一个基于Web的工作流管理系统*

史美林 杨光信 向 勇 伍尚广

(清华大学计算机科学与技术系 北京 100084)

E-mail: {shi,ygxin,xyong,wsg}@csnet4.cs.tsinghua.edu.cn

摘要 Internet/Intranet/WWW技术的日渐成熟以及资源、应用的日益丰富,为工作流管理系统的构造提供了一个理想的计算环境.在这种易于访问、无处不在的计算环境中,人们可以通过工作流管理系统将各种分布式的信息资源有效地集成、协调起来,以促使企业业务目标的高效实现.文章描述了一个基于Web而构造的通用工作流管理系统Wowww! (workflow on the world wide web),详细讨论了其体系结构的设计、工作流模型及过程实例的执行算法,并讨论了Wowww!为提高其可用性和灵活性而实现的工作流过程定义的自学习算法以及为提高系统的协作支持能力而采取的对同步协作的支持方法.

关键词 工作流管理系统,计算机支持的协同工作,workflow on the world wide web,Internet,WWW.

中图法分类号 TP393

进入90年代中后期,随着Internet技术的成熟,以企业Intranet计算环境为基础建立起了各种信息资源,如WWW、数据库、电子邮件、目录服务、电话、传真系统等.但这些资源基本上是相互独立的,意即它们很少能够意识到其他信息资源的存在(除非用特定的软件系统将它们连接起来).工作流管理系统[1](workflow management system,简称WfMS)着眼于协调企业内的各种资源(包括信息资源和人力资源),试图使各种业务活动在一定程度上自动进行,以高效地达成企业的业务目标.如果能够以Internet/Intranet为基础构造WfMS,实现企业Intranet环境下各种资源的高效协调,将能够最大限度地发挥这些资源的综合潜力,为企业的业务目标服务.同时,WWW为Internet/Intranet资源的访问提供了非常方便的手段,Java技术的成熟为开发跨平台的交互式应用提供了理想的开发工具.因此,基于Internet/WWW计算环境来构造WfMS将成为一种理想的选择.本文介绍一个基于WWW的工作流管理系统:Wowww! ——Workflow on the World Wide Web.

1 Wowww!体系结构

Wowww!体系结构的设计主要遵循如下几条原则:

· 基于Internet/Intranet/WWW分布式计算环境,面向跨企业或机构的大型分布式工作流管理.

· 集成已有的各种信息资源,如WWW、数据库、电子邮件、目录服务以及电话系统等,充分发挥这些资源的综合潜力.

· 与工作流管理联盟(workflow management coalition),参考模型[2~6]保持一致,以利于实现与其他WfMS系统的互连与互操作.

* 本文研究得到国家自然科学基金和国家863高科技项目基金资助.作者史美林,1938年生,教授,博士生导师,主要研究领域为计算机网络及其应用,CSCW.杨光信,1973年生,博士生,主要研究领域为计算机网络及其应用.向勇,1967年生,博士,讲师,主要研究领域为计算机网络,CSCW.伍尚广,1972年生,讲师,主要研究领域为计算机网络,CSCW.

本文通讯联系人:史美林,北京 100084,清华大学计算机科学与技术系

本文1998-04-03收到原稿,1998-12-14收到修改稿

1.1 Wowww! 总体结构

按照上述原则所设计的 Wowww! 的总体结构如图 1 所示,图中各宽带状的箭头表示各组成部分之间的通信方式.整个系统采用的实际上是一种客户机/服务器结构.在目前的实现中,Wowww! 服务器以一种服务的方式运行于 WinNT 环境下,它将完成工作流实例的执行、资源的集成以及各种控制数据和实例数据的管理.集成式工作流应用及开发环境(IWADE)给用户提供了一组用于开发工作流应用、处理工作项、对工作流实例进行管理的工具,具体包括过程及表单定义、工作项处理、服务器管理、过程实例监控以及 Internet E-Mail 客户端和联机帮助.它以 Java Applet 的形式运行于支持 Java 的 Web 浏览器中,并绕过 Web 服务器而直接与 Wowww! 服务器通信,以避免 CGI 或 ISAPI 或 NSAPI 实现所带来的低效率,但并不排除开发人员可以借助这些方式将工作流服务器中的各种对象和数据通过 WAPI[5](workflow API)以 HTML 的形式显示在 Web 浏览器中.

