Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, February 15-17, 2006 (pp145-152)

Matchmaking for Semantic Web Services with Constraints on Process Models

NATENAPA SRIHAREE[1] AND TWITTIE SENIVONGSE[2]
Department of Computer Engineering
Chulalongkorn University
Phyathai Road, Pathumwan, Bangkok 10330
THAILAND

Abstract: -. Service discovery is part of the service-oriented architectural model and supported by any of the realising technologies including Web Services. This paper presents an agent-based system for discovering semantic Web Services whose behaviour is described by OWL-S process model. Rules can be set to constrain service processes, and criteria for considering matching between a service process and the expected capability in a consumer's query are proposed. The matching algorithm follows the flexible match approach which involves ontological matching and evaluation of process constraints. Relevant services can be retrieved and ranked by consumer's preference criteria.

Key-Words: - Service discovery, Semantic Web services, OWL-S Process Model, Ontology.

1 Introduction

The current approach for Web Services discovery is towards describing various aspects of the semantics and behaviour of a Web Service by providing semantic layer on Web Service standard components such as WSDL [1] and UDDI registry [2]. Hence, it becomes Semantic Web Services discovery. To discover, an agent is proposed as a discovery engine for semantic matchmaking, and can either be used as an integrated part of the UDDI or be used as an autonomous agent which is available in a distributed network and collaborates with other agents for discovery. Semantics of Web Services can be described by ontology-based profiles which use ontology as a shared formal representation [3] to characterise services in particular domains. Search for Web Services becomes semantic search which alleviates the limitations in keyword search or search by attribute values in UDDI.

In this paper, we present an agent-based system that is autonomous and responsible for service discovery based on semantic profiles of Web Services. Here the semantic profile focuses on the Web Service process which is specified in terms of input, output, precondition, effect, and constraints upon the execution of the process. The paper extends our previous work [4] by representing the semantic profile in OWL-S process model [5] and allowing constraints in rule-based expressions. A matchmaking algorithm is proposed to determine whether the process models of any service providers match to the capability expected by a service consumer. Matchmaking involves ontological matching and evaluation of process constraints. Preference criteria for matchmaking and ordinal scale which represents the degree of matchmaking are proposed to determine how close a matched service is to a consumer's query. Service consumers can use the comparative ordinal scale assigned to each matched service as a suggestion for making a service selection.

Section 2 presents ontology-based profiles of Web Services especially OWL-S process model which can specify the behaviour of a Web Service with rule-based constraints. Section 3 considers the criteria for matchmaking of the query while preference criteria for matching and ranking by ordinal scale are in Section 4. An agent-based system for Semantic Web Services discovery is depicted in Section 5. The paper discusses related work in Section 6 and conclusion with future work can be found in Section 7.

2 Ontology-based Profiles for Web Services

2.1 Service Descriptions

An initial effort for semantic descriptions for Web Services is proposed by the DAML services coalition by which the service descriptions are described by ontology-based DAML-S profiles [6], the predecessor of OWL-S. In OWL-S, service descriptions consist of three profiles, i.e. Service Profile, Process Model, and Service Grounding.

Even so, only the Service Profile is aimed for service discovery purpose. The Service Profile is defined in terms of functional attributes such as input, output, precondition, and result of the execution of the Web Service, as well as some description that describes general attributes (which may be simple) such as service name, contacts, descriptions, and quality rating. More dynamic behaviour of the Web Service is represented in the Process Model which covers the process flow of the Web Service and the functional attributes mentioned above. These attributes are described as semantic attributes with ontological information and constraints can be put on them in terms of axioms.

