基于 XML 的多 Agent 系统的研究与实现*)

徐 明[1] 庄 毅[2]

(上海理工大学计算机工程学院 上海 200093)[1] (南京航空航天大学计算机科学与工程系 南京 210016)[2]

摘 要 作为构建开放和分布式应用系统的一种主流模式，多 Agent 系统有着广阔的研究前景和应用价值。在统一建模语言(UML)的支持下，面向 Agent 的软件工程研究开始走向成熟。一些面向 Agent 的方法学提供了开发多 Agent 系统的工具、应用方法或技术。随着 Web 服务技术的发展，XML 成为 Internet 上数据组织和交换的标准。现有研究工作所提出的多 Agent 系统对 XML 文档提供很少的支持。针对上述问题，设计了一个基于 XML 的多 Agent 系统——XMAS。该系统采用带根连通有向图来表示 XML 文档数据模型，并给出相应的文档模式提取算法，XML 文档数据的解析以及对 Web 服务的相关支持。在数据存储过程中的索引优化使得 XMAS 在数据查询上具有良好的性能。

关键词 多 Agent 系统，XML，数据模型，Web 服务

Research and Implementation of XML-based Multi-agent System

XU Ming[1] ZHUANG Yi[2]

(Institute of Computer Science and Engineering, University of Shanghai for Science and Technology, Shanghai 200093)[1]

(Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016)[2]

Abstract As a widely used pattern for building open and distributed applications, multi-agent systems are becoming a promising paradigm for research and application. Agent-Oriented Software Engineering becomes more and more mature under the support of the Unified Modeling Language (UML). As a result, several agent-oriented methodologies for developing multi-agent systems have been proposed, with the aim to provide tools, practical methods, and techniques. With the development of Web Service, XML becomes a standard for data organization and exchange. The current works have a little support to XML documents in their multi-agent systems. In this paper, we design a multi-agent system-XMAS based on XML. We use a Rooted Connected Directed Graph to present XML document data model and give XML schema extraction algorithm. We also offer function of parsing XML document and support to Web Service. During the data store process, index optimization makes XMAS have good performance on data retrieval.

Keywords Multi-agent system, XML, Data model, Web service

1 引言

随着网络技术与分布式人工智能理论的发展，多 Agent 系统(Multi-Agent System)的研究拓展到许多领域并得到广泛的应用。在统一建模语言(UML)的支持下，面向 Agent 的软件工程研究开始走向成熟[1,2]。一般而言，多 Agent 系统通过消息交换以实现 Agent 间通信[3]，在数据管理上缺乏丰富的数据模型。关于 Agent 方法学的讨论延伸到数据模型的扩展以及多 Agent 系统进化等问题上来[4,5]。XML 以其良好的数据存储格式、可扩展性、高度结构化、便于网络传输等特点，已成为事实上的 Internet 数据组织和交换标准，为多 Agent 系统的数据管理提供了新的数据模型。Amor 等人提出的基于模型驱动体系结构的多 Agent 系统 Malaca[6] 可以通过对 XML 文档的编辑来对 Agent 进行"编程"，但只是提供了对 XML 文档浅层次的兼容。Cabri 等人提出的基于角色的多 Agent 系统框架 BRAIN[7] 对 Agent 角色提供了 XML 的表示，可以通过 XML 文档来定义角色，但是没有提出相关的数据模型。

针对当前 Agent 方法学对 XML 支持的不足，本文提出了一个基于 XML 的多(Multi-Agent System)系统 XMAS。XMAS 采用带根连通有向图来表示 XML 文档数据模型，并给出相应的文档模式提取算法以及对 XML 文档数据的解析。在 XMAS 体系结构中，还设计了对 Web 服务的相关支持。

本文在第 2 节介绍 XMAS 的数据模型，文档模式，以及相应的模式提取算法。第 3 节介绍 XMAS 的体系结构，并对主要模块进行了功能分析。第 4 节给出了相关的实验结果和分析。最后是对全文总结，并展望了进一步的工作方向。

2 XMAS 数据管理

2.1 XMAS 数据模型

XMAS 中用带根连通有向图来表示 XML 文档，图中的节点和 XML 文档的元素、属性或元素值/属性值相对应，XML 文档元素间的关系则用带根连通有向图的边来表示。边分为三大类：元素边 E、属性边 A、引用边 R。每一个边都有一个标签，用来表示该边所对应节点的关系。每一个元素

