基于过程组件模型的流程引擎设计

朱小杰,徐 京,刘 华
(北京卫星信息工程研究所,北京 100086)

摘 要:针对目前工作流界标准众多、流程引擎和定义语言绑定、各业务领域重复开发流程引擎的现状,设计并实现一种基于过程组件模型的流程引擎框架。该框架采用流程虚拟机的架构设计,从多个流程语言提取流程的共性,摈除流程语言间活动类型的差异性,从流程定义中分离节点执行定义,从而支持多种流程定义语言。验证结果表明了该设计的可行性。这种过程组件框架能够降低从业务模型到流程定义语言之间的损耗,实现流程引擎与流程语言的解耦。

关键词:工作流;商业流程管理;工作流引擎;过程组件模型;流程虚拟机
中图分类号:TP311.52 **文献标识码**:A **文章编号**:1673-629X(2011)10-0108-04

Design of Process Engine Based on Process Component Model

ZHU Xiao-jie, XU Jing, LIU Hua
(Beijing Institute of Satellite Information Engineering, Beijing 100086, China)

Abstract: Aiming the defects of current workflow status, such as multifarious standards, process engine combing with defining language, designing process engine repeatedly in different domains, design and implement a workflow engine framework based on process component model. This framework better supports the transition from analysis process diagrams to executable processes, can handle multiple process languages, and give the benefit of loose coupling and higher extensibility in application. The feasibility was verified in the practical applications.

Key words: workflow; business process management(BPM); workflow engine; process component model; process virtual machine

0 引 言

在进行工作流系统的开发之前,首要解决的是流程定义模型的选择问题。由于工作流过程建模的复杂性,建立可读性强又可以被计算机接受的模型[1]变得尤为重要。所以目前工作流的规范都是围绕如何建立高效实用的流程建模方法而来的。但是总体而言,工作流标准还处于制定阶段,现在存在大量相互重叠的规范[2]。文章试图从新兴的过程组件模型的角度避开从规范出发来构建引擎,解决引擎构建对规范的强依赖关系。

1 引擎总体设计

1.1 总体结构设计

对于工作流界的规范,可谓仁者见仁智者见智,目前不少人对这些规范作了比较和对比。例如,有人对BPEL和XPDL作了分析和对比[3,4],得出了以下结论:BPEL才是"执行语言",而认为XPDL主要用来"建模";而有的专家认为XPDL是一个建模和可执行的混合体。

总的来说,可以将这些规范划分为两种:第一个层面是面向过程建模[1,5]的;第二个层面是面向可执行的,如表1所示。目前的标准和规范最大的问题都是不能二者兼顾。

表1 工作流规范分类

面向过程建模标记的规范	可执行规范
	jBPM的jPDL
BPMN	WfMC的XPDL
UML	OASIS的BPEL
EPC	BPMI的BPML
	OMG的BPDM

在建模者眼里,最初的焦点并非技术,而是描述人和系统协作方式的非技术的业务分析[6]。可执行过程是作为业务过程管理系统输入的软件部件。它通常由图形化的图标表示。面向定义的流程描述都是以图形为基础的,而面向可执行的流程描述通常是以XML描述为基础的。

这么多的流程定义语言形成的原因在于工作流在不同的环境和特性的应用。例如,BPEL关注于企业服

收稿日期:2011-03-28;**修回日期**:2011-07-04
基金项目:中国空间技术研究院自主研发项目(GCE-R);中国空间技术研究院CAST创新基金项目
作者简介:朱小杰(1986-),男,安徽安庆人,硕士研究生,研究方向为卫星信息管理技术;徐 京,研究员,研究方向为遥感信息处理。

务总线(ESB),非常适合将一组现有服务组合成一个新服务;BPMN[7]适合分析员做分析图,但不能执行。

在构建流程引擎的时候,一般会选择某一种图形定义描述流程后,再选择某一种过程语言负责执行。例如可以选择 BPMN 加 XPDL 的组合,BPMN 加 BPEL 的组合,或者 UML 加 jPDL 的组合。引擎的设计与实现都是基于这些标准而来的,节点的运行期行为与流程定义语言里定义的节点类型是一一绑定的,这也造成了流程引擎与特定流程语言的绑定,要支持其他的流程语言变得困难。即流程引擎和流程语言是绑定的。

