文章编号:1001 - 9081(2005)01 - 0085 - 03

面向 Web Services 动态复合的流程自动化系统的研究与实现

周 坤,邓保华,林齐圣,姚丹霖
(国防科学技术大学 计算机学院,湖南 长沙 410073)
(zknotes@sina.com)

摘 要:采用工作流复合 Web 服务,设计并实现了支持动态复合 Web 服务的业务流程管理系统 WSBPMS,给出了 WSBPMS 的总体结构并描述了其中的关键技术,包括 Web 服务与工作流活动的动态绑定技术、流程的自动和半自动执行,以及 Web 服务的 QoS 最优化选择等。

关键词:Web 服务;BPEL4WS;BPM;工作流;QoS;UDDIe
中图分类号:TP311.52 **文献标识码**:A

Research and implementation of process automation system oriented dynamic compounding of Web service

ZHOU Kun, DENG Bao-hua, LIN Qi-sheng, YAO Dan-lin
(School of Computer Science, National University of Defense Technology, Changsha Hunan 410073, China)

Abstract: Combining the Web services with the workflow, WSBPMS which supported dynamical compounding of Web services was designed and realized. General architecture of WSBPMS and its critical techniques were introduced, including dynamical binding Web services to the workflow, supporting automatic and half-automatic execution of process and optimizing choices of Web services by QoS.

Key words: Web Services; BPEL4WS; BPM; Workflow; QoS; UDDIe

0 引言

如何实现灵活的、方便的企业业务信息系统集成一直是信息领域的核心问题,传统的 EDI 和 EAI 技术,均不能有效地使企业后台业务应用和企业外部的客户、供应商业务应用有机的、动态的联系起来。Web 服务技术的出现,为跨组织边界的系统集成提供了有效手段,特别是基于过程复合 Web 服务的方法成为了集成分布式的、异构的、自治的应用系统的有力工具。Microsoft、IBM,以及 BEA 等业内主要厂商通力合作,将 IBM 的支持图形化的流程描述语言规范 WSFL[2] 和 Microsoft 的结构化流程描述语言规范 XLANG[3] 结合起来,于 2002 年 8 月推出 Web 服务的业务流程执行语言(Business Process Execution Language for Web Services, BPEL4WS)规范[1],该规范目前已经成为 OASIS 标准组织维护的 Web 服务标准。

Web 服务技术不受系统架构、地域和技术方案的限制和约束,可以非常快捷、灵活和方便地集成现有 IT 资源;以工作流方式对业务流程建模,实现业务流程处理的自动化,是目前较为成熟和合理的方式。因此,采用工作流方式复合 Web 服务来实现开放的、清晰的、松耦合的业务流程,是下一代商业软件的模式,为企业业务集成提供了一种新的解决方案。

本文结合现有工作流系统的研究,设计并实现了复合 Web 服务的业务流程管理系统(Business Process Management System, WSBPMS)。和已有的研究不同,WSBPMS 从 Web 服务的角度出发,阐述 Web 服务复合的问题,在 Internet 工作流中嵌入复合 Web 服务,提供了复合 Web 服务的设计、运行和监控平台。本文主要描述了 WSBPMS 的总体结构和关键技术,介绍了 WSBPMS 框架的核心功能和主要特点,并对下一步工作进行了展望。

