计算机科学2004Vol.31№.9A

基于 Agent 的自适应性工作流模型的研究与设计

刘大昕　孙立杰

（哈尔滨工程大学 计算机科学与技术学院　哈尔滨150001）

摘　要　该文在对代理技术和当前工作流技术所面临的问题进行研究的基础上，提出了基于 Agent 的自适应性工作流模型，研究了其中的一些相关算法和关键技术的实现机制。最后讨论了下一步要解决的若干问题。

关键词　代理，自适应，工作流

Research on the Adaptive Distributed Workflow Model Based on Agent

LIU Da-Xin　SUN Li-Jie

(School of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)

Abstract　On the basis of researches of Agent and work-flow, this paper presents a auto-adapt distributed workflow management system based on agents and provides the system architecture. This paper also discusses key algorithm. The paper discusses some questions in the future research.

Keywords　Agent, Adaptive, Workflow

1　引言

工作流是一类能够完全或部分自动执行的经营过程，根据一系列过程规则、信息或任务能够在不同的执行者之间传递、执行。而工作流管理系统是一个通过软件来定义、创建和管理工作流的系统。作为一项集成技术，工作流所要实现的目标就是使企业中大量的基于知识与规则的任务能够相互协调一致，高效运作，在正确的时间内能够将正确的信息传递给正确的人，从而完成正确的业务流程。

1.1　工作流技术面临的主要问题

(1)*适应性差*　现代企业业务流程通常由大量的活动组成和人员参与。这些活动有些可预知，有些则不可预知。因此在工作流的定义阶段，不能全部描述各种活动以及相应的处理措施，即使是一个定义好的流程在执行时也会发生变化。同时，由于流程模型的不完善、执行过程中发生变化等原因所导致的异常事件出现也是不可避免的。上述这些都要求工作流管理系统具有较好的自适应性：

·在不中断执行的情况下把某条事先没有放入模型的执行路径加进模型，并按该路径继续执行。

·自动识别不同类型的例外情形并对其做出适当的处理，保证工作流继续正常运行。

(2)*缺少自学习能力*　目前的工作流系统能力都还有限，或是适用面很窄，或是只能解决较简单的问题，其重要原因之一是不具有学习能力，系统不能从解题实践或用户提供的实例中自动获取问题求解所需的知识。

1.2　问题的分析及相应的解决方法

当传统的系统面临诸多问题时，代理技术为解决上述问题提供了一种新的思路。代理是一种处于特定环境下，能感知环境的计算机系统。它能够实现设计者和使用者的一系列目标，并能在那种环境下自主的运行计算实体或程序。它具有以下特征：

(1)*代理性*　Agent 的最基本功能是“代理”用户或软件完成某些任务。如“代理”用户查找因特网上的信息，“代理”软件与其它软件进行通信。

(2)*自主性*　它能根据当前动态变化的环境状态，在无需外界参与的情况下，独立的发现和利用完成任务所需的资源和服务，独立的制定完成任务的规则，最终实现规划，达到目标。

(3)*推理/学习/自适应能力*　Agent 的智能由三个主要部件来完成，即内部知识库、学习或自适应能力以及基于知识库内容的推理能力。

(4)*可移动性*　一个 Agent 在计算网络中漫游的能力。这一特性概括了基于 Agent 的分布式计算的特点，即将计算移往数据，而非将数据移往计算，这样做所带来的好处是可以减少网上数据的流量。

由于 Agent 具有上述优越性，使它在工作流应用中具有很大的潜力。它不仅可以将信息相关的操作封装在移动 Agent 之间，再利用后者的移动性和自治性，让信息在机构内部有效地流动。同时，其推

刘大昕　博士生导师，孙立杰　硕士研究生，主要研究方向：数据库与知识库。

理/学习/自适应能力使得基于 Agent 的工作流系统能够很好地适应动态环境的变化。

本文通过研究分析代理技术和当前工作流产品在应用于大规模,分布式的企业应用环境中所面临的问题,提出了一个基于 Agent 的自适应性工作流模型,讨论了过程执行中的相关算法和关键技术的具体实现。在保证系统可靠性的同时,增强系统的灵活性,以提高系统的自适应能力。