Recently, Web Services Modeling Ontology (WSMO) [7] provides the conceptual framework for semantically describing Web Services. Based on WSMO, Web Services Modeling Language (WSML) [8] implements this conceptual framework in a formal language for annotating Web Services with semantic information. WSML defines semantics in terms of four elements: ontologies, goals, Web Service descriptions, and mediators. Focusing on service discovery, WSMO shares with OWL-S the vision that ontologies are essential to support automatic discovery. It is possible to choose either of them to describe Web Services. However, at the moment OWL-S can be implemented without stipulating framework and several tools also exist. We then choose OWL-S for describing service descriptions in this paper.

2.2 Web Services Metadata in OWL-S

Applying matchmaking by considering behavioural descriptions of Web Services can support complex query and is closer to service consumers' needs than considering simple attribute matching alone. As mentioned earlier, dynamic behaviour of the Web Service is represented partly in OWL-S Service profile in terms of input, output, precondition, and result of the execution of the Web Service. Fig. 1 shows a Service Profile of an *ElectronicsEshop* service which sells electronic appliances online. The profile consists of input, outputs, precondition, and results which bundle together the output and effect. These behavioural aspects have correspondences in OWL-S Process Model in Fig. 2. Although the Process Model can also describe complex process components and flows of the Web Service, we do not consider that aspect in this paper.

In Fig. 2, the *ElectronicsEShop* service represents a function that requires customer information as an input (line 3) and produces some outputs such as the product with shipping fee (line

```
<profile:Profile rdf:ID="EShopProfile">
  <profile:serviceName rdf:datatype="& XMLSchema#string">
     ElectronicsEShop</profile:serviceName>
  <profile:textDescription …
  <profile:hasInput rdf:resource="&process#CustomerInfo"/>
  <profile:hasPrecondition
       rdf:resource="&process#AcceptedCreditcard"/>
  <profile:hasOutput
       rdf:resource="&process #OrderedProductWithShippingFee"/>
  <profile:hasOutput
       rdf:resource="&process #OrderedProductWithoutShippingFee"/>
  <profile:hasResult
       rdf:resource="&process#OrderConfirmedByCompany"/>
  <profile:hasResult
       rdf:resource="&process #OrderConfirmedByPartner"/>
</profile:Profile>
```

Fig. 1 Example of Service Profile of *ElectronicsEShop* service

```
 1.  <process:AtomicProcess rdf:ID="EShopAtomicProcess">
 2.  <process:hasInput>
 3.  <process:Input  rdf:IDresource="CustomerInfo"/>
 4.  </process:hasInput>
 5.  <process:hasOutput>
 6.  <process:Output rdf:ID="OrderedProductWithShippingFee"/>
 7.  ...
 8.  </process:hasOutput>
 9.  <process:hasOutput>
10.  <process:Output rdf:ID="OrderedProductWithoutShippingFee">
11.    <process:parameterType rdf:datatype="&XMLSchema;anyURI">
12.         &eshop ;Product</process:parameterType>
13.    <process:parameterValue rdf:parseType="Literal">
14.         &eshop;Desktop</process:parameterValue>
15.    <process:parameterValue rdf:parseType="Literal">
16.          &eshop;Laptop</process:parameterValue>
17.  </process:Output>
18.  <process:hasOutput>
19.  <process:hasLocal rdf:resource="#UsersCreditcardType"/>
20.  <process:hasPrecondition>
21.    <expr:SWRL-Condition rdf:ID="AcceptedCreditcard">
22.      <expr:expressionLanguage
23.         df:resource="&Expression.owl#SWRL"/>
24.      <expr:expressionBody rdf:parseType="Literal">
25.         hasCreditcard(#UsersCreditcardType, Amex) →
26.         hasCreditcardStatus(#CreditcardIsAllowed, "xsd:True")
27.      </expr:expressionBody>
28.    </expr:SWRL-Condition>
29.  </process:hasPrecondition>

30.  <process:hasResultVar rdf:resource="#DeliveryDay"/>
31.  <process:hasResultVar rdf:resource="#UserShippingLocation"/>
32.  <process:hasResult>
33.    <process:Result rdf:ID="OrderConfirmedByCompany">
34.     <process:inCondition>
35.       <expr:SWRL-Condition rdf:ID="LocationInArea">
36.         <expr:expressionLanguage
37.           rdf:resource="&Expression.owl#SWRL"/>
38.         <expr:expressionBody rdf:parseType="Literal">
39.           hasLocation(#UserShippingLocation, Bangkok) →
40.           IsInAreaStatus(#InAreaStatus, "xsd:True")
41.         </expr:expressionBody>
42.       </expr:SWRL-Condition>
43.     <process:inCondition>
44.     <process:hasEffect>
45.       <expr:SWRL-Condition rdf:ID="ProductDeliveredByCompany">
46.         <expr:expressionBody rdf:parseType="Literal">
47.           → deliveredProduct(#EShopAtomicProcess, #DeliveryDay)
48.             swrlb:lessThan(#DeliveryDay, 3)
49.         </expr:expressionBody>
50.       </expr:SWRL-Condition>
51.     </process:hasEffect>
52.     <process:withOutput>
53.       <process:OutputBinding>
54.         <process:toParam
55.         rdf:resource="#OrderedProductWithoutShippingFee"/>
56.       </process:OutputBinding>
57.  </process:withOutput>
58.  </process:Result>
59.  </process:hasResult>…
```

