一种基于 Agent 技术的 Web Services 匹配模型的研究*)

卞昭娟　任晓鑫　陶先平　吕　建

(南京大学计算机软件新技术国家重点实验室　南京大学计算机软件研究所　南京210093)

摘　要　Web Services 是当前 Web 应用的一种新的模式。Internet 将成为一个真正的分布式的计算平台，其上的服务可以协作完成某一任务，并且 Web 将成为可编程的。Web Services 的发现是很重要的，因此有必要研究 Web Services 的匹配。在本文中，我们提出了一种基于 Agent 的 Web Services 的两层次匹配模型。

关键词　Web Services 的匹配，Agent，UDDI，LARKS

On the Matchmaking Mechanism of Web Services Based on Agent Technology

BIAN Zhao-Juan　REN Xiao-Xin　TAO Xian-Ping　LU Jian

(The State Key Laboratory for Novel Software Technolgoy, Institute of Computers Software, Nanjin University, Nanjin 210093)

Abstract　Using Web Services is a new pattern of current Web applications. Internet will become a true distributed computation platform where services can be cordinated to finish one task and the Web will become programmable. The discovery of Web services is very important. So it's necessary to do research on matchmaking of Web services. In this paper, We put forward a two-level matchmaking mechanism which makes use of agent technology.

Keywords　Matchmaking of Web services, Agent, UDDI, LARKS

1　引言

随着 Internet 的快速发展和普及，“软件作为服务”的趋势深刻影响着软件技术的未来，在该理念的驱动下，Web Services[1]概念及相关技术应运而生并得到了迅速发展。在此基础上 Internet 将成为一个真正的分布式的计算平台，其上的服务可以协作完成某一任务，并且网络将可编程。其中需要定义最基本的动态服务交互的规则，包括如何将服务发布到某服务器上、如何查找服务以及如何程序化地绑定服务等。对于查找某个服务，除了要有在分布式的环境下灵活的查询机制外，此外关键的一点就是 Web Services 的匹配。

现有的 Web Services 匹配技术仅局限于 UDDI[2,3]全球注册处的搜索匹配，匹配方式单一且结果粗糙，对 Web Services 之间的自动集成与协作的支持不好，因而需要一种更为有效的匹配机制。从本质上来说，Web Services 是开放环境下的一种服务，除 UDDI 方式外，现有开放环境下还有使用 LARKS[4]语言的匹配以及在 RDF[5]图基础上的匹配等。然而现有服务匹配机制并不能直接用于 Web Services 的匹配，或多或少都存在某些方面的不足。将 LARKS 语言用于 Web Services 的匹配时，只能局限于较小范围和特定领域中，且由于 LARKS 语言自身的限制，匹配的内容有限且无法扩展；RDF 图的方式对 Web Services 的提供者和请求者的要求比较高，需要其提供蕴含丰富 Ontology 知识的 RDF 图，查询结果的精确性完全依赖于 RDF 图的内容[6]。

尽管现有服务匹配机制不能直接用于 Web Services 的匹配，但通过对其用于 Web Services 匹配的情况进行分析，可以得出有关 Web Services 匹配机制的一些结论：对于单一匹配方式其匹配深度与精度与匹配范围成反比；匹配信息的多少、匹配算法的复杂度以及 Ontology 库的设计与某种特定的中间表示形式相关。

本文在以上结论的基础上，借鉴 Agent 技术，提出了一种基于 Agent 的 Web Services 的两层次匹配模型，对 Web Service 不同描述层次上信息采用了不同的匹配形式进行两层次匹配，并且在匹配性能与代价之间进行折中，采用了一种用 XML 语法表示的 AMCDL 语言对匹配内容进行描述。

2　基于 Agent 的两层 Web Services 匹配模型

2.1　需求分析

考虑到以上两个结论，较为理想的 Web Services 匹配模型需满足以下两点：

(1) *对 Web Services 不同描述类型的信息采用不同的匹配形式*　对 Web Services 的联系分类等信息实现全球性质的匹配(必须符合 Web Services 的通用标准)，只能采用字符串匹配的形式。对服务更深层次的信息，如功能性及其他特征信息的匹配需要用到 Ontology 知识，匹配方式比较复杂。此时可采用两种策略提供 Ontology 知识，一是由匹配模型提供知识库，包含领域内的知识；二是由服务提供者和请求者提供。而对于有些难以匹配的有用信息，如服务的组织流程等，匹配模型中不直接匹配，而是在其他信息匹配的情况下将其作为结果的附加信息返回给用户做进一步筛选。