2 相关概念和定义

首先定义相关概念,在此基础上定义具有自适应性的工作流模型。

定义1(工作流) 它是一个大的活动。用 P 表示,每一个活动有其内在的运行机制,依据特定的输入,产生特定的输出。

定义2(活动或任务) 它是工作流中可调度的最小单位,可以表示为 Ts=〈N,T,R,Ap,St,Et,Pa,Fa,Id,Od〉。其中 N 表示任务或活动名称或标识;T 表示类型;R 表示参加者角色;Ap 表示指定活动的应用程序,包括程序所在的节点目录和位置;St 表示活动的开始时间;Et 表示结束时间;Pa 表示前驱活动集合;Fa 表示后继活动集合;Id 表示从前驱活动传来的数据集合,即输入数据集;Od 表示从当前活动传到后继活动的数据集,即输出数据集。

定义3(过程实例) 它是工作流的一次执行过程,可以表示为 Pe=〈N,W,S,E,T〉。其中 N 为过程实例的名称;W 为相应的过程定义的名称;S 和 T 分别为实例的开始和结束时间;T 为过程实例的当前运行状态。

定义4(组织属性) 它可以形式化表现为 Ha=〈N,I,T,K,R,A,L〉。其中 N 为参与者的名称或标识,应具有唯一性;I 为参与者所在主机的 IP 地址;T 为其目标集,描述所需完成的任务,并包含推理机制;K 为知识源,描述行为所需的知识;R 为参与者的角色;A 为信任度值,它是动态变化的;L 表示负载信息。

定义5(ECA 规则) 即 event-condition-action 规则。其中 E 代表出现一个异常事件;C 代表异常出现后,采取处理措施所满足的条件;A 表示条件满足后采用的处理措施,处理措施实际上是另外的一些可以执行的活动实例。ECA 规则的一般形式定义为:

Rule 〈规则标识〉
While 〈异常事件〉
If 〈条件表达式〉
Then 〈行为动作〉
…
End

定义6(补偿) 在工作流执行过程中,当执行到一个过程实例出现异常时,已执行的活动实例已经产生了一些影响,为了使整个流程能通过其它路径继续下去,必须对产生的影响进行消除。这就是通过补偿完成的。例如:有这样一个流程,如图1所示。

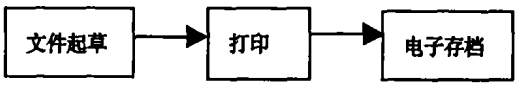

图1 起草文档

若"打印"活动发出一个打印进程后,发现打印机故障,且有可替换的打印机,则对"打印"活动的补偿活动就是杀掉打印进程,更换打印机,然后向新打印机发打印命令。

3 系统实现方案

基于系统的总体结构设计,采用了 Agent 技术,设计了基于 Agent 的自适应性工作流模型,以解决如下问题:

·用户能在运行中动态修改工作流模型,在某些节点之间加入一条迄今未预见的执行路径,以适应应用场合的需要;

·对工作流执行过程中条件不满足或发生变化引起的异常或中断进行处理。

3.1 系统结构

各部分的主要功能如下:

(1)*接口 Agent* 主要功能:(a)代替用户发出过程启动请求;(b)代替用户自动处理消息或等待用户处理,并将处理过程存入内部的知识库中;(c)利用自身的自学习能力,为用户提供处理建议。

(2)*调度 Agent* 主要功能:(a)负责接收用户的过程启动请求,指导建模工具进行过程分片,并根据分片结果进行任务调度;(b)向异常管理模块发出异常处理请求;(c)当用户需要修改当前工作流程定义时,向当前执行节点发出暂停要求,直至修改完毕,通知该节点继续执行;(d)接收任务执行过程数据,分析每个任务的执行情况,更新组织数据库中执行此任务的组织属性;(e)创建并分派携带有任务相关数据的移动 Agent。