Fig. 2 Fragment of Process Model of *ElectronicsEShop* service

6) and the product without shipping fee (line 10). The service puts constraints (line 11-16) on the product type such that if the ordered product is a desktop or laptop, the shop requires no extra shipping fee. (Note that, this requires an external ontology to define the kinds of products that are available at the shop). The precondition of the service (line 21) says that the user must hold a credit card that is accepted by the shop in order to make a purchase. This precondition is defined by a rule expression (line 24-27) which reads if the user's credit card is an Amex card, then the card is allowed. The result of the execution of the service corresponds to the output and effect altogether. A result of the *ElectronicsEshop* is that the order will be confirmed by the company (line 33). However, this is the case only when the specified incondition (line 34-43) is true; otherwise the order will be confirmed by a partner. The incondition specifies that if the user's location is Bangkok, then it is in the area of service by the company. When this incondition is true, the output will be the ordered product without shipping fee (line 55), and the effect will be that the product delivery is by the company itself (rather than by a partner shipping company) (line 45). Note that the effect is further constrained that the delivery by the company is guaranteed to be within 3 days of purchase (line 47-48).

These behavioural aspects and constraints can be seen as representing service semantics according to some business rules or policies [9][10]. In the Process Model, conditions and constraints can be described in terms of logical expressions which, for instance, may be KIF, PDDL, RDF, OWL, or SWRL expressions. In this paper we choose to represent conditions and constraints in SWRL [11] with OWL-based syntax since it is becoming a standard rule markup language for Semantic Web technology.

These expressions allow us to specify different behaviour of a Web Service when certain conditions are met, and therefore matchmaking can be performed by determining such constrained behaviour. To do so, it requires these SWRL expressions to be extracted and translated into a rule-based language and then a relevant rule engine is used to evaluate the expressions.

2.3 Behavioural Constraints in Process Model

From an example in Fig. 2, we can summarise behavioural constraints in OWL-S Process Model that are of interest to this paper as follows.

(i) Precondition and incondition constraint expressions. These refer to the constraint expressions that are defined in *process:hasPrecondition* and *process:inCondition* in the Process Model. The pattern of these constraint expressions will be:

Expressions → *True*

The constraint will be bound to a SWRL rule expression which has the body part (i.e. antecedent part) which may consist of some atoms and the head part (i.e. consequent part) which returns true as a result.

For the incondition, it triggers the associated output and effect to result when it is evaluated to true. The output and effect can then be considered as conditional output and conditional effect respectively.

