用户、应用程序或其它工作流执行服务组件不能对其进行直接读写操作，它们可以通过向工作流执行服务中的多 Agent 集合发送消息请求来获得工作流控制数据的内容。

2 工作流相关数据（Workflow Relevant Data）：工作流管理系统通过工作流相关数据来确定过程实例状态转换的条件，并选择下一个将执行的活动。工作流管理软件需要在活动实例之间传递工作流相关数据。这些数据被工作流执行服务中的多 Agent 集合使用，并可以被工作流应用程序访问并修改。

3 工作流应用数据（Workflow Application Data）：由应用程序操作的数据，它们是针对应用程序的，是完成企业具体的业务功能所需要的数据，工作流管理系统无需对他们进行访问。

在 MAWFMS 模型的工作流执行服务层面上，主要有工作流引擎和多 Agent 集合两大组成模块，以下我们将对它们进行简要介绍，具体设计将在之后章节予以说明。

1 工作流引擎是整个工作流管理系统的核心组成模块，根据工作流管理联盟给出的工作流参考模型，结合 MAWFMS 模型的体系结构，它在与多 Agent 集合进行交互的基础上主要完成以下任务：

(1)对过程定义进行解释。

(2)控制过程实例的创建、激活、挂起、终止等。

(3)控制活动实例间的转换，包括串行、并行、选择或循环操作的执行。

(4)提供支持 Agent 操作的接口。

(5)通过与多 Agent 集合的交互，维护工作流控制数据和工作流相关数据。

(6)提供用于激活外部应用程序的接口。

(7)提供控制、管理、监督工作流过程实例执行情况的功能。

2 多 Agent 集合与工作流引擎共同构成传统工作流管理系统的工作流执行服

务模块，它们根据用户所定义的工作流模型中对流程的定义以及各个活动的定义，实现工作流程的实例化。在这个实例化过程中，创建、执行、调度相应的 Agent，并通过相关 Agent 之间的交互协作保证工作流程的顺利完成。根据所要完成的功能不同划分为用户 Agent、任务 Agent、资源 Agent、决策 Agent、监控 Agent、扩展 Agent。我们将在接下来的本章第四节对他们的功能、结构以及相互通信的策略进行详细的介绍。

4.3 工作流引擎的设计

4.3.1 工作流程的路由设计

工作流的流程执行实际上就是业务过程中的任务按照一定的顺序关系被执行。下面将介绍MAWFMS模型中可以执行的路由方式。其中，方框表示任务，箭头表示任务之间的顺序关系。

1顺序路由：执行完一个任务后，继续执行下一个任务。如图4.2所示。

图4.2 顺序路由

2并行路由：执行完一个任务后，同时开始若干个任务，它们都完成后再触发后续任务。如图4.3所示。

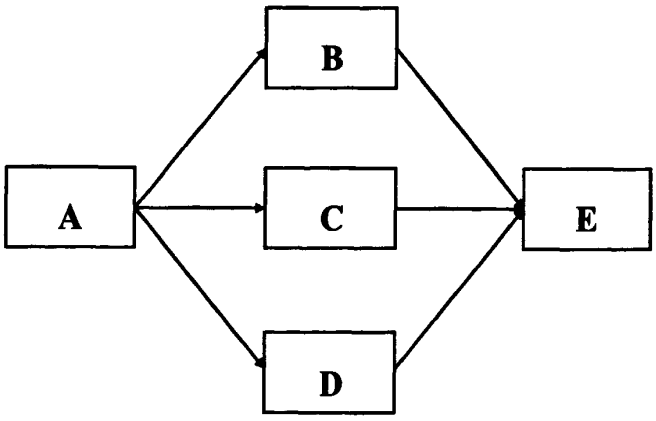

图4.3 并行路由

3条件路由：执行完一个任务后，根据一定的条件进行判断来选择执行后续任务。如图4.4所示。

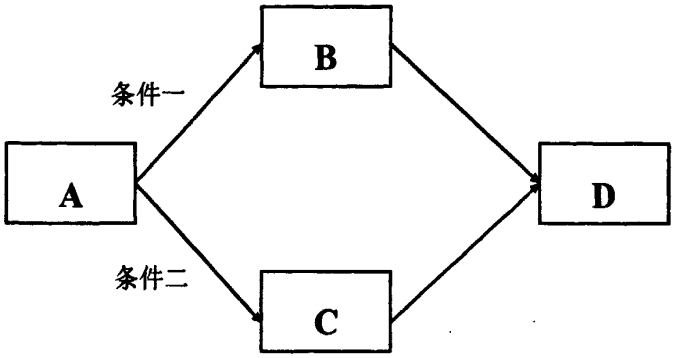

图4.4 条件路由

4循环路由：循环地执行某一个任务。如图4.5所示。

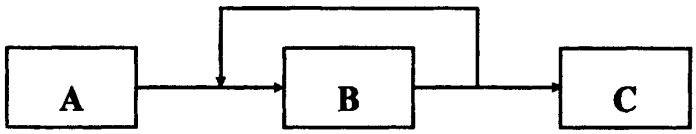

图4.5 循环路由

4.3.2 过程的执行状态设计

在工作流管理系统的参考模型中，流程、节点都是通过状态来反映业务的处理情况，工作流执行服务通过实例状态来控制实际任务的办理过程。下面我们将以过程实例状态的具体转换为例进行详细分析状态转换的规则。下图4.6首先描述了过程实例状态间的转换。

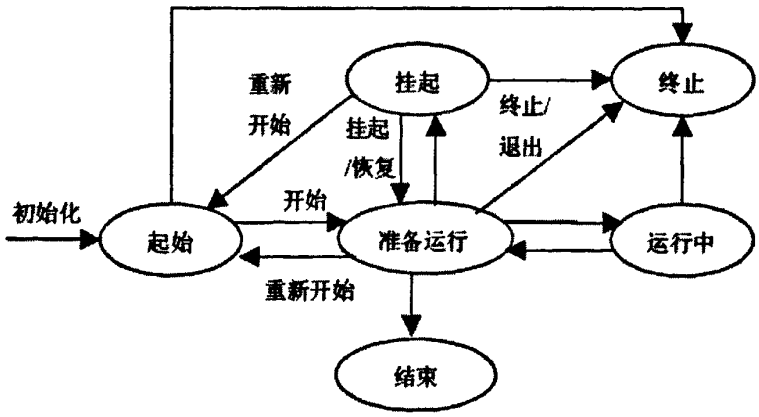

图4.6 过程状态转换图

1初始态：该流程实例已被创建，但此时还没有满足启动执行的条件并响应一个启动流程事件。处于初始态的过程，一定是通过启动过程的第一个节点而开始整个过程的。

2等待态：该过程的实例已经可以执行，等待执行第一个活动并生成一个任务项条件的满足。

3运行态：该过程实例已经成功启动执行，过程实例处于一种活动的状态，通过参与者的参与，按照一定的条件进行流程路由。

4挂起态：该过程实例正在运行，但处于静止状态，除非有一个“重启”命令或者其他外部事件触发该过程实例回到等待态，否则该过程实例的活动无法继续执行。

5完成态：该过程实例满足了一定的结束条件，流程成功地执行完成。完成状态是状态转换中的最后一个状态，从此不会有状态转换行为发生。但完成态意味着该结束不是出于任何强迫行为所导致的结果，而是过程实例按照预定方案进行正常的状态转换所形成的结果。

6终止态：该过程实例在流程正常完成之前，执行被某个事件强行结束。终止态也是过程状态转换中的最后一个状态，从此不会再有过程状态转换行为的发生。

我们以一个公文处理的实例来更加具体的说明各个状态。当客户根据工作需要起草发文呈批表的时候，发文流程即被初始化，过程处于初始态；当客户提交领导审批后，发文流程正式开始执行，过程的状态跳转为运行态；当文件已经成功发出时，发文流程也就结束，那么该过程的状态转换为完成态；如果文件被撤销等原因导致客户或管理人员认为该流程没有必要继续进行的时候，可以选择终止过程的执行，那么过程跳转至终止态。

4.3.3 过程实例的控制设计

工作流引擎的主要功能就是控制过程实例。它包括解释过程定义、创建以及终止过程实例。在该设计中，过程实例控制的基本流程如下：

1当工作流引擎接受到一个有效的数据，则创建一个过程实例，并执行它的初始化活动。

2将事务状态设置为初始状态，此时过程执行状态为初始态。

3过程实例接受数据，执行指定的活动，事务状态调转到指定的状态。

4如果此状态不是终止或者完成状态则过程的执行状态为运行态，继续循环步骤3。

5如果此时的事务的状态转到终止状态，则过程的执行状态为终止态。

6如果此时的事务的状态转到完成状态，则过程的执行状态为完成态。

以上1->2->3->4->5或6步骤完成了对一个过程实例的控制。系统会将所有被创建还没有被终止或完成的过程实例放入过程列表中，根据每个过程的优先级等属性统一调度控制。在路由选择的调度过程中引擎支持参数控制的自动流转也支持用户控制的非自动流转，具有很好的柔性路由选择。

4.3.4 任务分配方式的设计

任务分配是将当前需要完成的任务分派给合适的参与者，是工作流引擎的一个主要功能。在流程定义的时候，有时无法确定或没有必要指定某个具体的参与者，而是指定了一定范围内的人员来做这个任务，这种情况下就需要角色的柔性解析。柔性角色解析是指在流程定义的时候，不指定任务的具体执行者，只是指定执行该任务的人员角色，在流程运行时动态指定执行者。任务的柔性分配包含两种方式：

1、推模式：从满足条件的用户集中选取一个或多个参与者把任务指派给他们。指定参与者后，设置任务实例的属性，将这个任务实例“推”进参与者的任务列表中，等待参与者的执行。参与者可以通过“待办工作”模块获取自己所需要执行的任务。

2、拉模式：这种方式是将任务实例分配给一类参与者，通常这些参与者都具有相同的角色，他们中的任何人都可以将这个任务“拉”到自己的任务列表中。这种方式是建立在参与者都具有较大工作积极性的前提下的任务指派方法。这种方式要求任务的指派需要公开，属于该角色的参与者都能看到这个任务，一旦某个参与者从任务池中“拉”了这个任务之后，那么其他参与者就不能够再获取这个任务。

3、智能选推模式：这种方式是本文提出的创新点之一，通过Agent的智能决

策，从众多备参与者中挑选出一名最适合完成当前任务的参与者，将任务推进该参与者的任务列表并予以通知。有关智能决策的设计详见本文4.5节。

4.4 多 Agent 集合的设计

4.4.1 Agent 的分类

本文根据各 Agent 所要完成功能的不同，将 MAWFMS 模型中 Agent 具体分为以下六类：

1 用户 Agent：作为用户的助手，用来代理系统参与者的用户，维护用户的相关信息，并根据用户角色的不同来提供用户的个性化数据。此外，作为工作流程中的主要参与者，与任务 Agent 进行交互协作，完全或部分帮助用户完成一定的操作，最终完成整个工作流程。

2 任务 Agent：当启动一个工作流程时，相应的创建一个任务 Agent。任务 Agent 描述工作内容，提供工作数据以及一些其他资源。它的主要功能就是控制过程实例的执行、活动的调度、工作流控制数据的维护等。

3 资源 Agent：作为 MAWFMS 模型中的大管家，资源 Agent 负责管理系统中所要使用的数据、设备等资源，不同类型的资源划分为不同类型的资源 Agent。当其他 Agent 需要资源时，向资源 Agent 发出请求，资源 Agent 获取所需数据再打包送回。在 MAWFMS 模型中引入资源 Agent 的另外一个重要因素就是利用资源 Agent 来解决系统资源冲突问题，做到资源的统一管理、统一分配。

4 决策 Agent：作为 MAWFMS 模型中的组织部长，决策 Agent 负责 MAWFMS 模型中任务的分配，它通过与用户 Agent、任务 Agent、资源 Agent 的交互，获取决策所需要的数据，采用一定算法，选择出该项任务最合适的用户。决策 Agent 也是作为 MAWFMS 模型中智能决策的关键组成部分。

5 监控 Agent：作为 MAWFMS 模型中的监督者，监控 Agent 负责监控工作流引擎中各个流程实例的执行情况，并且在必要的时候，将信息提供给用户。监控 Agent 在应用服务启动的时随之启动并且保持持续工作。

6 扩展 Agent：为了加强 MAWFMS 模型的可扩展性，扩展 Agent 提供一组可供其它应用系统集成的接口。

4.4.2 Agent 的结构

不同功能的 Agent 在结构上各有不同，功能复杂则结构复杂，功能简单则结构简单。下面我们将 Agent 中共有的结构部分加以说明，图 4.7 介绍了 Agent 的结构图。

