ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.6, June 2006, pp.1401−1410
DOI: 10.1360/jos171401

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于工作流引擎的构件组装体系结构*

李海波[1,2+], 战德臣[1], 徐晓飞[1]

[1](哈尔滨工业大学 计算机科学与技术学院 企业智能计算研究中心,黑龙江 哈尔滨 150001)
[2](东北农业大学 工程学院,黑龙江 哈尔滨 150030)

Architecture of Component Composition Based on Workflow Engine

LI Hai-Bo[1,2+], ZHAN De-Chen[1], XU Xiao-Fei[1]

[1](Centre of Intelligent Computing of Enterprises, School of Computer Science & Technology, Harbin Institute of Technology, Harbin 150001, China)
[2](Engineering Institute, Northeast Agricultural University, Harbin 150030, China)
+ Corresponding author: Phn: +86-451-55116600, Fax: +86-451-86412664, E-mail: lihaibo@hit.edu.cn, http://www.hit.edu.cn

Li HB, Zhan DC, Xu XF. Architecture of component composition based on workflow engine. *Journal of Software*, 2006,17(6):1401−1410. http://www.jos.org.cn/1000-9825/17/1401.htm

Abstract: Nowadays, SA (software architecture) and CBSD (component-based software development), have shown great advantage in software development. CBSD is a successfully applied down-top methodology to software development; whereas SA still lacks a top-down component composition methodology, especially in process component. Based on workflow and the background of Web application, a component composition method is investigated. Being started with the key elements of business process, such as business object, control flow and data flow, characteristics and interactions between them are formalized. A workflow engine based SA of process component composition is proposed. Except for the control flow, the SA can deal with data flow between business activities, and provide a reference to others in applying archive workflow to component composition.
Key words: workflow; workflow engine; component composition; process component; control flow; data flow

摘 要: 近年来,软件体系结构(SA)、基于构件的软件开发技术(CBSD)等开始在软件开发中发挥出重要的作用.CBSD 应用的成功主要体现在自底向上构造系统的方法上.目前,对 SA 的研究还缺乏一种自顶向下的构件动态组装方法,尤其是过程构件的组装.以工作流技术为基础,以基于 Web 应用为背景,研究了基于工作流引擎的构件组装方法.从业务过程的基本要素——业务对象、控制流和数据流入手,详细分析了这些要素在构件环境下的特点和交互模式,并以此为基础提出了一种基于工作流的引擎的过程构件组装体系结构.该体系结构中除了考虑常规的控制流驱动的构件组装外,还深入研究了文档型的数据流驱动的构件组装机制,能够为其他研究者将文档型工作流技术应用于软件构件组装提供借鉴和参考.
关键词: 工作流;工作流引擎;构件组装;过程构件;控制流;数据流
中图法分类号: TP301 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant No.60573086 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2003AA4Z3210 (国家高技术研究发展计划(863)); the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No.20030213027 (国家教育部博士点基金)
Received 2005-12-14; Accepted 2006-03-13

基于构件的软件工程(component based software engineering,简称 CBSE)逐渐成为软件开发的主流范型,是软件开发工程化的现实可行途径.构件必须经过组装才能形成应用系统,所以构件组装技术是关键[1].构件组装的本质是,在构件之间建立关联,根据这种关联协调它们的行为,把它们组织成为一个有机的整体.构件组装策略之一是自顶向下的组装方法,它开始于系统的集中模型,而忽略底层细节,通过逐步求精把细节逐渐引入到框架中来;构件组装的策略之二是自底向上的方法,对各子系统单独建模而忽略它们之间的相互作用.近年来,软构件应用比较成功的模式就是基于标准化运行级构件的规范进行开发,如 CORBA,EJB 和 DCOM,中间件技术作为存在于系统软件与应用之间的特殊层次,抽象了典型的应用模式,是一种自底向上构造软件系统的途径.但当通过组装,构件的粒度由原子构件逐渐变大到过程构件(业务过程)的时候,构件间开始存在一种偏序关系,它描述了活动的执行顺序,是工作流视图[2].目前,对构件组装的研究大多侧重于较高层次的抽象描述,比如接口规约、功能规约等.从工作流角度、业务过程逻辑和业务活动实现分离的思想,使得过程的组装更贴近于自顶向下的组装途径,而当前的研究缺乏一种按照过程的抽象描述组装过程构件的方法.

构件按照执行顺序进行组装不能仅仅满足于构件的接口规约,还应该考虑运行时构件的状态.通过与用户以及其他构件的不断交互,构件的状态不断发生变化,在控制条件的约束下,不断地选择、实例化新构件.对执行顺序的处理虽然适合采用工作流技术[2],但在构件环境下,执行顺序的引入增加了过程维的考虑,为构件组装提出了新需求.过程视角体现在工作流技术的两个最基本的特征——控制流和数据流上,前者体现了工作流活动间的执行偏序;后者体现的是活动间的数据依赖关系.因此,从工作流的基本特征分析入手,必然能够准确地掌握基于工作流引擎的构件组装系统结构.因此需要研究:1) 工作流引擎和执行构件(活动)的交互模式,即控制流;2) 可执行构件间的数据交互模式;3) 基于角色的权限控制模式,体现了一种构件重用方法;4) 基于工作流引擎的过程构件组装的系统结构.

