一种支持分布式工作流模型的工作流管理系统

李　宁, 刘厚泉
(中国矿业大学 计算机科学与技术学院, 江苏 徐州 221008)

摘　要: 通过研究分析当前对跨企业工作流进行集成时所面临的问题, 提出了一种支持分布式工作流模型的工作流管理系统, 阐述了系统的设计原理和总体结构, 并给出了一个跨企业工作流管理的实例。

关键词: 工作流; 分布式工作流模型; 工作流管理系统
中图法分类号: TP319　　**文献标识码**: A　　**文章编号**: 1001-3695(2006)03-0205-03

A Workflow Management System Supporting Distributed Workflow Model

LI Ning, LIU Hou-quan
(School of Computer Science & Technology, China University of Mining & Technology, Xuzhou Jiangsu 221008, China)

Abstract: Based on the study of integration in cross-Enterprise workflow, this paper puts forward a workflow management system which supports the distributed workflow model. It also expounds the design principle and its main structure with an example.

Key words: Workflow; Distributed Workflow Model; Workflow Management System

工作流技术通过信息技术的支持为企业的业务过程提供了一个从模型分析、建立、管理、仿真到运行的完整框架, 是实现业务过程管理与控制的一项关键性技术。目前, 工作流技术已经成为了计算机应用的一个重要研究方向, 对工作流的技术研究以及相关的产品开发进入了一个繁荣的阶段。在当前工作流技术的研究领域中, 分布式工作流系统已逐步成为研究的热点。现有的分布式工作流系统可以在一定程度上解决集中式系统所固有的性能瓶颈、单点失败与可扩展性差等问题。

但是随着社会发展, 社会分工越来越明显, 市场逐渐细化。一个企业一般只有一个主导产品、或一个生产、技术优势。在一个企业的工作流过程中, 一个活动的运行, 往往需要使用别的企业所提供的工作流过程。另一方面, 在企业内部, 也需要将原先开发的各个子系统的工作流程集成起来。因而需要有一种柔性机制来将多个工作流过程连接起来。

为解决这个问题, 很多学者都进行了相关的研究。文献[1] 提出了一个柔性工作流系统的体系结构。在分析各个企业的工作流程的基础上, 直接建立统一的工作流过程模型。但是, 在实际使用过程中, 要将这些伙伴企业的相关业务过程全面了解清楚是一件费时费力的工作, 一个企业的业务流程一般不会对外开放, 所以建立一个全局的工作流模型就不合适了。文献[2] 将各个企业的过程包装为 Web 服务, 并利用服务的发现机制将这些服务连接起来, 从而达到工作流过程集成的目的。但由于所提供的服务的发现机制尚不完善, 一旦查找不到或查找出错, 将直接导致流程无法集成。

本文提出分布式工作流模型的概念, 即模型的分布式定义及模型的分布式执行。在定义阶段, 各个企业定义自己的工作流过程, 并通过顶层过程将这些过程连接起来。在执行阶段, 通过代理(Proxy) 控制各个企业执行自己的工作流过程。在此基础上, 开发了一个支持分布式工作流模型的工作流管理系统, 能够很好地解决跨企业流程集成的问题。

收稿日期: 2005-03-24; 修返日期: 2005-05-28

1　分布式工作流模型

传统的工作流管理系统集中式地定义工作流模型, 模型的建立一般采用自顶向下的建模方法。工作流模型由工作流定义人员使用工作流模型定义工具, 按照逐步分解、逐步细化的方法完成。这样做的前提是工作流定义人员熟悉全部的企业业务过程, 并且这些业务过程对他们都是开放的。因此集中式定义适用于描述某个组织结构下的流程。

如果所要建立的工作流模型描述的是涉及多个合作企业的工作流程, 则集中式工作流定义就不适用了。而分布式工作流模型(图 1) 就能解决这样的问题。

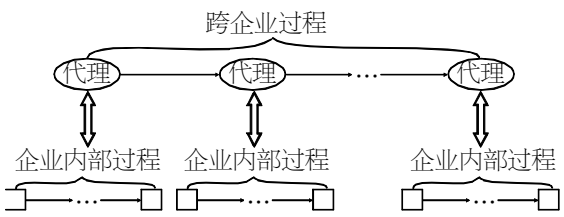


图 1　分布式工作流模型

本文中的分布式工作流模型包含两个方面的含义:

