文章编号:1004-5694(2006)03-0366-05

基于 RoleNet 模型的工作流引擎设计及实现*

程平,陈前斌,隆克平,朱道飞
(重庆邮电大学 光互联网及无线信息网络研究中心,重庆 400065)

摘 要:对动态流程的建模和支持一直是工作流引擎设计和实现中的难题。将过程建模方法划分为以活动为中心和以角色为中心 2 种类型,分析了传统的过程建模方法在对动态流程描述方面的不足,提出了一种以角色为中心的过程模型 RoleNet,并且基于该模型开发了工作流引擎 OM Engine。实际的应用情况表明,OM Engine 对动态流程具有较好的适应性,同时也验证了 RoleNet 模型的可行性。

关键词:工作流;活动;角色;工作流引擎;动态流程
中图分类号:TP273 文献标识码:A

0 引 言

工作流技术起源于上世纪人们对办公自动化技术的研究,经历了 70 年代理论雏形的诞生,80 年代商用产品的出现和 90 年代的飞速发展和标准化工作的开始[1,2],现在工作流在办公自动化、集成制造、电子商务等领域发挥了重大作用,并向更复杂和更灵活的方向发展[3]。

按照工作流管理联盟(workflow management coalition,WfMC)的定义,工作流是指能够完全或者部分自动执行的业务过程,在该过程中,根据一系列过程规则,文档、信息或活动能够在不同的执行者之间传递、执行。该定义被广泛引用,很明显,工作流的思想最初必然来自于工业自动化的"流水线"[4]。其特点是以活动为中心(activity-centric),基本的业务单元是活动,主要关注流程中的活动以什么样的逻辑或者规则串接起来,并以什么样的模型进行表示和计算,人只是作为一种资源,附属于业务过程。但这种模型并不适合企业所有的业务,由于用户需求是在不断变化的,企业需要不断地调整和优化自身的各种业务。很多时候需要在运行时由人来决定流程的走向,例如:通信运维中的故障处理,客户投诉处理,都需要依靠人根据实际情况来进行判断。传统的以活动为中心的工作流过程模型很难实现这样的要求,因为不可能穷举所有可能的情况。为此我们在本文中提出了一种新的以角色为中心的(role-centric)过程模型 RoleNet。此模型增强了工作流引擎流程控制的柔性,简化了工作流引擎内核设计和过程模板定义的复杂度。我们在此基础上开发出了一个轻量级的工作流引擎 OM Engine,并应用于实际的项目中。

1 工作流过程模型

1.1 Activity-Centric 过程模型

常用的过程建模技术有 Petri 网,FSM,UML 活动图等。无论采用什么理论方法,传统的过程模型有一个共同的特点就是,它的基本模型单元是活动,因此我们可以称它是以活动为中心的模型。图 1 是一个 activity-centric 过程模型的例子。

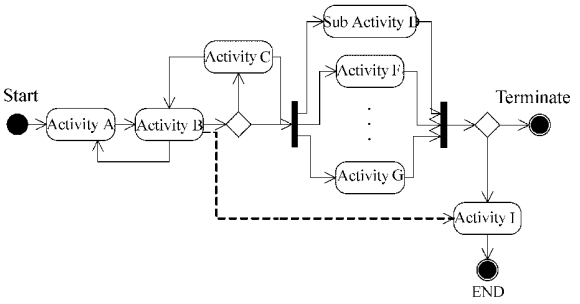

图 1 Activity-centric 过程模型图
Fig. 1 Activity-centric process model

WfMC 定义了 6 种基本的流程控制结构:顺序(sequence)、与分支(and-split)、与合并(and-join)、或分支(or-split)、或合并(or-join)、循环(iteration)。Aalst 在文献[5]中定义了 21 种控制模式,比较完

* 收稿日期:2005-11-23 修订日期:2006-03-29
基金项目:国家"863"高技术发展项目(2005AA122310),国家自然科学基金项目(90304004)、国家教育部新世纪优秀人才支持计划(教技司[2005]2 号)、重庆市科委项目(2005BB2006,2005AC2089)、重庆市教委项目(KJ060508,KJ060513)、重庆市科委自然科学基金项目(8477)
作者简介:程平(1980-),男,四川省南部县人,硕士,主要研究方向为互联网技术及应用,E-mail:chengp.cn@gmail.com;陈前斌 ,男,博士,教授,主要研究方向为无线网络技术、多媒体信息传输与处理。