为此,以基于 Web 的应用为背景,提出一种基于工作流引擎的过程构件组装体系结构.从分析工作流的基本特征入手,提出了工作流引擎和构件、构件之间的交互模式,以及基于这种交互模式的工作流引擎内部结构,最后给出了按照过程描述组装过程构件的系统结构,并通过一个具体实例加以说明.

1 相关工作

构件组装研究主要包括两部分:复合构件形成过程研究和基于体系结构(或者组装框架)的研究.其中,前者是为了得到性能稳定、有较强可靠性的复合构件,以用于构件组装,后者则是寻求良好的组装策略以实现开发目的.基于软件体系结构的构件组装方法是构件组装的主流研究方向,是实现构件生产与组装的技术基础,目前取得了一定的进展,如基于软件体系结构的构件组装技术——ABC 方法[1,3].

对构件组装的研究主要分为如下几个方面:1) 通过体系结构的建模和高层描述进行构件的组装方法[4,5],研究高层的一些性质;2) 通过过程结构[6]以及基于语义和知识[7]的方法,研究过程构件的组装方法;3) 分布式环境下过程构件组装研究主要包括服务组合[8]和构件调度优化[9]等方面.

Avoca[10]是软构件技术应用中比较成功的例子.该系统可以利用现有的构件和通信协议组装软件系统,但是,构件的修改必须了解足够的领域知识.SA 提供了一种自顶向下构造软件系统的方法,但目前对其研究大多局限于建模和高层描述[4,5]上.对于过程构件的组装方法,多数研究主要集中在过程结构[6]、优化[11]和适应性[9]等方面.对采用 SA 进行过程构件组装并映射到实现层的研究相对较少,并且,过程中的数据流是构件交互的基础,对数据流的交互模式则考虑得更少.

2 概 述

2.1 基本概念

企业日常经营活动中涉及到的相关信息实体与物理实体,如订单、报表、设备、员工等,我们称这类信息为业务对象.业务对象是指企业中所有物理资源和信息资源在信息系统中的抽象,通过一组连续的操作,使得业

务对象的属性不断改变,最终完成对一个业务对象的处理过程.业务对象之间的交互是工作流系统的数据特性之一[12].因此,首先给出几个相关概念.

定义 **1**(业务对象). 业务对象可以用一个三元组表示,记为 $bo=(ID,A,M)$.其中:

(1) ID 为该业务对象的唯一标识;

(2) A 为 bo 的属性 $a_i(i=1,2,\ldots,m)$的集合,$a_j(bo)$表示 bo 的属性集合 a_j;

(3) M 为作用于 bo 上的业务操作 $m_i(i=1,2,\ldots,n)$的集合.

定义 **2**(业务操作). 业务操作表示为 $op=(r,bo,A)$.op 是由一个特定角色 r 作用于 bo 上的一个原子性动作,并使 bo 的属性发生变化;A 表示 op 作用的业务对象 bo 的属性集合,业务操作与业务对象中的业务操作一一对应.

定义 **3**(业务活动). 业务活动定义为三元组 $ba=(r,OP,\prec)$,是由一个特定角色 r 不间断地执行的一组业务操作的集合 OP. $\prec$是业务操作构成的偏序关系.

定义 **4**(业务过程). 业务过程 bp 是为实现特定的业务目标,由不同角色执行的一组具有偏序关系的、逻辑上相互关联的、作用于一组相关业务对象上的业务活动,业务活动间的偏序关系由约束条件确定.业务过程可定义为五元组 $bp=(R,BA,BO,P,\prec)$:

(1) R 为角色的集合;

(2) BA 表示业务活动 $ba_i(i=1,2,\ldots,n)$的集合;

(3) BO 表示业务对象的集合;

(4) P 表示约束条件的谓词集合;

(5) $\prec$表示业务活动之间的执行偏序关系.

上述定义说明:业务过程是由一系列存在执行偏序关系的活动组成,体现了业务活动间的控制流.而业务活动可以进一步分解成一系列业务操作,使业务对象属性发生变化,数据流是在不同业务对象间交互时产生的.

2.2 交互式Web文档(Web document)的多重作用

Web 技术对于解决大规模且异构的系统具有其他技术所不可比拟的优势,所以,我们以 Web 应用为背景研究构件组装.

工作流活动的实现方法之一就是提供交互式的 Web 文档(Web page),所以,Web 文档最明显的一个作用就是与用户进行交互:接收用户的输入以及显示处理结果.此外,如果交互式 Web 文档的处理涉及到工作流,那么该文档接收的数据可以直接用于确定工作流的路由,或者作为工作流相关数据[13]被提交给工作流引擎,用于确定活动的状态或者计算后续活动.另外,交互式 Web 文档之间也随时交换各自的数据.所以在构件环境下,如果考虑工作流,交互式 Web 文档共有 3 个作用:与用户进行交互;与工作流引擎进行交互;活动间的数据交互.因此,Web 文档具有如下定义:

定义 **5**(**Web** 文档). Web 文档表示成 $WDoc=(C,D)$.其中:C 是构件的集合,包含两类构件,$C=(FC,IC)$,分别表示功能构件和界面构件.功能构件用于处理业务逻辑(业务操作),界面构件负责业务对象属性的表示,二者最终组装成业务活动 ba; D 是文档中的数据,包含 3 类数据,记为 $D=(UD,WfRD,WfID)$.其中:UD 是用户数据;$WfRD$ 是工作流相关数据,是用于计算活动间执行偏序的用户数据的子集[13];$WfID$ 是工作流实例数据,并用于控制 Web 文档中的数据流.一个 Web 文档可以实现一个业务活动.