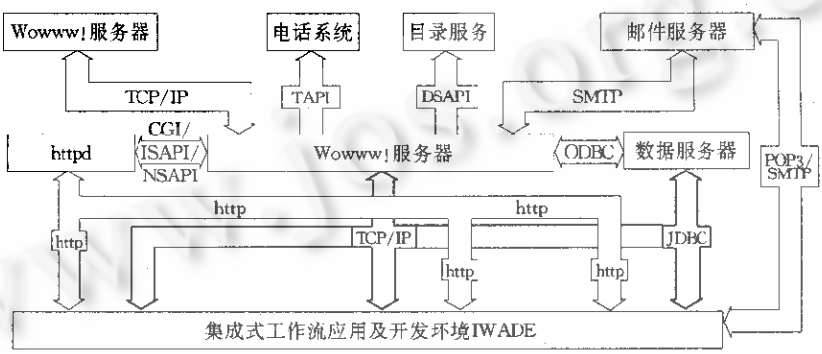

图1　Wowww!体系结构

1.2 Wowww! 服务器结构

处于 Wowww! 核心的是一个多线程的工作流服务器,它包括如图 2 所示的 5 个主要部分.其主要作用是,在业务流程的形式化表示的驱动下,实现流程中的各个环节之间的状态转换及数据的传递,以使这些环节按照一定的顺序在用户的参与下依次完成.通信管理模块接收网络数据包,并将其交给其他相应的模块进行处理;会话管理模块负责管理 IWADE 与服务器之间的会话连接以及同步协作组的维护,以实现对过程定义、表单和工作项的同步编辑.这些以及其他类型的数据都是以 C++对象的形式实现的,并由 Wowww! 服务器的面向对象的数据管理模块维护.这些对象被工作流执行控制模块用于过程实例的执行控制中,如实例的启动、不同活动状态之间的转换、用户工作项的生成以及对外部资源的访问等.不同的功能模块将分别运行于不同的线程中.各模块之间通过消息传递的方式进行通信.这种单进程、多线程的结构对于提高工作流过程实例执行的并发性、支持大规模的工作流管理起着至关重要的作用.

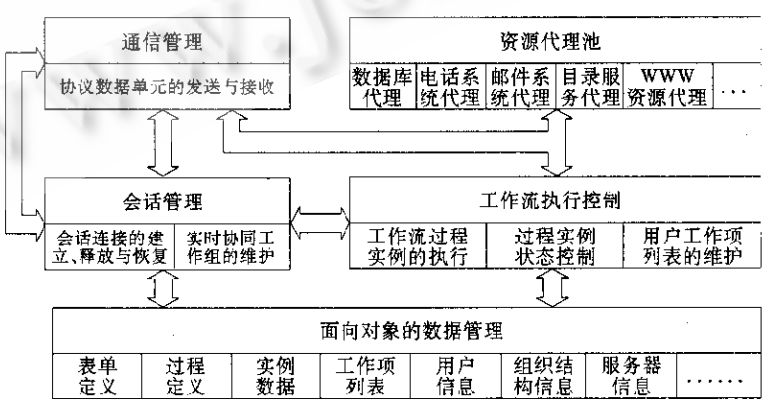

图2　Wowww!服务器结构

1.3 服务器端资源集成

前面提到过,在 Internet/Intranet 环境下已建立起大量的各种类型的信息系统,并且人们工作的完成将依赖于对这些信息系统的访问.为了实现协调多个人活动的目标,工作流管理系统必须提供一种访问各种类型异构资源的一致的方法.在 Wowww! 中使用了一种被称做"资源代理池"的机制.资源代理池实际上充当的是 Wowww! 工作流执行控制模块与其他信息系统的一个中介,它由一系列"资源代理"构成.每一种类型的资源代理负责完成对某种特定类型资源的访问,并给工作流执行控制模块提供相应的接口.例如,邮件系统代理可向其他 SMTP 服务器发送并收取邮件,它所提供的接口中包括 SendMail,GetMailList,GetMail 等操作.有了邮件系统代理之后,工作流执行控制模块就能够同邮件系统进行交互,以实现它的一些关键性的功能,如通知某个用户,他/她需要处理某个亟待处理的工作项等.

资源代理可以按同步及异步两种不同的方式来处理执行控制模块发送给它的请求.它们都是以 COM 组件的形式来实现的,因此可实现动态的"插拔"与配置.这种可动态扩展的特点一方面让人们可以根据需要将分布在 Internet/Intranet 环境下的资源纳入到 Wowww! 所协调的范畴中,另一方面可以突破 Java Applet 只能访问它们所来自的主机系统上的资源的限制.