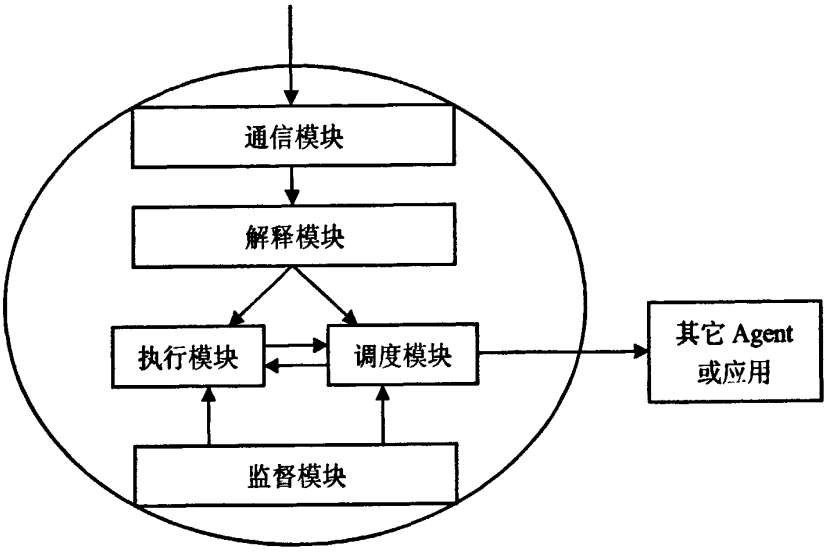

图 4.7 Agent 的结构图

1 通信模块：负责与其他 Agent 或应用程序进行数据交互，包括一对一、一对多的交互方式。

2 解释模块：负责解释发送给 Agent 的命令，然后根据需求交由执行模块或者调度模块，完成后续工作。

3 执行模块：任何完成具体功能的 Agent 都具有此模块，它在接受指令后完成本 Agent 的业务功能。

4 调度模块：当 Agent 对消息进行解释后，通过调度模块调用相应的应用程序完成任务。Agent 调度模块的构成与 Agent 所要完成的功能密切相关。

5 监控模块：也可以说是错误处理模块，主要功能是监控 Agent 的执行过程，当 Agent 的执行过程中发生错误时，通过一定的错误处理保证系统的正常运行。

4.4.3 Agent 间的通信

Agent 间的通信是 MAWFMS 模型的重要研究内容之一。Agent 间的通信是

建立在下层可靠的数据通信基础之上，但它与下层的数据通信语言不同，Agent 间的通信内容是具有含义的消息或知识。由美国高级研究计划署（ARPA）提出的交互语言 KQML（Knowledge Query and Manipulation Language）受到了广泛的关注，已经有成为 Agent 通信标准的趋势。KQML 语言强调了 Agent 的自主性，即没有预先限定 Agent 是客户还是服务器。此外，KQML 语言提供了一套简单有效的实用性技能描述语言，对通信内容表达语言没有作限定，它独立于网络传输机制、独立于内容语言、独立于内容实体。KQML 消息的三层结构如下：

1 内容层：在程序开发语言中,包含有消息的实际内容，KQML 支持 ASCII 码和二进制符号。

2 通信层：实现消息特性,将低级的消息参数，例如消息的传送者和接受者,采用特定的标识符进行标识.。

3 消息层：KQML 的核心，其基本功能是标识用以发送消息的协议，提供讲话动作或执行发送者在内容中附加的行为，另外，消息层还包括一些用以描述内容层中信息的可选部分，如语言，采用的术语，通信主题描述符等，以便 KQML 对要传递的内容进行分析、路由和发送。

在 MAWFMS 模型中，工作流程的执行主要依靠各个相关 Agent 以及工作流引擎的交互协作。因此，Agent 间的通信必不可少，模型中 Agent 间的通信主要发生在下列 Agent 之间：

1 任务 Agent 与用户 Agent 之间：这是系统中最频繁的通信。在工作流程的执行过程中，用户参与任务的执行通过用户 Agent 与任务 Agent 之间的交互协商确保任务的顺利完成。

2 任务 Agent 与决策 Agent 之间：当一项任务到达时，任务 Agent 为了选择最合适的人选，将任务的描述信息传递给决策 Agent，决策 Agent 找到最合适的人选返回任务 Agent。

3 任务 Agent 与资源 Agent 之间：在工作流程的执行过程中，工作流引擎通过任务 Agent 对工作流控制数据进行维护。而任务 Agent 又是在与资源 Agent 的通信基础上完成维护操作。

4 决策 Agent 与资源 Agent 之间：决策 Agent 在接受任务后，与资源 Agent 进行通信，要求相应的资源 Agent 提供决策所需要的数据。

5 决策 Agent 与用户 Agent 之间：决策 Agent 在接受任务后，与用户 Agent 进行通信，要求相应的用户 Agent 提供该用户的个性化数据以供决策 Agent 决策使用。

6 用户 Agent 与资源 Agent 之间：当用户需要某个具体资源时，通过用户 Agent 向资源 Agent 发出申请来获取。

4.5 系统智能决策的设计

4.5.1 智能决策分析法的设计

MAWFMS 模型的一个创新点就是在系统中引入了智能决策，它将为工作流过程中的每个活动选择最合适的用户进行任务分配。智能决策的引入，使 Agent 真正成为了人的“助理”，它将完全或者部分帮助用户完成一些操作。当然，Agent 只是一个软件实体，它的智能需要进行一定的设计，使其具有完成一定功能的智能分析。这就要求在流程中设定一些权重系数供 Agent 计算，包括用户个性化数据项、任务个性化数据项以及用户在系统中的相关数据项等。通过对这些数据的权重系数计算，得出满足任务需要的最优用户，即一次智能决策。目前关于权重系数的精确测度主要有“专家咨询法（Delphi）、层次分析法(Analytical Hierarchy Process，简称 AHP)、二项系数加权法、环比评分法”等。其中比较有代表性的、较成功的主要有 Delphi 法和 AHP。根据 MAWFMS 模型的实际需要，本文选择更为合适的层次分析法（AHP）来进行权重计算。

层次分析法(Analytic Hierarchy Process 简称 AHP)是美国运筹学家 T. L. Saaty 教授于 70 年代初期提出的， AHP 是对定性问题进行定量分析的一种简便、灵活而又实用的多准则决策方法。它的特点是把复杂问题中的各种因素通过划分为相互联系的有序层次，使之条理化，根据对一定客观现实的主观判断结构(主要是两两比较)把专家意见和分析者的客观判断结果直接而有效地结合起来，将同一层次元素两两比较的重要性进行定量描述。而后，利用数学方法计算反映每一层次元素的相对重要性次序的权值，通过所有层次之间的总排序计算所有元素的相对权重并进行排序。该方法自 1982 年被介绍到我国以来，以其定性与定量相结合地处理各种决策因素的特点，以及其系统灵活简洁的优点，迅速地在我国社会经济各个领域内，如能源系统分析、城市规划、经济管理、科研评价等，得到

了广泛的重视和应用。

这里用一个更为直观的例子来说明 AHP，使大家对 AHP 进行形象的理解，同时也使大家对本系统的智能决策模式有个更为清晰的认识。本文以一个用户要到市场中买一个电冰箱为例。他对市场上的 6 种不同类型的电冰箱进行了解后，在决定买哪一款式，往往不是直接进行比较，因为存在许多不可比的因素，而是选取一些中间指标进行考察。例如电冰箱的容量、制冷级别、价格、型式、耗电量、外界信誉、售后服务等。然后再考虑各种型号冰箱在上述各中间标准下的优劣排序。借助这种排序，最终做出选购决策。在决策时，由于 6 种电冰箱对于每个中间标准的优劣排序一般是不一致的，因此，决策者首先要对这 7 个标准的重要度作一个估计，给出一种排序，然后把 6 种冰箱分别对每一个标准的排序权重找出来，最后把这些信息数据综合，得到针对总目标即购买电冰箱的排序权重。最后根据这个权重排序，做出最后的决定。

4.5.2 MAWFMS 智能决策的流程

MAWFMS 的智能决策由决策 Agent 负责完成。决策 Agent 需要向三个 Agent 请求获取数据，分别是：请求任务 Agent 提供该任务的个性化数据，了解任务需要什么样的人选；请求各用户 Agent 提供该用户的个性化数据，了解该用户是否符合任务的需要；请求资源 Agent 获取用户的系统应用其他相关数据，了解各用户的工作状态。各个 Agent 的数据来源为：任务 Agent 的数据在任务创建时由管理员填写，任务不同数据不同；用户 Agent 的数据在各用户初始 Agent 时完成自己的个性化数据填写，在系统运行过程中，用户个性化数据如无特殊变动维持不变，如需改变则进入用户 Agent 设置窗口，进行数据设置并确定保存即可；资源 Agent 数据来源于即时请求数据库获取。决策 Agent 在获取决策所需要的数据之后，应用算法搜索最佳人选。本文采用层次分析法作为决策算法，通过计算得出各个指数值与其权重的乘积之和，指数值越大说明备选人在这个方面具有越大的优势，权重值越大说明该指数项在决策时占的比重越大，决策的最佳人选就是最终乘积之和最大的备选人。这里需要指出的，数据项以及数据项的权重在系统中指定并统一管理。在 MAWFMS 模型中，系统的一次决策中各 Agent 间的请求过程如图 4.8 所示。

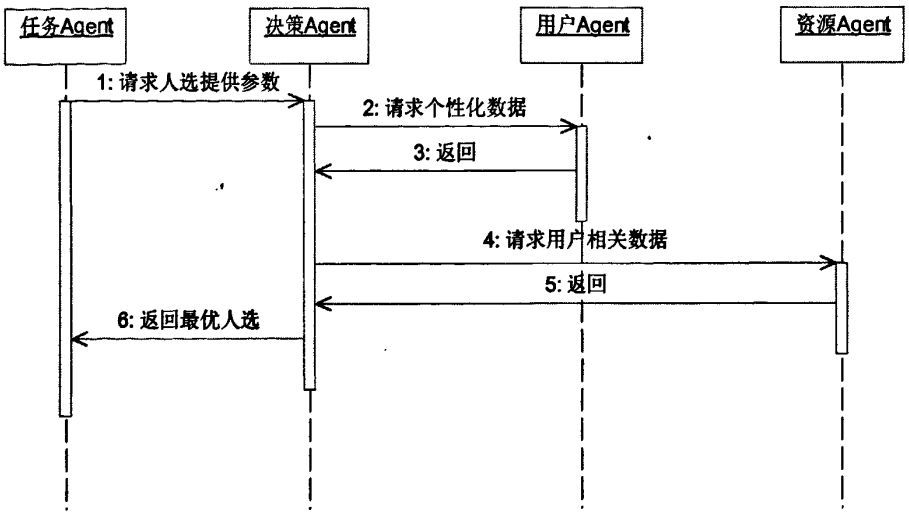

图 4.8 Agent 间请求协作过程图

在获取所需决策数据后，决策 Agent 应用 AHP 得出最优人选返回给任务 Agent，应用 AHP 进行决策计算的具体过程为[56]：

步骤 1：构造多级递阶结构，把复杂问题分解为元素的各个组成部分，并按照要素间的互相关联以及相互隶属关系形成不同的层次。一般情况下，最高层为目标层，代表问题的总目标；中间层为准则层，对下一层次的要素起支撑作用，同时又受其上层支配；最下层是解决问题的各种方案。图 4.9 为一个典型的完全相关层次结构图。

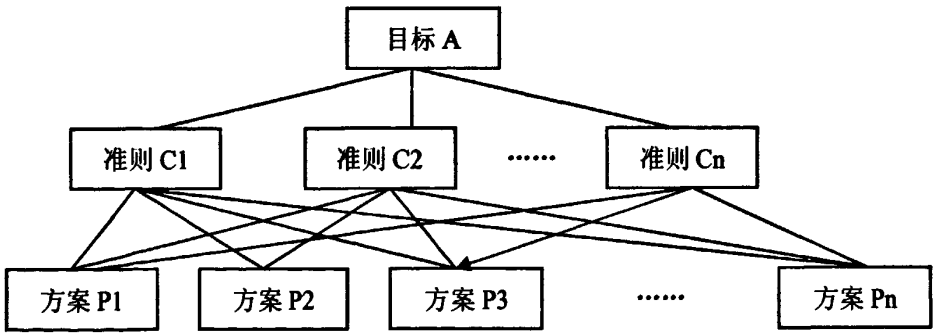

图 4.9 完全相关层次结构图

步骤 2：建立模糊判断矩阵。判断矩阵是层次分析法的基本信息，也是计算各要素权重的基本依据。经过上一步构造多级递阶结构的操作后，上下层次之间的要素就已经确定了，其格式如下：

R	c_1	c_2	$\cdots$	c_n
c_1	r_{11}	r_{12}	$\cdots$	r_{1n}
c_2	r_{21}	r_{22}	$\cdots$	r_{2n}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
c_n	r_{n1}	r_{n2}	$\cdots$	r_{nn}

模糊评判矩阵 R 表示针对上一层次某元素，本层元素与之有关元素之间相对重要性的比较，r_{ij} 表示元素 c_i 和元素 c_j 相对于元素 R 进行比较时具有模糊关系“…比…重要的多”的隶属度，其中 $0 < r_{ij} < 1$。为了使任意两个方案关于某准则的相对重要程度得到定量描述，可采用如表 4-1 所示的数量标度。

表 4-1 数量标度

标度	定义	说明
0.5	同等重要	两元素相比较，同等重要
0.6	稍微重要	两元素相比较，一个元素比另一元素稍微重要
0.7	明显重要	两元素相比较，一个元素比另一元素明显重要
0.8	重要的多	两元素相比较，一个元素比另一元素重要的多
0.9	极端重要	两元素相比较，一个元素比另一元素极端重要
0.1，0.2 0.3，0.4	反比较	若元素 c_i 和元素 c_j 相比较得到判断 r_{ij}，则元素 c_j 和元素 c_i 相比较得到判断 $r_{ji} = 1 - r_{ij}$