3 基于工作流引擎的构件环境

工作流管理技术的主要贡献就是业务过程逻辑和业务活动实现的分离.在运行阶段,工作流引擎根据过程模型调度业务活动的执行,在构件环境下,则是过程构件动态组装的集中体现.过程构件组装时不一定实例化模型中的所有构件,对构件实例化排序就是对过程构件的调度.基于对交互式 Web 文档的认识,工作流引擎和构件间的关系总体上概括为两个方面:执行调度和数据交互.下面从工作流技术的两个最基本的特征:控制流和数据流以及权限控制方面进一步分析构件环境下工作流引擎的调度和交互机制.

3.1 控制流

3.1.1 过程控制方式

从定义 4 可以看出,业务活动间存在一种执行偏序关系,这种偏序关系被抽象为过程描述,并作为调度规则由工作流引擎进行翻译,结果就是活动的执行顺序(工作流引擎的结构见第 3.5 节).用户不断地与 Web 文档进行交互,使业务对象的属性不断发生变化,并依据静态的过程描述和运行时业务对象属性的取值计算后续活动.

定义 6(执行偏序关系). 对于$\forall ba_i,ba_j\in BA$,设 bo_i 和 bo_j 分别为 ba_i 和 ba_j 操作的两组业务对象,如果在 4 种基本控制结构[13]的作用下,依据谓词 $P(a_1,a_2,\ldots,a_n)$, $a_i\in a_i(bo_i)$,能够确定出执行顺序,则称活动 ba_i 和 ba_j 之间具有执行偏序关系,记为$(ba_i\prec ba_j)p$;反之,记为$(ba_j\prec ba_i)p$.

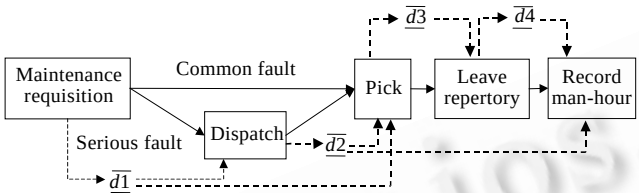


Fig.1 Equipment maintenance process

图 1 设备维修过程

工作流引擎依据业务过程中的谓词 $P(a_1,a_2,\ldots,a_n)$在运行时的取值计算后续活动,属性 $a_1,a_2,\ldots,a_n$ 称为工作流相关数据,其含义实质上与工作流管理联盟(WfMC)给出的定义相同.图 1 是制造企业典型的设备维修过程,其中,实线表示执行偏序关系.根据运行时故障的大小(谓词 P),确定维修申请审批的后续活动是派工还是领料.

在基于构件的 Web 环境下,过程构件实际上是集成在一起的、具有执行偏序关系的多个 Web 文档.因此,Web 环境下过程构件的定义如下:

定义 7(过程构件). 过程构件是由多个 Web 文档以及 Web 文档(活动)之间的执行偏序关系组成,表示成 $PC=(WDoc,\prec)$.

过程构件是一系列构件按照执行偏序关系组装而成的更大粒度的构件,而业务活动则是由若干无执行偏序关系的构件组装而成.过程构件必须解决工作流引擎和工作流相关数据交互问题.

3.1.2 与运行数据交互

采用 XML 作为 Web 应用间的数据交换格式具有许多优点,比如结构化、易于校验、人与计算机均可读等.加之采用 XSLT 样式表(style sheet)进行界面布局,则可以使 Web 文档数据与表示方式相分离,因而常利用 Web 及 XML 来解决分布式环境下的互操作问题.为了实现对过程的调度,所采用的描述方法必须有能力与工作流相关数据进行交互.

按照定义 5,工作流相关数据对确定活动执行顺序起着关键作用.在交互式 Web 文档中则具体体现为对业务对象操作的使能控制,比如,业务对象的“创建”按钮处于 Disabled 状态,就表示业务对象暂时不满足创建条件,这种工作流调度模式见第 3.3 节.所以,既然运行数据最终要体现到 Web 文档界面上,那么运行数据就可以生成 XML 文件,然后再利用 XSLT 样式表在构件组装时控制 Web 文档的操作.为此,我们在语法上给出基于 XSLT 样式表的相关描述方法.

(1) 获取工作流相关数据

当处理〈xsl: for-each〉元素时,处理过程依赖于工作流相关数据的变量 $x_1,x_2,\ldots,x_n$ 的取值,因此,获取 x_i 以及获取多个数据在语法上可以分别表示为

〈xsl: *variable name*=“*Function*(x_i)” *select*=“…”/〉,

〈xsl: *variable name*=“…” *select*=“*Function*(*query*)”/〉,

这里,*query* 以及 *Function*(*query*)依赖于具体的实现机制,一般来讲,大多采取 XPath 和 XQuery 语言.