(1) 工作流模型的分布式定义, 即在一个分布的环境下由参与人员协作完成工作流模型的定义。模型的建立过程是自顶向下分解和自底向上综合地结合。首先, 自顶向下的分解过程完成单个企业的内部工作流模型的建立。在各个伙伴企业完成了其内部流程模型后, 采用自底向上的综合方法进行模型的连接, 即建立跨企业的过程。建立跨企业过程时以代理作为中介, 将各个企业内部过程集成起来, 而不是直接将各个企业内部过程连接起来。跨企业过程中的每个代理相当于一个步骤, 它对应着一个企业内部工作流模型, 而企业内部工作流模型是存储在各个企业内部的, 属于企业的商业秘密, 由各个企

业分别维护。

(2) 工作流模型的分布式执行。跨企业过程中,每个代理负责执行相应的企业内部过程,并由各个企业自己的工作流引擎负责运行。代理作为一个中介,将跨企业过程和企业内部过程联系起来。跨企业过程运行时,将任务交给各个代理,由代理去推动相应的企业内部过程的运行,并将运行结果返回给跨企业过程。

2 系统模型

通过上述分析,提出了一种支持分布式工作流模型的工作流管理系统。本系统能够很好地支持多个企业的工作流集成,具有以下几个方面的优点:

(1) 各企业建立自己的工作流模型,然后由工作流定义人员根据需要将其连接起来,而无须在全面了解这些企业的工作流模型之后,再建立全局模型。这就大大提高了建立模型的速度。

(2) 各企业的内部过程仍然存放在各个企业内部,保护了企业的商业秘密。

(3) 系统具有良好的扩展性,随时可以加入新的企业过程。

(4) 企业子过程的定义被发布为 Web 服务,在执行到该子过程前才调用该 Web 服务以读入过程定义,因而在执行子过程前可以随时修改过程定义,提高了系统的柔性。

系统分为两个部分,定义态系统和运行态系统(图 2)。

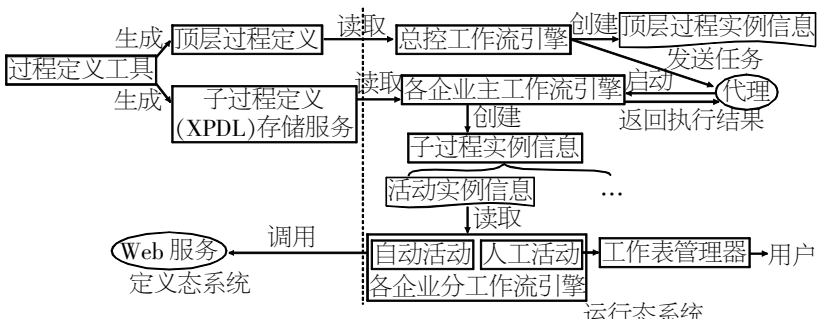


图 2 系统模型图

定义态系统包括工作流过程定义语言、工作流定义工具、顶层过程定义和子过程定义存储服务。

(1) 子过程是各个企业内部的流程,由各企业通过工作流定义工具自行定义,生成工作流程,最终发布为子过程定义存储服务。其他授权程序调用此 Web 服务后得到的是该子过程的定义。

(2) 工作流定义工具定义子过程后,将会在总控工作流引擎所在机器上进行注册,注册时,指明该子过程的作用、需要输入的数据、将会返回什么数据、所属企业和该子过程定义存储服务的调用地址。

(3) 有相关权限的用户可以根据这些子过程的描述,直接使用这些子过程来定义顶层工作流过程。顶层过程信息存储在总控工作流引擎所在机器上。

(4) 顶层过程是跨企业的工作流过程,由一系列代理组成,每个代理都有两个最主要的属性:①该代理的相关描述;②该代理所对应的子过程及其定义存储 Web 服务调用地址。

运行态系统分为两个层次,即高层和低层。

(1) 高层中由总控工作流引擎负责顶层过程的运行。总控工作流引擎读取顶层过程定义,然后创建顶层过程实例信息,并将任务发送给相应的代理。这里的代理作为一个独立的实体,能够负责顶层过程中相应步骤的执行,并接收执行结果。代理接收到任务之后会去执行,并将结果返回给总控工作流引擎。