R 具有如下性质：

（1）$r_{ii} = 0.5, i = 1,2,\cdots n;$

（2）$r_{ji} = 1 - r_{ij}, i, j = 1,2,\cdots,n$；即 R 是模糊互补矩阵。

步骤 3：模糊判断矩阵一致性检验。

在实际决策分析中，由于所研究的问题的复杂性和人们认识上可能产生的片面性，所构造的判断矩阵往往不具有一致性，这是就需要调整，具体调整步骤为：

（1）确定一个比其余元素的重要性比较有把握的元素，假设决策者任务对判断 r_{11} r_{12} … r_{1n} 有把握。

（2）用 R 的第一行元素减去第二行元素，若所得到的元素均为常数，则不需要调整第二行元素，否则对第二行元素进行调整直到第一行减第二行元素之差为常数为止。

（3）用 R 的第一行元素减去第三行元素，若所得到的元素均为常数，则不

需要调整第三行元素，否则对第三行元素进行调整直到第一行减第三行元素之差为常数为止。

（4）按上面的步骤继续下去，直至第一行元素减第 n 行对应元素之差为常数为止。。

步骤 4：单层次排序

所谓单层次排序是指根据模糊一致矩阵计算对于上一层某因素而言，本层次与之有联系的因素的重要性排序，即确定权重，模糊互补矩阵求元素 c_1、c_2...c_n 的权重值 ω_1、ω_2、...ω_n，当模糊互补判断矩阵具有一致性时，权重公式为：

$$\omega_i = \frac{1}{n} - \frac{1}{2a} + \frac{1}{na}\sum_{j=1}^{n} r_{jk}$$，其中 n 为 R 的阶数，$a = \frac{n-1}{2}$

步骤 5：层次总排序

通过计算每一组元素对其上层中某一元素的权重向量，我们最终是要得到各元素对总目标的相对权重，特别是最低层因素指标对总目标的合成权重。假定第 *k-1* 层上第 n_{k-1} 个元素相对于总目标的权重向量为 ω_{k-1}^1，第 *k* 层元素对 *k-1* 层上第 *j* 个元素权重向量为：$\omega_j^{(k)} = (\omega_{1j}^{(k)}, \omega_{2j}^{(k)}, \cdots, \omega_{nj}^{(k)})^T$，那么 $\omega_k^1 = \omega_k^{k-1} * \omega_{k-1}^1$，则综合各层权重矩阵，可得到 n 层递阶结构的指标因素层相对于总目标的合成权重为：

$$\omega_n^1 = \prod_{2}^{i=n} \omega_i^{i-1} = \omega_n^{n-1} * \omega_{n-1}^{n-2} \cdots \omega_3^2 * \omega_2^1 。$$

步骤 6：确定指标因素集。

确定各供应商指标体系中各指标的初始值，并且经过无量纲处理后建立指标因素集 $X = (x_1 x_2 \cdots x_m)^T$.

步骤 7：计算评价集。

在通过上述模糊层次分析法获得权重集 W，又确定了因素集 X 后，我们就可以计算评价结果，目标层评价集 Y 与因素集、权重集之间的关系为：

$$Y = (\omega_n^1)^T X = \sum_{i=1}^{m} \omega_i x_i$$

我们需要强调的是，如果所选的要素不合理，其含义混淆不清，或要素间的关系不正确，都会降低 AHP 的结果质量，甚至导致 AHP 决策失败。为了保证决

策的正确性，在分解简化问题时把握主要因素，注意相比较元素之间的强度关系，相差太悬殊的要素不能在同一层次比较。为了加快系统决策速度，我们可以在具体实施过程中体现层次分析法的基础上加以简化。

4.6 MAWFMS 模型的性能分析

1 可靠性

MAWFMS 模型采用数据集中存储，这样保存可以确保数据的一致性。在公共数据需要使用的时候，由资源 Agent 统一调度，有效避免资源的冲突。当然，这样做也会带来负面影响，就是数据处理的效率将成为系统性能提升的瓶颈，但正确的得到数据是最为关键的。

在工作流程中的任务分配中，通过决策 Agent、任务 Agent、用户 Agent 以及资源 Agent 的交互协作，决策 Agent 收集所需数据，使用 AHP 进行计算，寻找到最为合适的任务完成者，这将有效的提高企业的经营过程。

在工作流程的执行中，每个流程实例配有一个任务 Agent 负责其运行控制。工作流引擎在执行过程中，任务 Agent 携带工作流程中所需要的数据与工作流引擎交互完成任务。通过任务 Agent 的自我监控和监控 Agent 对工作流引擎的监控，确保系统正常运行，及时处理动态环境中不可预知的错误。

最后，由监控 Agent 对整个模型系统运行的监控生成的日志，可以方便的查看系统运行情况，找出存在的问题，优化系统中存在的缺陷。当然，我们也可以分析并优化具体的企业业务流程。

2 可扩展性

MAWFMS 模型中，在企业具体业务应用和工作流引擎之间，搭建了一座桥梁——多 Agent 集合。其中各个 Agent 将它们独自承担一些业务从系统中分离出来，实现业务分离，从而降低了系统的耦合性。各个 Agent 都有与外界进行通信的模块，如果需要设置新的 Agent，那么要将它的通信模块设置为一致的即可，也即多 Agent 集合的扩展性是无限的。

在工作流引擎的扩展层面上，MAWFMS 模型专门提供了扩展 Agent，它的主要功能就是为了加强 MAWFMS 模型的可扩展性，提供一组可供其它应用系统集成的接口。在工作流参考模型提供的五类接口的基础上，扩展 Agent 为外界的

扩展提供支持。

3 实用性

MAWFMS 模型中，每个业务流程由一个任务 Agent 负责，并且通过任务 Agent 携带工作流引擎所需要的数据控制工作流引擎的执行，推进业务流程的执行。这样工作流引擎在工作流执行过程中被动的解释执行过程定义的问题将有效解决。利用任务 Agent 可以在一些特殊的需要动态修改业务流程的工作流管理系统中提高灵活性，因为工作流引擎是通过任务 Agent 驱动的。

在 CSCW（Computer Supported Cooperative Work）中，我们强调用户参与协作过程的主动性。不同企业的员工日程安排自由度不同，在 MAWFMS 模型中，用户通过为自己的用户 Agent 设定一定的个性化数据，可以使用户 Agent 协助用户自主的安排日程，提高用户在系统中的主动性。

4.7 本章小结

本章在深入学习工作流与 Agent 的相关技术基础上，提出一种基于 MAWFMS 的模型，简称 MAWFMS 模型。从阐述 MAWFMS 模型的设计思想开始，结合工作流管理联盟给出的工作流管理系统体系结构，给出了 MAWFMS 模型的整体体系结构。接着详细设计了工作流执行服务、Agent 以及系统的智能决策，同时系统智能决策的设计也是本文的一个创新点。最后，研究分析了 MAWFMS 模型的性能。

第五章 基于 MAWFMS 模型的绩效考核系统的设计

在上一章节中，本文提出了一种基于 MAWFMS 模型。本章将 MAWFMS 模型应用到上海某信息科技有限公司的实际业务系统中，详细介绍基于 MAWFMS 模型的应用实例——员工绩效考核系统的设计。

5.1 需求分析

5.1.1 总体目标

由于上海某信息科技有限公司的员工队伍不断壮大，项目任务日益繁重，为了更好的对员工工作进行有效考核，希望投入使用一个绩效考核系统，全面考核员工任务完成情况，并对员工进行高效的任务分配。设计一个基于 B/S 的 Web 系统，管理角色的用户可以为手下员工安排任务，查看自己手下员工的任务完成情况，了解各个项目的进度情况，统计各个员工的工作实效；员工角色的用户通过绩效考核系统，获悉自己需要完成的任务，每日通过填写进度报告向上级汇报工作情况。

在该系统中，如果项目经理对员工的在项目开发中的特长不甚了解的话，那么他很难将项目开发中的各项任务安排到最合适的员工手中。目前公司管理层已经意识到现有的基于人人交互的任务安排存在一定的缺陷，影响整个公司的项目实施效率，因此希望在该绩效考核系统设置任务分配模块，进行集中化改造，以技术手段来辅助控制任务分配的流程，做到将各项任务分配到最合适的人选中。基于此，本文对绩效考核系统的目标说明如下：

1 采用 J2EE 技术开发绩效考核系统，重点开发任务分配子系统，使其成为一个独立系统集成在原系统中。

2 绩效考核系统中，员工每日汇报工作进度以便项目经理查阅，掌握项目进度以及员工工作情况。

3 任务分配模块中任务分配的模式将由人人交互转变为“经理提交任务给系统、系统选择最合适的员工、员工确认任务、经理确认开始”的工作流程模式。

4 系统在选择员工时，具有一定的智能性，根据一定的参数计算得出，从而改变员工参与项目的被动性。

5 绩效考核系统中，需要有用户管理等系统管理模块以及基于角色访问控制的机制确保系统安全性。

5.1.2 需求分析

1 功能的划分

本章重点说明本绩效考核系统的核心功能——业务操作模块。它的主要使用者是项目参与人员，主要功能是实现公司所洽项目开发中任务的分配与完成。具体功能包括：

(1)在项目开发中，项目经理将一项任务的参数在系统中发布，包括项目简述、所用语言、逻辑难度等参数。

(2)系统根据任务描述，收集数据并进行计算，得出最合适的人选并通知该用户。

(3)用户需要有一个助手性质的软件，在用户的设置下成为其秘书，可以向系统提供用户的个性化数据，以供系统选择用户时参考计算，同时在系统分配任务给该用户时，可以根据用户的设置直接接受任务，或及时提示以便用户决定是否接受；如果该用户不接受，则系统根据合适度排名顺序向下通知，直到任务完成分配。

(4)在任务分配完成后，系统通知项目经理，经项目经理确认后，该项任务正式开始；如果项目经理不确认，系统通知本次选中的员工并接着向下选择用户完成以上步骤。

(5)在任务向系统提交时，项目经理也可以指定项目完成人，这种情况下系统将不进行计算选择，用户也只是接到通知，不可取消任务。

(6)任务执行过程中，用户每日填写进度汇报。当任务完成时，流程将任务转向测试人员，测试人员对任务进行测试，测试通过则通知项目经理及用户项目完成；未通过则返回用户进一步完善任务。

此外，绩效考核系统还包括员工的每日汇报模块、系统安全认证模块以及系统用户管理、角色管理等管理模块。系统通过这些管理模块可以对系统运行的参

数项进行增、删、改等维护操作。

2 性能的规定

(1)正确性

该系统在任务分配选择人员的计算中，需要按照项目经理提供的任务描述数据、员工自我定义的个性化数据等进行精确的计算。

(2)灵活性

该系统在任务分配时，项目经理可选择由系统分配或由其自己选派；同样在员工接受任务时，可以由助手接受或助手提醒员工接受的模式。

(3)易用性

该系统的开发，是为了减轻项目经理、公司职员在项目任务分配时的工作复杂度，因此系统在使用过程中应有较好的用户体验度。

(4)可扩展性

鉴于计算机技术的飞速发展，面对未来不可预知的应用需求，该系统需要有良好的扩展能力。

(5)时间特性

该系统在任务分配时，各个环节需要有一个计时器，超过一定的时限将视为发生异常，重新开始。

(6)输入输出

为了减轻项目经理、公司职员在项目任务分配时的工作复杂度，系统需要最大限度地减少输入项，输入项要设置合理、清晰的展现给使用者。

(7)出错处理

该系统需要有一个统一且强大的错误处理机制，在错误发生时及时做出相应的处理，确保系统正常运行。

(8)数据管理

该系统需要有一个统一的数据管理模块，通过数据的集中管理，达到避免数据脏读、数据冲突等问题的目的。

5.1.3 业务流程设计

根据对绩效考核系统进行的需求分析，我们得出的绩效考核系统业务操作流程如图 5.1 所示。

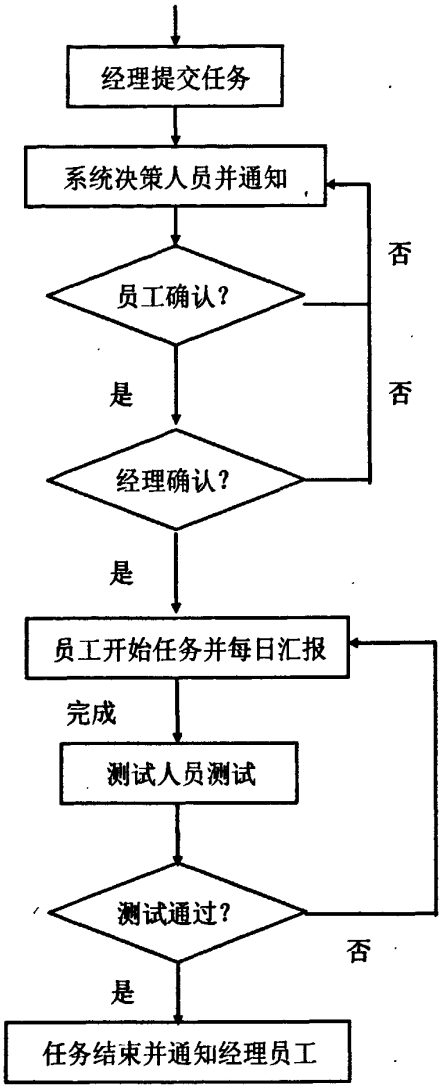

图 5.1 绩效考核系统业务操作流程图

5.2 系统结构设计

根据本章第一节的叙述，本文采用模块化的设计方式，对系统进行整体结构设计。如图 5.2 所示，该系统的整体结构由 Web 浏览器、Web 应用程序、工作流管理系统、数据层四个层面组成。

