基于 Agent 的 WfMS 的设计及应用

Design and Application of an Agent – Based WfMS

徐义峰　徐云青　诸葛理绣　（衢州学院　浙江衢州 324000）

摘要：工作流及工作流管理系统是 CSCW 领域研究的热点之一，Agent 技术也是人工智能领域的新兴课题。把 Agent 技术结合到工作流管理中，利用 Agent 智能、易于扩展的特点，可以使工作流管理系统更加灵活，适应性更强。本文提出了一种基于 Agent 的工作流模型，并给出了该模型的一个实际应用。

关键词：Agent　角色　工作流　工作流管理系统

1　引言

随着网络及通信技术的发展，工作流（Workflow）技术以其良好的适应性和灵活性在信息系统的开发方面受到广泛关注，工作流管理系统（WfMS）作为一个通用的支撑工具，为信息系统的开发提供了一种新的模式。但是面对现代企业业务流程灵活、高效、可扩展等新要求，传统 WfMS 的工作模式已显得力不从心。为了适应新的要求，为 WfMS 引入其它技术是一个很好的选择[1]。

本文提出的 WfMS 是在开发高校科研管理系统项目中，为解决实际问题而提出的。该项目主要业务流程是对科研项目的审批，所以设计系统的重点就放在解决审批流程上。该系统与传统 WfMS 相比，具有两个特点：① 面向角色：系统在执行工作流实例时，并不直接控制实际参与者之间的执行先后顺序，而是控制各角色执行任务的顺序。② 引入 Agent 技术[2]，利用 Agent 控制工作流系统的执行，充分利用软件 Agent 的优越性，避免了某些传统 WfMS 中存在的不足，提高了系统的性能。

2　面向角色的 WfMS

一般的审批流程都是从申请开始，然后经过几个不同的审批步骤，最后由最高负责人批准通过。具体到每个审批流程，也许它们经过的中间审批步骤不同，但却都是相对固定的，而每个步骤可能不止一个人来完成，也可能跳过该步骤直接进行下面的审批，我们根据审批流程的特点提出了面向角色的 WfMS[3]。该系统的特点是：在过程定义时，当一个参与者任务完成返回工作流引擎之后，工作流引擎并不是直接确定下一执行参与者，而是确定下一执行角色。当执行角色确定之后，再由控制该角色内部子流程的引擎管理、控制该角色所有参与者的执行顺序。这种机制的一个明显优点就在于利于工作流实例的扩展和更改。

在实际应用中，一个工作流实例可能历时很长，这期间难免存在人事变动。如果采用直接控制参与者执行先后顺序的方法，人员变动势必影响到过程定义，这就给工作流的正常运行带来了不便。而系统通过使用角色的概念避免了这个问题。因为过程定义只涉及到角色，具体的参与者的增加、减少或者变更对它不会产生影响，所以避免了工作流过程定义随着人员变动而改变。另外，我们通过增加参与者注册和注销的功能，方便工作流实例运行过程中参与者的增加、减少或更换。而每个角色的子流程引擎对参与者执行顺序的控制采用的方法是：为每个参与者编号，工作流正常运行时是按照编号顺序决定执行先后顺序的，而需要反复时，则由当前执行者选择决定下一执行者，这同样不会涉及到具体的参与者，达到了使用角色概念的目的。

3　基于 Agent 的 WfMS 模型

在面向角色的 WfMS 中，引入 Agent 技术，设计了一个基于 Agent 的 WfMS 模型[4]，系统包括：① 过程定义：为用户提供可视化过程定义接口。② 工作流引擎：WfMS 的核心部分，管理和控制工作流实例的执行。③ 角色 Agent：衔接工作流引擎与执行者之间的工作，