(ii) Effect constraint expression. This refers to the constraint expression that is defined in *process:hasEffect* in the Process Model. The pattern of the constraint expression will be either of the following:

→ *Expressions*

Expressions → *Expressions*

The constraint will be bound to a SWRL rule expression which require some atoms either in the head part only or in both the body and head parts.

(iii) Output constraint expression. The constraint on the output in the Process Model is defined as a set of ontological values that are associated with the output produced. In Fig. 2, line 11-16 defines a set of ontological values of the output product (i.e. Desktop, Laptop) that, if purchased, will be shipped free of charge (line 10). However, in this example, this output is also constrained by the incondition about the location of the user (line 35 and 55), therefore the user has to buy either a desktop or laptop and also has to be in Bangkok for the free delivery to take place.

3 Matching Criteria

In our approach, behavioural constraint matchmaking requires some input data from the query and the algorithm will consider a service to match the query if it can accept the input from the query, process its function, and return a result that satisfies the user's need. Matchmaking here takes the input data from the query to evaluate behavioural constraints. Such evaluation will be done by ontology reasoning, mathematical computation, and also rule reasoning using a rule engine. Using ontology as a knowledge representation, a query can be complex with some semantic constraints. For example, a service

consumer may want to find a Web Service that accepts American Express credit card for buying a laptop, and may need the laptop to be delivered to his/her place in Bangkok within 5 days of purchase.

In this section, we define matching criteria for matchmaking between the Process Model of a Web Service and a query as follows.

3.1 Matching Ontological Concepts

Matching by subsumption [12] and equivalence is the basis for matching ontological concepts in the query and the provider's Process Model. This approach has been adopted in [13][14]. In [15][16], a weaker match of ontological concepts, called partial match, is defined for two ontological concepts that have a shared node in an IS-A taxonomy and there is no subsumption relationship between them. The degree of matching is determined between two concepts as described below.

Let C_Q be the concept specified in the query and C_P be the concept in the Process Model:

(i) If $C_Q \equiv C_P$ *then* C_P *is an exact match for* C_Q, where

$\equiv$ means is equivalent to. For example, in Fig. 3 which is an ontology of the electronics shop domain, the provider who sells Desktop will be an exact match for the query that also requests for Desktop.

(ii) If $C_P \sqsubseteq C_Q$ *then* C_P *is a specialised match for* C_Q, where $\sqsubseteq$ means is subsumed by (i.e. C_P is more specific than C_Q). In this case, the query may specify a generic concept while the Process Model defines a specific concept. For example from Fig. 3, the provider who sells either Laptop or Desktop will be a specialised match for the query that requests for PC.

(iii) If $C_Q \sqsubseteq C_P$ *then* C_P *is a generalised match for* C_Q. This means the concept in the query is more specific than, and is subsumed by, the one in the Process Model. For example from Fig. 3, the provider who sells PC will be a generalised match for the query that requests for a Desktop.

(iv) If $(C_Q \not\sqsubseteq C_P) \wedge (C_P \not\sqsubseteq C_Q) \wedge (C_Q \sqsubseteq C_C) \wedge (C_P \sqsubseteq C_C)$ *then* C_P *is a partial match for* C_Q, where $\not\sqsubseteq$ means is not subsumed by and C_C is a node in the same IS-A taxonomy. This means it is acceptable for the concept in the Process Model to be a match for the concept in the query provided that the two concepts have common characteristics through a common parent concept. For example from Fig. 3, the provider who sells Laptop will be a partial match for the query that requests for Desktop.

(v) If none of the above relationships exist then C_P *is a fail match for* C_Q.

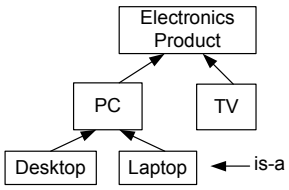

Fig. 3 Fragment of Ontology of the Domain

3.2 Matching Behavioural Constraints

According to Section 2.3, the details for considering matching for behavioural constraints are as follows.