1.4 客户端应用集成

在不同的领域中,人们用以完成他们的日常工作所用的应用程序也是各不相同的.工程设计人员应用某种类型的 CAD 工具去建立一些工程设计对象,而从事写作的人将使用某种类型的字处理软件去输入文档内容,并对其进行格式化.为了实现协调多个具有不同业务兴趣与范围的人的活动目标,工作流管理系统必须能够尽可能多地集成各种不同类型的应用.在 Wowww! 中,这一点是借助于"应用表单"的机制来实现的.应用表单实际上是供某个或某组用户用以处理某特定任务的一组应用程序的集合.所有这些应用程序都是以 JavaBean 组件的形式加以实现的.应用开发人员可以用 IWADE 所提供的表单定义工具,将这些应用组合起来,并设置各应用在表单中出现的格式.在过程实例的运行过程中,IWADE 将解释表单的定义,并根据相应的 JavaBean 的类定义生成相应的用户界面元素,供用户完成他们的任务.当然,为了能够将这些 JavaBean 组件形式的应用程序成功地集成到应用表单中来,它们必须实现一个特定的接口,以便 IWADE 能够设置其初始数据,并获得其最终的处理结果.这种特点使得 IWADE 成为一个能够包容大量不同应用、并可由用户进行扩展的框架.但目前的实现由于受 Java Applet 无法访问本地应用资源的限制而没有考虑对其他类型应用(如各种桌面应用)的集成.

2 Wowww! 工作流模型

WfMS 协调业务流程的完成是以业务流程的形式化表示——工作流过程定义为基础的.不同的 WfMS 用以描述过程定义的形式化模型各不相同.常用的有有向图[7]、PetriNet[8]、对象模型[9]、带有约束条件的文法表示[10]等.这些模型在描述能力、使用的方便程度以及易修改程度等方面各不相同.但一个好的模型既应能够描述结构化的过程,也应能够描述非结构化的过程,同时还应提供足够的灵活性以便能够处理可能发生的变化及例外情况.综合考虑这几方面的因素,Wowww! 提出并实现了一种"条件化有向图"的工作流模型.这种模型用有向图中的各结点表示工作流过程中的各个环节,而用有向边来描述各环节之间的控制及数据流动关系.下面给出此模型的形式化描述.

定义 1. 命名值 v 为一个四元组$\langle n,t,l,v\rangle$,其中 n 为一个字符串,它表示值的名称;t 为值类型;l 为值的字节长度;v 为值.命名值表示的是某种类型的简单数据对象.在目前的实现中,t 可以是 10 种类型的简单数据类型(即 boolean,char,short,int,long,string,float,double,datetime 和 binary)之一.用 $\mathscr{V}$ 表示所有命名值构成的集合,定义 $\varphi^0(\mathscr{V})=\mathscr{V}$;$\varphi^1(\mathscr{V})=\varphi(\mathscr{V})=\mathscr{V}\cup 2^{\mathscr{V}}$;$\varphi^n(\mathscr{V})=\varphi(\varphi^{n-1}(\mathscr{V}))$,$n>0$,则 $V^n\subseteq\varphi^n(\mathscr{V})$可以表示一个任意复杂的数据对象,其中 n 为任意非负整数.在 Wowww! 中,我们使用 V^0(简写为 V)来表示每一个活动中待处理的数据对象.

定义 2. 命名表达式 e 为一个三元组$\langle n,t,e\rangle$,其中 n 为表达式名称;t 为表达式的值类型(可为上述 10 种基本类型以及一个特殊的 void 类型);e 为表示表达式的字符串.表达式的构成元素可以是常量(如 boolean 类型的

true 和 false、字符串常量及各种类型的数值常量等)、变量(某个命名值的名称)、运算符、各种类型的预定义的函数(如字符串函数、日期函数、数值函数以及 Wowww! 服务器所提供的系统函数等)、资源代理所提供的接口中的操作、Wowww! 所能访问的 Windows 可执行模块或 COM/DCOM 组件所提供的接口中的操作等. 用 E 表示多个命名表达式构成的集合. 对于给定 e,定义一算子 $Eval:E\Rightarrow V$,$Eval(e)=v$,其中 $n_v=n_e;t_v=t_e;v_v$ 为根据 e_e 计算得到的值;而 l_v 则为此值的长度.