(2) *对匹配所需的描述信息采用合适的中间表示形式*　要对服务描述信息进行较深层次的匹配，可采用中间表示形式，从而对服务的匹配就可以转换为对中间表示形式的匹配。这种中间表示形式需要满足以下几个要求：首先，要包含足够多的信息，即可以完全描述用户的需求，这就要求该表示形式包含所有与服务匹配相关的信息或者具有可扩充性；其次，要

*)本文受国家重点基础研究发展规划973资助项目(2002CB312002)；国家自然科学基金项目(60273034)；国家863高科技发展计划资助项目(2002AA1160101)资助。卞昭娟　硕士生，研究方向：分布对象技术、移动 Agent 技术。吕　建　博士，教授，博导，研究方向：形式化方法、分布式对象技术、构件技术。

包含一定的语义信息以简化匹配算法；最后还要折中考虑代价和性能。匹配算法简单，性能高的表示形式(如 RDF 图的形式)，在形成时代价必然高，而简单的表示形式(LARKS 语言描述的形式)，其匹配时必然性能较低。

2.2 匹配对象

在基于 Agent 的两层 Web Services 匹配模型中，我们对 Web Services 的匹配对象(即 Web Services 的描述信息)进行了详细分析，根据其对服务匹配方式的影响可分为以下四个层次。

(1)联系和分类信息，主要是指服务提供者的信息，包括名称、地址、电子信箱以及其他一些描述信息；此外各服务都会在某一特定标准下属于一个或多个类别，如蔬菜或瓜类的种植服务在 NAICS 标准下的分类值为11121。

(2)服务的接口与实现信息，这是 Web Services 的功能性信息，包括服务提供的方法、出口和入口参数以及调用时刻的数据编码以及传输协议等信息。

(3)服务的质量、代价及安全、信任等信息，这些信息在开放环境下尤其重要，反应了 Web Services 的综合性能。

(4)工作流信息、访问信息及上下文信息等。一个服务可能由多个更小一级的服务(最小一级的就是原子服务)组成，这些服务之间有一定的控制流和数据流的依赖关系，服务之间可能是顺序执行或是选择执行等，一个服务的输出也可能是另一个服务的输入等等。

为满足通用性与集成性，软件协同联盟中所有服务都使用 Web Services 的通用规范标准进行描述。对于第一层次的信息用来供服务请求者做最简单的匹配，在我们的模型中这部分信息保存在私有的 UDDI 服务注册处，其信息的描述与维护由 UDDI 规范规定；对于第二层次的信息使用 WSDL 规范描述服务接口和实现信息；而对于第三层次的信息使用 WSEL 描述服务上下文、质量、代价等，使用 Ws-Trust，Ws-Security 描述安全、信任性等信息；不参与匹配但作为匹配结果返回的第四层次的工作流信息由 BPEL4WS 描述。

服务提供者在向 UDDI 服务注册处进行注册时(图1中 I)提供联系、分类信息以及 WSDL 等文件的 URL。而服务请求者查询服务时通过图形用户界面将需求提交给 Mid_Agent (图1中①)。

2.3 匹配模型

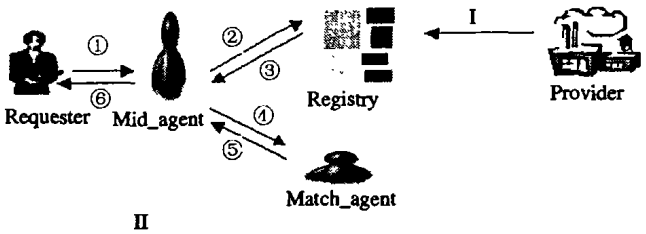

图1 基于 Agent 的两层 WS 匹配模型

第一步，对第一层次描述信息进行整个开放环境的字符串匹配。服务请求者将自己的需求以表单或客户端的方式提交给 Mid_Agent，Mid_Agent 按一定的查询机制进行迁移，向分布在开放环境中的多个服务注册处(如 UDDI 私有注册处或者是全球唯一的公共注册处)提交用户要求的联系和分类信息(如图1中②)；在服务注册处进行匹配后，Mid_Agent 得到结果(包括相应服务的 WSDL，BPEL4WS 等文件的 URL)(如图1中③)。