(2) 测试谓词真值

依据工作流相关数据以及谓词表达式,谓词条件表达成

〈xsl: for-each *select*=“*Function*($P(x_1,x_2,\ldots,x_n)$)”〉…〈/xsl: for-each〉,

其中,谓词 P 的表达依赖于具体的实现.如果谓词 P 中的参数 $x_1,x_2,\ldots,x_n$ 的取值有多种组合使得 P=true,则可以

借助于 if 子元素,表达式如下:

$$\langle \text{xsl: if } test=\text{“}Function(P(x_1,x_2,\ldots,x_n))\text{”}\rangle\ldots\langle/\text{xsl: if}\rangle,$$

把那些所有使 P=true 的参数取值都包含在子元素⟨xsl: if⟩中,而且,为了使每个判断分支能够执行一定的行为,我们允许每个子元素⟨xsl: if⟩分支中包含一个模板规则.

(3) 工作流活动和Web文档的绑定

工作流过程描述中的活动最终应被转换成 Web 文档,提交给用户执行.界面开发者可以在工作流活动和交互式 Web 文档中的 form 或者 link 建立起映射关系,语法上表示为

$$\langle \text{xsl: } submit\ action=\text{“}Function(x_1,x_2,\ldots,x_n)\text{”}\ actionElement=\text{“actionID”}\rangle\ldots\langle/\text{xsl: submit}\rangle,$$

元素 submit 将活动 Function 和 HTML 中的 form 元素绑定起来.其中:form 元素由 actionID 标识;变量 $x_1,x_2,\ldots,x_n$ 的取值来自于 HTML 中的 form 或者 XSLT 中的变量;XSLT 中的变量定义可以参见(1)中的情况.

3.2 数据流

根据业务对象的定义以及对交互式 Web 文档的分析可知,业务对象间的交互产生了数据流.因此,业务对象之间在数据上存在着依赖关系,下面给出定义:

定义 **8**(数据依赖关系). 设 $ba_j.bo$ 和 $ba_i.bo$ 分别是活动 ba_j 和 ba_i 所操作的业务对象,若 $ba_j.bo$ 的属性集合 $d_j(ba_j.bo)$的产生需要读取 $ba_i.bo$ 的属性集合 $d_i(ba_j.bo)$,则称业务对象属性 d_j 和 d_i 之间存在着数据依赖关系,记作 $d_i(ba_j.bo)\rightarrow d_j(ba_j.bo)$;如果不需要区分属性集合,则简称 $ba_i.bo$ 和 $ba_j.bo$ 之间存在着数据依赖关系,记作 $ba_i.bo\rightarrow ba_j.bo$.

数据依赖关系是个一般性的概念,只要业务对象之间存在读写操作,都称作具有数据依赖关系.图 1 中的虚线表示数据依赖关系,其中两个业务对象维修申请单 a 和派工单 b 分别由两个业务活动维修申请和派工进行操作,$d1$ 是数据集合,包括故障记录号 No 和故障发生时间 Time.此时,$A.a$ 和 $B.b$ 之间就存在数据依赖关系:$No(A.a)\rightarrow No(B.b)$和 $Time(A.a)\rightarrow Time(B.b)$.

在工作流引擎的调度下,同一类型的多个活动可能会同时运行.因此,在说明如何调度数据流之前,首先给出活动实例和过程实例的定义:

定义 **9**(业务活动实例). 业务活动实例表示为三元组 $bai=(id,u,ba)$,是特定用户 u 对业务活动 ba 的一次执行,id 是活动实例 ID.

定义 **10**(业务过程实例). 业务过程实例表示为三元组 $bpi=(id,U,bp)$,id 表示过程实例 ID,是特定用户集合 U 对业务过程逻辑 bp 的一次执行.对$\forall bai\in bpi$,其 id 都等于过程实例的 id.

在同一个过程实例中,活动实例 a_i 和 a_j 具有相同的过程实例 ID,数据流的调度控制主要是指控制同过程实例下的数据交互.

同一个 Web 文档内部业务对象间的交互可以直接进行变量引用,不同 Web 文档间业务对象的交互不仅需要传递数据,并且要求能够从共享数据中准确定位业务对象.所以,工作流引擎除了能够调度活动外,还应该发挥对数据流的控制能力.Web 文档间的数据交互模式实际上是一种多任务表之间的交互模式.

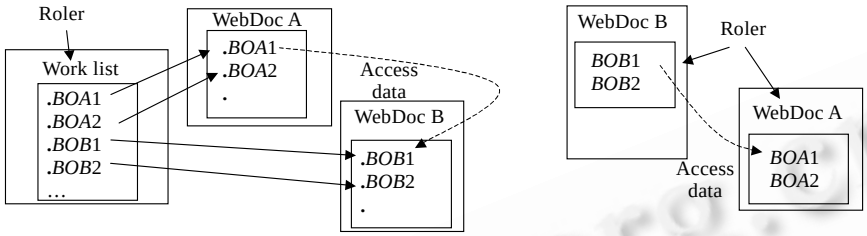
3.3 多任务表模式

按照 WfMC 的定义,任务表是指分配给某用户并等待用户处理的任务项队列.不同角色的用户登陆后,从任务表中选取任务项,然后再打开与之关联的应用对其进行处理.在这一管理模式下,任务表成了业务对象间交互的场所.

工作流任务表管理器可以作为工作流管理系统的一部分提供给用户,也可以是用户自己编写的程序[14].任务项的表示虽然可以采取多种形式,如列表、E-mail(Metro 项目)等,但其目的都是为了管理用户要执行的任务.