(2) 低层中由代理和各企业工作流主引擎进行交互。代理启动企业工作流主引擎以执行相应的子过程,执行完毕后接收结果,然后向总控工作流引擎发送执行完成消息及执行结果。企业主工作流引擎及分工作流引擎负责子过程的运行。企业主工作流引擎被启动后,就读取子过程定义存储服务,并进行解析,得到子过程模型,然后创建子过程实例信息。企业主工作流引擎对子过程实例进行分解,由多个分工作流引擎共同执行。模型中的活动分三类,即自动活动、人工活动和内部活动。自动活动在系统中实现为对一个具体 Web 服务的访问,它用于跟外部各类业务应用交互,将各类异构的应用结合进业务流程;人工活动,即定义了需要人参与的活动;内部活动则主要完成数据转换等内部操作。

3 一个例子

以供应链中的订单处理为例。购买企业制定购买计划,并提交订单。生产企业接收订单并进行处理,生产出产品后,提交配送请求。物流企业收到配送请求后进行处理,然后配送。整个流程由总控工作流引擎、企业主工作流引擎、企业分工作流引擎共同执行。总控工作流引擎负责推动顶层过程的运行。每个企业有一个工作流主引擎和多个工作流分引擎,负责推动各企业子过程的运行,如图 3 所示。

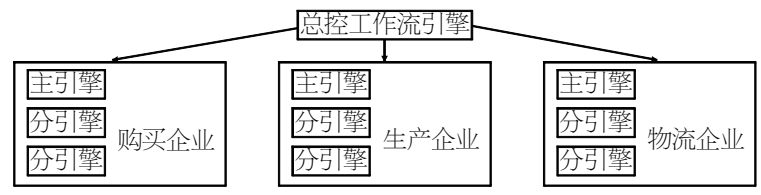


图 3 系统结构

3.1 工作流定义阶段

这几个企业分别使用过程定义工具定义了自己的子过程,并分别发布为过程定义存储服务。各个子过程的具体模型定义并不在总控工作流引擎上,而是在各企业内部。

购买企业定义了订购过程,如图 4 所示。

生产企业定义了生产过程,如图 5 所示。

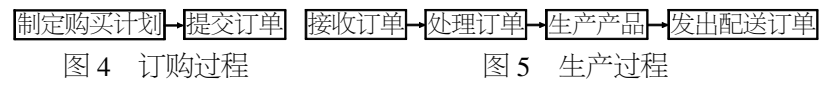


图 4 订购过程 图 5 生产过程

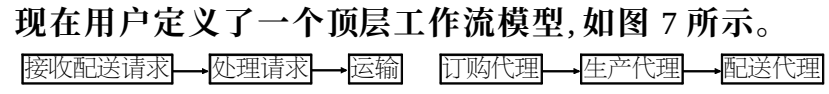
物流企业定义了配送过程,如图 6 所示。

经过程定义工具定义一个子过程后,则会在总控工作流引擎上的子过程表中添加一条与子过程相关的记录,其子过程表如表 1 所示。

表 1 子过程表

序号	子过程描述	需要输入的内容	将会输出的内容	所属企业	子过程定义存储服务调用地址
1	订购产品	订购要求	产品订单	购买企业	地址 1
2	生产产品	产品订单	配送请求	生产企业	地址 2
3	进行配送	配送请求	配送信息	物流企业	地址 3