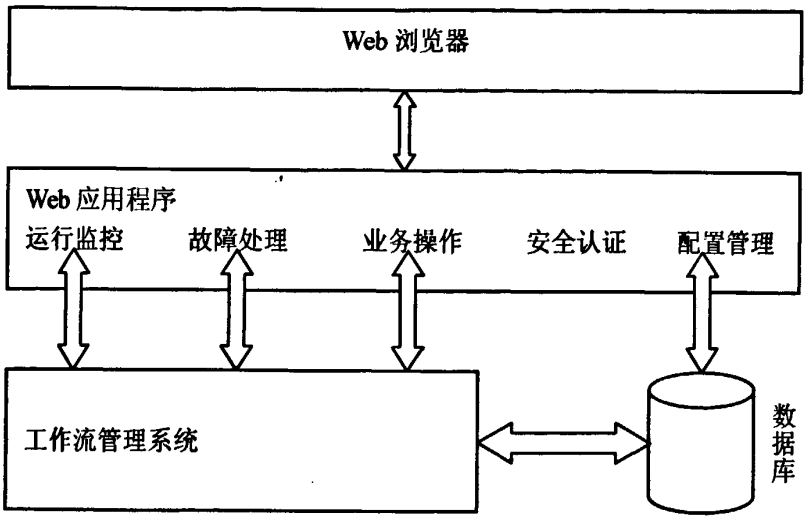

图 5.2 系统结构图

1 用户通过客户端，即 Web 浏览器，访问 Web 应用程序，通过 Web 应用程序与工作流执行服务及数据库的交互，完成相应的业务操作。

2 Web 应用程序通过 Http 与客户端 Web 浏览器进行通信，向其提供服务。Web 应用程序主要划分为四个功能模块，包括运行监控、故障处理、业务操作和配置管理。

（1）运行监控：通过与工作流执行服务的交互，对系统的业务完成情况及业务流程的执行状态进行监控查看。

（2）故障处理：对系统可能发生的错误进行统一及时的处理，并生成相关的日志提交给管理员。

（3）业务操作：通过与工作流执行服务的交互，按照相关流程完成本系统的关键业务功能，即任务分配与完成功能。

（4）安全认证：系统实现访问控制的模块，防止非法用户进入系统以及合法用户做出不符合权限的非法操作，确保系统的安全性。

（5） 配置管理：也是系统的关键模块，它将维护系统的一些关键数据，包括用户信息、角色信息、权限信息、任务个性化数据信息、用户个性化数据信息等，它通过直接与数据库进行读写操作完成。

3 工作流管理系统定义工作流，包括具体的活动、规则等，这些定义是同时被人以及电脑所“理解”的，遵循定义创建和运行实际的工作流。监察、控制、管理运行中的业务（工作流），例如任务、工作量与进度的检查、平衡等。根据

实际需要，工作流管理系统将按照 MAWFMS 模型进行构建。

5.3 系统 Agent 的设计

根据系统需求分析，结合上一章我们提出的 MAWFMS 模型中关于 Agent 的设计，本系统 Agent 分为六类，具体设计如下：

1 用户 Agent：作为用户的助手，用来代理系统用户，维护用户的相关信息，并根据用户角色的不同来提供用户的个性化数据。此外，作为工作流程中的主要参与者，与任务 Agent 进行交互协作，完全或部分帮助用户完成一定的操作，最终完成整个工作流程。该 Agent 运行在各个用户的客户机上，构建分布式应用。

2 任务 Agent：当启动一个工作流程时，相应地创建一个任务 Agent。任务 Agent 描述工作内容，提供工作数据以及一些其他资源。它的主要功能就是控制过程实例的执行、活动的调度、工作流控制数据的维护等。在系统中，任务 Agent 将任务描述提供给决策 Agent，并按照决策 Agent 选择好的员工进行通知。

3 资源 Agent：作为绩效考核系统的大管家，资源 Agent 负责管理系统中所要使用的数据、设备等资源，针对不同类型的资源划分为不同类型的资源 Agent。当其他 Agent 需要资源时，向资源 Agent 发出请求，资源 Agent 获取所需数据再打包送回。在绩效考核系统中引入资源 Agent 的另外一个重要因素就是利用资源 Agent 来解决系统资源冲突问题，做到资源的统一管理、统一分配。

4 决策 Agent：作为绩效考核系统中的组织部长，决策 Agent 负责绩效考核系统中任务的分配，它通过与用户 Agent、任务 Agent、资源 Agent 的交互，获取决策所需要的数据，采用层次分析法，选择出该项任务最合适的用户。决策 Agent 也是绩效考核系统中智能决策的关键组成部分。

5 监控 Agent：作为绩效考核系统中的监督者，监控 Agent 负责监控工作流引擎中各个流程实例的执行情况，并且在必要的时候，将信息提供给用户。监控 Agent 在应用服务启动的时随之启动并且保持持续工作。系统的运行监控模块主要信息来源就是监控 Agent。

6 扩展 Agent：为了加强绩效考核系统的可扩展性，扩展 Agent 提供一组可供其它应用系统集成的接口。

5.4 系统设计

5.4.1 工作流引擎的设计

所谓工作流引擎，是指工作流作为应用系统的一部分，根据角色、分工和条件的不同，为各应用系统提供对其具有决定作用的信息传递路由、内容等级等核心解决方案。例如，开发一个系统最为关键的部分不是系统的界面，也不是和数据库之间的信息交换，而是如何根据业务逻辑开发出符合实际需要的程序逻辑并确保其稳定性、易维护性（模块化和结构化）和弹性（容易根据实际业务逻辑的变化做出程序上的变动，例如决策权的改变、组织结构的变动和由于业务方向的变化产生的全新业务逻辑等等）。工作流引擎解决的就是这个问题：如果应用程序缺乏强大的逻辑层，势必变得容易出错（信息的路由错误、死循环等）。

目前市面上成熟的工作流引擎有 Shark、JBPM、OSWorkflow、WF 等，其中 Shark1.0、JBPM、OSWorkflow 等为开源工作流引擎，使用免费且有大量参考资料。为了提高系统开发效率，本系统中工作流引擎采用开源工作流引擎 OSWorkflow。OSWorkflow 是 opensymphony 组织开发的一套由 Java 写成的 workflow engine，目前的版本是 2.8。它有一套完整的 API 处理流程，并藉由其本身自行定义的 XML 来表示工作流程，可搭配多种 Database 作为存取的媒介。OSWorkflow 几乎提供了所有用户在实际流程定义中可能用到的工作流构成元素，如：环节（step）、条件（conditions）、循环（loops）、分支（spilts）、合并（joins）、角色（roles）等等。

5.4.2 Agent 通信模块的设计

不同功能的 Agent 的具体实现方式有所不同，比如任务 Agent、决策 Agent、资源 Agent、监控 Agent、扩展 Agent 寄存在 Web 服务器中，用户 Agent 则分布于各个客户机中。但是他们有相同的组成结构，他们之间的通信也具有相同的机制——基于 KQML 的多 Agent 间通信。本系统在实现通信模块时，采用可扩展标记语言 XML 描述 KQML 通信的消息，通过使用 JDK 中的 SAX 包中的相关 API 函数对消息进行解析。

在用户、应用程序、智能体之间可能有丰富的数据需要交换，数据的内容和

格式也可能较为复杂，给智能体语义表达造成很大困难。KQML 内容层规定了消息的内容，但不关心被交换的信息的格式标准化问题，也不精确指定这些信息如何被移动。从理论上讲，KQML 可以携带任何表达性语言，包括用 ASCII 串和二进制符号表示的语言。因此，本系统设计将 XML 嵌入 KQML 中的内容层，用 XML 来表达内容的语义信息，丰富 KQML 语义表达，方便地实现表达复杂语义。下图 5.3 展示了基于 XML+KQML 的多 Agent 通信的体系结构。

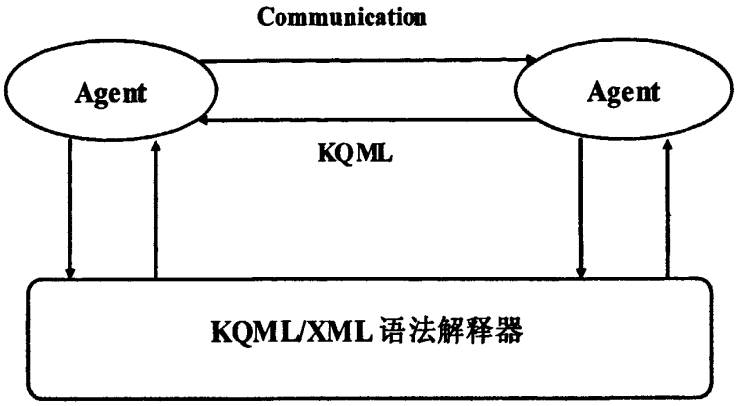

图 5.3 基于 XML+KQML 的多 Agent 通信

通过使用 XML，可以在协作的智能体之间交换结构化的内容信息。因为 XML 是一种元语言，组织者可以开发出符合 DTD 定义的新的属性和元素。XML 的语法是基于标签和属性的，它同时提供了一套丰富的语法来建立事务，允许智能体之间以一种平台无关的方式进行通信。因此，将 XML 集成到 KQML 中会增强智能体通信的语义信息，使基于 KQML 的智能体之间的通信更加灵活，而且 XML 可以在多种平台使用，可以用多种工具进行解释。因为文档的结构是相容的，所以解释它们的语法分析器可以以较低的费用建立。KQML 的内容层并不是固定格式，允许用不同的语言来表示。用于表示智能体间知识共享的语言必须能够表达一种行为、语句、元素或对象。内容层所用的语言在 Language 域指定。通常其句法限制在 ASCII 码或 Comma Lisp 通用波兰前缀表示法。如果将 XML 集成到 KQML 中，XML 元素类型的含义在文档类型定义中说明，则基于 XML 的内容较容易实现语义分析。

5.4.3 安全认证模块的设计

通过引入用户、角色、权限三个实体概念的 RBAC 系统已逐步成为中小企业实现访问控制的首选方案。但是，目前很多权限验证模块的实现放在各个实现核心业务的页面中，在页面每次加载时，通过执行权限验证代码实现对用户的访问控制。这种方法导致代码的大量冗余，尤其是在权限验证模块发生变动的情况下，需要对各个页面的代码进行修改，可维护性与可扩展性非常低。本系统的设计以 RBAC 的基本思想为基础，牢牢抓住 AOP 的核心思想，通过将系统的权限验证模块封装成单独的模块，与核心业务模块解耦合，在调用核心业务模块时借助动态 AOP 框架将权限验证模块动态的织入，从而实现一种通用的、可维护的、易扩展的权限验证模块。

依照上述设计思想，基于 AOP 的 RBAC 模块对任意业务模块的访问处理流程如图 5.4。

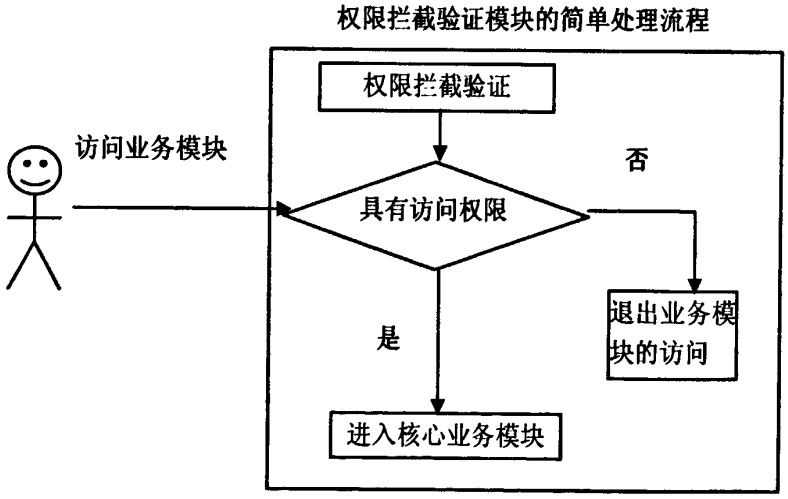

图 5.4 安全认证模块处理流程图

基于 AOP 的 RBAC 模块主要由以下 4 个子模块构成：

(1)用户管理子模块：维护用户的基本信息，并通过将相应角色赋予用户，实现用户与角色的动态关联。其中一个用户可以被赋予多个角色，一个角色可以赋予多个用户，二者是一种多对多的关系。

(2)角色管理子模块：维护角色的基本信息，并通过将相应权限赋予角色，实现角色与权限的动态关联。其中一个角色可以被赋予多个权限，一个权限可以赋

予多个角色，二者是一种多对多的关系。

(3)权限管理子模块：维护权限的基本信息，并通过将访问资源的入口赋予权限，实现权限与资源的动态关联。其中一个权限只能对应一个入口，二者是一种严格一对一的关系。

(4)权限验证子模块：本模块为系统的核心所在。当用户访问系统资源时，进行拦截，并通过对用户是否已经登录被访问资源是否需要验证，以及用户是否拥有访问该资源的权限的判断，实现权限验证功能。