同时,在将图形化描述流程后,将其转化为过程语言的时候,由于标准本身存在的,BPMN 和 jPDL 以及 XPDL 都是基于图的,而 BPEL 是基于块的,这造成了当将业务人员使用 BPMN 所建立的流程模型向 BPEL 执行模型进行转换时,出现许多的不匹配,最初的流程模型会扭曲变形。分析模型与实现模型之间的转化存在很多难题,到现在依然无法实现它们之间无损耗的平滑过渡,而扭曲的后果就是业务人员与开发人员之间的协作困难[8]。直到现在,还有许多人关注于怎么样建立一个最好的流程定义语言,或者怎样很好地将一个定义语言转化为另外一个。

如果从另外一个角度来看待以上两个问题,从这些过程语言中抽取公共部分,将这部分作为流程定义语言的公共层,通过一套通用的流程模型可以支持多个流程定义语言,也可以自定义可执行的流程语言,这样就能很好地解决引擎实现对标准的强依赖关系。过程组件模型就是这样的工作流新技术。这为将分析过程图更好地转换成可执行过程的过程语言提供了支持,解决流程引擎与流程语言绑定的问题。这无疑能够分离分析员和开发者之间的关注点,跨越分析模型与实现模型之间的鸿沟。

流程虚拟机[9,10]要在各种流程语言之间寻找它们之间的共同点,建立起一个总体的模型,流程语言是在此模型之上作为一个流程结构实现,在流程虚拟机之上,可以根据具体的业务需求选择合适的流程定义语言,并可以自定义流言。在一个理想的基于过程组件模型的流程引擎中,流程虚拟机和可执行流程语言之间的关系[11]如图 1 所示。

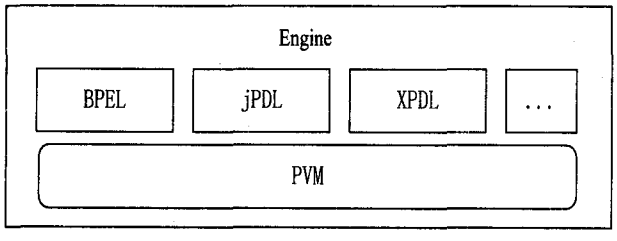

图 1　总体结构图

1.2　流程定义对象模型设计

对于各个流程语言,它们流程组织的原理和方式是不同的,例如 BPEL 是基于树形结构的[12],强调的是“信息交互”;XPDL 是基于图形结构的,强调的是“活动关系”;jPDL 是基于 UML 活动图结构的。

但总的来说,在一个流程中,最基础的结构是由节点和连接弧组成的,连接弧拥有方向,节点还可能拥有嵌套的节点。节点的类型在不同定义语言之间是不同的。如果摒除这些差,只求共同点,就可以得到所有定义语言一个共同的结构,如图 2 所示。

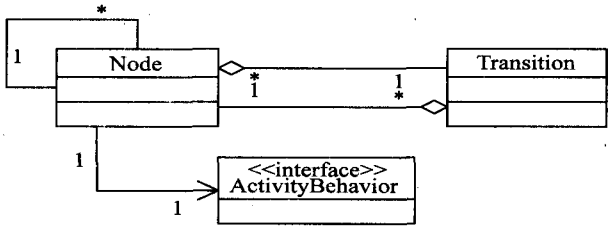

图 2　定义对象模型设计

对于一个节点来说,从定义角度,其实只关心这个节点的转入 Transition 和转出 Transition。对于一个连接弧来说,只关心连接弧两端的节点是什么,即源节点和目的节点分别是什么。这样就简化了流程定义对象的设计,不同节点的执行动作的类型放在 ActivityBehavior 对象中。

这样最大的好处就是为后续加入不同流程定义语言提供了极大的方便。对于不同流程语言有不同节点类型的情况,可以在外围进行扩展实现,此例中展示的是一个基于 BPMN 的自定义流程定义语言。若加入 XPDL 流程定义语言,即继承 ActivityBehavior,形成 XPDLActivity,以此类推,可以形成 jPDLActivity 等等。实际上,过程组件模型的最大意义就是从流程定义中分离了节点执行的定义。

1.3　内核设计

通常来讲,一个引擎的设计,涉及到流程定义、流程调度和执行、流程实例对象等多个方面的问题。流程定义是如何用一套定义对象,来诠释所定义的流程;流程调度指提供什么的机制,可以确保流程能够处理复杂的“流程图结构”,调度机制有很多种实现方法,而大部分与所依赖的模型有关;执行机制是指当流程运行到某个节点的时候,是否执行此节点,并如何执行此节点的问题,并维持节点状态生命周期;流程实例对象用来描述流程实例运行的状态和结果。