(3)*建模工具* 负责定义和修改工作流模型的基本要素和它们之间的相互关系。经过授权的用户还可以动态改变运行中的工作流模型。监控器出现在建模工具中,它负责监视工作流程模型是否被修改,以便及时通知调度 Agent。由于工作流模型将会在一个分布式的环境中运行,这就需要对现有过程模型进行合理的整理与分片,然后再分别把整理后的各个部分传递给相关的工作流引擎,因此本文的工作流建模工具还提供过程分片功能。

(4)*任务配置信息库* 负责存储与流程中各任

务有关的信息。把任务看作是一个对象，每个对象有独立的变量用来记录它的前驱节点，后继节点和执行该任务的角色等。同时，由于对象的独立性和所有信息都由对象本身的变量存储，所以，只要该任务没有被执行，都可以任意灵活地更改这些信息，具有足够的柔性，且不会影响到整个工作流的运行。

(5)选择 Agent　接收调度 Agent 传来的有关下一步任务的信息，选择最适合执行此任务的节点，传递该节点的相关信息至调度 Agent。

(6)组织数据库　记录所有参与流程执行的组织属性，为选择 Agent 提供相关数据。

(7)移动 Agent 服务设施　提供 Agent 的各种功能支持，包括创建、运行、挂起、终止 Agent、接收 Agent、保护验证 Agent 等工作，为移动 Agent 创造一个位置透明、便于控制、安全可靠的运行环境。