1 WSBPMS 总体结构

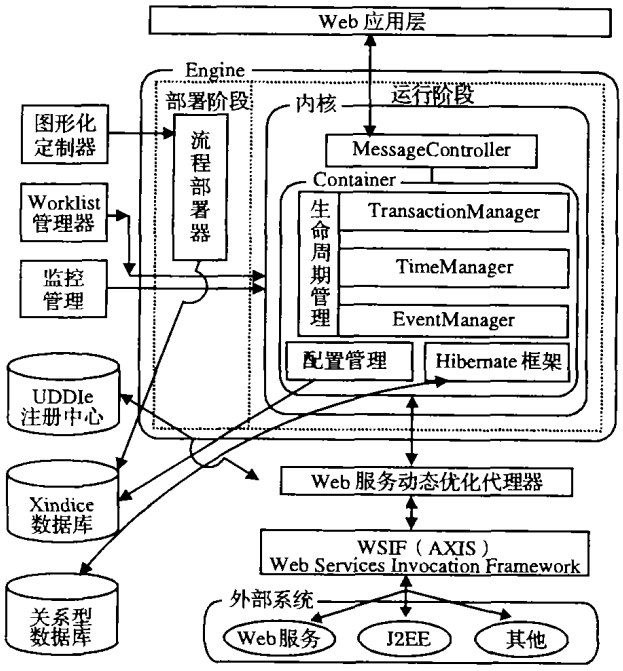

图 1 WSBPMS 总体结构图

WSBPMS 涉及到两个关键技术:基于工作流的业务流程自动化和基于 Web 服务的企业应用集成。通过 Web 服务技术本身的 UDDI 的运行时态查询、WSDL 的动态解析绑定等机制,使企业应用系统在实时条件下被动态地集成。通过工

收稿日期:2004 - 06 - 15;**修订日期**:2004 - 09 - 07

作者简介:周坤(1978 -),男,硕士研究生,主要研究方向:软件工程、分布式计算; 邓保华(1981 -),硕士研究生; 林齐圣(1980 -),男,硕士研究生; 姚丹霖(1964 -),男,副教授,博士,主要研究方向:软件工程、计算机网络.

作流技术来描述业务流程的过程重组和整体规划,实现业务流程的自动化。上述两个关键技术的结合,实现了 Internet 上的、具有动态特性的、支持自动化的业务流程集成系统。

WSBPMS 的整体结构遵循 WFMC(国际工作流管理联盟)在工作流领域的标准化定义[6],WSBPMS 的流程定制语言将严格遵循 2003 年 5 月发布的 BPEL4WS 1.1 规范;流程过程节点即参与业务流程的企业应用作为 Web 服务动态的绑定。WSBPMS 中的引擎(Engine)按照 WFMC 的标准被定义为"一个能够给过程实例提供运行时的执行环境的软件服务",它可作为独立的 Web 服务发布或者集成在企业应用中。用户或参与者通过任务列表管理器与 Engine 在运行时态进行人工交互来实现流程的半自动化执行。

WSBPMS 按功能划分为:图形流程定制器,引擎(包括流程部署器和引擎内核)、任务列表管理器、监控管理和 Web 服务动态优化代理器等模块。其整体体系结构图如图 1 所示。