对于流程虚拟机的设计,考虑到要兼容各种流程语言,所以在设计之初应尽量将一些诸如“时间调度”、“消息服务”等功能性的设计放到外围,而只涉及到流程语义和运转的基本特征。按照流程定义对象模型设计[13],其仅包含最基本的对象和服务,以及用于

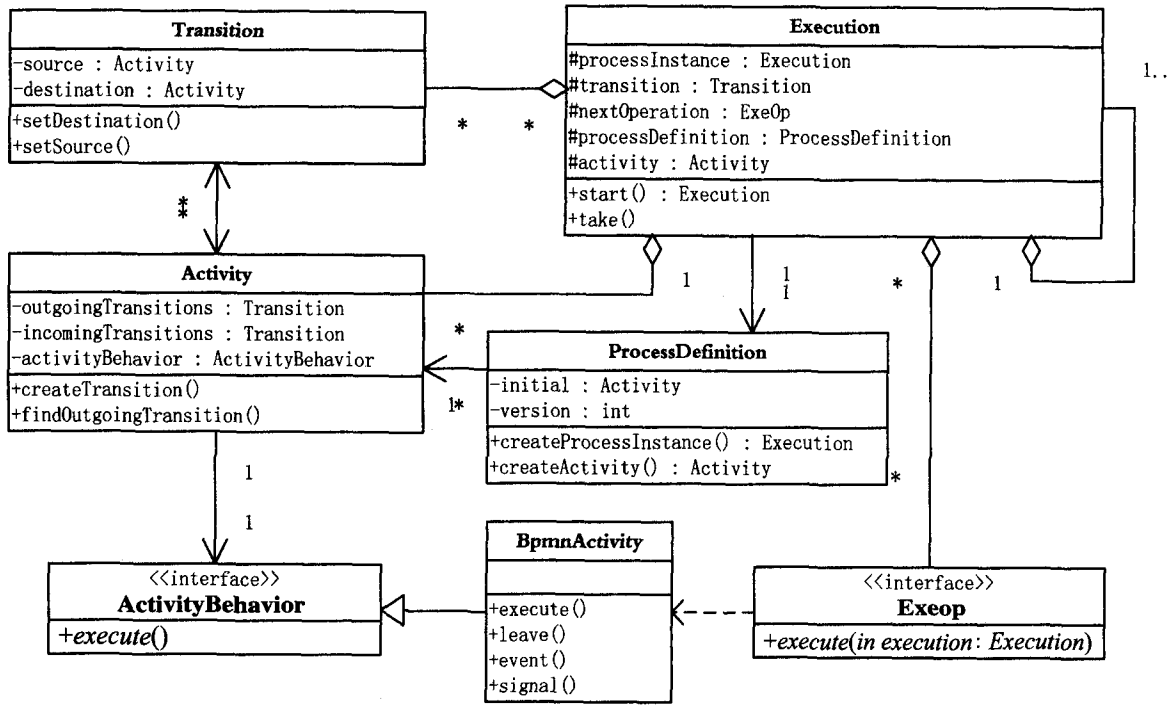

图 3 虚拟机内核简略类图

解决流程运行问题的调度机制和执行机制[14]，如连接弧和节点，连接弧和节点组成的流程定义和流程实例。对虚拟机设计更应尽可能简略，其职责是非常“精简”的：确保流程按照既有的定义，从一个节点运行到另一个节点，并正确执行当前节点。内核简略类图如图 3 所示。对于不同节点类型的定义和执行机制关系到不同流程定义语言，将在后文介绍。

2 调度与执行机制

2.1 节点的调度

各节点的调度和执行，即确保流程按照预先的逻辑定义进行流转和执行。Execution 根据 ProcessDefinition 的定义，先创建一个流程实例并设置开始节点，然后维持此流程实例上下文环境。为力求在完成调度与执行功能的基础上尽可能地轻量级，设计时将上下文环境和流程实例都放在 Execution，摒除了很多引擎采用的 Token 机制[11]的调度方法。当流程流转到某个 Activity 节点，Execution 依据不同的 Activity 类型获取其相应动作并执行后，再将环境对象移交给 Execution，由 Execution 根据上下文环境再获取此节点的转出连接弧，再转交 ExeOp 对象执行完连接弧的相应事件信息后，Execution 由此弧线得出其下一步的 Activity，将它移交给上下文环境。具体的流转和执行由 ExeOp 调用不同的 ActivityBehavior 来完成，以此往复，直至流程运行结束。图 4 为调度机制顺序图。