管理该角色内部子流程的工作。④用户 Agent：用户与工作流之间的人机接口。系统模型如图1所示。

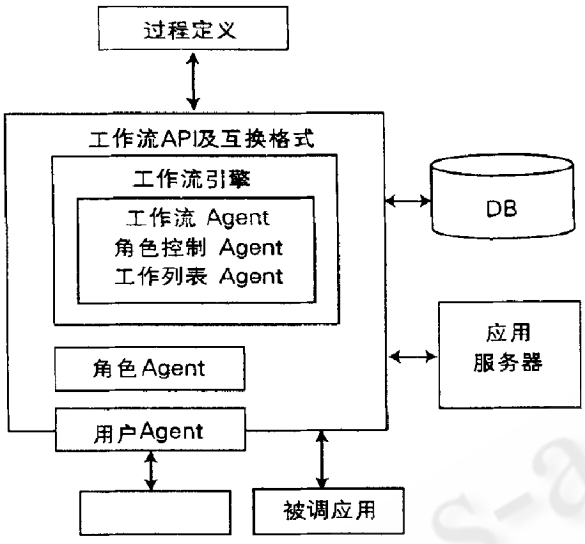

图 1　系统模型

3.1 过程定义

WfMS 完成业务流程自动化的基础是将业务流程描述为计算机可识别的形式化表示，也就是工作流的过程定义。过程定义包含多种必要的信息，以保证这个过程能够被工作流执行软件执行。考虑到工作流的灵活性，即不能在工作流实例执行之前就预先决定了它的执行顺序，而是要在执行过程中根据执行的具体情况动态的控制工作流的执行顺序，我们采用确定有限自动机理论来描述工作流模型。

工作流定义为一个五元组：<R, V, f, Ro, F>。其中：R 表示从业务流程抽象出来的角色的集合；V 表示输入事件的集合，即代表各角色完成任务后可能采取的动作集；转换函数 f 是用来表示工作流中执行角色的转换关系的。它是从 R×V 至 R 的部分映射，$f: R \times V \rightarrow R$（×表示叉乘），或者表示成函数形式：$f(r_i, v_j) = r_k (r_i, r_k \in R, v_j \in V, i, j, k \in N)$，例如对于角色 r_1，$r_2 \in R$，$v \in V$，如果有函数 $f(r_1, v) = r_2$，则表示当前执行角色为 r_1，输入事件为 v 时，工作流将转到角色 r_2 执行；$r_0 \in R$，表示工作流中第一个执行角色，是唯一的；F 是一个终态集，在系统中元素个数必须大于等于1，它是 R 的一个子集，用来表示有权结束工作流的角色集。一个由申请者、修改者、审核者和批准者四个角色参与的项目审批过程可以定义为：[examine_item] = {R, V, f, Ro, F} 其中：R = {申请者，修改者，审核者，批准者}；V = {a = 申请提交，b = 流程继续，c = 修改提交，d = 审核提交，e = 反复修改，f = 反复审核，g = 返回申请，h = 批准完成}；f = 状态转移图（即工作流流转图，见图2）；Ro = 申请者；F = {申请者}。

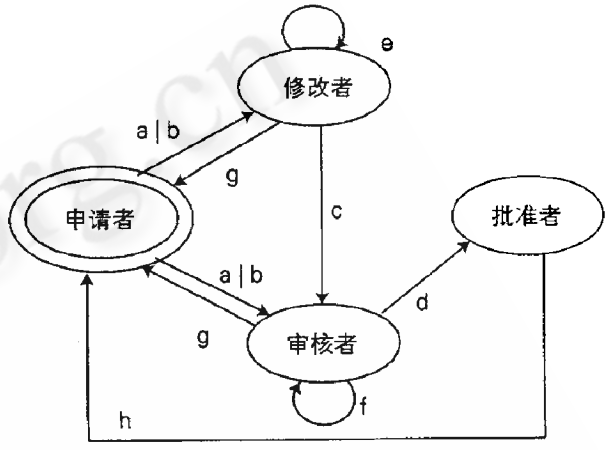

图 2　工作流流转图

实现时可以设计一个可视化过程定义窗口，用户只须将业务流程中涉及到的角色相关信息按照窗口提示提交系统即可，而不必懂得系统是如何完成工作流建模的。建模过程将由系统内的相应程序根据用户提交的数据完成。然后通过工作流引擎与过程定义之间的接口将业务流程的形式化描述即过程定义存储到工作流 Agent 中，为工作流的正常运行提供依据。

3.2 工作流引擎

负责解释过程定义，控制过程实例的创建、激活、挂起、结束等，在角色间导航，加入或者撤销参与者，为用户指定工作项、维护工作项列表，维护工作流相关数据和工作流控制数据等，调用外部应用和连接工作流相关数据的接口等。工作流引擎有三个 Agent：工作流 Agent、角色控制 Agent、工作列表 Agent。

(1) 工作流 Agent。负责解释并存贮过程定义，并根据过程定义控制过程实例角色间的导航。当接收到来自外部的触发信息时，它将创建一个工作流过程实例，读入相关过程定义。管理和控制过程实例的执行，在执行过程中根据执行情况将产生的工作项写入相应参与者的工作列表中。同时，它还必须负责参与者加入时注册和离开时注销的功能。