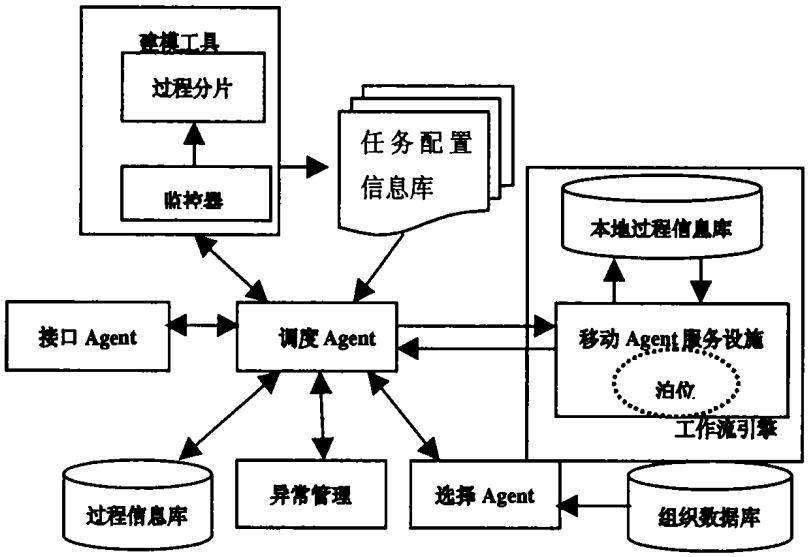

图2　系统结构图

(8)泊位　它是移动 Agent 服务设施上的一个虚拟的位置，负责管理移动 Agent 的到来和离开，并为其提供可以访问的资源。

(9)本地过程信息库　记录和维护本地工作流引擎执行过程实例中产生的相关数据。

(10)异常管理　接收并处理工作流执行过程中产生的异常或中断，保证工作流在发生意外的情况下能够及时处理异常并继续正常工作。它的内部结构如图3所示。

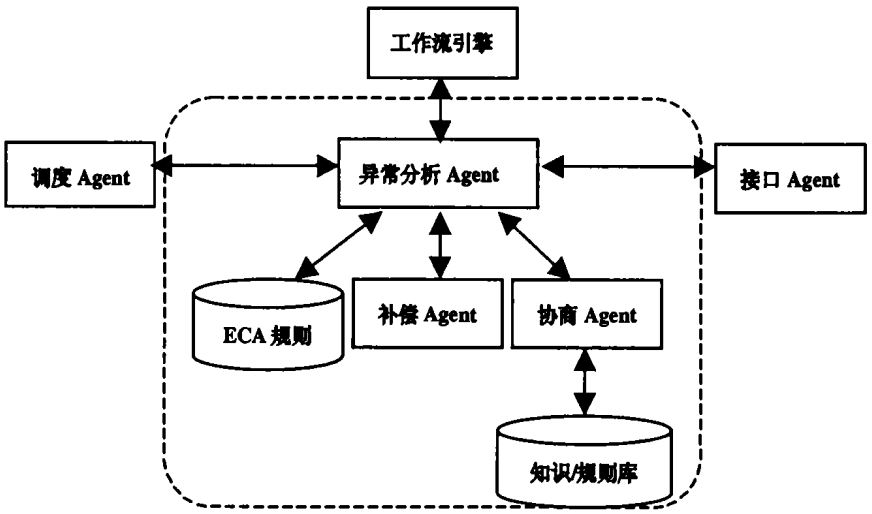

图3　异常管理 Agent 的内部结构

3.2　过程执行及自适应算法

1 过程配置分解：

当调度 Agent 收到客户发出的启动请求后，向工作流建模工具发出请求，调用其中的过程分片模块，将该过程定义分解成为一个个任务的配置信息，并存在任务配置信息库中。

2 实例启动：

a)调度 Agent 获取某个任务的配置信息，将任务名称，执行该任务所需的角色等信息传递给选择 Agent。

b)选择 Agent 根据接收的信息，在组织数据库中查找最适合(角色匹配，信任值高且负载低)解决此类任务的节点。如有，返回节点参数(节点 ID，节点 IP)至调度 Agent；否则，向调度 Agent 发出异常请求。

c)调度 Agent 接收到节点的信息后，利用 ATP (agent Transfer Protocol)协议创建一个任务移动 Agent，并对其进行状态，知识库等的初始化：将下一个任务的配置信息和控制信息(过程实例启动命令)写入 agent 的知识并打包，同时对包加入安全性措施保证。然后按照接收到的节点 IP 地址分派此 agent 到目标系统。

3 实例执行：

由调度 Agent 分派的任务移动 Agent 到达目标系统后，移动 Agent 服务设施对此 Agent 进行口令和身份验证等合法性检查工作，然后对其解包；该任务移动 Agent 根据任务配置信息调用具体的应用程序，在目标系统中执行指定的任务；执行过程中，如产生异常(资源冲突，执行超时等)，移动 Agent 服务设施保存当前工作流状态参数至本地过程信息库并挂起过程实例的执行，然后发送异常消息给调度 Agent，转5；否则，将任务执行过程数据，当前节点负载信息以及执行完毕的控制信息等数据写入任务移动 Agent 中，由它携带这些信息返回到调度 Agent，转4；

4 实例调度

调度 Agent 接收任务移动 Agent 返回的已完成任务相关信息，包括任务名称、代号、执行时间、完成时间、执行中产生的数据和执行结果等实例的数据信息和负责执行此任务的节点负载信息等，将实例的数据信息保存到过程信息库中；根据这些信息对任务执行情况进行分析，重新计算此节点执行该任务的信任值并修改组织数据库中对应节点的信任值和负载信息值，以供下次调度参考；最后读取任务配置信息库，取得下一个任务的配置信息，转2；