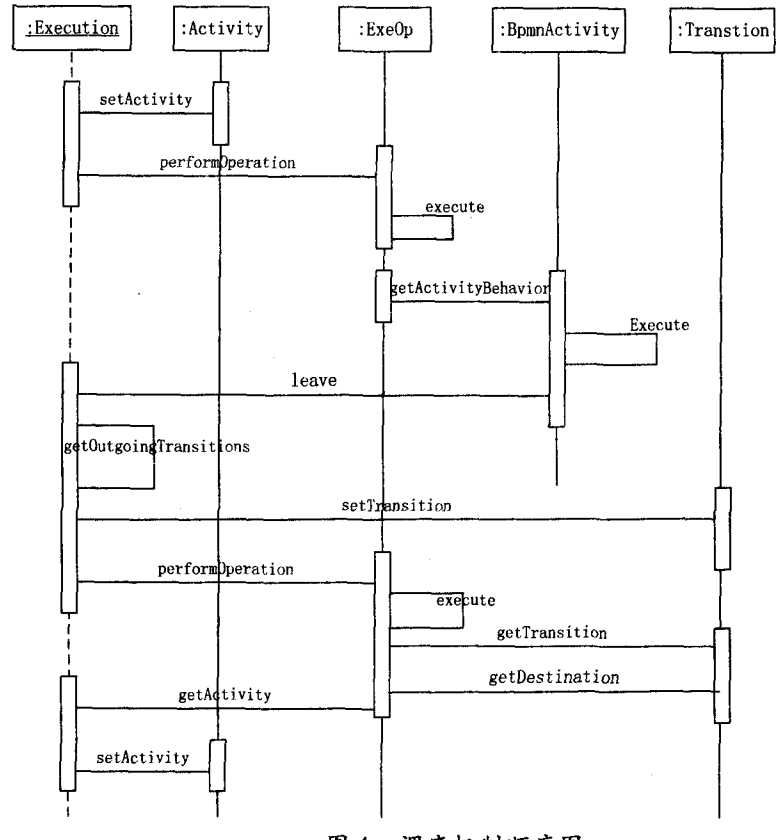

图 4 调度机制顺序图

2.2 节点的执行机制

在执行机制的设计上，仍然依照流程定义对象模型的设计。对于一个流程而言，总有开始和结束，开始

总会做一些初始化工作,结束时总会做消除对象工作;对于一个节点而言,总有进入节点、执行节点、离开节点这几个动作;对于一个连接弧来说,仍然有进入、执行和离开这三个动作。对于以上,采用事件监听的方式来进行的,事件监听器允许执行外部的 java 代码或表达式,事件可以通过以下几种方法被捕获:开始和结束一个流程实例,进入一条连接弧,开始和结束一个节点。对于每个活动点具体的动作,由继承自 ActivityBehavior 的各节点类型决定。

3　节点类型设计

对于不同的流程定义语言,其流程结构是不同的,活动类型和功能、执行机制是不同的。流程虚拟机考虑到对于多种定义语言的兼容性,将各种语言的活动类型的执行抽象为 ActivityBehavior,各个流程语言都需要继承这个接口,如图 5 所示,以基于 BPMN 的自定义过程语言为例说明在流程虚拟机上如何进行节点类型的设计。过程语言由多个活动类型组成,活动类型决定了可以实现的额外运行时行为。对于每一个活动,运行时行为可以用如 java 的通用语言实现,此活动对应一个实现组件,例如一个 Web 服务调用,一次人工操作,或一个算法计算等。可执行过程语言就是一组活动类型的实现,例如 BPMNActivity 可以细分为 StartActivity、EndActivity、RouterActivity、TaskActivity、ProcessActivity 等。ProcessActivity 代表子流程节点;RouterActivity 代表路由节点,又可以细分为并行型的和条件型的;TaskActivity 代表有任务的节点,可以细分为人工任务节点、脚本任务节点、服务任务节点等等。一个活动组件可以定义一组配置属性。例如,一个电子邮件活动可以有接收者、主题和正文的配置。这样,同一个活动实现在每次被使用的时候可以进行不同的配置。