2 关键技术

2.1 动态绑定技术

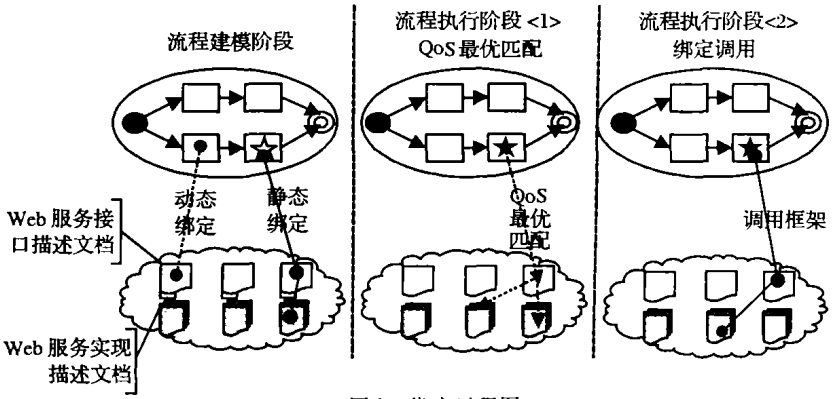

图2 绑定过程图

在 BPEL4WS 1.1 规范中,整个业务流程中扮演实现具体活动的角色(包括流程本身)都被标识为 Web 服务,通过 BPEL4WS1.1 描述为 Web 服务之间的流模型,而在这个流模型中,存在多个表示具体 Web 服务的结点,这些结点通过业务流程逻辑组织起来。在传统的工作流中,在组织和描述业务流程逻辑的同时,结点与其结点的实现就已经绑定,这存在着很大的局限性。在 WSBPMS 系统中,采用的 BPEL4WS1.1 流程定义规范是建立在 WSDL 规范之上的。在 WSDL 规范中,portType 和 binding 元素用来表示可重用的 Web 服务的抽象定义,它们可以被多个 Web 服务实现。service 和 port 元素用来表示 Web 服务的具体实现定义。利用 WSDL 规范提供了抽象功能描述与具体实现细节的分离机制,结合工作流模型的建模时机,对流程结点与具体的 Web 服务之间的绑定可采取两种方式:

(1) 静态方式:建模时绑定具体的任务,即传统的工作流任务绑定形式。在建模的过程中,如果任务对应的 Web 服务的抽象功能描述和具体实现描述都已明确且不会轻易改动,则把该任务静态地绑定到相关的流程结点上(如图 2 中空心星号所示)。具体的静态绑定实现除了指定流程定义中 < invoke > 活动的 Web 服务的功能性描述信息 portType 和 operation 属性之外,还需明确给出具体 Web 服务的实现信息 port 中的端点引用信息,最后引擎运行流程时将这些属性指定给 Axis 的 call 调用对象自动完成具体 Web 服务的调用。

(2) 动态方式:建模时静态绑定任务的功能性描述,在执行时动态绑定任务的实现性描述,如图 2 中实心星号所示。建模时,事先并不知道哪些潜在的 Web 服务实现可能满足任务的要求,仅仅为流程的 < invoke > 活动指定 Web 服务的功能性描述信息 portType 和 operation 属性,并不与具体的 Web 服务实现绑定。只有在流程模型被引擎解释执行时,引擎通过流程模型定义中的 Web 服务的功能性信息和选择匹配策略(比如 QoS 选择策略)来从 UDDI 注册中心获取满足需求的 Web 服务具体实现,引擎根据功能性信息 portType 从 UDDI 中获取实现该接口的所有 Web 服务的实现信息,通过选择匹配策略选择最符合需求的 Web 服务具体实现来调用。

虽然在静态方式中过死的绑定任务存在缺陷,但如果能够确定某个任务在应用过程中不会变更,那么建模时直接绑定该任务,可以节省动态绑定时查找任务所带来的系统开销。

动态方式中充分利用 WSDL 规范提供的分离机制和工作流建模的绑定时机来对流程结点与具体的 Web 服务进行绑定。Web 服务的抽象描述信息 portType 和 binding 映射为 UDDI 中 tModel 数据结构。Web 服务的实现信息 service 和 port 映射为 UDDI 中的 businessService 和 bindingTemplate 结构,这些实现定义结构通过对 tModel 的引用与抽象定义结构建立关联。在流程执行时,根据结点的抽象定义和该抽象定义与其实现定义的关联关系动态查询 UDDI 注册中心获取该抽象定义的所有 Web 服务实现定义,通过一定的选择策略(如 QoS 最优化机制)来过滤选择最优的 Web 服务实现,完成任务执行。

2.2 自动化与半自动化执行技术

在工作流模式中,任务是业务活动的形式化表示,既可表示自动化活动也可表示人工活动。在 BPEL4WS 1.1规范中,人工任务经常也被集成到业务流程过程中,用户参与的交互很自然的作为异步来处理,BPEL4WS1.1 规范采用 < invoke > 和 < receive > 来实现异步活动。由于 BPEL4WS 1.1规范并非定位于工作流描述语言,没有对工作流中的任务列表管理提供支持,为此必须通过现有 BPEL4WS 规范组合实现人工交互过程。BPEL4WS 流程中人工交互过程如图 3 所示。