5.5 本章小结

在上一章提出 MAWFMS 模型设计的基础上，本章详细介绍了一个基于 MAWFMS 模型的应用实例——员工绩效考核系统的设计。首先说明了该系统的总体目标以及需求规定，从需求出发设计了该系统的业务流程，然后根据需求进行了系统的总体结构设计以及系统 Agent 的设计。在系统设计一节，详细阐述了该系统在参考 MAWFMS 模型基础上进行了系统工作流引擎的设计、系统 Agent 通信模块的设计以及系统安全认证模块的设计。

第六章 基于 MAWFMS 模型的绩效考核系统的实现

在上一章节介绍基于 MAWFMS 模型的绩效考核系统的设计基础上，本章介绍该绩效考核系统重要模块的实现环节，并通过图文并茂的形式介绍系统的运行，验证 MAWFMS 模型的可用性。

6.1 OSWorkflow 源码的分析与改进

6.1.1 OSWorkflow 存在的不足

1、OSWorkflow目前推出的版本较少、升级速度慢、网络上有价值的资料也相对有限，使用者需要有较高的代码开发经验才可以较快的上手。

2、OSWorkflow可以支持多种数据源持久化配置方法，但 OSWorkflow不支持目前广泛流行的structs+hibernate+spring框架的配置方式。

3、OSWorkflow在权限认证方面存在一定的不足。OSWorkflow按组划分action的所有权，组即为权限的最小单位，这在实际应用中可能会碰到权限划分不够细致的问题。并且虽然对action的所属组进行判断，但在流程流转过程中，它并不对当前使用者的组别进行判断从而对action进行操作权限控制，而是全部显示，这就造成流程操作权限的混乱。

4、OSWorkflow自带数据库表格中对用户数据存储的表格os_propertyentry表的结构分布为纵向的，这样的设计方便用户可以根据自己的需要定义列的属性，同时增加和删除列也非常方便，不需要改变表的结构。但这同时带来一个问题，那就是在该系统需要进行查询的时候，需要使用较为复杂的SQL查询语句，同时，多次自连接会导致查询速度较慢。

5、OSWorkflow没有好的可视化工具来开发流程，这就意味着要手工书写和定义这些XML流程描述文件。

6、如果开发者使用SpringHibernateStore的存储方式，在前台jsp页面中的用户管理模块将会抛出异常，原因是OSUser中引用了jdbc propertyset模块，但hibernate propertyset中所用的os_propertyentry表和jdbc中所用的os_propertyentry

表有很大的不同，导致如ITEM_KEY找不到这样的异常。

7、当用户创建用户和群组时，用户名称和群组名称不能重名，一旦重名便抛出com.opensymphony.user.DuplicateEntityException: group test already exists.这样不仅需要用户id、组id不能重复，且所有id中任何两个都不能重复，给用户带来很大的不便。

8、OSWorkflow的核心API在JDK1.3及以上环境中即可运行，但是它附带的GUI设计器则需要JDK1.4及以上版本才能工作。

6.1.2 OSWorkflow 源码的改进

1、增加 OSWorkflow 对 spring 和 hibernate 的支持，修改了原有的配置方式。

官方提供的 OSWorkflow 最新版本 OSWorkflow2.8.0 不支持 structs+spring+hibernate 持久化方式，对源代码进行改进后，支持该功能。图 6.1 为修改的 OSWorkflow 源码目录：

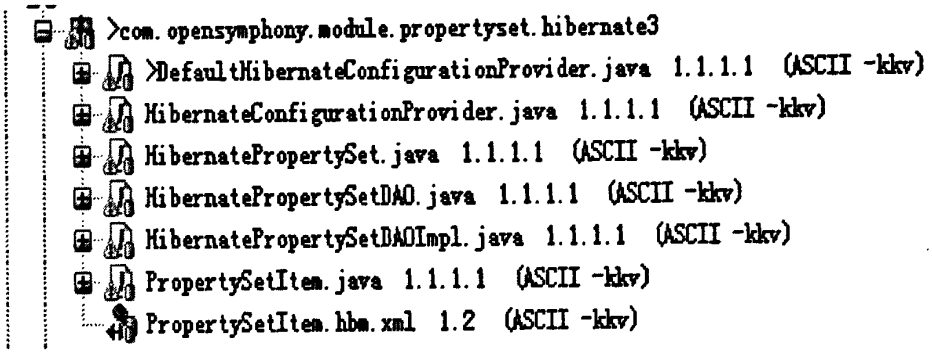

图 6.1 升级的 OSWorkflow 接口目录

2、修改 OSWorkflow 的持久化方法，图 6.2 为分离 OSWorkflow 自带表中的实体持久化的代码目录：

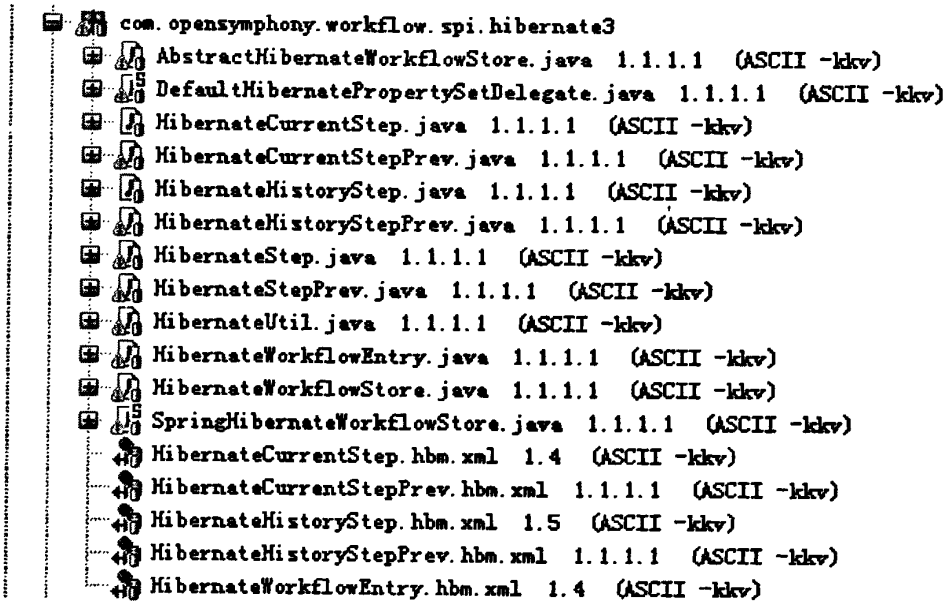

图 6.2 分离 OSWorkflow 自带表中的实体持久化目录

3、增加了 OSWorkflow 对权限角色的管理。

OSWorkflow2.8.0 对 action 的操作有组的权限控制，考虑到系统的需求，用户可能携带多种身份，需要对源代码进行升级以适应需求。升级的代码目录如图 6.3 所示：

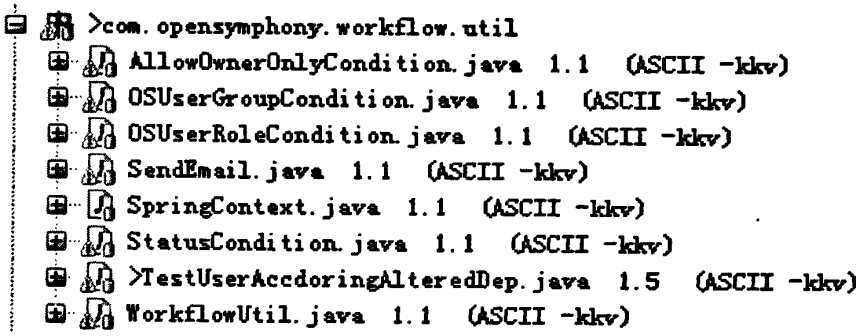

图 6.3 升级包 com.opensymphony.workflow.util 类的目录

升级后的 OSWorkflow 在配置流程时可具有多重身份权限控制，如下面代码所示：

```
<condition type="class">
    <arg name="class.name">
    com.opensymphony.workflow.util.OSUserRoleCondition</arg>
      <arg name="group">BG</arg>
      <arg name="role1">manager</arg>（此处为添加的角色）
</condition>
```

代码说明：<arg name=" ">.....</arg>为流程限制条件，升级前只支持<arg name="group"></arg>的权限管理，升级后可以添加<arg name="role1"></arg>的权限。

4、增加了 OSWorkflow 对流程步骤验证限制条件的限制。

OSWorkflow在流程流转时，并不根据当前action的权限拥有者来分派action。根据需求，该管理系统必须是一个点对点的流程、数据传输系统，所以需要对源代码进行修改来控制action的操作权限。新建类MyWorkflow 继承 Workflow类扩展 AbstractWorkflow 类。扩展代码包如图6.4所示：

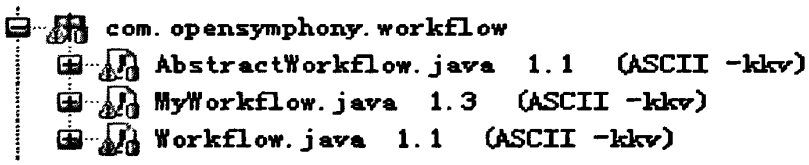

图 6.4 升级包 com.opensymphony.workflow 类的目录

5、利用 spring 框架，改进了 OSWorkflow 对流程和数据事务的支持。

原 OSWorkflow 可以对流程进行事务控制，但不能对与流程无关的数据操作进行事务控制，需求中除了需要处理流程，还涉及到与流程无关的数据操作，所以在整个 spring 框架内修改数据操作方式，实现流程和数据的事务支持，在 spring 添加配置代码如下：

```
<beanid="transactionInterceptor"class="org.springframework.transaction.interceptor.
TransactionInterceptor">
<property name="transactionManager" ref="transactionManager" />
<property name="transactionAttributes">
<props>
 <prop key="insert*">
PROPAGATION_REQUIRED,ISOLATION_READ_COMMITTED,-Exception</pr
op>
 <prop key="update*">
   PROPAGATION_REQUIRED,ISOLATION_READ_COMMITTED,-Exception
</prop>
 <prop key="save*">
PROPAGATION_REQUIRED,ISOLATION_READ_COMMITTED,-Exception</pr
```

```
op>
 <prop key="set*">
PROPAGATION_REQUIRED,ISOLATION_READ_COMMITTED,-Exception</pr
op>
 <prop key="add*">
    PROPAGATION_REQUIRED,ISOLATION_READ_COMMITTED,-Exception
</prop>
 <prop key="find*">PROPAGATION_REQUIRED</prop>
 <prop key="get*">PROPAGATION_REQUIRED</prop>
 <prop key="*">PROPAGATION_REQUIRED,-Exception</prop>
     </props></property></bean>
```

代码说明：将事务添加入 spring 框架，从而从系统全局的角度对系统数据进行安全性保护。例如：<prop key="insert*">为所有的 insert 操作进行事务管理，如果出现操作异常则 rollback 数据，抛出 Exception。

6.2 系统实现

6.2.1 流程执行模块

流程执行过程主要由任务 Agent、资源 Agent、用户 Agent 之间的交互协作来完成的。系统管理员、用户登录系统，在 Web 层面上请求实例化一个流程，Web 服务器收到请求之后，调用 CreateProcess()方法，实例化流程。同时在 CreateProcess()方法中，实例化一个跟随该流程的任务 Agent。任务 Agent 描述工作内容，提供工作数据以及一些其他资源。它的主要功能就是控制过程实例的执行、活动的调度、工作流控制数据的维护等。在任务 Agent 创建之后，该流程转由任务 Agent 控制，通过与资源 Agent 交互获取流程执行所需的各种资源，与用户 Agent 交互推动流程的转向。在流程执行过程中，任务 Agent 发挥了重要的作用，以下是任务 Agent 类的核心组成部分。

```
public void OnInit(Object init){
//初始化任务 Agent，进行流程注册以及相关数据的存储
    try{
```

```
            this.registry(init.getProcessId());//流程注册
            this.sendMessage("ResourceAgent",this);
            ...//将流程执行所需要的相关数据存放于数据库中
            this.startTask(init.getWorkflow(),init.getProcessId());
        }
        Catch(Exception e){
            e.printStackTrace();
            throw new InnerException(e.getMessage());
        }
    }
    public boolean doTask(Task task){
    //在流程执行过程中主动与其他 Agent 交互以推动流程的执行
        if(task.equalKind("start Task")){
            this.sendMessage("DecAgent",this);
            return ture;
        }
        else if(task.equalKind("Inform User")){
            this.sendMessage("UserAgent",this);
            return ture;
        }
        ...
        else
            return false;
    }
    public boolean handleMessage(Message message){
    //该方法为任务 Agent 接受消息后的分发处理函数
        if(message.equalKind("relative data")){
            this.setData(message);
            return true;
        }
```

```
    ...
    else
        return false;
}
public Step getCurrentStep(){
//获取流程当前的步骤
    Map mapState = osdao.getStepsByProcessId(this.getProcessId());
    if(mapState!=null){
        Object[] a =mapState.keySet().toArray();
        for(int i=0;i<a.length;i++)
            idList.add(a[i]);
        Step step = mwf.getAvailableSteps(idList, map);
    return step;
}
```