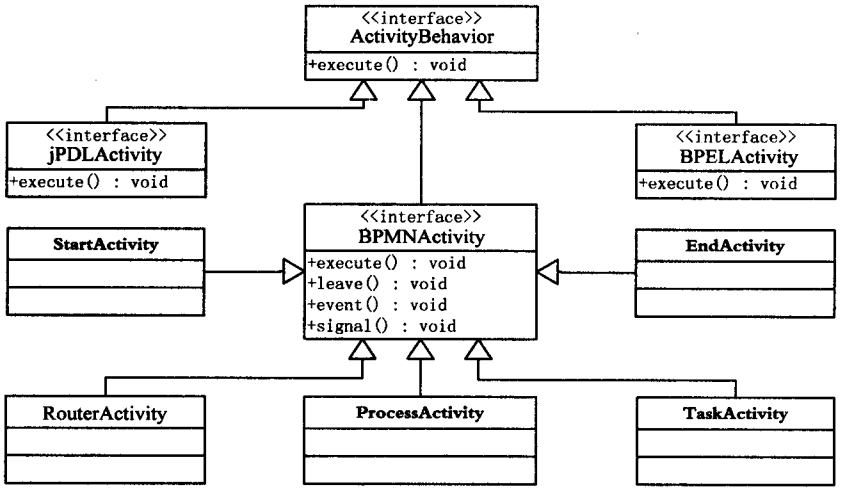

图 5　活动类型简略类图

4　结束语

对该流程虚拟机进行了实际测试[15],采用了一种基于 BPMN 的流程语言和 jPDL,对这两种不同的定义语言进行了不同的节点类型设计和二次开发,然后针对同一个流程,得出不同的流程定义,将二者交给引擎处理,能够得到相同的结果。

流程虚拟机的实现增加了流程语言的灵活性。其一是可以扩展诸如 BPEL、jPDL 或者其他的流程语言,也可以只暴露经常使用的语言的子集,创建简单的领域特定过程语言。其二以将业务领域中经常出现的功能单元进行封装,设计为一个有特定功能的活动类型,当需要的时候直接调用,即对活动类型进行扩展。其三是为一个系统中同时使用多个流程语言提供了框架基础。

文章基于过程组件模型的框架设计目前已在卫星地面信息系统中使用,但有待于对多个标准进行更深入研究,从而获取更多的共性和特性,在此基础上对流程虚拟机进一步完善,也使得后续增加流程语言更加快捷、简易。

参考文献:

[1] 汤双权,金可音,余　青,等. 工作流过程模型研究[J]. 计算机技术与发展,2007, 17(12):44-47.

[2] Baeyens T. State of workflow[EB/OL]. [2010-08-10]. http://www.jboss.com/products/jBPM/stateofworkflow/.

[3] 杨洪波. BPEL 与 XPDL 的区别[EB/OL]. [2010-10-10]. http://wenku.baidu.com/view/843be3649b6648d7c1c746ee.html.

[4] van der Aalst W M P. Patterns and XPDL:a critical evaluation of the XML process definition language[EB/OL]. [2010-08-13]. http://wwwis.win.tue.nl/~wvdaalst/publications/p201.pdf.

[5] 曹健辉,李　峰. 柔性工作流动态建模机制的研究与设计[J]. 计算机技术与发展,2009, 19(10):47-50.

[6] Baeyens T. Process Component Models:The Next Generation[EB/OL]. [2009-11-15]. http://www/infoq.com/articles/process-component-models.

[7] OMG. Business Process Model and Notation(BPMN)[EB/OL]. [2011-02-14]. http://www.omg.org/spec/BPMN/2.0.

[8] 2010 BPM & Workflow Handbook [EB/OL]. [2011-02-14]. http://www/futstrat.com/.books/book_

(下转第 115 页)