首先,流程定义中 < receive > 活动接受初始化消息,然后用 < assign > 活动为创建人工活动的 workItem 准备请求消息。人工活动的请求信息可以附加任意的 XML 文档作为人工任务的负载信息。通过构造好的请求消息来创建人工活动 workItem 并传送相应的负载数据,然后在 listWorkItem.jsp 中列出创建的 workItem,并通知与这个 workItem 相关的用户。

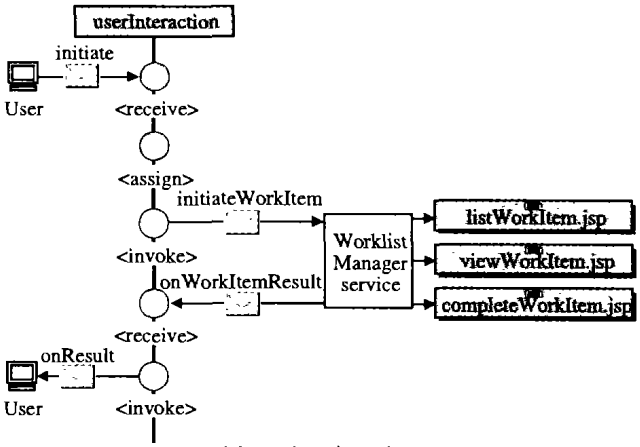

图3 人工交互过程

用户登录后,选择查看属于自己的 workItem。viewWorkItem.jsp 根据 BPEL4WS 流程定义的人工任务的响应消息中的数据类型动态生成 form,该 form 为用户提供的用户参与活动所需修改和处理的数据信息,用户通过响应该 form(如输入相应订单数目并确定订单)来提供响应消息需要的负载信息。Engine 定时等待(在流程定义时就指定好延迟时长 duration)人工任务 WorkItem 的完成,用户处理完人工任务 workItem 并将数据返回给流程的 < receive > 方法,从而结束人工交互过程。其 BPEL4WS 的片断描述如下:

```
< variables >
    < variable name = "workItemRequest"
      messageType = " workitem: workitem Message"/ > . . .
</variables >
< assign name = "setPayload" >
    < copy >
      < from variable = "reviewTask"/ >
      < to variable = " workItemRequest " part = "payload"/ >
    </copy >
</assign >
< invoke name = "initiateWorkItem"
  partnerLink = "review"
  portType = "workitem: workList"
  operation = "initiateWorkitem "
  inputVariable = "workitemRequest"/ >
. . .
< variable name = "workitemResponse"
  messageType = " workitem: workItem Message"/ > . . .
< receive name = "receiveWorkItem Result"
  partnerLink = "review"
  portType = "workitem: WorkListCallback"
  operation = "onWorkItemResult"
  variable = "workItemResponse"/ >
< assign name = "readPayload" >
    < copy >
      < from variable = "workItemResponse" part = "payload"/ >
      < to variable = "reviewWorkItem"/ >
    </copy >
</assign >
```

2.3　QoS 最优化问题

Web 服务具有高度自治性,用户在真正执行之前并不知道需要执行到哪些 Web 服务以及何时执行、可靠性如何等。业务流程中一个任务的质量将影响整个业务流程的质量。为了满足用户的期望,在 Web 服务的 WSDL 描述中扩展对 QoS 指标的描述。在流程执行时刻的动态绑定中,根据用户输入的 QoS 需求查找 Web 服务,确定对用户而言最优化的 Web 服务。

当前版本的 WSDL 对于 Web 服务的描述能力非常有限,它没有对服务的 QoS 度量进行具体的定义和描述,UDDI 服务发现机制也没有根据 QoS 指标查找服务的功能。为此,我们对现有 WSDL 服务描述和 UDDI 的数据结构进行扩展,在保证与现有规范兼容的同时,支持 QoS。