所以,除了给用户分配任务以外,任务列表还起两个作用:角色的权限控制和业务对象管理,而基于 Web 文档的业务对象管理模式更能体现这两个作用.基于以上理解,Web 文档就是任务表管理器,Web 文档中待处理的业务对象就是任务项.因此,这是一种多任务列表模式,集成到一起就是一个过程构件,表示一个业务过程.多任

务列表模式一方面简化了用户操作,使用户对业务对象的操作更加直接;另一方面也便于数据流的控制.图 2(a)中,业务对象 *BOA*1 和 *BOB*1 之间的数据交互需要两个 Web 文档都处于打开状态,而我们提出的多任务表模式(图 2(b))更加适合数据流的控制.与传统的人工检索获取数据的方式相比,基于工作流引擎的数据流控制能够自动完成数据的正确交互.这里所说的“正确”是指两个交互的业务对象应该属于同一个过程实例.



(a) Work list pattern defined by WfMC
(a) 工作流管理联盟定义的任务列表模式

(b) Multi-Work-List pattern
(b)多任务列表模式

Fig.2 Difference of work list pattern
图 2 任务列表模式的差别

3.4 权限控制

企业人员的权限最终表现为对业务对象的操作控制上.一个业务对象经常由多个角色共同操作完成,每个角色负责一类属性,比如,一个设备对象的 3 类属性——基本属性、使用属性和固定资产属性,就分别由设备管理员、车间人员和固定资产管理员分别处理.角色和业务对象的映射关系可以表示为 $M_{rd}=\{m_{rd}|m_{rd}\in(2^R\rightarrow D)\cup(R\rightarrow 2^D)\}$,其中 R 和 D 分别表示角色集合和业务对象属性集合.

除了这种现实的业务模式,不同角色在同一个 Web 文档中所能执行的操作也不同,比如只读、可写权限.考虑组装后的 Web 文档的空间视图,我们提出一种基于 Web 文档全局视图、按照角色的权限配置进行过滤的权限控制模式.这种模式大大增加了构件本身的可重用性.比如,采用传统的方法处理设备对象,3 类属性需要开发 3 个 Web 文档(用户视图),而我们提出的权限控制模式只需要开发一个全局视图,通过权限配置就可以直接对 Web 文档的布局产生影响,而且当 M_{rd} 需要局部调整时,只需修改配置文件即可.采用 XSLT 对 Web 文档进行权限过滤的描述方法举例如下:

```
〈xsl: template match=“INPUT//input”〉
〈xsl: for-each select=“child::display”〉
〈td〉〈input type=“text” name=“…” value=“…”/〉〈xsl: value-of select=“display”/〉〈/td〉
〈/xsl: for-each〉
```

上述描述对 Web 界面中的每一个 input 进行处理,依据变量 display 确定该 input 属性是只读还是可写.

3.5 工作流引擎

多任务表管理模式、权限控制模式决定了工作流引擎对构件的调度控制机制,工作流引擎的内部结构如图 3 所示.调度的结果最终体现在 Web 文档上,包括业务对象属性和功能操作.

矩形框表示计算,矩形框间的箭头表示过程实例的处理过程.图中涉及了活动状态,考虑到现实情况,由于业务活动未必都能顺利地执行完毕,因此首先引入并解释活动状态.从定义 5 可知,如果活动执行前需要满足谓词 P,则 P 称作活动执行的前提条件,记为 P_{pre},例如,判断某些数据是否已经准备好.若活动执行完毕后需要满足谓词 P,则称 P 为活动执行的后置条件,记为 P_{post},例如,某些数据是否全部生成.活动在前提条件不满足或者未被执行完时需要等待,直到条件满足为止.活动状态变迁过程如图 4 所示.

当活动的状态从 READY 变化至 COMPLETED,表示活动已经执行完毕;或者变化至 ABORT,表示活动被取消.活动状态的变迁主要依据 P_{pre} 和 P_{post} 这两类判定条件,如果用户考虑活动执行的时间因素,可以扩充增加 OVERTIME 和 ABORT 两个状态,活动执行时通过和用户交互确定活动状态的变迁.

活动状态变迁至 COMPLETED 后,依据过程逻辑定义开始计算后续活动,根据人员角色定义确定出后续活动的执行角色,写入工作流实例数据库,同时,根据工作流数据模型(型)构造工作流相关数据(值),如图 3 所示.计算出的后续活动用来准备产生 Web 文档的 XSL,工作流实例数据对产生 Web 文档的 XSL 起 3 个作用:1) 根据工作流实例数据,控制业务对象的创建.比如,某 Web 文档下的一个业务对象,当其状态是 CONSTRUCT 时,该业务对象的创建操作由 disabled 变为 enabled,表明该活动的执行已经满足了前提条件;2) 获取工作流实例信息,用于控制数据流,如工作流实例 ID.比如某 Web 文档下,一个业务对象属性的产生需要读取其他 Web 文档下的业务对象属性,此时就需要工作流实例 ID 作为控制条件;3) 部署视图.根据登陆用户的角色所对应的权限配置,对界面构件全局视图进行权限过滤,以及控制对业务对象属性的操作方式.