第二步，为第二层次的匹配做准备工作。Mid_Agent 分析服务请求者的描述内容，为第一步匹配所得到的 Web Services 搜集相应的描述信息(如服务请求者对服务的要求仅仅是服务接口信息，那么有关服务代价、质量等信息就不用搜集)，也即从③返回的各服务的 WSDL，WSEL 等文件的 URL 出发搜集具体的描述信息。并将其组织成中间表示形式；并在整个开放环境中按一定的查询机制搜集与服务请求者所需求服务相关领域的 Ontology 知识，构成 Ontology 库。Mid_Agent 将这两部分内容提交给 Match_Agent(如图1中④)，Match_Agent 就形成了查询中心。

第三步，Match_Agent 对中间表示形式根据匹配算法进行多步匹配，其匹配算法类似于对用 LARKS 语言进行描述的服务匹配算法，根据匹配度的大小，将满足需求的服务按次序返回给 Mid_Agent(如图1中⑤)。

第四步，Mid_Agent 将结果以图形方式返回给用户(如图1中⑥)。返回给用户的信息，是有关服务的比较全面的信息，其中包括服务上下文信息，工作流信息等与匹配无关的信息以及用户没有指定的匹配信息(如用户在提交请求时没有指定服务提供者等)。根据这些信息，用户可以做出进一步的选择，以确定自己使用哪一个服务。

3 匹配模型的实现

基于 Agent 的两层服务匹配模型实现于 HawkEye 查询模型，HawkEye 查询模型是由南京大学软件所提出的基于 Agent 的软件服务协同联盟中所使用的服务查询模型。

3.1 模型实现环境

在整个开放环境中，分散有多个服务注册处，当服务提供者要发布服务的时候，可以自己选择某种方式(如功能，地域等)注册到某个服务注册处；而服务请求者要找到满足自己要求的服务时，首先向 Mid_agent 提交查询要求(可通过 Web 用表单的形式提交，也可以利用客户端程序提交)，由 HawkEye 查询系统提供一种机制以确保 Mid_agent 在整个开放环境中进行有效的查询。

3.2 中间表示形式 AMCDL 语言

在匹配的整个过程中包括两次匹配行为：Mid_Agent 向分散在开放环境中的服务注册处提交第一层次的服务描述信息，进行第一次匹配，由于各服务注册处就是一个局部的 UDDI Registry，其匹配完全利用开源的 UDDI API 来实现。第二次匹配的实现与利用 Larks 语言进行的匹配相似，为满足系统的需要，我们采用了一种自定义的 Agent 匹配能力描述语言(AMCDL)，在 AMCDL 语言的基础上进行第二层次的匹配。对于其中的服务关键字，输入、输出参数及其限制条件部分进行匹配，访问知识库，利用 Ontology 知识，返回一个相似度的值；而对于服务质量、代价及安全部分的匹配，直接对数值大小进行匹配。

AMCDL 对中间表示形式进行描述，完全符合 XML 语法规范，其 Schema 如下：

```
〈?xml version="1.0" encoding="UIP-8"〉
〈xs:schema xmlas:xs="http://www.w3.org/2001/XML Schema"
    elementFormDefault="qualified"〉
    〈xs:element name="ImputConstraints"type"=xs:string"/〉
    〈xs:element name="Inputs"〉
        〈xs:comlexType〉
            〈xs:sequence〉
                〈xs:element mane="name"type="xs:string"/〉
                〈xs:element mane="type"type="xs:string"/〉
            〈/xs:sequence〉
        〈/xs:complexType〉
    〈/xs:element〉
    〈xs:element name="OutputConstraionts"type="xs:string"/〉
    〈xs:element name="Outputs"〉
```

```
        〈xs :complexType〉
            〈xs :sequence〉
                〈xs :element name="name"type="xs :string"/〉
                〈xs :element name="Type"type="xs :string"/〉
            〈/xs :sequence〉
        〈/xs :sequence〉
    〈/xs :element〉
〈/xs :simpleType name="stringList"〉
    〈xs :list itemType="xs :string"/〉
〈/xs :simpleType〉
〈xs :element name="Cost"type="xs :string"/〉
〈xs :element name="Qos"type=xs :string"/〉
〈xs :element name="Security"type="xs :string"/〉
〈xs :element name="serviceEntity"〉
〈xs :complexType〉
    〈xs :sequence〉
        〈xs :element name="serviceKeywords" type="stringList"/〉
        〈xs :element ref="Inputs"minOccurs="0"maxOccurs="unbounded"/〉
        〈xs :element ref="Outputs"minOccurs="0"maxOccurs="unbounded"/〉
        〈xs :element ref="InputConstraints"minOccurs="0"maxOccurs="unbounded"/〉
        〈xs :element ref="OutputConstraints"minOccurs="0"maxOccurs="QoS"/〉
        〈xs :element ref="QoS"/〉
        〈xs :element ref="security"/〉
    〈xs :sequence〉
    〈xs :attribute name="xmlns :xsd"type="xs :anyURI"use="required"/〉
    〈xs :attribute name="serviceKey"type="xs :string"use="required"/〉
〈/xs :complexType〉
〈/xs :element〉
〈/xs :schema〉
```