由于同一 Web 服务功能性描述文档可能被多个具有不同 QoS 性能的 Web 服务实现文档所引用,为了区分不同的 Web 服务实现,在 Web 服务接口实现描述文档中加入 QoS 描述机制:扩展 Web 服务接口实现描述文档中的 service 元素,在其中加入 property 子元素,同时,对相应的 UDDI 规范中的数据结构也进行扩展。现有的 UDDI 规范中包含五种数据结构:businessEntity、businessService、bindingTemplate、publisherAssertion 和 tModel。为了映射 WSDL 文档中的 QoS 描述,在 UDDI 规范中扩展了一个新的数据结构,构成了 UDDIe(Universal Description, Discovery and Integration Extension)。扩展的数据结构为 porpertyBag,由 porpertyName、porpertyType 和 porpertyValue 三部分组成。由于不同的服务提供商群体之间可能有不同的 QoS 标准,因此必须对 QoS 标准进行分类。UDDI 是一种非常灵活的机制,UDDI 中的 tModel 是一种可以自定义的技术指纹,通过使用 tModel 注册规范的信息,以达到在一个 UDDI 注册中心中建立一个唯一的技术标识的目的。一旦通过这种方法注册之后,就可以方便地在 categoryBag 中增加一个相应的 tModel 标识符(称为 tModelKey)的引用来表示提供了符合某种规范的 Web Service,因此,tModel 可以成为一种很好的描述 QoS 需求的方法。采用 tModel 作为 QoS 标准分类的载体,其最大的优点就是它具有很好的扩展性。为了满足不同商业的需求,可以定义更多、更详细的 tModel 作为 QoS 标准模板,通过不同 tModel 键值的组合来定位不同级别的 QoS 需求。扩展后的 WSDL 到 UDDIe 注册中心的映射关系如图 4 所示。

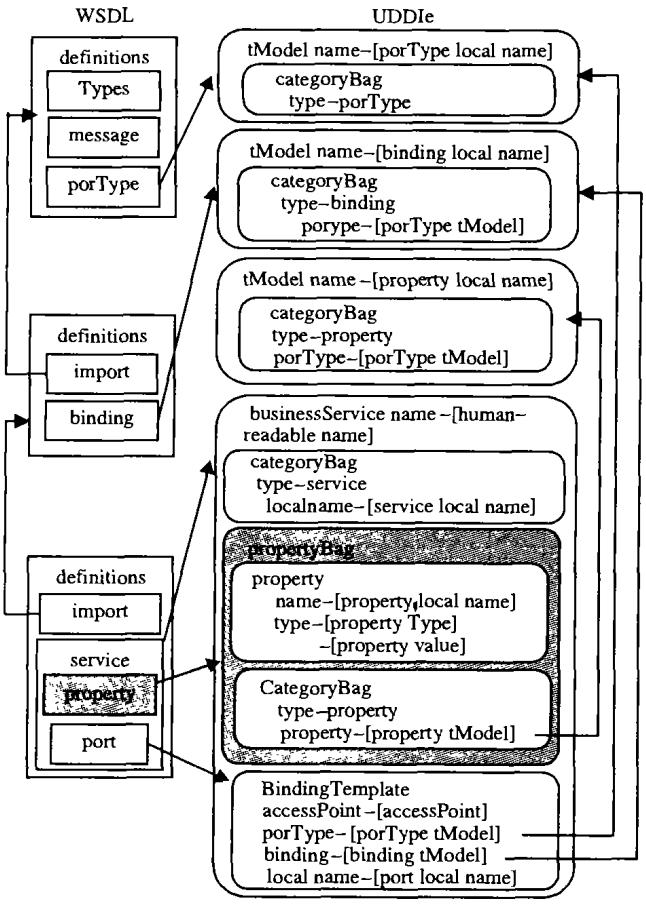

图 4　WSDL 与 UDDIe 之间映射图

根据上面的扩展,将 Web 服务与 QoS 机制有机地结合在一起。为了实现基于 QoS 机制的动态绑定,还需要提供一套可扩展的、灵活的 QoS 模型及其公平的、开放的 QoS 计算实现框架。在相应的计算实现框架中将这些标准分为两种类型:可确定的和非确定的。可确定性 QoS 标准就是当一个 Web 服务被调用时就能知道 QoS 标准的值,如执行价格等;非确定性 QoS 标准是指当 Web 服务被调用时还不能确定的,如信誉等级,执行时间等。在实现时,针对确定性 QoS 标准,服务提供者在WSDL中发布QoS的值;对于非确定性QoS,通