整地提出了工作流的控制结构。另外,在实际应用中,还有着"回退"、"自由流"、"取回"这样的非标准模式存在。例如:在图1中的虚线流程代表了一种自由流的模式,流程在B处可以在C,I,D,F,G中任意选择下一个活动,需要在运行时动态决定,这在故障处理、客户投诉处理等业务中是很常见的。

这些非标准模式,文献[5]中提到的任意循环、多实例等模式以及用户对流程变更重组的要求,正是体现着用户对引擎"柔性支持"的需求,然而,从实际应用来看,传统的建模方法将各参与者执行的活动交错分布在一个模型中,强调活动之间的逻辑关系却忽略了其主体角色的交互[6]。这样的模型是复杂而且不完整的,在实际应用中难以应付流程的动态性。

1.2 Role-Centric 过程模型

事实上,角色是流程最活跃的因素。文献[6]中的研究表明,工作流的变化经常可以找到角色方面的原因。流程的变化管理实质上是拆散原流程的各种角色,借助IT塑造新的角色能力,并把他们重新装配成为顾客创造价值的过程。可以说正是因为传统的按照活动来建模的方法对角色变化重视不足,才导致了工作流引擎对柔性支持的不足。

长期以来,人们对角色的研究更多地是关注于组织模型和授权模型方面[7,8],近年来也出现了一些基于角色的流程描述方法,例如通信的工作流模型、角色活动图(RAD)、消息顺序图等描述方法。这些方法多是从传统的流程图演变而来的,例如通信的工作流模型和角色活动图,由于将角色间的交互关系与活动逻辑关系揉在一起,增加了表述的复杂性,不适用于表达复杂的模型,而消息顺序图则并没有体现出流的概念,只是对角色间信息交换顺序的描述。

1.3 RoleNet 过程模型

对于工作流引擎的设计,需要的是一种更加抽象、简洁的过程模型描述方法。通过上述的分析,我们认为,传统的以活动为中心的过程模型和现有的基于角色的过程模型,都无法很好地解决工作流引擎对动态过程的支持问题。因此,我们试图从另一个角度来思考工作流。传统的工作流的定义把流当作是由一个个活动衔接而成的序列,把角色描述为流中活动的执行者。实际上,从角色建模的角度来看,一个组织中各种不同职能的角色之间的工作关系构成了一个"角色网",在业务执行过程中,活动和信息在不同的角色间交换,在业务执行中,文档、信息和活动等元素在不同的角色之间进行交换,工作流就是这些元素在"角色网"中流转所经过的路径。

基于以上的想法,我们提出了RoleNet过程模型,该模型属于以角色为中心的role-centric模型,具有简单、清晰的特点,能够满足柔性工作流引擎设计的需要。下面我们给出该模型的形式化描述。

2 形式化定义

定义1 过程 P 是一个有限状态机。$P=(Q,\sum,f,q_0,E)$ 是一个五元组,其中,$Q=\{q_0,q_1,q_2,\cdots,q_n\}$ 是流程所处状态的集合,$q_n=(st_n,ss_n)$ 是一个二元组,我们用步骤 st_n 和步骤所处状态 ss_n 来表示流程所处的状态;$\sum=\{\sigma_0,\sigma_1,\sigma_2,\cdots,\sigma_n\}$ 是输入控制信息的集合;f 是状态转移函数,根据过程模板和路由表中定义的转移约束,产生控制信息调度过程状态的转移;q_0 是初始状态,代表流程的开始;E 是终结状态的集合,只有2个元素:abe 和 ne,abe 代表异常终止,ne 代表流程正常结束。

定义2 RN 是一个有向图,$RN=(R,E)$ 是角色所构成的角色网,而 $R=\{r_0,r_1,r_2,\cdots,r_n\}$ 为所有角色节点的集合,$E=\{e_0,e_1,e_2,\cdots,e_n\}$ 是所有连接弧的集合。