定义 3. 应用表单 f 为一个三元组$\langle n,p,T\rangle$,其中 n 为表单名称;p 为表单的各种属性(如大小、背景、字体等);T 为包含在 f 中的所有应用程序工具构成的集合. 对于 T 中的每一个元素 t(称之为 f 中的一个域),可用一个三元组$\langle n,p,t\rangle$表示,其中 n 为用户给 t 指定的在 T 内唯一的名称;p 为域的格式;而 t 为域的类型. 每种可以包含到表单中的域均是以 JavaBean 组件的形式实现的,并提供相应的接口,供 IWADE 运行时系统设置其初始值、取得其最终结果和值类型以及处理键盘和鼠标消息等. 目前已实现的域包括单行或多行编辑框、RTF 编辑器、HTML 编辑器、列表框、复选框、单选钮、数据库控件(用以访问各种类型的数据库中数据)、电子白板(用以支持同步讨论)等. 每一个域 t 实际上提供了显示和修改某个命名值 v 的一个用户界面,而应用表单 f 则表示了在某活动中需要处理的一项任务. 对于给定的 f,可定义一算子 $GenValue:F\Rightarrow 2^V$,$GenValue(f)=V$,并满足如下条件:$\forall\ t\in T_f$,若 t 所产生的值不空,则$\exists\ !\ v\in V$,使得 $n_v=n_t,t_v=t.GetValueType(),v_v=t.GetValue()$,并且 l_v 为 v_v 的长度.

定义 4. 接收者 r 描述的是可以参加到某活动中的用户或用户组. 其形式化表示为一个四元组$\langle r,t,n,a\rangle$,其中 r 为接收者名称;t 为类型,如单个用户、工作组或角色. 工作组指的是,具有某种特定语义的一组用户的集合. Wowww! 所支持的工作组包括普通工作组、带权重工作组(工作任务按一定的权重在成员之间分配)、串行或并行工作组、同步协同工作组(成员可以通过实时协作的方式完成对某工作项的处理)等. 角色则指系统中具有某种特定职责的用户,它实际上是一个由系统目录和 Wowww! 所提供的系统函数所定义的表达式. n 为通知方式,它描述的是在发生了某种类型的事件时可按什么方式(如用电子邮件或传呼)发出通知;a 指定了可将通知发至何处(如某个电子邮件地址或传呼号等). 当 t_r 不是单个用户时,r_r 可以是某个工作组的名称或一个角色表达式. 用 R 表示可以参加到某个活动中的所有用户的集合.

定义 5. 活动 a 表示业务过程中一个实际或抽象的工作步骤. Wowww! 将其形式化为一个六元组$\langle n,t,s,e,R,L\rangle$,其中 n 为活动名称;t 为活动类型(如起始活动、原子活动或某个子过程等);s 和 e 均为一个布尔表达式,分别表示活动的开始和终止条件;R 为可参与到此活动的所有用户集合. L 中的每一个元素 l 表示的是 a 同其前驱活动之间的连接关系. l 可以形式化表示为一个三元组$\langle a,f,M\rangle$,其中 a 表示前驱活动的名称,f 为用于处理从 a 传到 a 的数据对象的应用表单的名称,M 描述的是如何将这些数据对象映射到 f 中所包含的那些域中以及 R 中的那些用户可按何种方式去访问这些域. 对于 $m\in M$,可以将其形式化表示为一个四元组$\langle t,e,o,p\rangle$,其中 t 为 f 中某个域的名称,e 为某个描述从 a 传到 a 的数据对象的命名表达式的名称,o 指定如何用 e 所产生的值去设置 t 的初始值(例如,可以用此值去替换 t 的已有值,可将此值同 t 的已有值进行合并),p 指定的则是 R 中的用户可按何种方式去访问 t(如只读、可修改等). 从上面的定义可以看出,L 实际上描述的是从各前驱活动中传来的数据可以按何种方式被 a 中的用户进行处理. 用 A 表示多个 a 构成的集合.

定义 6. 工作流 w 是实际业务过程中的数学描述. Wowww! 将工作流形式化为一个三元组$\langle n,A,F\rangle$,其中 n 为工作流指定的名称,A 表示从业务过程中抽象出来的所有活动构成的集合,而 F 为 $A\times A\times C$ 的一个子集,它描述的是 A_w 中各活动之间的数据流动关系和控制流动关系. 对于 C 中的每一个元素 c,它是一个二元组$\langle c,E\rangle$,其中 c 为一个布尔表达式,E 为多个命名表达式构成的集合. c 描述的是在条件 c_c 得以满足时,激活后续的哪一个环节并将什么数据对象传给它. 例如,若$\exists\ f=(a_1,a_2,c)\in F$,并且 c_c 的计算结果为真,那么,$V=Eval(E_c)$将被传递给 a_2 进行处理. 显然,在 c_c 和 E_c 中可以引用的变量是根据 a_1 中的应用表单所生成的命名值. w 满足如下两个条件:(1) $\forall\ a\in A_w,\forall\ a'\in\Gamma\ (a),\exists\ !l\in L_a$,使得 $a_l=n_{a'}$. 此条件说明,不论哪个方向传来的数据都有相应的应用表单对其进行处理;(2) $\exists\ !\ a\in A_w,t_a=$"起始活动",即每个工作流都有唯一的起始活动. 用 W 表示系统所维护的所有工作流定义.