其中包含以下几部分：

serviceKeywords(sk)：服务关键字，如排序服务就用 Sort 或者 Order 来表示。

Inputs(i)：服务的输入参数，包含两部分，参数名以及参数类型。对输入参数的匹配只进行类型匹配，参数名在输入限制条件的描述中使用。

Outputs(o)：服务的输出参数，包含两部分，参数名以及参数类型。对输出参数的匹配只进行类型匹配，参数名在输出限制条件的描述中使用。

InputConstraints(ic)：输入限制条件，对输入参数的限制条件进行描述。可以用 Horn Clauses 进行描述。

OutputConstraints(oc)：输入限制条件，对输出参数的限制条件进行描述。可以用 Horn Clauses 进行描述。

QoS(q)：服务质量，用数值表示。具体的数值大小依据评价标准。

Cost(c)：服务代价，用数值表示。具体的数值大小依据评价标准。

Security(s)：服务安全性，用数值表示。具体的数值大小依据评价标准。

serviceKey(k)：UDDI 注册处提供的关于该 Web Service 的唯一标志，不进行匹配，前述匹配成功后 serviceKey 作为结果返回。

用该语言对排序服务进行描述如下：

```
〈?xml version "1. 0" encoding="DIF-8"?〉
〈ServiceEntity xmins : xsd = "http://www. w3. org/2001/XMLSchema" serviceKey = "E85A2C0B-297B-43E0-B371-86E00F8CBC1B"〉
xmlns :xsi="http://www. w3. org/2001/XMLSchema-instance"xsi :sschemalocation="F:/amcd1. xsd"〉
    〈serviceKeywords〉Sort〈/serviceKeywords〉
    〈InPuts〉
        〈name〉xs〈/name〉
        〈type〉list of initeger〈/type〉
    〈/Inputs〉
    〈Outputs〉
        〈name〉ys〈/name〉
        〈type〉list of integer〈/type〉
    〈/Outputs〉
    〈InputConstraints〉
        le(length(xs),100)
    〈/Inputconstraints〉
    〈OutpuConstraints〉
        le(before(x,y,ys),-ge(x,y))
    〈/OutputConstraints〉
    〈OutputConstraints〉
        le(in(x,ys),-iin(x,xs))
    〈/OutputConstraints〉
    〈QoS〉4〈QoS〉
    〈Cost〉5〈/Cost〉
    〈security〉4〈/security〉
〈/serviceEntity〉
```

3.3 匹配算法

第一层次的匹配由开源的 UDDI API 实现，第二层次的匹配在中间表示形式的基础上进行。AMCDL 中描述的每一匹配项的依据其对匹配结果的影响分配权重。如 ServiceKeywords 的权重比 OutputConstraints 大($Wsk > Woc$)，其中 serviceKey 的权重为一极大值。设匹配过程中的每一步的结果用 $Rx(0<=Rx<=1)$ 表示，如 ServiceKeywords 的匹配结果为 Rsk，OutputConstraints 的结果为 Roc，serviceKey 的结果为 Rk 等等。对于 serviceKeywords、Inputs、InputsConstraints、Outputs、OutputConstraints 利用知识库中的信息以及统计规律进行模糊匹配，得到近似值，而余下的几项为精确匹配，结果非0即1。最终的匹配结果可以用下式表示：

$$R = WkRk + Rs * Rc * Rq * (Wsk * Rsk + Wi * Ri + Wic * Ric + Wo * Ro + Woc * Roc)$$