3.2.1. Matching precondition and incondition constraints

The service will match to the query if, by applying a set of input values from the query into a SWRL rule expression, the rule evaluation hits and returns true as a result.

In Fig. 2, if an input from the query specifies that the user's credit card is an American Express, the rule for the precondition (line 25-26) will fire or hit, and it will return true as a result of the evaluation. In the same manner, if an input from the query specifies the location of the user as Bangkok, the rule for the incondition (line 39-40) will fire and return true.

The expression in the head atom of the rule may be a numerical constraint or constraint on some data values, and these may require ontological reasoning, numerical computation, and also rule reasoning. We consider a match for precondition and incondition constraints only when such evaluation returns true; this means the Process Model satisfies the query for such constraint.

3.2.2. Matching effect constraints

An effect will state what happen as a consequence to the execution of the service (such as the product will be delivered by the company) and may also have additional semantics such the guarantee of delivery time (line 47-48 in Fig. 2). For the query of a user in Bangkok who requires a delivery of product within 5 days, the Process Model in Fig. 2 will match since the incondition fires for Bangkok as described above, and the 3-day delivery guarantee is within the range of the query.

Matching two numerical constraints compares the intervals of the possible values that are defined in the constraints. The degree of matching for

numerical constraints can be determined as described below.

Let N_Q be a nonempty set of numerical constraint values of the expression in the query (E_Q), and N_P be a nonempty set of numerical constraint values of the expression in the Process Model (E_P):

(i) If $N_P \subseteq N_Q$ then E_P is an exact match for E_Q.

(ii) If $N_Q \subseteq N_P$ then E_P is a plug-in match for E_Q.

(iii) If $(N_P \cap N_Q \neq \phi) \wedge (N_P \subsetneq N_Q) \wedge (N_Q \subsetneq N_P)$ then E_P is a weak match for E_Q.

(iv) If $N_P \cap N_Q = \phi$ then E_P is a fail match for E_Q

It is also possible that an effect constraint may involve both numerical constraint and ontological constraint, and hence the evaluation requires both ontological reasoning and rule reasoning. For example, for the following effect constraint:

hasProduct(#EShopAtomicProcess, TV) → deliveredProduct(#EShopAtomicProcess, #DeliveryDay)
swrlb:lessThan(#DeliveryDay, 3).

This constraint says that the delivery time is within 3 days for any purchase of a TV. Suppose that, in the query, the user orders an LCDTV which is a subclass of TV, the matching process will perform an ontological reasoning on the product first (refer to Section 3.1). When it is determined as a match by subsumption, the rule on delivery day will fire in the rule reasoning step.

3.2.3. Matching output constraints

An output may be bound to a set of ontological values which are associated with the semantics of the output. The product output which requires no shipping fee as in line 10 of Fig. 2, for instance, is associated with the set of the product of type Desktop and Laptop only. We consider matching two sets of ontological values as ontological matching of each of the values in the two sets [17]. The sets must concern the same ontological type (e.g. both are the sets of product).

Let D_Q be a nonempty set of ontological values in the query(Q), and D_P be a nonempty set of ontological values in the Process Model(P):

$SetOfOntoValsMatch(Q,P) = true \Leftrightarrow$

$\forall i, \exists j: (i \in D_Q) \wedge (j \in D_P) \wedge (i \otimes j)$

where $\otimes$ means having a kind of the ontological match as in Section 3.1 (i.e. *exact*, *specialised*, *generalised*, *partial*).

3.3 Matching Process Model

To check if the Process Model of a Web Service satisfies the query, we consider matching on all functional properties specified in the query against the corresponding properties in the Process Model, i.e. input, output, precondition, effect, and associated constraints. The Process Model will match the query if it satisfies the following:

(i) input, unconditional output (i.e. output with no constraint), and unconditional effect (i.e. effect with no constraint) satisfy ontological match in Section 3.1, and

(ii) precondition, conditional output (i.e. output with constraints) and conditional effect (i.e. effect with constraints) satisfy matching criteria in Section 3.2.