*)基金项目：航空十五预研基金项目(编号：41801150201)资助；航空科学基金项目(编号：No. 01F52036)资助。徐 明 硕士，助教，研究方向为分布式计算，Web 数据库，多 Agent 系统。庄 毅 硕士，副教授，研究方向为信息安全，分布式计算(含网格)，网络。

节点都要求有父节点。对于文档中的根节点，假设有一个虚拟的 root 节点作为最上层节点，则在边的集合 E 中，文档根节点也作为 root 虚拟节点的值节点出现。同时，边也表示元素和属性之间的关系。

定义 2.1 在 XMAS 中，值节点、元素节点 V 是一个三元组(oid，type，value)，其中 oid 用来表示该元素的唯一标识号，该标识号由系统自动产生和维护；type 表示该元组的类型，如果是元素节点(内点)，那么 type 总是 element，如果是数据节点，那么 type∈{String、Int、Date 等}；value 对于值节点和元素节点，它们的定义各不相同，值节点中的 value 用来存储数据值，元素节点中的 value 则用来存储元素名。

定义 2.2 在 XMAS 中属性边、元素边用一个四元组(parent，child，name，type)表示，其中 parent 是指该边的起始节点，child 是指该边所指向的节点，name 是指该边的标识名，Type 指该边的类型。

节点构造操作中根据输入的参数构造节点。在一般情况下，节点构造操作不能独立作为一个表达式，而是把该操作和边的构造操作结合在一起，这样不至于产生一个孤立的节点。节点构造操作的语法如下：

$\nabla[type](value)$

其中 type 是指所构造节点的类型，如果构造的是值节点，那么 value 就是该值节点的值，如果是元素节点，那么 value 值为空。

边的构造操作形式如下：

$\nabla[edgetype,name,child\ ](parent)$

其中 edgetype 表示边的属性，它的取值或者是 A 或者是 E，也就是只允许是元素边和属性边。Name 用来指定边的标签，也就是指定边属性名或元素名。Child 是边指向的属性节点、元素节点或值节点。Parent 则指定了 child 节点的父元素节点。

由于同一父节点的子元素之间是顺序相关的，因此在创建一个元素和子元素间的边时，还需要指定边和其它同一父节点边之间的关系。在缺省的情况下，新构造节点加入到所有同一个父节点的子节点末尾。

XMAS 利用关系数据库存储 XML 文档数据模型中的节点和边，而不是把 XML 数据平坦化为结构数据进行存储，因此能更完全地保存 XML 文档中的信息。此外，还基于每一个元素节点、属性节点、各种边的 oid 建立了索引，以提高查询和连接的效率。主要加入了以下两种索引来提高系统的查询效率。一是根据 XML 文档的原子数据，在所有元素或属性可转换为数值的数据上建立索引；二是根据 XML 数据的所有元素名建立索引。

2.2 XMAS 文档模式

为了能够从 XML 文档中提取模式信息，必须先确定模式信息的表示方法。XML 文档可以用不同的模式定义语言定义，而这些语言定义又存在相似之处。XMAS 采用 XGrammar[8] 作为 XML 的模式定义语言，以通用语法作为输出，而不依赖某种具体的语言标准，在此输出的基础上，可以容易地得到所需文档模式信息。

在定义 XML 文档模式表示方式之前，先定义一些基本符号。G 代表 XGrammar 中的一个模式，$\hat{N}$ 是非终结符集合，$\hat{T}$ 是终结符集合，$\hat{E}$ 是元素名集合，$\hat{A}$ 是属性名集合，$\hat{\tau}$ 是原子数据类型集(如：String，integer 等)，包括属性的一些特殊类型：ID、IDREF 和 IDREFS[9]。ε 表示空集，“+”表示并操作，“,”表示串联，“$a^?$”表示 a 出现一次或不出现，“a^*”(Kleene Star)表示 a 可以出现任意次，“a^+”相当于“a,a^*”，表示 a 至少出现一次。下面先介绍 XGrammar 的抽象形式定义：