(下转第 90 页)

提高了 CADS 回归测试的自动化程度。通过它,我们不必在测试之前把每一个需要运行的测试用例都写进自动化测试框架中,而只需要编辑配置文件,让配置文件来控制具体需要运行的测试用例。自动化框架只要在进行测试前读取配置文件就可以自动化地完成回归测试。

图 3 是自动化回归测试系统的框图。回归测试之前,测试人员在 CVS 中准备好测试库和配置文件。回归测试开始后,自动化测试框架会从 CVS 中获取最新的测试库以及配置文件并开始自动测试。测试结束以后,自动化测试框架会记录并发布测试结果。测试人员可以根据测试的结果以及测试的日志来更新 Bugzilla 中的错误信息。在整个回归测试中,除了需要手动编辑配置文件以及向 Bugzilla 报告错误外,其余的过程都是全自动完成的。需要特别指出的是,在 Bugzilla 中报告的每一个错误,它们的标题都需要包含相应测试用例的名字,这是 4.3 节中需要的配置文件的来源。

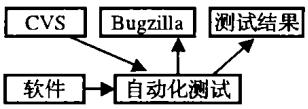

图 3 自动化回归测试

4.3 特殊的部分回归自动化测试

CADS 在一些项目节点,除了要进行全回归测试外,还需要运行一种特殊的部分回归测试。这种部分回归测试要求测试在此之前曾被发现的所有错误。它更关注软件的变化对于已经发现过的错误产生的影响。对 CADS 项目而言,它需要知道软件新版本是否能修复原有的错误,或者已被修复的软件错误是否重新发生。它与全回归测试的不同之处在于测试框架的配置文件。全回归测试的配置文件包含所有的测试用例,而部分回归测试只需要包含在错误管理软件中记录过的错误的测试用例。

图 4 是这种特殊需求的部分回归测试的自动化框图。可以看出它和图3不同的是,部分回归测试多了来自Bugzilla的输入。这个输入用来生成特定的配置文件。在测试开始之前,测试框架首先向 Bugzilla 数据库获取所有记录过的错误。由于测试用例命名规范,而且在每个软件错误的标题中均有与之对应的测试用例名,因此自动化框架可以容易地提取这些测试用例名,重新生成配置文件,用新生成的配置文件替换掉原有文件。自动化框架不需要额外改动,就可以完成这种特殊要求的部分自动化回归测试了。最后的测试结果也一样通过网页显示,并手动更新 Bugzilla 中错误的状态。

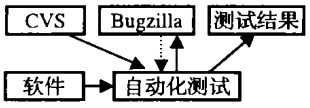

图 4 特殊的自动化回归测试

5 结语

自动化的测试大大减少了测试人员的人为失误,提高了测试效率。Java 语言的反射机制以及规范化的开发自动化测试框架,使得全回归和部分回归测试都能使用同一平台,提高了开发自动化平台的效率。自动化回归测试虽然需要额外的前期投入,但是可以及时地反馈软件版本的质量,帮助开发人员迅速发现错误,随着软件开发的不断前进,实际上反而大大节省了资源。实践证明自动化回归测试对于保证 CADS 项目的质量起了非常好的作用。综上所述,自动化回归测试非常适合于开发周期长,而且开发人员比较多的大型软件开发项目。

参考文献:

[1] KANER C, FALK J, NGUYEN HQ. Testing Computer Software [M]. Canada: Wiley Computer Publishing, 1999.