在匹配算法执行过程中，先计算 Rk，如果 $Rk=1$，则直接返回为其权重设置的极大值，不进行进一步计算；接着计算 Rs、Rc、Rq，其中出现任何一项为0，则匹配失败；在进行最后五项的匹配时，一旦结果为0，匹配也失败；最终按匹配值 R 的大小将相应的 serviceKey 依次返回。

4 模型评价

在 HawkEye 查询模型中，在 Mid_agent 和 Match_agent 处各要进行一次匹配，两次匹配的对象不同。在 Mid_agent 处进行匹配的是服务的第一层次信息，具体的形式只是字符串，匹配范围广(从 UDDI 服务注册处中获取信息)；而在 Match_agent 处进行匹配的是服务的第二、三层次信息，用基于 XML 语法的 AMCDL 语言描述，匹配精度高。其表述内容并不包含大量的 Ontology 知识，这与 LARKS 语言的思想是一致的。这是由于一方面中间表示形式是由机器根据服务提供者提供的信息动态形成，其中包含的语义信息仅来自于各标准的服务描述语言的 XML 语法形式下各标记的语义；另一方面，采用这种方式，对服务请求者的要求不苛刻，不需要服务提供者提供额外于其自身服务的领域内的知识。

与 LARKS 语言方式不同的是，第一，采用了 XML 语法，对 LARKS 语言进行了扩充，描述信息丰富，不局限于 LARKS 的七个描述内容，服务匹配可能用到的多方面信息都包含其中；第二，根据服务请求者的内容，各服务的相应中间表示形式动态形成，一个服务的中间表示形式并不唯一，对于不同的服务请求者有不同的中间表示形式；第三，Ontology 库动态生成，Match_Agent 并不需要包含各领域的 Ontology 知识，仅仅需要根据需求动态装配 Ontology 知识。

总结 本文分析了现有的比较有代表性的三种开放环境下的服务匹配机制，得出了关于 Web Services 匹配的几点结论。并将 Agent 技术用于 Web Services 的匹配，提出了一个基于 Agent 的两层次匹配机制。通过在两个层次上对不同的信息采用不同的匹配方式，使得在匹配范围不变的情况下，匹

(下转第107页)

首先，我们对所有需要阅卷的知识点进行分类整理，为每一个知识点设置了唯一的编号；其次，必须详细分析每一个知识点，整理该知识点阅卷必须获取的各种参数，比如对于“字体设置”，计算机的自动阅卷必须知道所设置文字所在章、节号，所在段落，出现次数等信息，才有可能准确定位到对应的位置并读取用户的操作结果。

形式化系统的分析求解系统就是阅卷模块的主体，在分析求解系统中，系统根据对形式化描述信息的分析，从中获取阅卷时所需要的所有信息，并根据参数信息获取考生的考试结果，与标准答案进行比对，最终判断是否得分。

一个阅卷信息典型的形式化描述如下：

X,Y=	{*知识点编号[|参数1|参数2|.....]*}	运算符	标准答案
①	②	③	④

其中①部分用来说明在当前试题中该知识点所属的题号和分数分配，题号信息是为了后期数据分析时能够给出更加细致的信息，分数信息的存在主要使出卷人员可以自由控制分数的分配；②、③、④部分组成一个布尔型数学表达式，当表达式结果为“真”时，该知识点得分，否则失分。

进一步细化，②部分是形式化描述的关键部分，在这部分中详细描述了知识点的类别及其从考生文档中获取考生结果的所有参数，阅卷系统首先分析该部分内容，并根据参数从考生文档中获取指定的信息（即考生对该知识点的操作结果），然后与④部分给定的标准答案进行指定运算。

为了提高阅卷的准确度，提高构建阅卷信息的灵活度，我们在③部分提供了大量丰富的运算符。此外，我们还在阅卷信息的构建中部分引入了程序的概念，允许在阅卷信息的描述过程中设置变量、引用函数和宏替换实现复杂信息的描述。