6.2.2 Agent 通信模块

扩展支持 XML 的 KQML 通信原语实现步骤如下：

1 智能体利用 KQML 语法解析器对所收到的 KQML 原语消息进行解析，提取相应参数内容。

2 当关键字 Language 的值为 XML 时，将 KQML 消息的内容层信息转换形成 XML 文件，提交给 XML 语法解析器。

3 XML 语法解析器采用 SAX（ Simple API for XML）编程接口基于事件方式，根据 DTD 定义实现该 XML 文件的解析，解析结果以特定格式字符串存放。

4 智能体根据协商协议对 KQML 所述行为及内容层解析结果进行相应的处理。

在实现扩展支持 XML 的 KQML 原语过程中，XML 语法解析器是其核心部分。JAVA 提供了两种不同类型的编程接口（XML API）用于提供开发者建立自己的 XML 解析器：一种为基于树结构的 API，主要建立文档内容的树状结构，形成文档对象模型即 DOM，提供解析；另一种为基于事件的 API，SAX 属于基于事件的 API，SAX 解析器处理事件的方式与用户界面程序处理事件的方式类似。它使用回调的方式用解析器读取文档，当解析器发现标签时告知程序所发现

的标签。由于 SAX 可以解析任意大小的文件，而基于树结构方式仅能解析有限大小的文件，因此本文中的 XML 语法解析器采用 SAX 接口编程实现，采用基于事件方式对 XML 文档进行解析。XML 语法解析器主要由 XMLSaxParser 类实现，核心代码如下：

```
public XMLSaxParser(String filename){
    this. result = " " ;                    //存放解析结果
    this. firstelementname = " " ;          //存放第一个元素的名称
    this. first = 0;                        //存放计数
    try{
        XMLReader parser = XMLReaderFactory.createXMLReader( " org.
    apache. xerces. parsers. SAXParser");
        XMLSaxParser XMLSaxParserInstance = new XMLSax-Parser();
        parser. setContentHandler(XMLSaxParserInstance);
        parser. Parse(filename);
    }catch(IOException ioe){
        ioe. printStackTrace();
    }catch(SAXException saxe){
        saxe. printStackTrace();
    }
}
```

Agent 之间的通讯往往是由某一个 Agent 发起的，因此需要利用一种主动的通讯手段，而不是被动的等待请求。由于 XML 是以文本格式存在的，从而大大扩展了系统传送数据的手段。鉴于 XML 文档便于传输，甚至可以专门开发它的通讯协议，但这样的通讯成本相对提高。在这里本文使用简单邮件传输协议 SMTP 来作为传送 XML 文档的手段，Agent 发送和接受邮件队列，通过对发送者和发送时间等参数的处理来完成对 KQML 消息的确认。当然我们也可以采用 C/S 模式，发送者作为客户，而接收者作为服务器，这需要一个 Agent 同时拥有客户和服务者的双重身份。

6.2.3 智能决策模块

MAWFMS 模型的一个创新点就是在系统中引入了智能决策，它将为工作流过程中的每个活动选择最合适的用户进行任务分配。智能决策的引入，使 Agent 真正成为了人的“助理”，它将完全或者部分帮助用户完成一些操作。当然，Agent 只是一个软件实体，它的智能需要进行一定的设计，使其具有完成一定功能的智能分析。其中参与计算的各个指标以及权重的选取对智能决策的成败起着非常重要的作用。本系统各个指标及权重的选取实现动态可维护，管理员及用户可以根据需要对数据进行更新。

本文选取层次分析法作为系统智能决策的算法，决策 Agent 根据所获取的指标值与权重值进行计算，得出总目标值最大的一项即为系统做出的决策。以下介绍本系统层次分析法实现的核心代码。

构造对比矩阵的核心代码如下：

```
for(int i=0;i<args.length;i++)
    A[i][j] = 1;
for(int i=0;i<args.length;i++){
    storageData = (StorageDataType)Helper.getStorageData(id,i);
    data_1 = storageData.getData();
    for(int j=i+1;j<args.length;j++){
        storageData = (StorageDataType)Helper.getStorageData(id,j);
        data_2 = storageData.getData();
        if(data_1 > data_2){
            A[i][j]=2;
            A[j][i]=0;
        }
        else if(data_1 < data_2){
            A[i][j]=0;
            A[j][i]=2;
        }
        else{
```

```
                A[i][j]=1;
                A[j][i]=1;
            }
        }
    }
```

计算重要性排序指数以及间接判断矩阵的核心代码如下：

```
for(int i=0;i<args.length;i++)
    r[i]=0;
for(int i=0;i<args.length;i++)
    for(int j=0;j<args.length;j++)
        r[i]=r[i]+A[i][j];
max=min=r[0];
for(int i=0;i<count;i++){
    if(r[i]>max)
        max=r[i];
    if(r[i]<min)
        min=r[i];
}
b=(float)max/min;
for(int i=0;i<args.length;i++)
    for(int j=0;j<args.length;j++){
        if(r[i]>=r[j])
            B[i][j] = (float)(((r[i]-r[j])*(b-1))/(max-min+1));
        else
            B[i][j] = (float)(1/(float)(((r[i]-r[j])*(b-1))/(max-min+1)));
        if(max = min)
            B[i][j] = 1;
    }
```

6.2.4 安全认证模块

本模块的数据库表及其相互关系建模图如图 6.5 所示。

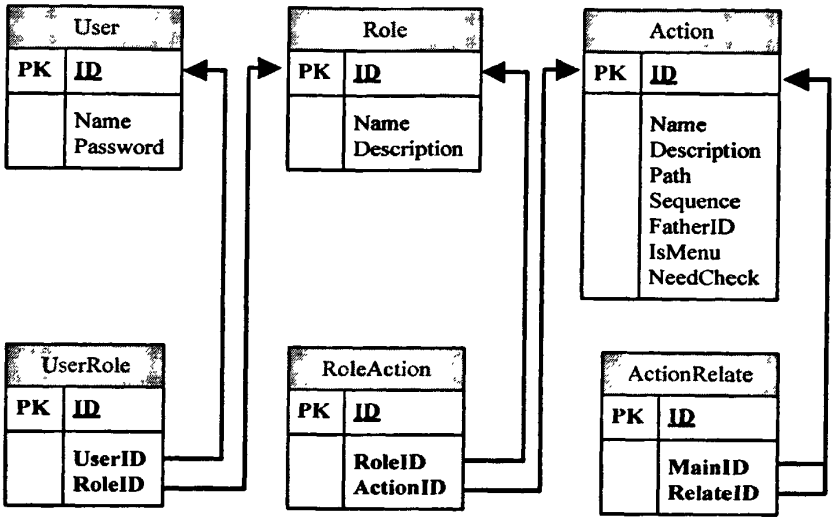

图 6.5 安全认证模块数据库关系建模图

Action表存放每个Web页面访问时触发的Action的详细信息，其中IsMenu记录此Action是否为菜单，如果是，则称之为菜单型Action；如果否，则称之为功能型Action。ActionRelate表存放各个菜单型Action与功能型Action相关联的信息，其中MainID字段中一定是Action表中IsMenu为1的元组的ID，RelateID字段中一定是Action表中IsMenu为0的元组的ID。一个菜单型Action关联若干个功能性Action，表示通过该菜单进入的模块中，可以通过相关联的功能性Action完成一定的业务逻辑。

本系统的实现采用目前流行的Struts框架，以Web页面访问中触发的各个Action（包括菜单型Action和功能型Action）为权限关注点。菜单型Action作为权限与角色直接关联，在用户登录时从会话中取得角色，根据角色动态地加载菜单并保存到会话中，同时与各个菜单相关联的功能性Action也加载到会话中。这样的效果是用户只要能看到访问某个业务模块的菜单，就拥有其中处理相关商业逻辑的权限，避免了看得到却用不了的尴尬局面，提升了用户体验。在进行权限验证方面，扩展Struts的RequestProcessor类，重写其中的权限判断方法processRoles()，并在web.xml文件中添加为controller属性，由此实现权限验证的拦截功能。这样在进入任何一个action类之前，首先进入RequestProcessor类进行权限验证。其中的逻辑是：获取将要提交的acton路径，如果为login或者logout则直接return true；接着判断用户是否登录，即判断session中是否有User的信息，

如果在这里session中没有相关信息，则return false，并抛出异常告知用户未登录，并转到登录页面；如果用户已经登录，则判断此action是否需要验证，如果不需要直接return true；最后判断用户是否拥有此action，如果有则return true，否则return false，并转到消息页面告知用户不具有此权限。

核心代码如下：

```
HttpSession session = request.getSession();
String strPath = mapping.getPath();
if(strPath.equals("/login") ||
strPath.equals("/logout"))
    return true;
User user = session.getAttribute(USER);
if(user == null)
//如果没有登录，则返回false
    return false;
if(User.needCheck(strPath) == false)
//如果该path不需要做权限验证，则返回true
    return true;
if(user.getAction(strPath) != null)
//如果用户拥有此权限，则返回true
    return true;
return false;
```

6.3 系统运行

本节我们将通过系统的运行来详细介绍本系统的业务流程以及 Agent 在其中的作用。系统在使用前，需要对进行决策判断的任务个性化数据项以及用户个性化数据项进行维护，包括增加、删除、修改。各个数据项还需要相应的指定数据类型。具体操作在图 6.6 展示的页面中进行。