从定义 6 可以看出,w 实际上是一个有向图,图中各结点表示的是工作流中的各个活动步骤,与常规的有向

图的区别在于:与每一有向边相对应,分别有一个逻辑表达式和一个命名表达式集合.我们使用此种条件化的有向边来表示活动之间的控制关系和数据流动关系.此种模型的最大特点在于,它是以数据为中心的,即在各环节之间传递的是数据对象集,并且数据流动是受这些数据对象本身自己控制的.应用表单只是提供了一个供用户访问这些数据对象的手段.与其他工作流形式化表示方法比较起来,条件化的有向图具有直观、灵活、易于修改、描述能力强的特点,同时充分体现了以对数据流动控制为中心的工作流思想.这种思想也是我们构造协同工作支撑环境的核心[11].

IWADE中集成的过程定义工具使得应用开发人员可以按一种可视化的方式对业务过程进行建模.直观的图形化表示将被转换成上面所描述的内部表示(以C++对象的方式加以实现),并被保存在Wowww!服务器上.在后面的各节中,我们将详细讨论这种内容表示是如何被用于过程实例的执行的.

3 工作流实例的执行

在得到业务流程的形式化描述之后,即可让Wowww!使用此形式化表示来协调业务活动的完成.一般地,我们将工作流过程的一次执行称做一个工作流实例.在说明实例的具体执行细节之前,我们先定义如下一些在过程实例执行中会用到的一些结构.

定义 7. 活动实例s是过程实例中的一个步骤.Wowww!将其形式化表示为一个七元组$\langle n,U,P,b,e,s,V\rangle$,其中$n$表示与此活动实例相应的工作流活动名称;$U$为参与到此活动中的所有用户的名称构成的集合;$P$为当前前驱活动的$U$分量;$b$和$e$分别表示s的开始和终止时间;$s$记录的则是s的当前状态,其可能的状态分别为未就绪、就绪、正在运行以及完成等.一般而言,活动实例状态的转换是由工作流执行服务根据过程实例的实际运行情况而维护的;V为被s处理的所有命名值构成的集合,它表示此活动实例中需要处理的各种数据.用S表示由多个活动实例构成的集合.

定义 8. 工作流过程实例p是一个工作流的一次执行过程.它可以形式化地表示为一个六元组$\langle i,w,b,e,s,S\rangle$,其中$i$为过程实例名称;$w$为相应过程定义的名称;$b$和$e$分别为过程的开始和结束时间;$s$为过程实例的当前运行状态,这些状态可以是正在运行、被禁止、暂停以及完成等;S为自p开始运行以来,此工作流执行过程中生成的所有活动实例构成的集合.与s_s描述的是活动实例的状态不同,s_p描述的是整个过程实例的状态.管理人员可以使用管理工具对过程实例的状态人为地加以控制,如暂停或恢复某过程实例的运行、强行终止某实例等.

定义 9. 工作项i表示的是需要某个或某组用户进行处理的一项任务.它可以表示为一个二元组$\langle p,s\rangle$,其中p为相应的过程实例名称;而s为此过程实例中某活动实例的名称.用I表示多个工作项构成的集合.

定义 10. 用户工作项列表l记录的是需要某个用户进行处理的所有工作项的集合.它可以表示为一个二元组$\langle u,I\rangle$,其中u为负责处理I中那些工作项的用户名称.

以上面的定义为基础,下面,我们给出一个简化的工作流过程实例的执行算法.此算法可以分成两个部分,首先是过程实例的启动,然后是活动实例的提交.工作流过程实例的执行过程实际上是实例启动后反复执行活动实例提交的过程.

算法 1. 用户u启动工作流w的实例.

1. 从系统所维护的所有工作流定义W中选择待启动的过程w.
2. 根据用户指定的过程实例名称name,建立一个新的过程实例结构p,并使:
   $w_p\leftarrow n_w$; $b_p\leftarrow$系统当前时间; $i_p\leftarrow$name; $S_p\leftarrow\varnothing$.
3. 根据w的起始活动a,建立一个新的活动实例s,并使:
   $n_s\leftarrow n_a$; $U_s\leftarrow\{u\}$; $P_s\leftarrow\varnothing$; $b_s\leftarrow b_p$; $V_s\leftarrow\varnothing$; $s_s\leftarrow$就绪.
4. $S_p\leftarrow\{s\}$.
5. 建立一个工作项结构i,并使:
   $p_i\leftarrow i_p$; $s_i\leftarrow n_s$.
6. $I_{1|u_j=u}\leftarrow I_{1|u_j=u}\cup\mathbf{i}$; //将此工作项加入到用户$u$的工作项列表中.