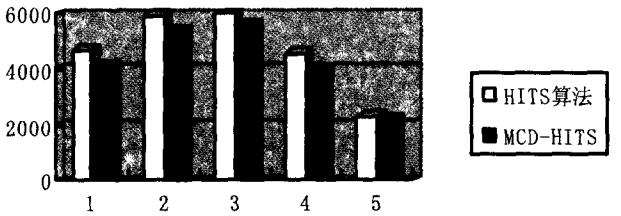

图 1　基集 web 页面数目比较

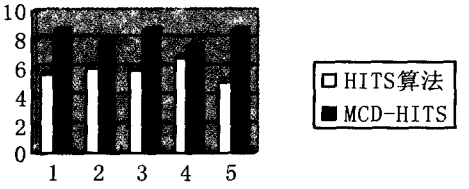

图 2　权威页面主题相关性比较

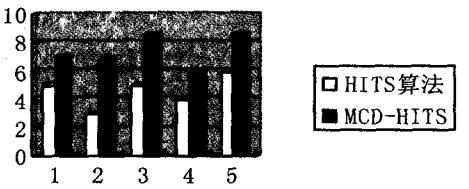

图 3　中心页面主题相关性比较

从上图 1 可以看出,MCDHITS 算法在优化了基集之后获得的 Web 页面数目明显少于传统 HITS 算法,这很大程度上节约了算法运行的系统开销。

图 2、图 3 则可以看出,改进后的 MCDHITS 算法大大提高了查询结果的精确度,同时较大程度上降低了传统 HITS 算法发生主题漂移问题的可能性。

5　结束语

文中在深入研究了网页超链接结构之后,在传统的 HITS 算法基础上,针对原算法出现的问题提出了一种改进算法 MCDHITS。该算法一定程度上解决了 HITS 算法的主题漂移问题,并提高了 HITS 的精确度。算法通过对 R 集进行的两层扩展而得到一个比原来范围更广的 Web 网络子图,之后采用最大流最小割 S-T 算法思想来发现以 R 集为中心的社区。社区很好地优化了基集,在高质量的基集上利用页面相似度的空间向量投影算法思想,最终获取高的 Authority 页面和 Hub 页面。实验结果表明 MCDHITS 算法提高了查询结果的查全率和查准率,并降低了算法运行的系统开销。

参考文献:

[1] 欧阳柳波,李学勇,李国徽,等. 专业搜索引擎搜索策略综述[J]. 计算机工程, 2004(13):32-33.

[2] 杜光芹,张化祥,赵瑞东. 主题 Web 挖掘研究 [J]. 计算机技术与发展, 2008,18(2): 94-97.

[3] 卢虹宇. Web 结构挖掘中 HITS 算法的研究 [D]. 成都:西南交通大学, 2008.

[4] 王宇新,刘海峰,郭　禾,等. 一种有效的专题信息集中和检索策略[J]. 计算机应用研究, 2010(6): 2106-2108.

[5] 常　庆,周明全,耿国华. 基于 PageRank 和 HITS 的 Web 搜索[J]. 计算机技术与发展, 2008,18(7): 77-79.

[6] 罗林波,陈　绮,吴清秀. 基于 Shark-Search 和 Hits 算法的主题爬虫研究[J]. 计算机技术与发展,2010,20(11): 76-79.

[7] 邱东洋,汤小春. 一种基于超链和锚文本分析的主题发现算法[J]. 微电子学与计算机, 2009(6):125-128.

[8] Flake G W,Lawrence S, Giles C L. Efficient identification of web communities[C]//Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. USA: ACM Press, 2000: 150-160.

[9] Flake G W,Lawrence S, Giles C L,et al. Self-organization of the web and identification of communities[J]. Communities, 2002, 35(3): 66-71.

[10] 杨　楠,弓丹志,李　忱,等. Web 社区发现技术综述[J]. 计算机研究与发展, 2005(3):439-447.

[11] 罗彩君. Web 社区结构挖掘的研究与应用[D]. 西安: 西北大学, 2008.

[12] Deng R M A G. A mixed weight HITS in link analysis[C]// Proceedings of the 2010 International Conference on Circuit and Signal Processing (ICCSP 2010) & 2010 Second IITA International Joint Conference on Artificial Intelligence (IITA-JCAI 2010). Shanghai:[s. n.],2010

[13] 王艳华,张　纪. Web 结构挖掘及其算法 [J]. 计算机工程, 2005(S1):125-127.

(上接第 111 页)

images/2010_BPM_Handbook_Free_Chapters. pdf.

[9] Baeyens T. Process Virtual Machine [EB/OL]. [2009-11-21]. http://docs. jboss. com/jBPM/pvm/article.

[10] Activiti Team. Activti User Guide[EB/OL]. [2010-09-11]. http://www. activiti. org/userguide/index. html.

[11] jBPM Team. jBPM User Guide[EB/OL]. [2010-08-11]. http: //www. jboss. org/jBPM.

[12] 胡长城. 开源工作流平台 jBPM:过程组件模型与 PVM[J]. 程序员,2008(5):116-118.

[13] Workflow Management Coalition. The Workflow Reference Model[EB/OL]. [2010-06-20]. http://www. wfmc. org/Published-Research/View-category. html.

[14] 叶　娜,李　健. 工作流引擎推进过程中 m 选 n 问题的研究[J]. 计算机应用研究,2009,26(11):4098-4100.

[15] 余　阳,汤　庸,潘茂林,等. 时态工作流过程模型及其合理性验证[J]. 软件学报,2010,21(6):1233-1251.