(2) 角色控制 Agent。主要功能是存储并维护角色相关信息。在工作流实例开始时,它将读入此次流程涉及到的角色的相关信息,创建各角色 Agent。在执行过程中根据工作流 Agent 的导航信息激活相应角色 Agent 并为其提供相应信息。

(3) 工作列表 Agent。主要是为工作流实例各参与者维护工作项列表。当一个参与者完成任务之后,将由工作流 Agent 将产生的工作项写入工作列表中相应参与者的工作项中,而当激活一个用户 Agent 执行任务时,工作列表 Agent 为其提供工作项。

3.3 角色 Agent

系统为每个角色创建一个角色 Agent,用于管理和控制子流程的执行。它从工作流 Agent 那里获取该角色所有参与者的相关信息,在工作流执行过程中,主要是根据参与者的编号来控制其执行先后顺序的,当遇到特殊情况时把信息返回工作流 Agent,由工作流 Agent 决定工作流的下一执行步骤。

3.4 用户 Agent

工作流实例中的每个参与者在加入工作流时都需要通过工作流 Agent 的注册,工作流为每个注册用户创建一个用户 Agent,并将这些用户 Agent 交给相应的角色 Agent 管理。用户 Agent 是工作流与工作流参与者之间的人机接口。从工作流中为参与者获取相关数据、工作项列表及其它所需信息,同时将执行实体的结果反馈给工作流。

4 工作流 Agent 定义

Agent 完整的定义应该包括局部数据、历史经验库、处理过程和处理机几个部分。其中,局部数据是 Agent 私有的,只有该 Agent 具有存取权限,外部用户只能通过请求 Agent 的相应服务来访问这些数据。历史经验库定义 Agent 处理事务的经验、知识规则等,Agent 的处理过程可以根据历史经验库中的知识规则或历史经验有效地完成任务或者不断地完善自身。处理过程是 Agent 对外的窗口,Agent 所提供的服务都是通过对处理过程的调用请求而实现的。处理机提供 Agent 的处理能力,Agent 的处理过程以进程的形式在处理机上执行。限于篇幅,本文不给出各 Agent 的完整定义,仅给出工作流 Agent 的历史经验库和用于交互的处理过程部分。

工作流 Agent 的处理过程包括两个:决定下一执行角色的处理过程和生成用户 Agent 的处理过程。历史经验库包括过程定义规则,形式化定义如下:

```
Agent <Workflow_Agent>
  Knowledge_base 过程定义规则
  Process < 下一执行角色 >
    On < 工作流开始 || User_Agent 返回 action >
Do
    (1) 如果 event =工作流开始
    nextRoleName←R0. Role_Name
    (2) 如果 event = User_Agent 返回 action
      访问 Knowledge_base
    依据知识规则和当前条件决定 R
      nextRoleName←R. Role_Name
    移交 nextRoleName 给 RoleControl_Agent
    Process < 创建 user_Agent >
    On < RoleControl_Agent 返回值 > Do
    (1) 为执行实体生成 User_Agent
    (2) 将角色相关信息移交 User_Agent
    (3) 为 User_Agent 提供数据访问接口
    (4) 启动 User_Agent
    End Agent
```

从工作流 Agent 的定义中可以看出,工作流执行顺序是相应处理过程通过访问历史经验库中的过程定义规则决定的。当实际的业务流程有变动的时候,只要简单地改变过程定义规则即可,而工作流的其它部分并不会受到影响。这样设计的结果不仅实现了工作流实例执行过程中执行顺序可以根据执行实体具体操作灵活地改变,同时也允许企业业务流程自由地变更,而不用修改系统。

5 系统工作流程及 Agent 之间的关系

各 Agent 分别具有自己的功能并负责各自的工作,它们之间通过互相协作来完成整个工作流的执行。图 3 给出了各 Agent 之间的关系。同时从图 3 可以总结出系统的工作流程。

(1) 当工作流 Agent 感应到外部事件的触发时,它将生成一个工作流实例。在工作流实例开始时,工

作流 Agent 读入过程定义，激活角色控制 Agent。被激活的角色控制 Agent 将读入该实例中涉及到的各角色的信息，并为每一个角色生成一个角色 Agent。工作流 gent 同时接受每个参与者的注册，为每个参与者生成一个用户 Agent，并根据角色分类为这些用户 Agent 编号，将其信息通知相应的角色 Agent。最后工作流 Agent 将工作流实例初始时的工作项信息移交给工作列表 Agent，写入相应参与者的工作项列表中。经过上述步骤之后，工作流实例创建完成。此时，工作流 Agent 还将再次激活角色控制 Agent，通知其当前执行角色。