此后,IWADE将根据w的定义生成与新建立的工作项相应的应用表单,用户u可以同此应用表单中的工具进行交互,以完成对此工作项的处理.处理的结果可以暂存起来,也可以将其提交.提交一个工作项表示对它

处理的完成.此时,Wowww! 将执行下面的算法.

算法 2. 工作项 $i\langle p,s\rangle$ 的提交.

1. 计算 $V=GenValue(\mathrm{f})$,并将 V 同 V_s 进行合并,其中 f 为与 s 相应的应用表单.
2. 对于每一个 $\mathrm{a}'\in\Gamma^{+}(\mathrm{a})$,其中 $\mathrm{a}\in A_w\wedge n_a=n_s$
2.1. 根据 V_s 计算 c_e,其中$(\mathrm{a},\mathrm{a}',\mathrm{c})\in F_w$
2.2. 若 c_e 为 true,则
2.2.1. $V_{next}=Eval(E_e)$;
2.2.2. 若 p 中不存在与 a′相应的活动实例,则
2.2.2.1. 建立一个新的活动实例 s′,并使:
$n_{s'}\leftarrow n_{a'}$; $P_{s'}\leftarrow U_s$; $V_{s'}\leftarrow V_{next}$; $b_{s'}\leftarrow$系统当前时间; $s_{s'}\leftarrow$未就绪
2.2.2.2. $S_p\leftarrow S_p\cup\{\mathrm{s}'\}$,否则
2.2.2.3. 根据 L_a 将 V_{next} 同 $V_{s'}$ 进行合并
2.2.3. 计算 $s_{a'}$
2.2.4. 若 $s_{a'}$ 计算结果为 true,则 //s′可以被启动
2.2.4.1. 建立一个新的工作项结构 i′,并使:
$p_{i'}\leftarrow p_i$; $s_{i'}\leftarrow n_{a'}$
2.2.4.2. 对于 $R_{a'}$ 中的每一个 u
2.2.4.2.1. $I_{1|u_j=u}\leftarrow I_{1|u_j=u}\cup\mathrm{i}'$
2.2.4.2.2. $U_{s'}\leftarrow U_{s'}\cup\{u\}$; //$U$ 中的所有用户均可处理 i′
2.2.4.3. $s_{s'}$—就绪.
3. $s_s\leftarrow$完成; $e_s\leftarrow$系统当前时间.
4. 若对于 S_p 中的每一个 s,s_s 均为完成,则
4.1. $s_p\leftarrow$完成
4.2. $e_p\leftarrow$系统当前时间.

4 过程定义的自学习

协同应用环境的动态多变的特点要求协同工作系统中的控制知识能够随系统的运行而动态地增长[12].工作流系统中的过程定义实际上就是一种控制知识.在传统的 WfMS 中,过程定义一般是事先定义好的,对它的修改只能由建模人员完成,WfMS 本身并不能根据工作流程的实际运行情况而动态地修改、完善这些定义.它给用户带来的是一些非常"僵硬"的操作步骤,阻碍了 WfMS 在更为广泛的领域内的应用.

过程定义自学习的基本思想是将工作流的有关参数推迟到工作流实例的运行时再加以确定,并由 WfMS 自动捕获这些参数,而得到相应的过程定义.这样,过程定义能够根据实际的工作流程而动态地生成,从而降低建模人员的工作负担,提高系统的可用性.由于自学习的过程实际上是在运行时对过程定义进行修改的过程,因此,可以通过此种机制而实现对过程定义的动态修改,WfMS 系统的灵活性将由此而得到显著的提高.

由于学习目标——过程定义的结构是非常明确的,因此,学习过程实际上是如何在合适的时机提取出用户在实际执行工作流程时所提供的信息中那些过程定义所需要的参数,以填充过程定义结构中的相应部分.根据这种思想,学习算法的设计主要需考虑在何时提取出哪些参数.下面给出 Wowww! 所实现的过程定义自学习算法.从具体的算法过程可以发现,自学习的机制对于即席工作流也将能够提供很好的支持.

算法 3. 工作流定义的自学习过程.

1. 生成一个空的过程定义结构 $\mathrm{w}=\langle n,A,F\rangle$,并使:
$n\leftarrow$用户提供的过程名称;$F\leftarrow\varnothing$;$A\leftarrow\{\mathrm{a}_{\mathrm{Start}}\}$.
2. 初始化 $\mathrm{a}_{\mathrm{Start}}$:
$n\leftarrow$用户提供的起始活动名称;
$t\leftarrow$起始活动;
$s\leftarrow OnDemand()$; $e\leftarrow OnSubmit()$;
$L\leftarrow\{(\text{""},f,\varnothing)\}$;
$R\leftarrow\{$〈开始此学习过程的用户名称,单个用户,E-MAIL,用户的邮件地址〉$\}$.