5 异常处理

(1)异常分析 Agent 接收调度 Agent 传来的异常信息，扫描 ECA 规则库，如果是可预测异常，则触发相关的可执行规则；

(2)如果是不可预测异常，分析该异常信息并确定异常处理措施；具体如下：

a)若该异常所产生的行为不影响其它任务的执行，则采取忽略策略，发送处理结果至调度 Agent；调度 Agent 通知当前执行引擎中的移动 Agent 服务设施恢复到异常发生前的状态，继续执行当前任务；同时形成新的 ECA 执行规则，并保存到 ECA 规则库中；

b)若该异常对整个流程的执行产生影响，但通过补偿可以使流程继续执行下去，则通知补偿 Agent 采取补偿措施，转3；

c)若该异常使当前任务不能继续，但存在另外一条可选执行路径时，通知调度 Agent 执行 Rollback 动作，到上一步正确的状态，重新调度，转3；

d)若该异常需要人工处理，通知相应的接口 Agent 处理该异常，转3；

e)若该异常的解决需要借助其它节点的知识或与其它节点相互协商才能解决时，通知协商 Agent 与其它节点进行谈判，转3；

(3)异常处理完成后，发送处理结果至调度 Agent，调度 Agent 创建一个移动 Agent，将异常处理结果写入该 Agent 的知识库中，分派其到目标系统，执行引擎中的移动 Agent 服务设施接收该移动 Agent，恢复到异常产生前的状态，继续执行。

3.3 过程模型的动态修改过程

过程模型的动态修改是指在工作流执行过程中，对过程定义模型的修改。随着市场的全球化，企业业务流程的复杂性正在一步步地加大，在工作流执行过程中，往往会出现这样一种情况，即在工作流执行期间，由于某种原因希望对该工作流程增加一个任务节点，从而处理各种异常或紧急情况。但是传统的修改方法是终止当前过程实例的执行，进行修改，然后按照新模型重新运行。这种方法比较死板，不能满足企业动态变化的需要。因此本文提出了下面的修改方法：

(1)工作流建模工具中的监控器实时监视当前工作流程模型是否被修改，如果发现授权人员修改模型，传递一个消息给调度 Agent；

(2)调度 Agent 接收到消息后，通知当前执行引擎中的移动 Agent 服务设施保存当前工作流状态参数至本地过程信息库，并挂起当前过程实例的运行；

(3)监控器监控修改过程直至完毕，启动过程分片模块对该过程模型重新分片并更新任务配置信息库；

(4)向调度 Agent 发送修改完毕的控制信息，调度 Agent 通知该移动 Agent 服务设施恢复到挂起前的状态，继续执行过程实例，同时按照修改后的过程模型进行任务调度。

从上面的修改方法中可以看到，对原先定义好的过程模型的修改是在工作流执行过程中进行的。实际上，本文提出的动态修改不单指在某一条路径上处理一些紧急情况，还包括其他一些修改，如修改

任务节点的属性、增加新的执行路径等，使得本文提出的工作流模型能够有效地支持某种'随机应变'的机制。

3.4 移动 Agent 的具体实现

工作流应用的特点之一是支持企业信息流动。对此，移动 Agent 技术能够提供很好的支持，本文提出的工作流模型正是借助移动 Agent 的移动性和自治性，将执行每个工作流活动所需要的行为和数据封装到移动 Agent 中，然后将它通过网络派遣出去。在该系统的具体实现中，采用基于 Java 的移动 Agent 系统 Aglet 来实现有关移动 Agent 的移动性和安全性(利用 com.ibm.aglets.Security 包)、系统设计样式(利用 com.ibm.aglets.pattern 包)、互操作性(利用 com.ibm.maf.MAFAgentSystem 包)等要求，并给出框架代码：

```
public abstract class Master extends Aglet{
    public void getResult();//得到移动 Agent 返回的信息
    }
public abstract class Slave extends Aglet{
    public void run(){
      initializeJob(); //初始化参数
      dispatch(destination);//分派到目标系统
      doJob(); //在目标系统中执行任务
      dispose(); //返回结果，然后清除
    } }
```