现在用户定义了一个顶层工作流模型,如图 7 所示。



e

图 6 配送过程 图 7 顶层过程流程

然后为每个代理配置对应的子过程。顶层过程表如表 2 所示, 顶层过程代理表如表 3 所示。

表 2　顶层过程表

序号	顶层过程描述
1	采购产品

表 3　顶层过程代理表

顶层过程序号	代理描述	对应子过程序号
1	订购代理	1
1	生产代理	2
1	配送代理	3

工作流模型如图 8 所示。

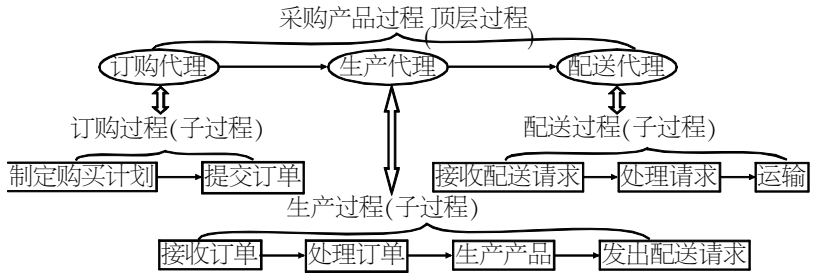


图 8　工作流模型

3.2　工作流运行阶段

工作流运行的主要过程如下:

(1) 用户发出启动顶层过程的指令。

(2) 总控工作流引擎创建一个顶层过程实例, 并给订购代理发送启动消息, 由订购代理负责订购过程的执行。

(3) 订购代理启动后先读子过程表, 发现订购过程属于购买企业, 则向购买企业的主引擎发送消息, 要求该主引擎创建并启动订购过程的实例, 消息中还包含该过程定义存储服务的调用地址。

(4) 购买企业主引擎收到该消息后, 首先调用订购过程定义存储服务, 得到过程定义后, 进行解析, 得到该过程的执行路径信息。

(5) 购买企业主引擎将订购过程的执行路径信息进行分解, 得到一个个节点的配置信息。这些配置信息将被发送到该公司的各个分工作流引擎上, 然后由这些分引擎共同执行订购过程。

(6) 执行活动时, 如果该活动是自动活动, 则由主引擎调用相关 Web 服务, 并接收相关数据。如果该活动是人工活动, 则由主引擎将该活动的描述和相关数据推送到相应用户界面的工作表上, 由该用户进行相应的处理, 并通过用户界面将结果反馈给主引擎, 然后执行下一个活动。

(7) 订购过程执行完毕后, 由购买企业主引擎将产品订单反馈给订购代理, 订购代理将产品订单反馈给总控工作流引擎。

(8) 总控工作流引擎创建生产代理, 并把产品订单传递给生产代理。生产代理负责生产过程的运行。具体过程同订购代理。最后生产代理将配送请求传递给总控工作流引擎。

(9) 总控工作流引擎创建配送代理, 以负责配送过程的运行, 配送代理执行完毕后, 整个流程结束。

4　系统实现方案

基于系统的模型设计, 我们设计实现了支持分布式工作流模型的工作流管理系统。下面将从过程定义语言、代理、对自动活动的处理等几个方面讨论系统的实现机制。

4.1　过程定义语言

本系统采用 XPDL(XML 过程定义语言) 作为过程定义语言。XPDL 是工作流管理联盟于 2002 年 7 月发布的基于 XML 的工作流过程定义语言。通过 XPDL, 可以方便工作流过程定义在不同的工作流产品之间进行交换。

下面是 XPDL 工作流元模型的主要组成部分:

(1) 过程定义。定义过程的模型, 包含过程的详细信息、过程所含活动、转移的列表等。

(2) 活动定义。描述工作流过程中一个步骤的信息, 包含活动的基本描述、活动参与者、活动优先级、最终期限等。

(3) 转移定义。定义该转移的源活动、目的活动和转移条件。

(4) 工作流参与者说明。描述活动执行者资源的信息。

(5) 工作流相关数据。定义在流程运转期间各个流程实例内创建及使用的数据。

4.2　代理

代理在定义顶层过程时被创建。代理作为一个独立的实体, 能够负责一个子过程的运行及接收运行结果。下面是代理类的基本定义:

```
Class ProcessProxy
{
public:
        int MainProcessID; //顶层过程 ID
       int SonProcessID; //子过程 ID
       AnsiString WebservicesURL; //子过程定义存储服务
       bool stratcondition; //代理启动条件
       void Start( ); //启动代理
        int GetMainEngineID( );
       //读子过程表, 以取得该子过程所对应的企业主引擎
       void StartMainEngine( ); //启动企业主引擎
       void SendMsgToMainEngine( );
       //向企业主引擎发送子过程定义存储服务等相关信息
       void SendResultToMainControlEngine( );
       //将子过程执行结果反馈给总控工作流引擎
       void GetMsgOfMainEngine( bool);
       //接收企业主引擎发来的消息
       void ManageError( ); //进行异常处理
       ProcessProxy( int, int, AnsiString);
};
   ProcessProxy: : ProcessProxy( int mpID, int spID, AnsiString wsURL)
{
    MainProcessID = mpID;
   SonProcessID = spID;
   WebservicesURL = wsURL;
}
   void ProcessProxy : : Start( )
{
       if( stratcondition = true)
{ //满足启动条件
int MainEngineID;
MainEngineID = GetMainEngineID( );
StartMainEngine( MainEngineID);
SendMsgToMainEngine( MainEngineID);
}
else
{
ReturnErrorMsg( ); //返回失败信息
      }
   }       void ProcessProxy : : GetMsgOfMainEngine( bool complete) {
      if( complete = true)
      { //如果成功完成, 则将过程执行结果反馈给总控工作流引擎
       SendResultToMainControlEngine( );
}
   else
{
      ManageError( ); //进行异常处理
}
}
```

(下转第 213 页)

pression > 通过如下范式来定义

< Criterion-expression > : : = < Item > | < Criterion-expression > or < Item >

< Item > : : = < Factor > | < Item > and < Factor >

< Factor > : : = < Criterion-expression >) | < Ontology-factor > | not < Criterion-expression >

< Ontology-factor > : : = < Concept > | literal | relation (< Ontology-factor >)

< Concept > : : = class | instance

过滤条件本质上是一个逻辑表达式, 但是这里的 < Factor > 基于 Ontology 的词汇集和操作符进行了扩展: 它可以是概念、关系或者 Literal(即关键词) 。

Ontology 计算引擎就是将基于 Ontology 词汇集和操作符描述的过滤条件表达式解释为基于关键词描述的布尔表达式, 其解释方法如下:

(1) 计算概念关系子表达式 Relation(< Ontology-factor >) , 它将被解释为具有 Relation 关系并且基于 < Ontology-factor > 表达的概念对应的所有 Classes(类) 和 Instances(实例) 。重复该步骤直到条件表达式中的所有关系表达式都计算完。现在过滤条件表达式中仅仅包含 Classes(类) 、Instances(实例) 和 Literals (即关键词) 。

(2) 将上一步得到的结果中的所有 Classes 和 Instances 用 Ontology 中 Classes 和 Instances 对应的标签词汇来替换。

这样, 过滤条件就被转换成了基于关键词(Literals) 的普通布尔表达式, 该布尔表达式即可通过 Jena RDQL(RDF 查询语言) 来检索符合条件的元数据。

4　结论和进一步工作

基于元数据来描述网络资源使得机器对网络资源的智能过滤更加有效, 而基于 Ontology 计算使得概念相关的过滤成为可能。例如用户如果想过滤掉 "法轮功 "相关的网络资源, 传统基于关键词分析的过滤不能将 "李洪志 "相关的网络资源过滤掉, 而基于 Ontology 计算不仅可以将 "李洪志 "相关资源过滤掉, 而且通过对概念所属领域的进一步限制, 可以只过滤宣传法轮功的网络资源。总体来看, 基于元数据和 Ontology 计算的信息过滤具有较高的准确率和召回率。目前对于网络资源中的元数据的挖掘还不精确, 领域 Ontology 主要还是通过领域专家手工构建和验证, Ontology 的描述还不够准确, 因此, 下一步我们将进一步研究更加准确和高效的元数据挖掘方法和 Ontology 的构建方法, 从而不断改进系统的过滤效率和准确率。

参考文献:

[1]　im Berners-lee. The Semantic Web[EB/ OL] . http: / / www. w3. org/ 2001 / sw/ #spec . Scientific American, 2001.

[2]　Tom Gruber. What is an Ontology? [EB/ OL] . http: / / www-ksl. stanford. edu/ kst/ what-is-an-ontology. html.

[3]　Dublin Core Metadata Element Set 1. 1: Reference Description[EB/ OL] . http: / / dublincore. org / documents / 1999 / 07 / 02 / dces / .

[4]　Tim Berners-lee. Semantic Web Road Map[EB/ OL] . www. w3. org/ DesignIssues / Semantic. html , 1998.