3. 根据用户构造的起始活动中所用表单 f 初始化 $f_{L_{a_{Start}}}$:$f_{L_{a_{Start}}} \leftarrow n_f$.

4. $a_{Current} \leftarrow a_{Start}$.

5. 在用户处理完与 $a_{Current}$ 相应的活动实例并将其提交时,向用户询问如下信息:

5.1. 一个条件表达式 c,一个命名表达式集合 E,下一个接收者名称 u;

5.2. 生成活动 a_{Next},并初始化其 n,t,s,e,R 等部分;

5.3. $A \leftarrow A \cup \{a_{Next}\}$;

5.4. $F_w \leftarrow F_w \cup \{\langle a_{Current}, a_{Next}, c, E\rangle\}$.

6. 若没有指定任何向后续活动传送的数据,学习过程完成一个工作流分支的学习.

7. 生成与 a_{Next} 相应的工作项,待指定用户处理此工作项时,他可以定义一个表单 f_{Next},并指定如何将传来的数据与 f_{Next} 中的各域对应起来,由此种对应关系设置 $L_{a_{Next}}$.

8. $a_{Current} \leftarrow a_{Next}$,转步骤 4.

值得注意的是,由于上述算法需要多次会话过程才能结束,并且一次只能学得工作流中的一个分支,因此,对于有分支的工作流程,在以后此流程的实例执行过程中,可能需要从第 4 步开始重新执行上述学习算法,以最终得到完整的过程定义.另外,学习得到的过程定义某些方面的信息可能仍然是不完全的,此时系统建模人员可以使用过程定义工具加以完善.

5 对同步协作的支持

在对群件系统进行分类时,通常是将 WfMS 划归为支持"异地异步"协作的系统[12].由于工作流实例中的各工作项一般是由不同用户异地异步进行处理的,因此,这种划分是合理的.但出现了要求将实时协作系统(如桌面会议系统[13],协同设计[14]、协同编著[15]等)中的一些特性引入到 WfMS 中来的新的应用需求,以使之成为一个支持各种类型协作的协同工作环境.

同步协作功能的引入可以从 3 个不同的方面着手.首先是过程定义的设计.由于应用系统的建模是一个复杂的过程,但现有 WfMS 中的建模工具无一例外地均为单用户版本,因此也就无法支持多个具有不同业务专长的用户按同步协作的方式完成过程的定义.Wowww! 的过程定义工具提供了一个与会议系统中"电子白板"类似的"虚拟设计板",它可以供多个应用系统建模人员共享,使他们能够以同步协作的方式共同完成某个应用系统的建模.这样,过程中不同步骤的建模可以由最熟悉此业务的人员完成,因此可以保障建模结果的准确性与全面性,使所得到的过程模能更准确地反映实际的业务情况.限于篇幅,"虚拟设计板"的具体细节此处就不再讨论了.

需要引入同步协作能力的第 2 个方面是对工作项的处理.传统 WfMS 中的工作项都是由某一个用户负责完成的.Wowww! 则可以让多个用户同时共同来完成对某个工作项的处理.为实现此种能力,一方面,Wowww! 允许多个用户同时分别处理表单中的各个域;另一方面,在表单中可以加上一种特殊类型的域:同步控件,如协同编辑框、虚拟绘图板等,同步控件所维护的内容可以供处理此工作项的那些用户同步地加以处理,这样他们将可用同步协作的方式完成某篇文档的撰写或某幅图纸的设计等.这两方面的功能都是在全复制的结构下实现的.IWADE使一种类似于 CoDesign[14]所实现的并发控制方法去保证多个用户访问同一对象时的数据一致性.与 CoDesign 的不同之处在于,由于 Java Applet 无法实现数据包的多播,因此,Wowww! 借助于其服务器上的会话控制模块来完成数据包的分发.

为了进一步增强同步协作的能力,还可以引入实时视频和音频服务.这些服务可以通过单独的工具提供,也可以用 Plug-in 的方式嵌入到 Web 浏览器中.但在 Internet 环境下,不同的机器上视频和音频信息的采集方式、编码格式、压缩方式等都有很大的不同,这给在 Wowww! 中引入这些服务带来一定的困难.事实上,在不同平台上有关会议系统互连的研究中也同样遇到这个问题.对此,我们以后将另文详述.