我们在《浙江省中小学信息技术等级证书考试系统》的形式化描述系统和分析求解系统中设计了以下10类符号：

1）算术运算符：+，-，*，/；2）字符串运算符：+，Include，In；3）关系运算符：>，>=，<，<=，<>，=；4）布尔运算符：Not，And，Or；5）括号运算符：(，)；6）变量：可以以A～Z的任意字符为变量名；7）字符串常量：以”为分隔符的任意字符串；8）数值常量：可以是整型、实型数值；9）宏替换：～～〈变量名〉，可以替换出变量当前的值；10）函数：可以在阅卷信息中调用若干函数，如MID等。

此外，为了降低阅卷信息构造的语法限制，分析求解系统允许输入非法符号和空格，并自动进行过滤。

通过运算符的灵活组合和简单程序的编制，可以表达复杂的阅卷信息，灵活控制各种阅卷要求的表达。对提高系统的阅卷能力，系统的独立性和灵活性起到了很大的支持作用。

以下首先以Word中一个常见的加下划线的操作为例进一步说明，假设试题内容是“为文档第3段中第2次出现的‘中华人民共和国’文字加上下划线，下划线类别不限”，对此我们用以下方法进行形式化描述。

4,1={*242|3|中华人民共和国|2*}!=-1

其中4,1表示这是Word操作的第4小题，分值为1分（系作者假设），242是描述系统中添加下划线知识点的代号，3表示段落编号，“中华人民共和国”表示需要添加下划线的文字内容，2表示匹配的序号，即要求设置第2次出现的文字，根据以上信息，阅卷系统可以从文档中获取考生的操作结果，对该知识点而言，如返回-1，表示考生没有进行此操作，1表示添加了单下划线，11表示添加了波浪线……所以可以用!=-1表示不限类别的要求。

以下简单说明利用程序设计思想进行阅卷信息的描述。假设试题内容为“文档后插入任意一幅图片，并设置成无环绕效果”，我们用以下方法进行形式化描述：

```
A={*421|Picture*}  //获取文档中图片对象的数目，并保存在变量A中
4,2=(A>0)  //如果变量A的值>0，则该知识点得分
4,3={*451|～～A*}=3  //获取序号为A的图片的环绕效果，并比较是否为3
```

通过注释，读者不难理解以上代码的含义，也可以发现程序设计思想应用于阅卷信息的描述具有很好的灵活性和强大的功能。

此外必须说明，分析求解系统和形式化描述系统是不可分割、相辅相成的两个部分，形式化描述系统建立在分析求解系统的基础之上，是分析求解系统功能的外在体现，其功能的强弱完全取决于分析求解系统。如果没有强大的分析求解系统为支持，要想构建灵活、强大的形式化描述系统是不可能的。

通过以上实例和实际的应用，我们发现阅卷信息的形式化描述在阅卷系统中具有以下明显优点：

1）独立性。对阅卷信息的描述和阅卷系统的代码完全独立，一方的修改对另一方没有任何影响，尤其可以满足考试系统中对试题内容、要求、分值等频繁变化的要求。

2）灵活性。通过丰富多变的数学运算符进行灵活的组合计算，可以使阅卷结果更加准确，阅卷方式更加灵活，而且可以根据考试主办单位对阅卷的要求，对阅卷的严格程度进行一定程度上的控制，具有较强的适应性。

3）可扩充性。形式化的描述系统可以根据实际的需求不断进行扩充，而且对已有的系统和代码无需修改，可以方便后期的维护，延长一个系统的生命周期。

参考文献

1 金炳尧，马永进，骆红波. Word文档中若干图片类对象的分析与自动阅卷的实现[J]. 浙江师范大学学报(自然科学版)，2003，26(4)：365～369

2 蒋庆，孙林夫. 形式化语言对实例-模型库的规约[J]. 计算机应用，2000(8)：143～144

（上接第100页）

配深度与精度有所改进。在进行第二层次匹配时使用了一种自定义的Agent匹配能力描述语言(AMCDL)对匹配信息的中间表示形式进行了描述。

参考文献

1 IBM Web Services Architecture Team. Web Services Architecture Overview

2 IBM Corp And Microsoft Corp. UDDI Technical White Paper

3 Tidwell D. Matchmaking for Web services Interacting with a UDDI server

4 Sycara K, Klusch M, Widoff S, Lu J. Dynamic Service Matchmaking Among Agents in Open Information Environments

5 Trastour D, Bartolini C, Gonzalez - Castillo J. A Semantic Web Approach to Service Description for Matchmaking of Services

6 卞昭娟. Web Services描述与匹配机制研究. 南京大学学报，Vol. 38：151～156