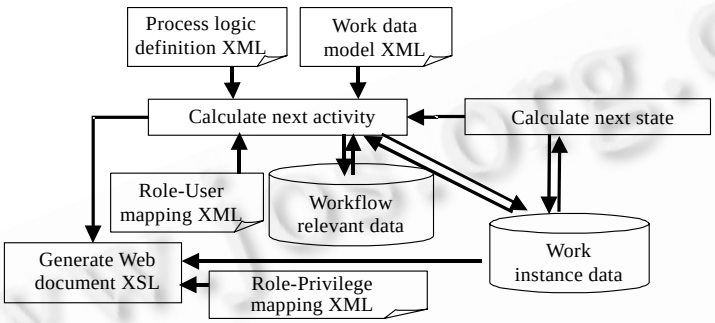


Fig.3 Structure of workflow engine

图 3 工作流引擎结构

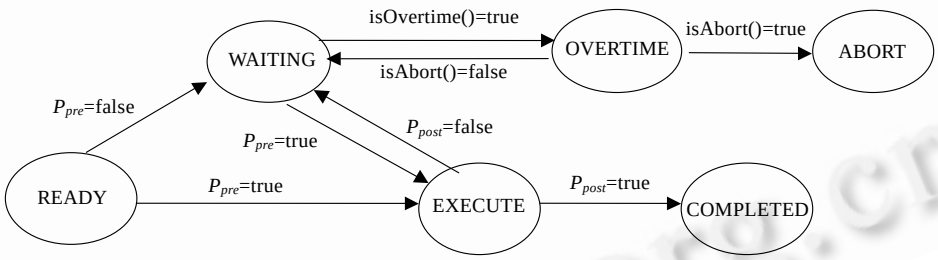


Fig.4 State transformation schema of workflow activity

图 4 工作流活动状态变迁图

4 系统结构

基于对构件环境的分析,我们开发了一个开放式、支持企业复杂应用系统建模的开发平台 ERP-Modeling. ERP-Modeling 完成了从业务建模到软件系统建模,最终到软构件生成以及应用系统部署的全过程,实现了从高层抽象描述到具体实现的映射.通过对实际应用的大量研究,我们总结出了若干种处理业务对象的典型界面模式.这些界面模式能够覆盖企业应用的大部分需求,并且可以由程序自动生成,实现了代码的批量生产.无论按照典型界面模式生成的构件还是由开发人员编写的构件,最终在工作流引擎的调度下组装(集成)在一起,实现业务过程,系统结构如图 5 所示.

很明显,组装过程构件时,工作流引擎起到调度、协调的作用,主要体现在 3 个方面:

(1) 根据用户登陆角色对全局视图进行权限过滤,形成用户视图提供给用户.一个业务对象往往由多个角色共同完成,除了第 3 节中设备对象的例子,还有维修计划对象、派工单对象等都属于这种情形.我们的方法不是按照角色生成构件,而是先生成构件的全局视图,再根据用户权限配置形成用户视图.因此,这种权限控制模式提高了构件的可复用性,如图 5 中左侧部分所示.这种设计为修改业务对象属性的访问权限带来了巨大的方便,用户只需修改系统配置,就可以快速完成用户权限的设置,并在下一次打开 Web 文档或者刷新后生效.

(2) 在运行时,计算活动的状态体现在 Web 文档的功能操作上,形成控制流.工作流引擎根据工作流相关数据模型的定义,在运行时测试过程控制条件,再根据测试结果和过程定义计算后续活动,并把调度结果直接体现在交互式 Web 文档(活动)的功能操作上.这主要依靠对界面操作的使能来控制.在图 5 中,工作流引擎把运行时的数据传送给 XSL 文件中的变量,对用户视图实现动态控制.

(3) 为数据流控制代码提供数据基础.这些数据来自于工作流引擎的处理结果,并且用于数据流的控制代码,以此保证同一过程实例间业务对象的交互.因此也可以说,对数据流的控制是工作流引擎功能的延伸.并且,通用的控制代码可以在构件开发阶段由 ERP-Modeling 直接生成并嵌入到交互式 Web 文档中,降低了构件的开发成本.