6 结 论

Internet/Intranet/WWW 及 Java 技术的成熟及应用的日渐丰富为工作流管理系统的构造提供了一个理想的计算环境.本文详细描述了一个基于 Web 的工作流系统 Wowww!.其设计的基本出发点是集成各种已有的

Internet 资源,使之能够被一致地协调起来,以促使企业业务目标的高效实现.文中讨论了 Wowww! 的体系结构、工作流模型、工作流实例的执行算法以及为提高其灵活性、可用性及协作能力而引入的两种新特性:工作流过程定义的自学习方法及对同步协作的支持.我们的最终目标是将 Wowww! 发展成一个能够支持各种协作方式的协同工作环境和协同应用的研究开发环境,并在此基础上进一步深化对于协同工作本质的研究.

致谢　本项目的研究开发得到国家 863 高技术研究发展计划智能计算机主题专家组的大力支持,在此谨表示诚挚的谢意!另外,李红臣、郑小虎、冯磊、颜宏等同志为本项目的开发做了大量的工作,在此一并致谢!

参考文献

1 史美林,杨光信,向勇等.WfMS:工作流管理系统.计算机学报,1999,22(3):325～334
(Shi Mei-lin, Yang Guang-xin, Xiang Yong *et al*. WfMS: workflow management system. Chinese Journal of Computers, 1999,22(3):325～334)
2 WfMC. Workflow Process Definition Read/Write Interface: Request for Comments. 1995
3 WfMC. Workflow Management Coalition Workflow Standard-interoperability Abstract Specification. 1996
4 WfMC. Workflow Management Coalition Audit Data Specification. 1996
5 WfMC. Workflow Management Coalition Workflow Client Application (Interface 2) Application Programming Interface (WAPI) Specification. 1996
6 WfMC. The Workflow Reference Model. 1994
7 Du Wei-min, Davis J, Shan Ming-chien. Flexible specification of workflow compensation scopes. In: Hayne S C, Prinz W eds. Proceedings of ACM SIGGROUP Conference on Supporting Group Work. Phoenix: ACM Press, 1997. 309～316
8 Ellis C A, Nutt G J. Modeling and enactment of workflow systems. In: Ajmone Marsan ed. Application and Theory of Petri Nets. Berlin: Springer-Verlag, 1993. 1～16
9 Peter C L, Walter H-D. Object-oriented protocol hierarchies for distributed workflow systems. Theory and Practice of Object Systems, 1995,1(4):281～300
10 Natalie S G, Daniele S P, Remo P. Generalized process structure grammars (GPSG) for flexible representations of work. In: Ackerman M S ed. Proceedings of ACM Conference on Computer Supported Cooperative Work. Boston: ACM Press, 1996. 180～189
11 Yang Guang-xin, Shi Mei-lin. Dicse: a platform for developing CSCW. In: Xue Chun-pei ed. Proceedings of the International Conference on Communication Technology. Beijing: Chinna Construction Materials Publishing House, 1998
12 Ellis C A, Gibbs S J, Rein G L. GroupWare: some issues and experiences. Communication of ACM, 1991,34(1):39～58
13 Wu Shang-guang, Shi Mei-lin. Support environment for CSCW research-design and implementation of a desktop computer conferencing system. In: Xue Chun-pei ed. Proceedings of the International Conference on Communication Technology. Beijing: Chinna Construction Materials Publishing House, 1996. 187～190
14 史美林,杨光信.一个协同设计支撑系统原型——CODESIGN.清华大学学报(科学与技术),1998,38(S1):30～35
(Shi Mei-lin, Yang Guang-xin. CODESIGN: prototype of a cooperative design support system. Journal of Tsinghua University (Science and Technology), 1998,38(Si):30～35)
15 Ellis C A, Gibbs S J. Concurrency control in groupware systems. In: Clifford J, Lindsay B, Maier D eds. Proceedings of the ACM SIGMOD Conference on Management of Data. New York: ACM Press, 1989. 399～407

A Web-based Workflow Management System

SHI Mei-lin　YANG Guang-xin　XIANG Yong　WU Shang-guang

(*Department of Computer Science and Technology　Tsinghua University　Beijing*　100084)

Abstract　As the Internet/Intranet/WWW technologies become more and more mature and web-based applications become richer and richer, the application developers are provided with an ideal computing environment to construct WfMSs (workflow management systems). Web-based WfMSs can help business goals to be achieved effectively by means of integrating various distributed resources in an easy to access environment. A web-based general purpose workflow management system named Wowww! is described in this paper, detailing its architecture, the workflow model and the algorithms used for workflow instance execution, the self-learning algorithm for automatic process definition generation, and the strategies to support synchronous cooperation.

Key words　Workflow management system, computer supported cooperative work, workflow on the world wide web, Internet, WWW.