定义3 $FP\subseteq RN$ 是 RN 的一个子图,$FP=(R,E')$,$E'\subseteq E$ 代表一个工作流,即角色与角色之间的工作关系所构成的路径。

定义4 节点 $r_n=(ra,type,RT)$ 是属于该业务过程某一个状态 $q_n=(st_n,ss_n)$ 的一个角色,它是一个三元组,其中 ra 是节点地址,可以和实际组织模型中的用户映射起来;$type$ 描述角色节点的类型,分为auto(自动)和manual(人工交互节点)2种类型;$RT=(cond,st,ss,act,nst,nss,nr)$ 是该角色节点的路由表,定义了该角色在不同的条件下,能够完成什么样的操作和将活动转发到哪些节点;其中 $cond$ 是外部传入的条件,可以是应用程序根据相关数据生成的条件,也可以是角色之间相互发送的消息;st 是流程所处的步骤;ss 是步骤所处的状态;act 代表角色能完成什么样的操作,包括人的交互和调用用户程序自动执行的操作;nst,nss 表示完成这个操作后的当前流程的步骤和步骤的状态;nr 是转发目标角色节点。

定义5 边 e 代表角色之间的工作关系,具体地说,代表角色之间的活动转移关系,其方向是由角色节点路由表来决定的。

根据以上的定义,我们可以改写图1给出的activity-centric过程模型,为了直观,我们还是采用形象化的描述,RoleNet过程模型如图2所示。RoleNet模型图分为2部分:过程的状态转移图和角色网,图2中的虚线代表角色之间相互的工作关系,由角色节点路由表定义。角色和这些关系共同

构成了角色节点网，一个工作流实例就是活动和信息在这些角色节点之间转移所走过的路径，如图 2 中的实线所示。

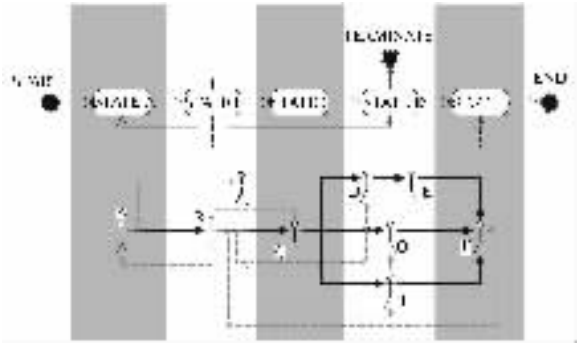

图 2　RoleNet 过程模型

Fig. 2 RoleNet process model

3　RoleNet 模型的特点

我们提出的 RoleNet 模型方法与其他的建模方法相比，RoleNet 模型有以下特点。

(1) RoleNet 中，活动不再是基本单元，取而代之的是角色节点，工作流也不再表现为活动之间的迁移，而是抽象为一个个的业务步骤以及活动在角色节点之间的转移路径。

(2) RoleNet 不仅可以描述静态业务过程，还可以描述动态业务过程，它将工作流运行实例引入到模型中，一个动态的业务过程可以有多个不同的工作流实例。如图 2 所示，其中的虚线路径代表工作流所有可能的路径，实线代表运行时工作流的路径，另外，人不再是流程的客体，而成为流程的主体，这个特点使得 RoleNet 特别适合动态流程的建模和分析。

(3) RoleNet 模型简化了工作流引擎的设计。由于将过程抽象为有限状态机，将复杂的业务逻辑分布式定义在各个节点路由表中，并且将角色看作是具有信息接收转发功能的节点，使得工作流引擎内核的设计非常简单，只需要根据节点路由表和过程当前的状态进行判断，完成任务转发就可以了，如图 3 所示。

(4) 流程定义简单。RoleNet 模型与传统的流程定义语言(如 XPDL)不同，RoleNet 模型采用过程模板和路由表来定义流程。将复杂的业务逻辑分解，分别定义在各个角色节点的路由表中，符合问题分离的原则，过程定义简单清晰，并且业务执行逻辑既可以在运行时决定(动态路由表)，也可以在定义时决定(通过设定自动转发角色节点)。