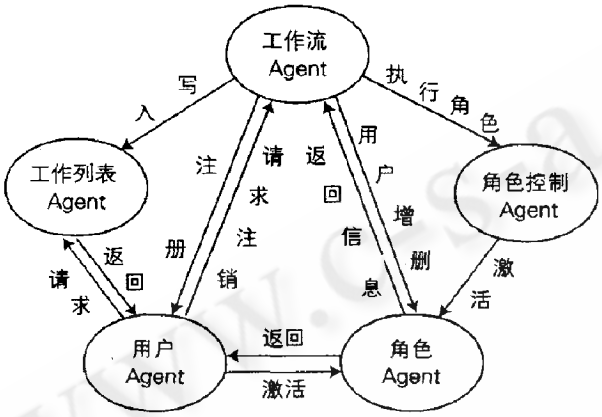

图 3　系统工作流程

(2) 角色控制 Agent 接收到当前执行角色后，将激活相应的角色 Agent。角色 Agent 将根据子流程内部的控制信息控制属于该角色的用户 Agent 工作。角色内部子流程一般是按照用户 Agent 的编号顺序执行的，所以当一个用户 Agent 任务执行完毕后返回角色 Agent，角色 Agent 无需请求工作流 Agent 来决定下一个执行者，而是由它自己按照自身规则决定，此时它只需将可能产生的工作项提交给工作流 Agent 即可。但是，如果某个用户 Agent 在执行过程中请求流程转出该角色或者该角色所有用户 Agent 都已完成任务，则角色 Agent 激活工作流 Agent，提交相应信息。

(3) 当用户 Agent 由角色 Agent 激活开始工作时，它将向工作列表 Agent 发出请求，工作列表 Agent 将该用户 Agent 的工作项列表移交给该用户 Agent。用户 Agent 任务完成之后返回角色 Agent。

(4) 在工作流执行过程中，工作流 Agent 在接到角色 Agent 提供的新产生的工作项后仍需要激活工作列表 Agent 写入相应用户 Agent 的工作项列表。而当工作流 Agent 接到角色 Agent 提交的工作流导航信息时，它需要按照过程定义的规则确定下一执行角色，并激活角色控制 Agent 工作。

如果在工作流执行过程中，某个用户 Agent 因为某种原因(例如有新任务，或者没有时间及时处理该工作流实例的工作等)要退出该工作流实例的执行时，它将向工作流 Agent 提出注销请求，工作流 Agent 注销该用户 Agent 并通知相应的角色 Agent。同样的，如果某个参与者要加入该工作流实例，则在工作流 Agent 上注册，工作流 Agent 为其生成用户 Agent 并通知相应角色 Agent。

6　应用实例

高校科研管理系统是基于 win2000 网络平台，主要采用 Java 语言开发的，专为某高校科研管理量身订做的办公自动化系统。它实现了高校从项目申请到立项，再到项目实施，最后到结题全程自动化。其中项目管理模块的项目申请审批应用了以上系统模型。项目申请审批流程是项目从申请到最后审批通过的一个过程，即申请者完成表格填写之后，提交修改人员(可以是本人)进行修改(可能由多个修改人员多次修改，也可能没有修改)，修改后提交系部和科研处审核人员审核(可能由多个审核人员多次审核也可能没有审核)，审核通过后提交领导最后批准，流程结束。该流程用自动机的状态转移图表示见图 2。在该实例执行过程中，我们针对各种不同的情况均做了模拟测试，运行结果正确，成功地解决了实际问题，同时也达到其灵活性的要求。

参考文献

1 解放、曹江辉，基于 Agent 的产品开发工作流管理系统的研究[J]，计算机应用研究，2002(12)：85－87。
2 何炎祥，Agent 和多 Agent 系统的设计与应用[M]，武汉大学出版社，2001。
3 赵卫东、黄丽华，面向角色的多 agent 工作流模型研究[J]，管理科学学报，2004(4)：55－61。
4 毛新军、闫琪，面向 Agent 的软件开发方法[J]，计算机科学，2003(5)：94－96。