定义 2.3 XGrammar 用六元组表示，$G=(N,E,A,S,P,\Sigma)$。其中各元素含义分别表示如下：

N 表示非终结符集，$N\subseteq\hat{N}$；E 表示元素名集；A 表示属性名集；S 表示起始符集；P 表示形如“$X\rightarrow x(RE)$”的元素产生式规则，其中 $X\in N$，$x\in E$，正则表达式 RE 定义为：$RE::=\varepsilon|\tau|Y|(RE)|(RE+RE)|(RE,RE)|(RE)^?|(RE)^*|(RE)^+$，其中 $\tau\in\hat{\tau}$，$Y\in N$；Σ 表示模式 G 中的约束集，这里定义三种约束：

IDREF 约束：考虑属性 $a\in A$，如果 a 是 IDREF 类型，则定义 a 的目标类型为 $a::IDREF\rightarrow RE_1$，这里 $RE_1::=X|(RE_1+RE_1)$。如果 a 是 IDREFS 类型，则定义 a 的目标类型为 $a::IDREFS\rightarrow RE_2$，这里 $RE_2::=\varepsilon|X|(RE_2+RE_2)|(RE_2,RE_2)|(RE_2)^?|(RE_2)^*|(RE_2)^+$，其中 $X\in N$；

主键约束：定义主键约束 $key(X)=(p_1,p_2,\cdots,p_n)$，其中 $X\in N$，$p_i\in PE$，$1\leqslant i\leqslant n$；

外键约束：定义外键约束 $X(p_1,p_2,\cdots,p_n)REFERENCES\ Y(q_1,q_2,\cdots,q_n)$，其中 $X,Y\in N$，$p_i,q_i\in PE$，$1\leqslant i\leqslant n$。

2.3 算法设计

访问 XML 文档的 API 有两种：SAX 和 DOM[10]。SAX 是解析 XML 文档的一种标准接口，它读入 XML 数据流，当处理标记时检查定义的事件，并将该事件返回给应用程序，用户在此加入对这些事件的处理。DOM 则是先解析 XML 文档，为其构造一棵文档树。DOM 提供了对文档树的各种操作(如访问文档元素、子元素，得到文档节点数据等)，应用程序就是利用 DOM 提供的操作访问 XML 文档。利用 DOM 对文档建立文档树时对系统的内存要求较高。而 SAX 则是采用边读入边解析的方法，所以对系统要求较低，解析速度也较快。在 J2EE 平台里，增加了处理 XML 文档的 API 函数：Common DOM API 和 Simple API for XML Parsing (SAX)。

算法 2.1 描述了 XMAS 的模式提取算法。输入是 XML 文档，利用 SAX 解析 XML 文档，同时进行相应的模式提取工作，输出是该 XML 文档树所对应的模式——XGrammar。

算法 2.1

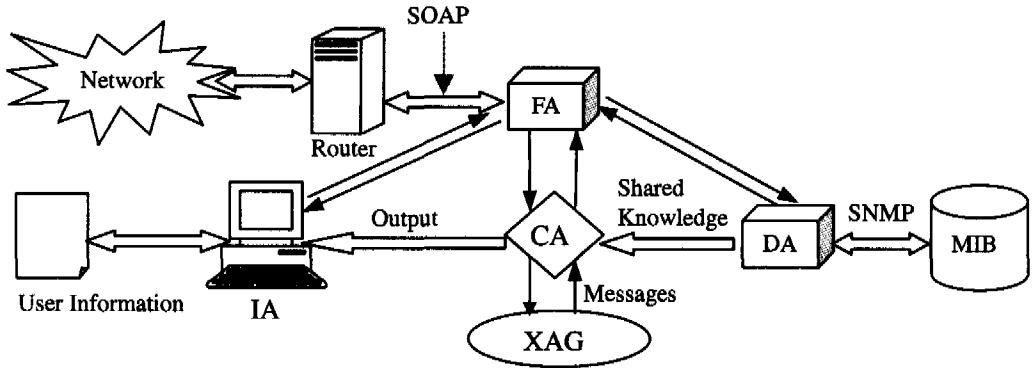
```
class CHandlerClass implements ContentHandler
{
  private Locator locator;
  Stack estack=new Stack();//作为访问的元素栈
  Stack ExprVector=new Vector();//作为元素产生式向量
  public void setDocumentLocator(Locator locator)
  {this. locator = locator;}
  public void startDocument()throws SAXException
  {…}//初始化 XGrammar 的各
  //个部分
  public void endDocument()throws SAXException
  {…}//XML 文档模式提取结束
  public void startElement(String namespaceURI,String localName,
  String rawName,Attributes atts)throws SAXException
  { if(estack. empty())
    {把元素名加入到 XGrammar 的 N、E、S,入 estack 栈}
    else {把元素名加入到 XGrammar 的 N 、E,
    ExprVector. add(new
    SingleExpr(estack. peek(),localName)));
    元素入 estack 栈}
    for (int i = 0; i < atts. getLength(); i++)
    {把 atts. getLocalName(i)加入到 XGrammar 的 A 中。}
    把元素产生式加入到 XGrammar 的 P 中。
  }
  public void endElement(String namespaceURI, String localName,
  String rawName)throws SAXException
  {根据元素名 localName,取出 ExprVertor 中元素的属性 parent-
```

```
name为localName的元素集,分析元素集得到元素产生式,如果没
有相关元素,那么把"localName->empty"加入到产生式集中。
estack.pop();
}
public void characters(char [] ch,int start,int end)
throws SAXException
{String sss = new String(ch,start,end);
estack.peek()->Data作为元素产生式加入元素产生式集中。
}
.....
}
```