RoleNet 模型非常适合描述那些主要参与者是人，并且业务流程需要在执行时动态决定的工作流。

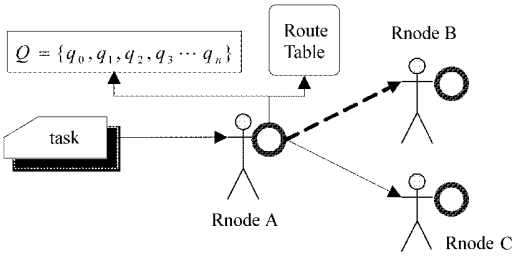

图 3　基于 RoleNet 的工作流引擎工作原理

Fig. 3 Principle of RoleNet based workflow engine

下面介绍我们基于该模型设计实现的一个轻量级的工作流引擎 OM Engine。

4　OM 工作流引擎的设计和实现

OM engine 的全称是 Operation and Maintenance engine，顾名思义，该引擎是专门针对通信网络运行维护业务的特点所设计的。通信网络的运行维护业务具有典型的动态业务过程的特点[9]：(1)每个业务步骤都需要人的干涉和参与；(2)业务的处理人并不固定；(3)业务的流转与应用数据相关，并且需要根据人的判断来决定下一步的走向；(4)具有很多回退、自由流等非标准模式；(5)随着新业务的推出，业务流程需要不断进行调整。

由于这样的特点，如果采用传统的以活动为中心的过程模型和引擎设计模式，很难满足客户对自由流，流程动态调整的要求。我们采用 RoleNet 过程模型，参照 WfMC 定义的接口和规范模型，在 .Net Frameworks1.1平台上设计并实现了 OM Engine，并应用在重庆联通运维管理信息系统[10](OMMI)中，较好地满足了客户对动态过程的需求。

4.1　引擎的体系结构

OM Engine 的体系结构如图 4 所示。图 4 中的各部分的功能介绍如下。

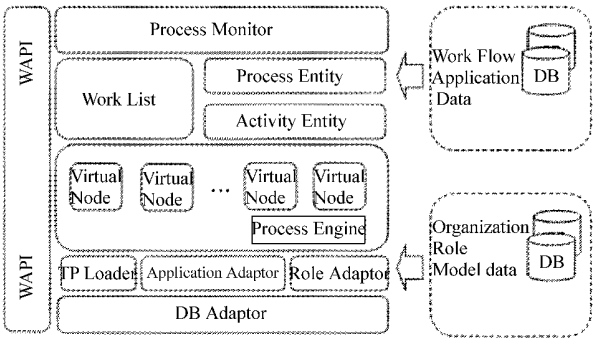

图 4　OM Engine 体系结构图

Fig. 4 Architecture of OM Engine

Process Engine 是 OM Engine 的核心，其主要的功能是负责创建虚拟角色节点、分配活动、调度和

控制流程运转。

TP Loader 为 Process Engine 提供过程模板和路由表解析服务。

Application Adaptor 为引擎调用外部应用程序提供接口和适配器。

Role Adaptor 提供组织模型到角色地址之间的映射,OM 引擎有自己的角色地址编码方案,在实际应用中必须实现组织模型到角色地址的映射。

Virtual Node 是 Process Engine 在流程调度时创建的虚拟节点对象,协助引擎完成角色任务操作、路由转发的功能。

Word List 为任务表,根据过程实例和活动实例产生工作项并负责和用户交互。

Process Entity 过程实例,即运行中的过程对象。

Activity Entity 活动实例,即运行中被角色处理的任务对象。

Process Monitor 流程监控器,通过对 Process Entity 和 Activity Entity 的管理,监控运行中的流程及活动,控制和更改流程状态。

DB Adaptor 数据库访问接口,屏蔽底层数据库特性以支持多个数据库。在 OM Engine 目前的版本中支持 SQL Server2000 和 Oracle9i。

OM 引擎并没有提供过程定义的工具,因为根据 RoleNet 模型建立的过程模板和路由表非常清晰和简单,完全可以直接进行编辑。

4.2 角色编址方案

在 RoleNet 中,必须为每一个角色分配一个惟一的地址。另外,角色的编址方案必须便于和实际的组织结构进行映射转换。现实中主要有树型的层次模型和矩阵模型两种组织模型[11],为了适应这两种模型,我们设计了组 ID 结合成员 ID 的方式来对角色进行编址。