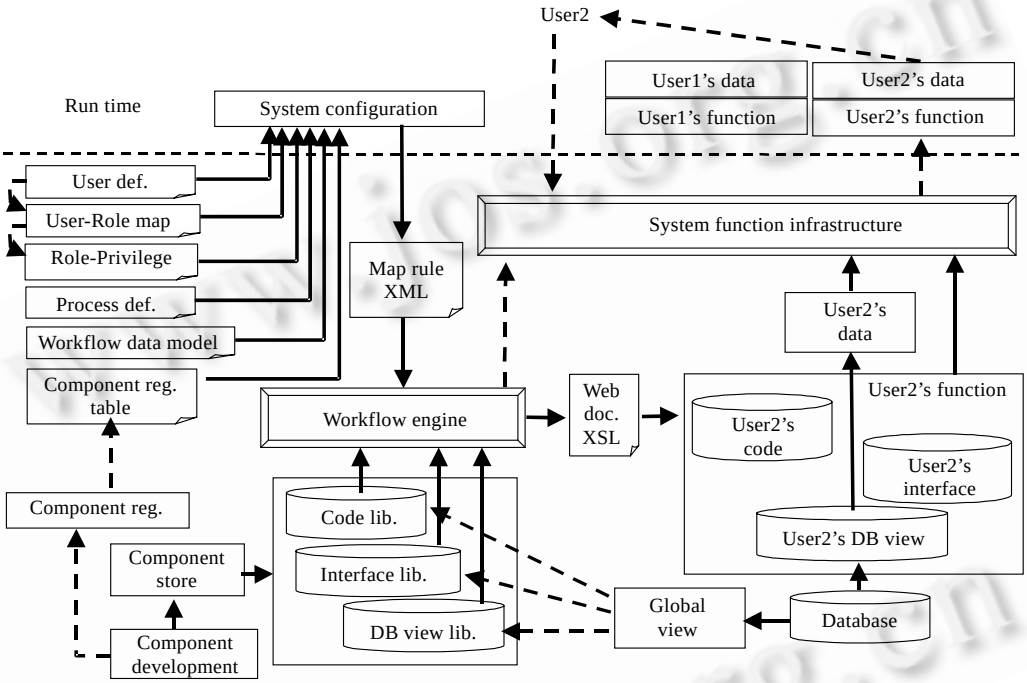


Fig.5 Workflow engine based component composition architecture

图5 基于工作流引擎的构件组装总体结构图

5 应 用

我们开发的应用系统,绝大多数代码都是基于 ERP-Modeling 生成的.这里以设备维修过程为例,说明基于工作流引擎的构件组装方法.以如图 1 所示的业务过程为例,首先获取工作流相关数据,并依据谓词确定后续活动.图 6 是相应的 XSL 片断.

```
〈script language="JavaScript"〉
    function EnableAction(String WfInstanceID,String ActionName){
    〈!- code here for the function-〉
〈xsl: template match="/approval"〉
    〈xsl: variable name="approval-result" select="approval-result"/〉
    〈xsl: variable name="Instanceid" select="//InstanceID"/〉
    〈xsl: if test="equal($approval-result,'Maintained by Workshop')=true"〉
      EnableAction($Instanceid,"Record man-hour");〈/xsl: if〉
    〈xsl: if test="equal($approval-result,'Professional maintenance')=true"〉
      EnableAction($Instanceid,"Dispatch");〈/xsl: if〉
    〈xsl: template〉
```

Fig.6 Segment of control flow specification in xsl

图 6 控制流 xsl 描述片断

图 7 是派工活动的 Web 文档的 HTML 文件片断,〈%isshow%〉和〈%isreadonly%〉作为变量,在运行时替换成具体用户的权限,即权限过滤.用户角色和操作的映射关系 XML 以及 XSL 描述略.例如,如果设置当前派工人员无权修改维修材料用量,则后 3 行代码的〈%isshow%〉根据用户和权限的映射关系取值为“hidden”.

```
〈td〉派工单号: 〈input type=〈%isshow%〉 value=“〈%=dispatched%〉” size=12〈%isreadonly%〉 〈td〉
〈td〉故障原因: 〈input type=〈%isshow%〉 value=“〈%=Faultrsn%〉” size=12〈%isreadonly%〉 〈td〉
〈td〉维修部门: 〈input type=〈%isshow%〉 value=“〈%=Mnt-dept%〉” size=10〈%isreadonly%〉 〈td〉
〈td〉技术需求: 〈input type=〈%isshow%〉 value=“〈%=Tech-req%〉” size=12〈%isreadonly%〉 〈td〉
〈td〉检查标准: 〈input type=〈%isshow%〉 value=“〈%=Chk-std%〉” size=25〈%isreadonly%〉 〈td〉

〈td〉维修员: 〈input type=〈%isshow%〉 value=“〈%=Person%〉” size=13〈%isreadonly%〉 〈td〉
〈td〉工种: 〈input type=〈%isshow%〉 value=“〈%=WorkType%〉” size=13〈%isreadonly%〉 〈td〉
〈td〉工时: 〈input type=〈%isshow%〉 value=“〈%=MntHour%〉” size=15〈%isreadonly%〉 〈td〉

〈td〉维修材料: 〈input type=〈%isshow%〉 value=“〈%=MntItem%〉” size=15〈%isreadonly%〉 〈td〉
〈td〉单位: 〈input type=〈%isshow%〉 value=“〈%=Unit%〉” size=15〈%isreadonly%〉 〈td〉
〈td〉数量: 〈input type=〈%isshow%〉 value=“〈%=Num%〉” size=15〈%isreadonly%〉 〈td〉
```

Fig.7 Segment of access control specification in HTML

图 7 体现权限过滤的 HTML 片断

图 8 表示了控制流、业务对象间的数据流交互过程.首先说明控制流:工作流引擎根据角色的权限定义,确定该角色能够访问过程的两个活动——派工和领料.因此,另外两个活动——出库和完工被过滤掉.一个派工业务对象处理完毕后,领料任务表中自动出现一个待处理的领料业务对象与它相对应,便于用户区分业务过程实例.数据流方面:制定维修派工单时,需要填写故障记录号,而故障记录号则是故障报告的处理结果.工作流引擎产生的控制数据中包括了所有活动实例产生的业务对象的定位信息,这些定位信息被传送到 Web 文档中,很容易从业务数据库中自动地提取业务对象属性,用户也可以打开故障记录界面,直接读取全部属性,而不必再通过检索的方式查询.