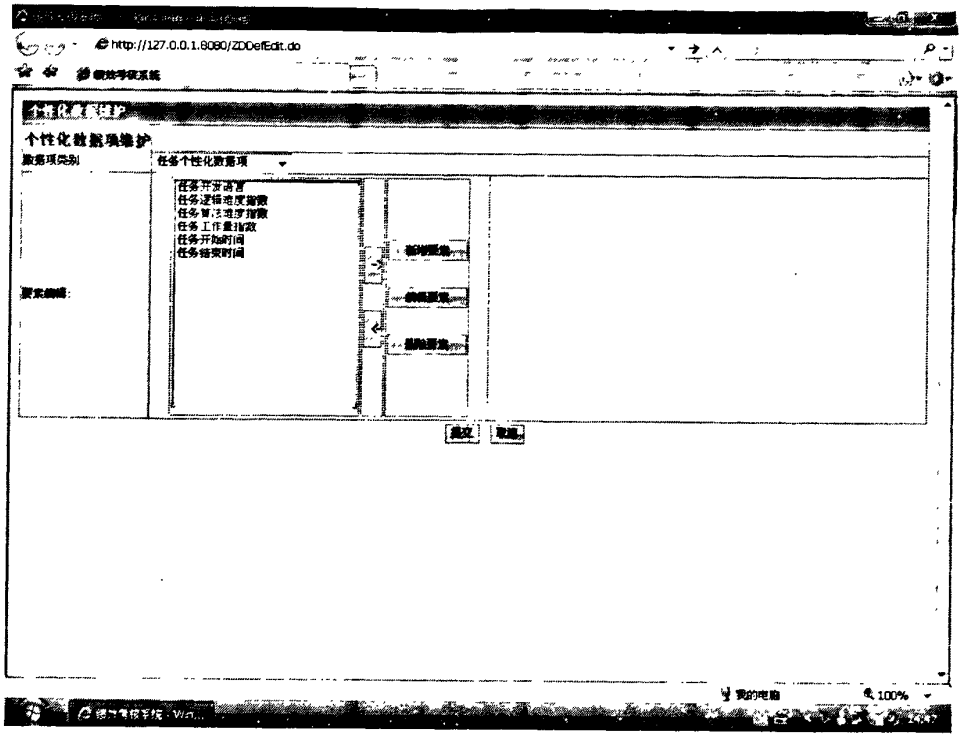

图 6.6 系统数据项维护页面

在对个性化数据项维护好之后，还需要对各个数据项的权重进行维护，在图 6.7 所示的页面中进行。

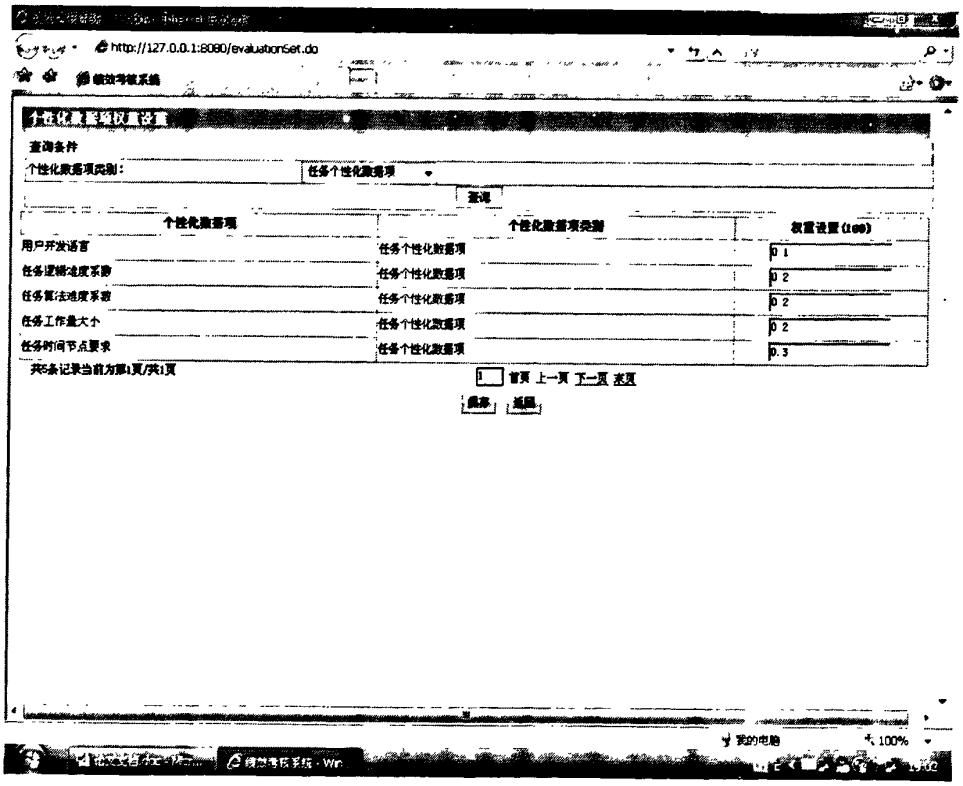

图 6.7 个性化数据项权重维护页面

在完成以上操作之后，系统可以开始正常运行，并对各个任务进行准确有效的分配。整个流程的第一步是项目经理在图 6.8 中所示的页面中，输入将要发布的任务的各个要素，然后点击提交。这时流程开始，任务 Agent 开始接管工作，并将描述任务的个性化数据收取并转送给决策 Agent，然后等待决策 Agent 为本次任务进行人员决策；决策 Agent 在接到任务 Agent 提出的请求之后，读取各个用户的个性化数据以及各个用户当前的工作状态数据，结合任务描述的个性化数据，使用层次分析法进行决策，并保存决策结果；决策 Agent 将当前最适合此项工作的员工 ID 回发给任务 Agent，由任务 Agent 根据 ID 通知相应的员工；用户 Agent 在接受到通知后，弹出通知，要求用户对当前任务是否接受进行确认，如图 6.9 所示。点击“是”，则该员工确认接受本次任务，点击“否”则任务 Agent 将要求决策 Agent 把下一个适合该任务的员工 ID 传入，继续上述过程，直到有一个用户接受任务或者无一员工接受任务通知项目经理，由项目经理采取工作流任务分配模式中的“推”模式将该任务指定分配给一个员工。这里需要指出的是，用户 Agent 可以通过用户配置，自动接受系统认为员工合适的任务，实现一个助手的角色。

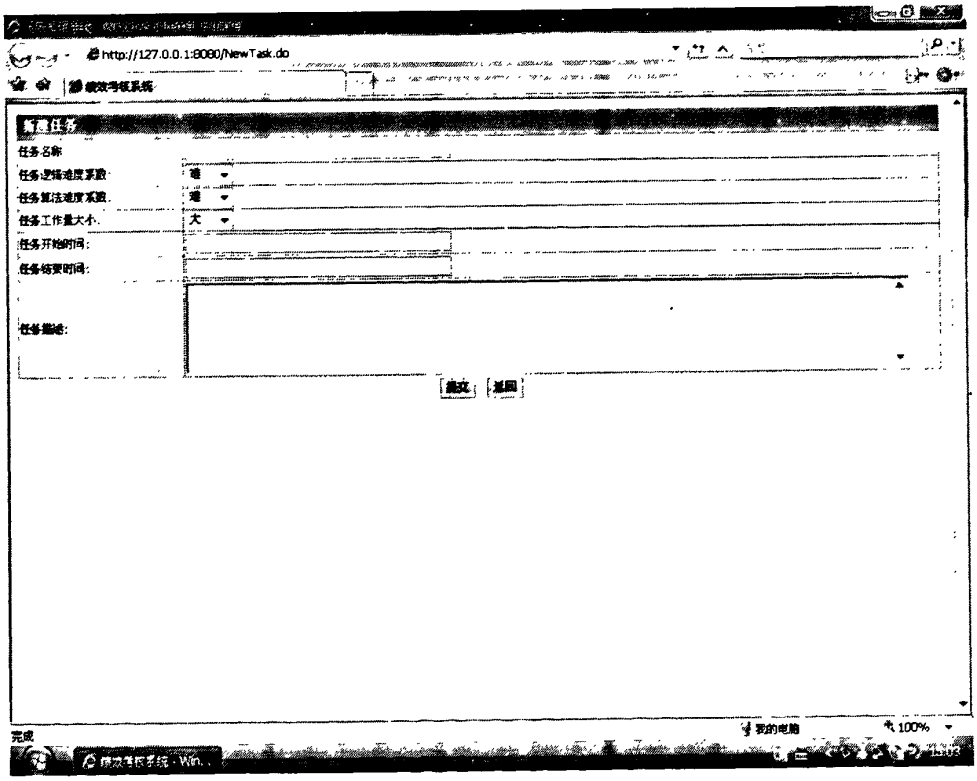

图 6.8 新建任务页面

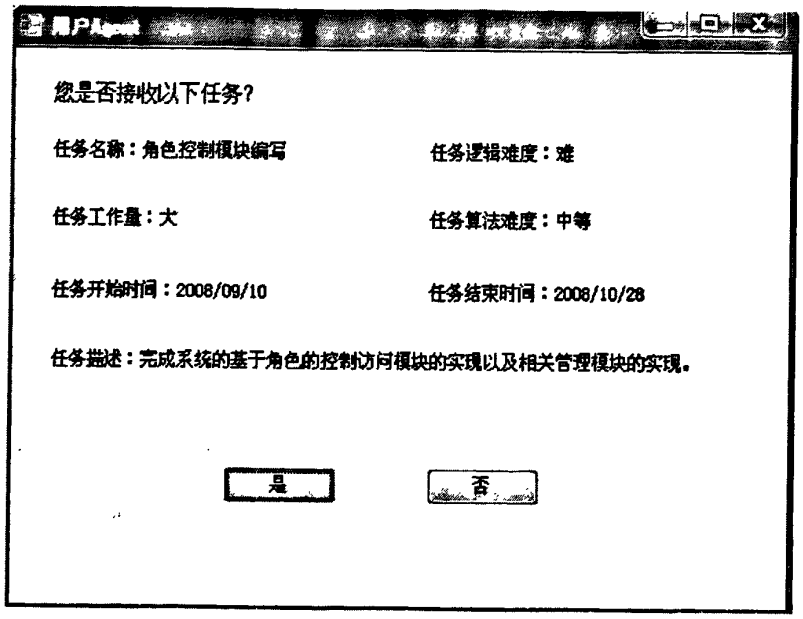

图 6.9 员工的用户 Agent 提示窗口

当员工通过用户 Agent 确认该任务后，任务 Agent 将告知项目经理请求确认，项目经理的用户 Agent 收到通知后，弹出窗口告知项目经理，如图 6.10 所示。点击“是”，则该任务进入员工的工作列表中；点击“否”，则通知任务 Agent 重新分配该任务，循环上述操作。

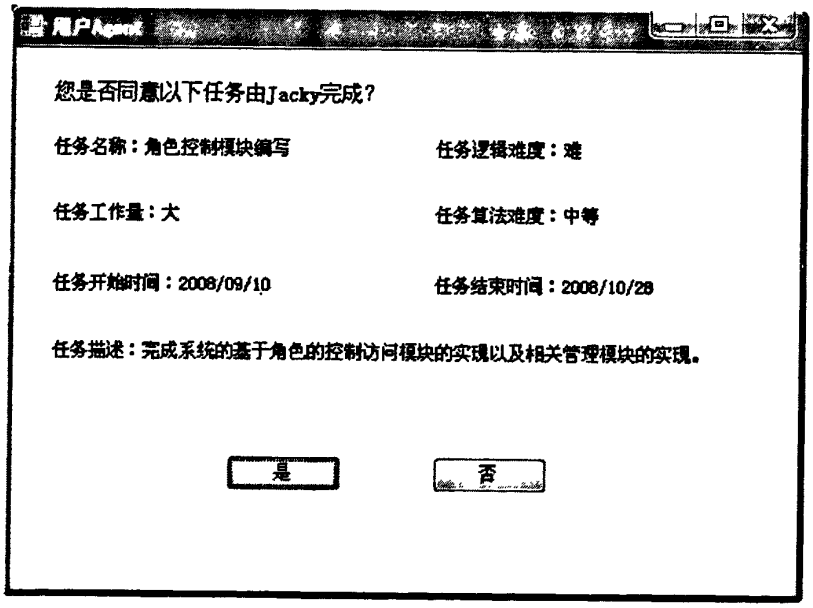

图 6.10 项目经理的用户 Agent 提示窗口

一项任务通过员工以及项目经理的确认后，正式进入员工的任务列表，即正式开始执行。员工在工作过程中，每天需要对自己的工作情况进行汇报，填报页面如图 6.11。通过进行每日汇报，一来可以使项目经理对员工当前任务的完成情

况加以了解；二来通过汇报工作情况，可以使系统掌握该员工的工作状态，从而在任务分配决策时作为参数影响分配结果。

当用户在每日汇报中将任务完成进度填写为 100%时，提交该页面将触发流程进行前行，将该任务转到测试人员的工作列表中。测试任务的分配采用工作流任务分配模式中的“拉”模式，任何测试人员都可以看到该任务，由第一个“拉”下该任务的测试人员接受任务。测试人员经过测试，不通过则提供一份测试报告，指出存在的问题，由负责该项任务的员工进行完善，完善后继续转向测试；通过则告知项目经理，该项任务圆满结束。

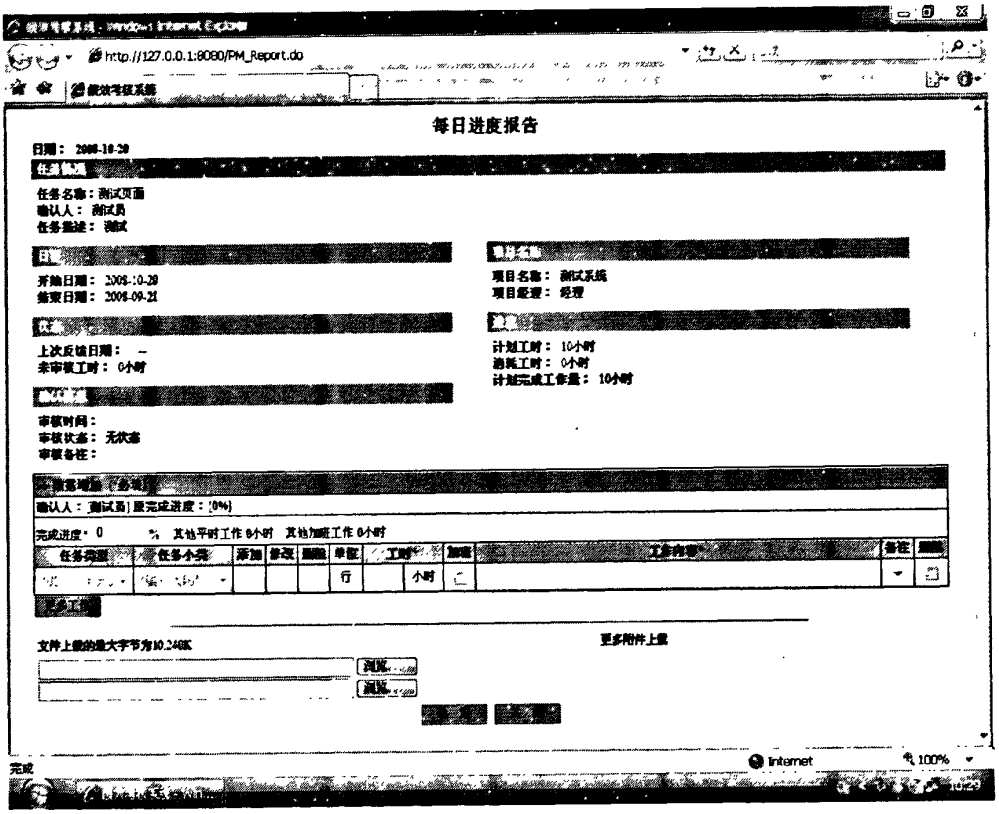

图 6.11 用户每日进度报告页面

6.4 本章小结

本章主要介绍员工绩效考核系统的实现。首先对 OSWorkflow 的源代码进行分析，点出目前最新版本中存在的问题，然后根据本系统开发的实际需要，进行相应的升级改造。然后在系统实现一节中，对系统的几个核心模块的实现分别予以介绍。最后在系统运行一节中，通过截图辅以语言解释的形式，形象的描述了系统的主要功能及作用。

第七章 总结与展望

7.1 本文总结

工作流技术是近年来在计算机应用领域中发展较为迅速的新技术之一。工作流管理技术是实现企业业务过程重组、过程管理和过程自动化的核心技术。工作流管理系统是一种发展趋势，它的主要目标是通过调度和分配有关的信息资源与人力资源来协调业务过程中的各个环节，以促使业务目标的高效实现。

本论文完成的主要工作有：

1 介绍了当前工作流管理系统的国内外研究现状，分析了传统的工作流管理系统并指出其中存在的问题，并针对存在的问题，提出了基于多 Agent 交互协作的工作流管理系统的设计思想。

2 阐述了工作流技术的一些重要概念，包括工作流及工作流管理系统的概念、工作流建模、工作流参考模型、工作流管理系统体系结构及主要功能等，讨论了工作流管理系统的实施及其意义，研究分析了工作流产品中存在的问题以及今后的发展方向。