GID 由 8 位整数组成,为角色组 ID,例如 01010201。

MID 也由 8 位整数组成,为属于某一个组的成员 ID,例如 00000001。

RID=GID. MID,即角色 ID 为组 ID 和成员 ID 的组合。例如 01010201. 00000001

4.3 过程模板与路由表

由参考文献[12],我们定义 OM Engine 的运行至少需要如下 3 个描述文档:(1) Workflows. xml 描述所有过程模板的信息(包括名称、类型号、存储位置等);(2) Process_Name. xml 具体某个过程的模板,描述过程的各个迁移状态;(3) Roles. xml 描述系统中所有的角色及其路由信息。

文档(2),(3)是 OM Engine 运行的过程描述语言,采用 XML 定义,图 5 给出了其 UML 描述。

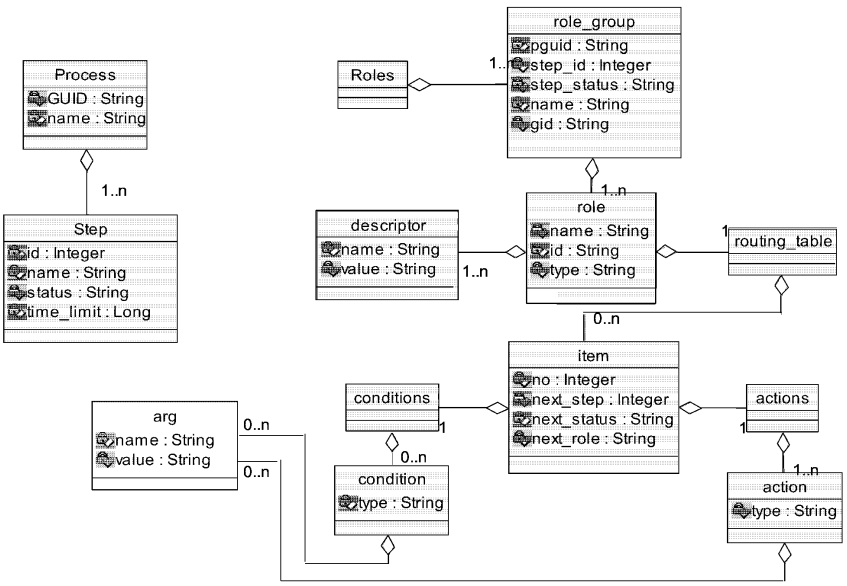

图 5 OM Engine 过程定义语言的 UML 结构图

Fig. 5 UML diagram of OM Engine process definition language

图 5 中的 Process 实体中,可能包含多个步骤(Step),每个步骤的 ID 和 Status 的组合代表流程的一种状态。过程按照路由信息表中定义的逻辑在各个状态间跳转。每个步骤都会有一些参与的角色,称为角色组(role_group)。该组定义了流程在当前步骤参与活动的角色(role)。角色的类型(type)属性将角色划分为自动执行角色(auto_node)和人工交互角色(manual_node),每个角色都

有一个路由表(routing_table)，描述了角色在该步骤状态下可以执行的动作。路由表由多项(item)组成，每一项就是一条路由，包括：next_step 和 next_step_status，指当前活动结束后流程的下一个状态是什么，如果 next_step 为 0 时，代表当前操作执行后流程状态不改变(可以用来处理与合并的情况)，如果 next_step 为 −1，代表当前活动结束后，流程就终止(Terminate)；next_role 指当前活动结束后，要转发的目标角色。元素 Condition 给出了一种扩展机制，使得引擎可以按照用户的要求做一些条件判断，决定流程的走向。条件(Condition)和 next_step，next_step_status 是有约束关系的，也就是说，相同的条件，其 next_step，next_step_status 必须是一致的。元素 Action 指出了角色在当前 Step 要执行的操作，可以调用外部应用来完成一些处理。