Fig.8 Workflow engine based process scheduling

图 8 基于工作流引擎的过程调度

6 结束语

本文从分析业务对象、控制流、数据流等最基本的特征入手,以业务过程的动态组装为目的,首先分析了构件之间数据流、控制流的特点,并且给出了工作流引擎的内部结构,以适应这些特点;最后,提出了一个以建模平台为高层描述工具,最终把模型描述映射成可执行软件的系统结构.文章还以一个具体例子说明了构件组装的过程.与国外一些学术研究机构所提出的体系结构和构件组装方法相比,我们提出的构件组装方法的主要优

点在于:1) 工作流引擎不仅作为过程调度的核心机构,而且发挥了其对数据流的控制作用,能够很好地体现构件组装的动态性;2) 借助于建模平台,自动生成典型的功能构件、界面构件,自动生成数据流控制代码(胶合代码),减少构件组装的出错几率;3) 权限过滤方式极大地提高了构件的可复用性,数据流的自动控制也使工作流引擎的调度控制能力得到更大的发挥.

由于在本文提出的过程构件组装方法中,组成业务活动的构件之间依靠静态调用的方式进行组装,所以,工作流引擎以某种调度规则直接调度构件实体是进一步的研究内容.

References:

[1] Xiang JL, Yang J, Mei H. ABC-Tool—An architecture-based component composition. Journal of Computer Research And Development, 2004,41(6):956–964 (in Chinese with English abstract).

[2] Mayer A, McGough S, Furmento N, Lee W, Newhouse S, Darlington J. ICENI dataflow and workflow: Composition and scheduling in space and time. In: UK e-Science All Hands Meeting. 2003. 627–634. http://www.nesc.ac.uk/events/ahm2003/AHMCD/pdf/132.pdf

[3] Mei H, Chen F, Feng YD, Yang J. ABC: An architecture based, component oriented approach to software development Journal of Software, 2003,14(4):721–732 (in Chinese with English abstract). http://www.jos.org.cn/ 1000-9825/14/721.htm

[4] Scheben U. Hierarchical composition of industrial components. Science of Computer Programming, 2005,56(1–2):117–139.

[5] Gössler G, Sifakis J. Composition for component-based modeling. Science of Computer Programming, 2005,55(1–3):161–183.

[6] Sadaoui S. Composition of structured process specifications. Electronic Notes in Theoretical Computer Science, 2003,82(5):1–12. http://www.elsevier.nl/locate/entcs/volume82.html

[7] Kim J, Spraragen M, Gil Y. An intelligent assistant for interactive workflow composition. In: Vanderdonckt J, Nunes NJ, Rich C, eds. Proc. of the Int'l Conf. on Intelligent User Interfaces (IUI-2004). New York: ACM Press, 2004. 125–131.

[8] Alda S, Cremers AB. Towards composition management for component-based peer-to-peer architectures. Electronic Notes in Theoretical Computer Science, 2005,114:47–64.

[9] Regehr J. Scheduling tasks with mixed preemption relations for robustness to timing faults. In: Baruah S, ed. Proc. of the 23rd IEEE Real-Time Systems Symp. Austin: IEEE Press, 2002. 315–326.

[10] Tokuda L, Batory D, Evolving object-oriented designs with refactorings. Journal of Automated Software Engineering, 2001,(8): 89–120.

[11] Caballero R, Demurjian S. Towards the formalization of a reusability framework for refactoring. In: Gacek C, ed. Proc. of the 7th Int'l Conf. on Software Reuse. LNCS 2319, Berlin: Springer-Verlag, 2002. 293–308.

[12] Russell N, ter Hofstede AHM, Edmond D, van der Aalst WMP. Workflow data patterns. QUT Technical Report, FIT-TR-2004-01, Brisbane: Queensland University of Technology, 2004. http://is.tm.tue.nl/staff/wvdaalst/publications/p220.pdf

[13] Hollingsworth D. The workflow reference model, Document No. WfMC-TC-1003. Workflow Management Coalition, 1995. http://www.wfmc.org/standards/docs/tc003v11.pdf

[14] Fan YS. Foundation of Workflow Management Technique. Beijing: Tsinghua University Press, 2001. 63–66 (in Chinese).

附中文参考文献:

[1] 向俊莲,杨杰,梅宏.基于软件体系结构的构件组装工具 ABC-Tool.计算机研究与发展,2004,41(6):956–964.

[3] 梅宏,陈锋,冯耀东,等.ABC:基于体系结构、面向构件的软件开发方法.软件学报,2003,14(4):721–732. http://www.jos.org.cn/1000-9825/14/721.htm

[14] 范玉顺.工作流管理技术基础.北京:清华大学出版社,2001.63–66.

李海波(1972－),男,黑龙江哈尔滨人,博士生,讲师,主要研究领域为工作流技术,软构件技术.

徐晓飞(1962－),男,博士,教授,博士生导师, CCF 高级会员,主要研究领域为企业智能计算,计算机集成制造 CIMS,管理与决策信息系统,数据库,敏捷虚拟企业集成技术,企业资源计划 ERP 与供应链管理 SCM 技术,电子商务与商务智能,知识工程及应用.

战德臣(1965－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为现代集成制造系统,现代企业管理与企业资源计划系统,虚拟企业集成,电子商务系统.