该算法实现了XML文档模式的提取,使用SAX解析遍历XML文档。读入XML文档数据的同时对遇到的各种情况进行操作。该算法的空间复杂度与XML文档的大小成正比,在时间上是对XML文档的一次文件扫描。

3 XMAS体系结构

图1给出了本文的多Agent系统的体系结构。



FA:Facilitator Agent;IA:接口Agent;CA:协调者Agent;DA:数据Agent;XAG:XML Agent组;MIB:管理信息库

图1 XMAS体系结构

各功能模块的分析如下:

• Facilitator Agent (FA) 在XMAS体系结构中,为了从UDDI[11](通用描述、发现和集成协议)注册中心获得远端Web服务的调用信息,路由器通过FA来缓存信息并使用相关的调用规范,通过该Web服务注册的地址来访问这个Web服务。当使用从UDDI注册表中获得并缓存下来的信息调用失败时,正确的做法是去查询当初获得该数据的UDDI注册中心并获取与其对应的更新了的信息。如果返回的数据与缓存的信息不同,那么应该使用新的调用信息来重新尝试服务调用。如果这一重试操作获得成功的话,新的信息应当取代原先缓存的信息进入当前的缓存。

XMAS可以在UDDI注册节点注册自己的服务。方法是把服务的WSDL[12](Web服务描述语言)文件打包成SOAP[13](简单对象访问协议)消息通过HTTP发送给注册节点,注册节点的HTTP服务器判定为SOAP消息后转发给SOAP处理器,SOAP处理器判定消息为注册信息后把它交给UDDI注册服务器,注册服务器根据服务的类型把服务归入相应的类型中。

FA还可以维护服务名字注册表。路由器可以利用FA来帮助它找到信息的目的主机。此外,FA还可以直接将信息发送到服务的提供者,或者是在信息的提供者和消费者之间进行匹配。一般情况下,每一组本地的Agent拥有一个FA,但是也可以提供多个FA以增加冗余度。在为多个网络中的Agent同时提供服务的情况下,通过一组分布的FA可以获得比集中式更高的效率和可靠性。

• 协调者Agent (CA) 是一个XML Agent组的管理者(从XML Agent组中产生)。可以接受从数据Agent解析的模式信息或者XML文档。当某个子任务被分配到该组时,CA通过检查组内XML Agent的状态以及通过与Facilitator Agent协商来决定该组是否可以满足需求。在子任务的执行过程中,CA通过通信接口将异常情况反馈给Facilitator Agent。

• XML Agent组(XAG) 是加入了XML对象的一组Agent,可以将逻辑和数据封装在一起,在网络间移动(也可以驻留本地处理任务),能在具有Java运行环境的目的节点直接处理。

• 接口Agent(IA) 是用户与系统之间的交互界面。从功能和地位上考虑,它相当于客户/服务器模式的瘦客户端。为了方便用户与不同的Web接口进行交互,XMAS采用了扩展用户接口协议[14](Extensible User-interface Protocol)。该协议支持基于XML语言的用户界面开发,具有跨平台的特点。

• 数据Agent(DA) 其主要功能解析XML文档,从XML文档中提取模式信息并将XML文档中的信息保存到管理信息库(MIB)中。DA还可以从内部的知识库中获取相关的规则并进行信息交换。此外,还可以分析从其它知识库中获取的有用信息。