[2] ECKEL B. Thinking in Java [M]. 京京工作室, 译. 北京: 机械工业出版社, 1999.

[3] BARNSON MP. The Bugzilla Guide - 2.16.5 Release [EB/OL]. http://www.bugzilla.org/documentation.html, 2004-3-21.

[4] Cederqvist P, *et al.* Version Management with CVS [EB/OL]. http://www.cvshome.org/docs/manual/, 2004-5-1.

(上接第 87 页)

过激活的执行监控和用户反馈机制来获取 QoS 标准的值。限于篇幅,有关 QoS 最优化机制在此不作详细讨论。

3 结语

本文针对跨企业级的企业业务流程的过程重组中传统工作流技术绑定过死问题和目前系统集成技术中紧耦合、高成本以及灵活性差等问题,从整体规划的角度,结合现有的工作流技术和 Web 服务技术,为系统开发人员给出了一种以全局的角度来调用 Web 服务来实现业务流程管理系统的模型,提出了 Web 服务与工作流活动的动态绑定技术、自动和半自动执行技术,以及 Web 服务的 QoS 最优化选择机制。

按照动态程度的递增,业务流程自动化的动态性分为三个级别:第一级,Web 服务本身技术特性提供的动态性,如 UDDI 的运行时态查询、WSDL 的动态解析绑定等特性;第二级,依据工作流节点中绑定的时机和策略的选择,动态地选择参与者(Web 服务应用实例),即每次执行过程中参与角色的具体实例可以不同;第三级,Web 服务工作流不仅可以动态选择参与者,业务流程本身也可动态选择,并且可引入支持最新发布的基于 WSRF 的网格服务来以工作流方式协同共享 Internet 上所有资源。本文重点讨论了第二级动态性实现机制,下一步重点将集中开展第三级动态性机制的研究与实现。

参考文献:

[1] IBM, Microsoft, Bea. SAP Business Process Execution Language For Web Services 1.1 [EB/OL]. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel, May 2003.

[2] Web Services Flow Language (WSFL) Version 1.0 [EB/OL]. http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf, 2002.

[3] XLANG: Web Services for Business Process Design [EB/OL]. http://www.gotdotnet.com/team/xml wsspecs/xlang-c/default.htm, 2002.

[4] OASIS. Universal Description, Discovery and Integration of Bunsiness for the Web - Specifications [EB/OL]. http://www.uddi.org/specification.html.

[5] WSDL [EB/OL]. http://www.w3.org/TR/wsdl.html

[6] Workflow Management Coalition. Published on Web (2003) [EB/OL]. http://www.wfmc.org.

[7] [QoS4WS] Understanding quality of service for Webservices [EB/OL]. http://www-106.ibm.com/developerworks/library/wsquality.html.

[8] RAN S. A Model for Web Services Discovery With QoS [J]. ACM SIGecom Exchanges, 2003, 4(1): 1-10.

[9] AL-ALI R, RANA O, WALKER D, *et al.* G-QoSM: Grid service discovery using QoS properties [J]. Computing and Informatics Journal, Special Issue on Grid Computing, 2003, 21(5).

[10] SHAIKHALI A, RANA OF, AL-ALI R, *et al.* UDDIe: An Extended Registry for Web Services [A]. Workshop on Service Oriented Computing: Models, Architectures and Applications at SAINT Conference [C]. IEEE Computer Society Press. Florida, US, January 2003.

[11] AL-ALI RJ, RANA OF, WALKER DW. G-QoSM: A Framework for Grid Quality of Service Management [A]. Proc UK e-Science Programme All Hands Meeting [C]. Cox SJ, ed. Nottingham, UK, 2003.

[12] ZENG L, BENATALLAH B, DUMAS M, *et al.* Quality Driven Web Services Composition. In Proceedings of the 12th international conference on World Wide Web (WWW), Budapest, Hungary [M]. ACM Press, May 2003.