下面是具体的调度 Agent(DispatchMaster 类)和具体的任务移动 Agent(ExecuteTask 类)代码：

```
public final class DispatchMaster extends Master{
      public getResult(Object result){
   //重载调度 Agent 中的 getResult 函数，把得到的结果存入相应的数据库中
      }
  }
public final class ExecuteTask extends Slave{
  SimpleItinerary itinerary=null;
  protected void initializeJob()throws Exception{
   //具体的初始化工作，如封装任务配置信息
   }
  protected void dispatch(String des){
   try{
    itinerary=new SimpleItinerary(this);
    itinerary.go(des,"doJob"); //到达目标系统，执行下面的 doJob 操作
   }catch(Exception ex)
   {ex.printStackTrace();}
   }
  public void doJob()throws Exception{
   //在目标系统中执行具体任务
}}
```

结束语 随着企业应用规模和功能的不断增大和调整，基于工作流技术的应用系统必将是将来软件技术的发展方向之一。该文通过分析当前工作流系统的一些不足之处，提出了一种基于 Agent 的自适应性工作流管理系统，研究了其中的一些关键算法和实现机制。论文未来的研究工作是与知识工程等领域的研究结合起来，进一步完善基于 Agent 的自适应性工作流管理系统。另外，增加工作流的自学习功能也是未来的研究方向之一。

参 考 文 献

1 罗海滨，范玉顺，吴澄. 工作流技术综述[J]. 软件学报，2000，(11)
2 Workflow Management Coalition. The Workflow reference model [R], WFMX TC00-1003, 1994
3 Chin Dickson K W, LI Qing, Karlapalem Kamalakar. A meta modeling approach to workflow management systems supporting exception handling [J]. Information Systems, 1999, 24(2): 159～184
4 Arpinar S, Dogac A, Tatbul N. An Open Electronic Marketplace Through Agent-based Workflow: MOPPET. Intl. Journal on Digital Libraries, 1998
5 Koksal P, Cingil I, Dogac A. A Component-based Workflow System with Dynamic Modifications. In: Proc. of the Next Generation Information Technologies and Systems (NGITS99), Istael, 1999

(上接第245页)

GSA 算法，对 W_{Que} 中的服务分组并选择一个分组提交给服务机构处理。分组调度结束后，服务机构忙，因此 GSA-SA 进入"忙等"状态，一直到服务机构完成服务并产生服务机构"闲"事件，此时转入"空等3"状态，等待与缓冲区有关的事件发生。根据缓冲区产生的不同事件 Lower、Empty、Normal 而分别转入"空等1"、"空等2"、"分组调度"状态。

图8 分组调度算法状态转换图

结束语 成批服务的系统是一类特殊的随机服务系统，它有较广泛的应用背景。但是在实际当中，成批服务一般是作为生产或者是商务流程中的一个或者数个环节而存在。工作流管理作为支持商务流程处理的有力工具，受到了学术界与产业界的广泛关注，但是现有的工作流管理没有考虑到该问题，因此也未提供相应的支持机制。本文总结了工作流中出现成批服务的8种不同的 X/Y/Z 形式，建立可支持成批服务的工作流模型。分析了支持成组处理的工作流执行服务系统设计与实现会带来的新问题，利用状态图与事件的方法分析了 K/1/1模型的优化调度问题。

参 考 文 献

1 徐光辉. 随机服务系统. 北京：科学出版社，1980
2 王秀利，吴惕华. 一种求解两机成组作业流水车间优化调度问题的遗传算法. 系统仿真学报，2001，13(8)：88～90
3 胡锦敏，张申生，余新颖. 基于 ECA 规则和活动分解的工作流模型. ，软件学报，2002，13(3)：761～767
4 刘建勋，胡涛，张申生. 基于工作流与 XML 的敏捷供应链管理系统集成框架研究. 计算机集成制造系统－CIMS，2001，7(6)：6～35
5 罗海滨，范玉顺，吴澄. 工作流技术综述. 软件学报，2000，11(7)：899～907
6 Goh A, Koh Y K, Domazet D S. ECA rule-based support for workflows. Artificial Intelligence in Engineering, 2001, 15(1): 37～46
7 Liu Jianxun, Yao Yingxiong, Tang Xinhuai. Pre-dispatching of tasks in workflow: the concept, model and implementation. Chinese Journal of Electronics, 2004, 13(2)