3 介绍了 Agent 的定义及其基本特性，阐述了软件 Agent 是什么。深入介绍了基于 Agent 的系统。在阐述多 Agent 系统概念，分析多 Agent 系统的特性的基础上比较了多 Agent 系统与单 Agent 系统的差别，进而对多 Agent 系统的体系结构进行研究，详细介绍了多 Agent 间的协作机制。

4 提出一种基于多 Agent 交互协作的工作流管理系统的模型，简称 MAWFMS 模型。阐述了 MAWFMS 模型的设计思想，结合工作流管理联盟给出的工作流管理系统体系结构，提出了 MAWFMS 模型的整体体系结构并进行详细设计，并分析了 MAWFMS 模型的性能。

5 通过一个基于 MAWFMS 模型的应用实例——员工绩效考核系统来验证 MAWFMS 模型的可用性。

在论文完成之际，经过分析，我认为本论文的创新点如下：

1 本文将 Agent 技术引入到工作流管理系统的运行控制阶段，通过一个 Agent 集合交互协作来推进流程的执行，同时也有效提高了用户参与的主动性。

2 在第四章的任务分配方式设计一节中，本文将管理学中的层次分析法思想引入模型的任务分配过重中，拓展传统的工作流任务分配模式——“推”模式和“拉”模式，实现了一个“智能选推”模式，通过引入具有智能的 Agent，进行决策判断，选择最合适的用户完成相应的任务，真正加强工作流管理系统。

3 在第四章的多 Agent 集合的设计一节中，本文通过设计完成各种功能的 Agent，将相应的业务逻辑从系统中分离，有效降低了系统的耦合度，提升了系统的可扩展性和易维护性。

4 在第四章的多 Agent 集合的设计一节中，本文通过设计资源 Agent，并由资源 Agent 采用 Java 的消息机制，统一有序的进行数据库的读取，有效避免了系统资源冲突问题。

5 在第五章的系统设计一节中，关于 Agent 通信模块的设计，本文通过使用 XML 技术包装 KQML 语言，将 XML 嵌入 KQML 中的内容层，用 XML 来表达内容的语义信息，丰富 KQML 语义表达，方便地实现表达复杂语义，从而简化了 Agent 之间通信的复杂度。

6 在第五章的系统设计一节中，关于安全认证模块的设计，本文将面向切面（AOP）的设计思想引入到设计中，通过将系统的权限验证模块封装成单独的模块，与核心业务模块解耦合，在调用核心业务模块时借助动态 AOP 框架将权限验证模块在运行时动态地织入，从而实现一种通用的、可维护的、易扩展的权限认证模块。

7.2 研究展望

本论文主要提出了一个基于多 Agent 交互协作的工作流管理系统模型。由于本人的学识、时间和开发环境所限，系统还存在一定的不足。要把 Agent 技术更好的融入到工作流管理系统当中，今后还需要做好以下工作：

1 提出基于多 Agent 交互协作工作流管理系统开发的标准和规范。

2 引入 Agent 自我学习机制，真正实现完全智能。

3 图形化过程定义的研究与实现。

4 模型运行效率的研究与优化。

5 模型安全性的分析。

6 由于时间所限，模型应用实例只实现了工作流管理系统的部分功能，今后需要加以完善。

由于本人水平所限，论文中存在不足与不妥之处，恳请各位老师和同学批评、指正。

参考文献

[1] 范玉顺.工作流管理技术基础.清华大学出版社，2001，4

[2] 于海斌，朱云龙.协同制造——E 时代的制造策略与解决方案.清华大学出版社，2004,4

[3] 何炎祥，陈莘萌.Agent 和多 Agent 系统的设计与应用.武汉大学出版社，2001，6

[4] Alonso G, Mohan C, Gunthor R, et al. Exotica/FMQM: a persistent message-based architecture for distributed workflow management [EB/OL]. http://www.almaden.ibm.com/cs/exotica.

[5] Dath S, Kochut K, Miller J, et al. ORBWork: a reliable distributed CORBA-based workflow enactment system for meteor2 [EB/OL]. http://lsdis.cs.uga.edu/lib.

[6] Chan D H K, Vonk J, Sanchez G, et al. A specification language for the WIDE workflow model [EB/OL]. http://dis.sema.es/projects/ WIDE/Documents.

[7] Weissenfels J, Wodtke D, Weikum G, et al. The mentor architecture for enterprise-wide workflow management [EB/OL]. http://paris. cs.uni-sb.de/public_html/papers

[8] 工作流技术.http://www.javafox.org/

[9] Foundation for Intelligent Physical Agents.FIPA00023，FIPA Agent Management Specification.http://www.fipa.org/specs/fipa00023/，2002.

[10] Shen W，Norrie D.H..A Hybrid Agent-oriented Infrastructure for modeling Manufacturing Enterprise.In proceeding of KAW' 98，Banff，Cananda，1998(Agent-5:1-19)

[11] Jeff Y.C.p，Tennbaum J.M..An intelligent agent framework for enterprise integration.IEEE Trans On Sydtems.Man and Cybernetice，1991.21(6):1391-1408

[12] 许青松，范玉顺，吴澄等.支持动态联盟的车间控制信息系统框架模型.信息与控制，2000.29(4):289-296

[13] 韦文斌，杨建军，曾波等.基于多代理的分布式车间控制系统的研究.机械设

计与制造工程，2001.30(1):28-33

[14] 赵卫东，黄丽华，蔡斌. 工作流过程模型研究. 系统工程理论方法应用，2002, 11（3）：212-217

[15] 李红臣，史美林. 工作流模型及其形式化描述. 计算机学报, 2003, 26（11）：1456-1463

[16] 夏长虹，陈文博. 工作流系统过程建模与应用生成环境研究. 计算机工程，2003，29（5）：59-61

[17] 牛军钰，赵宏，赵大哲. 基于 Petri 网的工作流建模方法. 控制与决策，1999，14（增刊）：521-525

[18] 黄世秀，高飞，胡小华. 基于工作流的电子政务系统. 合肥工业大学学报（自然科学版）, 2004, 27（2）：140-143

[19] 赵刚，杨宗凯. 基于工作流和 Web 技术的 OA 系统设计. 计算机工程与应用，2002.09：235-238

[20] GAO Chun-ming, XIAO Wei, CHEN Yue-xin, LI Zhu-chao. A Evaluation Management Information System Based on Workflow. Jour Nat Scie Hunan Norm Uni, Dec. 2001; Vol. 24 No. 4:25-28

[21] Jiang Hao, Dong Yisheng, Luo Junzhou. Research on Petri Net Based Modeling and Analyzing Methods for Workflow Process. Journal of Southeast University(English Edition), Dec. 2000; Vol. 16 No. 2:66-68

[22] Workflow Management Coalition. The workflow reference model.WFMC TC00-1003,1994

[23] C.Ellis,K.Keddara and GRozenberg,Dynamic change within workflow systems,ACM Conf.on Organizational Computing Systems(COOS 95),August,1995

[24] Peter Lawrence.Workflow Handbook.Chichester,West Sussex,England;New York:John Wilev & Sons,c1997.

[25] Workflow Management Coalition. Workflow management coalition terminology & glossary. WFMC TC00-1011,1994

[26] Workflow Management Coalition. http://www.wfmc.org.

[27]刘蕾,刘厚泉.基于工作流的 B/S 模式 OA 系统设计与实现.微计算机信息，

2008，2-3：233-235

[28]WilVan DerAalst, KeesVan Hee. 工作流管理:模型、方法和系统[M]. 王建民, 闻立杰译. 北京:清华大学出版社, 2004.

[29] 史忠植.智能主体及其应用[M].北京：科学出版社，2000:1-11

[30] FIPA Agent Management Specication.2001-10

[31] 姚郑,高文.软件 Agent.计算机科学[J].1996，23(l):10-13.

[32] Wooldridge M J, Jennings N R. Intelligent agent:theory and Practice.Knowledge Engineering Review, 1995, 10(2):115~152

[33] 程莉,毛莺池,王志坚.基于移动 agent 的 B2B 协作电子商务系统.计算机工程, 2003.4

[34] H S Nwana. Software Agent: an overview, Knowledge Engineering Review, 1997, 58 (2): 205~245

[35] Y Shoham. Agent-based programming. Artif Intell, 1993,60(1): 123~154

[36] 钟凌燕.基于 Agent 联邦的开放式工作流管理系统的研究.浙江大学博士毕业论文，2003.

[37] Michael W，Nicbolas RJ. Intelligent Agents theory and Practice[J].Knowledge Enginering Review，1995, 10(2):115-152.

[38] Fushiung, Hsieh. Design of evolvable manufacturing processes [A].Proof the 2002 Congression Evolutionary Computation[C], 2002, 339-244

[39] Goldman CV, Rosenscbein JS. Evolutionary patterns of agent organizations [J]. IEEE Trans Systems, Man and Cybernetics, PartA, 2002, 32(1):135-148.

[40] Luo Y,Liu K,Davis DN. A multi-agent decision support system for stock trading[J].IEEE Network,2002,16(1):20-27.

[41] Jia D,Krogh BH,Talukdar S.Distributed model predictive control[J].IEEE Control Systems Magazine,2002,22(1):44-52.

[42] 王一宾,李心科,刘桂江.软件 Agent 技术与软件体系结构.河南科技大学学报(自然科学版)，2005 年 4 月:30~34

[43] 吴建林,姜丽红,薛华成.专家系统与多 Agent 协作系统计算机科学[J],1998,voL.25 No.4

[44] 李建民,石纯一.DAI 中多 Agent 协调方法及其分类.计算机科学[J],1998,Vol.25 No.2

[45] 王映辉. 分布构件模型技术比较研究[J]. 计算机应用, 2003, 20 (7): 3-9

[46] Ravi S Sandhu , Edward J Coyne , Hal L Feinstein. Role-Based Access Control Models[J]. IEEE Computer, 29 (2): 38-47

[47] 黄建, 卿斯汉, 温红. 带时间特性的角色访问控制[J]. 软件学报, 2003, 14(11): 1944-1954

[48] 张敏, 陈曦, 龚育昌. 面向方面系统的耦合度评估[J]. 中国科学院研究生院学报, 2008, 25 (2)：244-250

[49] 赵成勇, 周南, 张晓泉. 基于AOP与MVC模式的Web应用架构的设计与实现[J]. 南京大学学报(自然科学), 2005, 41(21): 718-723

[50] Sandholm T. Automated Negotiation.[J]Communications of the ACM,1999,42(3):84-85.

[51] Jennings N，Faratin P,Norman T. Autonomous Agents for Business Process Managemen [J] . Internatio International Journal of Applied Artificial Intelligence,2000,14(2):145-189.

[52] Wooldridge M,Jennings N. Intelligent Agents:Theory and Practice[J]. Knowledge Engineering Review,1995,10(2):115-152.

[53] 史忠植. 智能主体及其应用[M]. 北京:中国科学出版社, 2000.

[54] 左伟明. XML 数据标记语言参考手册. 北京:人民邮电出版社, 2007:309

[55] 李随成,陈敬东,赵海刚.定性决策指标体系评价研究[J].系统工程理论与实践，2001,9:22-28。

[56] 李刚军,李娟,李怀恩等.基于标度转换的模糊层次分析法在宁夏灌区水权分配中的应用[J].自然资源学报,2007,11:872-879。

[57] 孙卫琴 著.精通 Struts：基于 MVC 的 Java Web 设计与开发[M]. 电子工业出版社. 2005/12

[58] 杨易 等著.JSP 网络编程技术与实例.人民邮电出版社[M]. 2005/10

[59] 周智仁等 著. Java 网络应用编程[M]. 高等教育出版社. 2004/4

[60] [美] Hans_Erik Eriksson 、Magnus Penker 著. 夏昕、何克清译 UML 业务建

摸[M].机械工业出版社. 2004/3

[61] 杨磊、陈凌云等著 精通 Eclipse Web 开发—Java 体系结构、工具、框架及整合应用[M]. 人民邮电出版社. 2006/10

[62] Eissen H N. OSWorkflow.A.Guide.for.Java. 北京：清华大学出版社，2008：215

[63] OSWorkflow Team. OSWorkflow-Opendoc V2.8. 2008:50

攻读硕士学位论文期间发表的学术论文

[1] 基于 AOP 的 RBAC 系统的设计与实现,计算机安全，已录用，拟 2009 年 5 月出版

[2] 工作流在面向订单型企业 ERP 中的应用，微计算机信息，已录用，拟 2009 年 12 月出版

致 谢

论文写作完成之际，在此诚挚的感谢在两年半的硕士研究生学生生活中给与我帮助的老师同学们。

首先衷心的感谢我的导师孙莉老师。在我的研究生学习生涯中，孙老师不仅教授给我宝贵的科学知识，还传授给我丰富的人生经验。感谢她在论文完成过程中给予我悉心的指导和关怀，为我付出了许多心血，她渊博的学识、严谨的科研态度使我受益终生。

此外，我还要感谢在工作和学习中给予我很大帮助的吴国文老师，吴老师在论文的选题和完成过程中，给我提出了很多宝贵的建议和参考资料，使得论文得以顺利完成。

最后，我还要感谢实验室的所有老师和同学们，是你们在学习、生活、工作中给了我莫大的帮助，让我在这段求学生涯中更充实。

谢谢大家。