图2是一个XML文档的片断,包含了XML的一些基本元素、属性、引用等。

```
<root>
<person name='N1' city='C1' state='S1'>
  <book btitle='T1' ISBN='I1' BID='B1>
  <book btitle='T2' ISBN='I2' BID='B2>
      <review Aref='P1' rating='9'>
      <review Aref='B1' rating='9'>
    </person>
    <person name='N2' zip='200093'>
  <paper ptitle='T3' PID='P1>
      <review Aref='B1' rating='9'>
      <review Aref='B1' rating='10'>
      </person >
< root >
```

图2 XML文档片断

用图2的XML文档作为输入,经过数据Agent的分析得到的G=(N,E,A,S,P,Σ)如下:

N = {Root,Person,Book,Paper,Review}
E = {root,person,book,paper,review}

(下转第215页)

6 Clouqueur T, Saluja K K, Ramanathan P. Fault-tolerance in collaborative sensor networks for target detection, Computers. IEEE Transactions on, 2004, 53(3): 320
7 Christian J, Feser K. Procedures for Detecting Winding Displacements in Power Transformers by the Transfer Function Method, Power Delivery. IEEE Transactions on, 2004, 19(1): 214
8 Ayoubi R A, Ziade H A, Bayoumi M A. Fault-tolerant Hopfield associative memory on torus, Defect and Fault-Tolerance in VLSI Systems, 2003. In: Proc. 18th IEEE International Symposium on, Nov. 2003. 369~376
9 Choi M, Park N, Lombardi F. Modeling and analysis of fault-tolerant multistage interconnection networks, Instrumentation and Measurement. IEEE Transactions on, 2003, 52(5): 1509~1519
10 Stoustrup J, Blondel V D. Fault-tolerant control: a simultaneous stabilization result, Automatic Control. IEEE Transactions on, 2004, 49(2): 305~310
11 Huang J, Tahoori M B, Lombardi F. Fault-tolerance of p rogrammable switch blocks, [C] Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings, Volume: 2, Feb. 2004. 1358~1359
12 Stanley-Marbell P, Marculescu D. Local decisions and triggering mechanisms for adaptive fault-tolerance, [C] Design, Automation and Test in Europe Conference and Exhibition, 2004, proceedings, Volume: 2, Feb. 2004. 968~973
13 Jackson A H, Canham R, Tyrrell A M. Robot fault-tolerance using an embryonic array, [C] Evolvable Hardware, 2003. Proceedings. NASA/DoD Conference on, July 2003. 91~100
14 杨军,杨晨,段朝阳,等. 现代导弹制导控制系统设计. 北京: 航空工业出版社, 2005
15 宋闯,魏毅寅. 非线性系统理论在导弹控制中的应用研究进展与展望. 战术导弹技术, 2003(6): 48~53

(上接第 207 页)

A = {name, city, state, zip, BID, btitle, ISBN, PID, ptitle, art, rating}
S = {Root}
P = {Root→root (Person*),
Person→person (@name, ((@city, @state)+ @zip)(Book+ + Paper+), Review+),
Book→book (@btitle, @ISBN, @BID),
Paper→paper(@ptitle, @PID),
Review→review(@ARef, @rating)}
Σ = IDREF constraints:
{ARef::DREF→(Book + Paper)}
Key constraints:
{key(Person)= (@name),
key(Book)= (@ISBN),
key(Paper)= (@ptitle),
key(Review)= (parent :: person/@name, @Aref)}

4 实验分析

作者用 J2EE 平台和 Oracle9i 数据库管理系统在 Windows 2000 上实现了 XMAS,个人电脑的配置为 P4 2.1G 处理器,内存 256 M。采用的实验数据来源于 Jon Bosak 提供的部分 XML 文档[15]。在进行查询实验之前,需要把 XML 文档导入到数据表中存储并建立索引,表 1 列出了数据分析及导入所用时间,以及对相应文档元素属性的统计。