4.4　过程实例与活动实例

过程实例是流程启动后的运行实例，活动实例是流程启动后分配给角色节点的任务实例，在 OM Engine 中，它们都在数据库中持久化，工作流引擎在流程调度的时候，不断更新这 2 个实例表，推进流程运行。Worklist 与 Process Monitor 也会访问这 2 个实例表，生成工作项同用户交互，或者对运行中的工作流进行监控和管理。

4.5　核心调度算法

由于引擎在每次调度的时候只考虑单个节点的操作和转发动作，因此 OM Engine 的核心调度算法相当简单，OM Engine 的调度算法过程如下：

(1) 创建当前过程实例 $P \leftarrow q_i$；

(2) 初始化虚拟角色节点 $r_i \leftarrow RT$，活动实例 A_i；

(3) 初始化引擎 $OM\_Engine \leftarrow (r_i, P, A)$，锁定过程实例 P，检查 r_i 所属的过程状态是否与当前过程实例状态 $P.st$，$P.ss$ 匹配，如果匹配则继续执行，否则抛出异常；

(4) 根据当前角色节点的路由表 $r_i.RT$，执行每一项 item 中 condition 定义的外部函数，检查条件是否匹配，执行 Actions 中的操作，执行完毕后，引擎根据 next_step 和 next_status 来推进 Process 的状态，并更新过程实例 P 及活动实例 A_i，分配新的活动 A_{i+1} 给 next_role 中的角色 r_{i+1}；

(5) 如果 $r_{i+1} \cdot type = auto$，启动新的线程，根据 next_roles 创建新的虚拟节点 r_{i+1}，活动实例 A_{i+1} 将 r_{i+1}，A_{i+1}，P 分配给新的引擎；

(6) 引擎结束，释放 P。

4.6　对回退及自由流的处理

OM Engine 对回退和自由流的处理很简单，由于回退实际上可以看作是自由流的一种特例，OM Engine 对自由流的处理机制为：OM Engine 提供了给角色节点加载动态路由表的机制，在初始化一个虚拟节点 $r_i \leftarrow RT$ 的时候，可以不调入角色节点本身的物理路由表，而是根据实际情况，动态生成路由表进行加载。这样引擎在调度的时候，可以将流程导向任意的角色节点(只要满足过程模板 P 的要求)。值得注意的是自由流对系统状态变化的影响，这需要在需求分析时决定，同时应当小心处理动态路由表的数据。

5　结束语

工作流引擎的柔性一直是引擎设计中的难题，所谓柔性，一是指支持在运行中决定流程的走向，二是指引擎应当对业务流程的变动具有一定的适应性。在本文中，我们提出了一种 role-centric 的过程模型 RoleNet，并根据该模型设计实现了 OM Engine，目前，我们以 OM Engine 为核心开发的电子工单已经成功地应用在中国联通重庆分公司运维管理信息系统中，该系统已在重庆联通正常运行 1 年多时间，实践表明 OM Enigne 能够很好地支持动态流程的要求，并能灵活应对客户业务流程变化的需求，验证了 RoleNet 模型的价值。

参考文献：

[1] HOLLINGSWORTH D. The Workflow Reference Model [EB/OL]. (1995-07-27) [2005-09-20]. http://www.aiim.org/wfmc/.

[2] zur MUEHLEN M. History of Workflow Research [EB/OL]. (2003-04-22) [2005-09-20]. http://www.b2b-workflow.de.

[3] REICHERT M, BAUER T, DADAM P. Enterprise-Wide and cross-enterprise workflow management-challenges and research issues for adaptive workflows [EB/OL]. (2004-01-05) [2005-10-11]. http://ftp.informatik.rwth-aachen.de/publications/CEUR-WS/Vol-24/rbd99.pdf.

[4] 胡长城. 工作流之星光 [EB/OL]. (2005-08-10) [2005-09-20]. http://javafox.vip.myrice.com.

[5] AALST Van der. Workflow Patterns [EB/OL]. (2003-03-22) [2005-09-20]. http://www.workflowpatterns.com.

[6] 赵为东. 面向角色的工作流模型研究 [EB/OL]. (2004-08-16). [2005-09-20]. http://www.amteam.org.

(下转 415 页)