[5]　Resource Description Framework Model and Syntax Specification[EB/ OL] . http: / / www. w3. org / TR / 1999 / REC rdf syntax 19990222/ , W3C Recommendation , 1999.

[6]　Ting Wang, Ji Wang, Yang Yu, *et al*. Metadata Pro: Ontology-based Metadata Processing for Web Resources[C] . Proceedings of the 5th International Conference on Web Information Systems Engineering, the International Workshop on Intelligent Networked and Mobile Systems, Brisbane, Australia. Published by Springer in Its Lecturer Notes in Computer Science 3307.

[7]　H-M Haav, J F Nilsson. Approaches to Concept-based Exploration of Information Resources [A] . W A bramowicz, J Zurada. Knowledge Discovery for Business Information Systems[M] . Kluwer Academic Publishers, 2001. 89-109.

作者简介:

刘金红(1978-) , 男, 博士研究生, 研究方向为人工智能、数据挖掘; 陆余良, 男, 教授, 博士生导师, 研究方向为计算机软件与理论、网络工程; 夏阳, 男, 博士研究生, 研究方向为软件工程、网络安全; 张旻, 男, 副教授, 研究方向为信号与信息处理、数据挖掘; 胡友志, 男, 硕士研究生, 研究方向为网络工程。

(上接第 207 页)

5. 3　对自动活动的处理

自动活动在系统中实现为对一个具体 Web 服务的访问。系统提供静态绑定和 UDDI 绑定两种定位方式。在静态绑定方式下, 可以在活动定义时直接给出服务的调用地址。有时, 服务的定位需要在流程运行时根据流程执行状态动态执行, 这就可以采用 UDDI 绑定方式: 用户在定义活动的执行时给出 Web 服务的查询条件, 在运行时根据该查询条件动态地到 UDDI 注册中心进行查询, 得到服务的调用地址。

6　总结

本文通过研究分析当前对跨企业工作流进行集成时所面临的问题, 提出了一种支持分布式工作流模型的工作流管理系统模型。本系统的优点在于: 能够提高模型的建立速度; 各企业维护自己的工作流模型, 保护了企业的商业秘密; 具有良好的柔性, 随时可以加入新的企业工作流模型。实践证明, 本系统能够很好地解决跨企业工作流集成的问题。

参考文献:

[1]　盛步云, 罗丹, 倪小平 , 等. 支持动态联盟的工作流系统研究[J] . 计算机集成制造系统—— CIMS , 2004, 10(5) : 497-501.

[2]　陈廷斌, 吴伟. 基于 Web Services 的跨企业分布式工作流管理[J] . 计算机工程 , 2004, 30(16) : 36-38.

[3]　刘洋, 葛声. 一种基于 Web 服务的分布式工作流系统的研究与实现[J] . 计算机工程与应用 , 2003, (1) : 208-211.

[4]　陈廷斌 吴伟 张光前. 一个基于 CBR 的跨企业 Web 服务工作流模型[J] . 计算机工程与应用, 2004, (14) : 25-27.

[5]　朱文华 王茜. 企业动态联盟中柔性工作流的研究与实现[J] . 小型微型计算机系统, 2003, 24(4) : 758-762.

[6]　周丹晨, 殷国富. 基于 Web 服务面向虚拟企业的柔性工作流管理技术研究[J] . 计算机辅助设计与图形学学报 , 2004, 16(4) : 427-433.

[7]　盛昭瀚, 张传芹. 基于 Web 的工作流管理系统及企业应用集成技术研究[J] . 系统工程理论与实践, 2002, (10) : 123-127.

[8]　范玉顺. 工作流管理技术基础——实现企业业务过程重组、过程管理与过程自动化的核心技术[M] . 北京: 清华大学出版社; 施普林格出版社, 2001.

[9]　Wil Van Der Aalst, Kees Van Hee. 工作流管理—— 模型、方法和系统[M] . 北京: 清华大学出版社, 2004.

[10]　Workflow Management Coalition. The Workflow Reference Model [R] . WFMC-TC-1003, 1995.

作者简介:

李宁(1980-) , 男, 江苏南京人, 助教, 硕士, 研究方向为工作流系统; 刘厚泉(1963-) , 男, 江苏徐州人, 副教授, 博士, 研究方向为协同与分布式计算技术。