表 1 实验数据分析

实验数据名	元素节点数	值节点数	元素边数	属性边数	时间(秒)
Four Religious Works	6637	5472	12109	2718	22.3

对数据查询,本文根据查询目标所在的目录层次挑选了 5 条查询语句,根据优化前后对查询的执行情况,可以得到图 3 的实验结果。

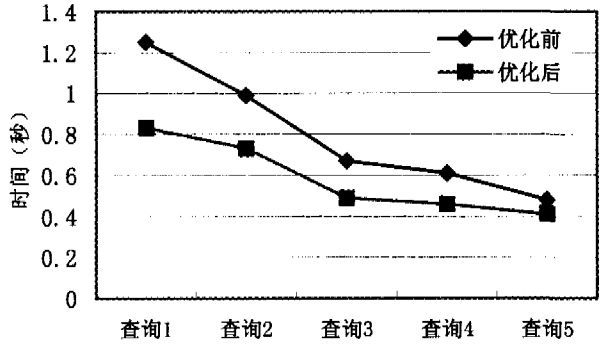


图 3 查询性能对比图

查询 1: tstmt [contains='christ']
查询 2: tstmt/bookcol [contains='christ']
查询 3: tstmt/bookcol/book [contains='christ']
查询 4: tstmt/bookcol/book/chapter [contains='christ']
查询 5: tstmt/bookcol/book/chapter/v[contains='christ']

表中的数据都是针对同一个查询语句运行 3 次所用时间的平均值。通过优化前和优化后查询执行所用时间的对比可知,优化后的查询时间普遍少于优化前的查询时间。随着查询目录层次的深入,两者查询时间的差值在减少。

在大部分情况下,基于索引的查询优化都能够使查询效率得到一定的提高。由于对于索引的维护也需要一定的系统开销,因此加入索引后在导入数据时会使其效率降低。但总的来说,对那些有频繁查询要求的 XML 数据,索引的引入还是必需的。

结论 本文介绍一个基于 XML 的多 Agent 系统——XMAS。该系统采用带根连通有向图来表示 XML 文档数据模型,并对 XML 文档的原子数据和元素名建立索引,能更完全地保存 XML 文档中的信息。XMAS 采用 XGrammar 作为 XML 的模式定义语言,以通用语法作为输出,而不依赖某种具体的语言标准,在此输出的基础上,可以容易地得到所需文档模式信息。在数据存储过程中的索引优化使得 XMAS 在数据查询上具有良好的性能。

进一步的研究工作包括对 XMAS 中 Web 服务的扩展以及查询优化策略的细化。

参 考 文 献

1 Kavi K, Kung D, Bhambhani H. Extending UML to Modeling and Design of Multi-Agent Systems. In: Proc. of ICSE 2003 Workshop on Software Engineering for Large Multi-Agent Systems, Portland, Oregon, 2003
2 Wooldridge M, Ciancarini P. Agent-Oriented Software Engineering: The State of the Art. In: First Int. Workshop on Agent-Oriented Software Engineering, LNAI 1957, 2000. 1~28
3 Tanenbaum A S, Van Steen M. Distributed Systems Principles and Paradigms. Prentice Hall Inc, 2002. 57~66
4 Sturn O S. A Framework for Evaluating Agent-Oriented Methodologies. In: Int. Workshop On Agent-Oriented Information Systems, 2003
5 Dam K H, Winikoff M. Comparing Agent-Oriented Methodologies. In: Int. Workshop On Agent-Oriented Information Systems, 2003
6 Amor M, Fuentes L, Troya J M. A Component-Based Approach for Interoperability Across FIPA-Compliant Platforms. In: Cooperative Information Agents VII, LNAI 2782, 2003. 266~288
7 Cabri G, Leonardi L, Zambonelli F. BRAIN: a Framework for Flexible Role-based Interactions in Multi-agent Systems. In: The Conf. on Cooperative Information Systems, Catania, Italy, November 2003. http://www.w3.org
8 Murata M, Lee D, Mani M. Taxonomy of XML Schema Languages using Formal Language Theory. In: Extreme Markup Languages, Montreal, Canada, Aug. 2001
9 W3C. XML-Schema Working Group. http://www.w3c.org/XML/Schema.html.
10 Document Object Model (DOM) Technical Reports. http://www.w3.org
11 UDDI Version 3 specification. www.uddi.org
12 Web Services Description Language (WSDL) version 1.1. W3C Note, Mar. 15, 2001. http://www.w3.org/TR/wsdl.
13 SOAP Version 1.2 Recommendation documents. http://www.w3.org/TR/soap.
14 World Wide Consortium's Technical Reports. Boston. 2003: http://www.w3.org/TR/
15 http://www.ibiblio.org/bosak/xml/eg/