In other words, let $\mathbb{R}_Q$ and $\mathbb{R}_P$ be the sets of functional properties of the query and Process model respectively, each comprising inputs, outputs, preconditions, and effects:

$ProcessModelMatch(\mathbb{R}_Q, \mathbb{R}_P)$ = true

$\Leftrightarrow (\mathbb{R}_Q \subseteq \mathbb{R}_P) \wedge (\forall i, \exists j :$

$(i \in \mathbb{R}_Q) \wedge (j \in \mathbb{R}_P) \wedge (i \Theta j))$

where Θ means having a kind of match as in Sections 3.1 and 3.2.

3.4 Ordinal Scale of Matching

Table 1 lists the types of matching and ordinal scale which represents match scores, from the strongest to the weakest match. This ordinal scale will be used further for ranking purpose.

Table 1 Types of matching and match scores

Match Type	Match score
Ontological match (c.f. Section 3.1)	exact = 4, specialised = 3, generalised = 2, partial = 1, fail = 0 Note: if reverse subsumption preference is set (see next section), generalised=3 and specialised = 2.
Numerical constraint match (c.f. Section 3.2.2)	exact = 3, plug-in = 2, weak = 1, fail = 0
Set of ontological values match (c.f. Section 3.2.3)	satisfied = 1, fail = 0

4 Preference Criteria for Matching and Ranking

Service consumers can specify any of the following preference criteria for matching and ranking:

(i) *Match Preference*. This criterion can be set to define the weakest acceptable match type. For example, the service consumer may query for the product PC and set a match preference to

specialised. So the services that publish the product with the same concept, i.e. PC (by *exact* match), and more specific concepts, i.e. Desktop and Laptop (by *specialised* match), will match to the query.

(ii) *Functional Property Priority Preference*. This criterion can be set to define a significance that one matching functional property has over the others. This preference setting can help overcome a problem of conflicting ordinal scale of match types among several functional properties. Suppose two candidate services have ordinal scale match for input and output as (*specialised*, *generalised*) and (*generalised, specialised*) respectively. This can be problematic for ranking. For example, the query may specify the output to have significance over input. By specifying that the output should have significance over the input, the latter would be ranked higher than the former.

(iii) *Reverse Subsumption Preference*. This criterion can be set to define that, in some cases, a generalised match is preferable to (i.e. stronger than) a specialised match in ontological subsumption. For example, a candidate service may expect an input that is more generalised than the query's input while the other service expects a more specialised input. In this case, the former may be a better match.

5 Agent-based System for Semantic Discovery

We design and implement an agent-based system for Semantic Web Services discovery as depicted in Fig. 4. A service provider will define the Process Model of the Web Service as well as any necessary local ontology (1), using an ontology editor (e.g. Protégé). The definition may be based on shared ontology of the domain, which is defined by service domain experts. The service provider maintains the Process Models and the local ontology, but also registers the Process Model with the agent via the registering proxy (2). The registering proxy will store the URL of the Process Model in the ontology repository (3). The Process Model will be processed to extract knowledge and to reason further by the parser module in discovery time (4). The parser module is integrated with an inference engine (e.g. Jena [18]) and a SWRL2Jess parser. The agent may preprocess to extract knowledge and to reason from the shared ontologies prior to the matchmaking; the results are stored in a knowledge base. The agent can provide the service consumers with a GUI template that corresponds to the ontologies of the domain so that the consumers can specify query onto the Process Model more easily (5). Internally, the query will be in the form of RDF-based expressions and will pass through the querying proxy to the matching module. While performing matching, the matching engine may interact with the constraint evaluation module which is integrated with a rule engine (e.g. Jess engine [19]) (6). Constraint expressions defined by SWRL will be parsed by SWRL2Jess parser into a rule script (e.g. Jess script) which will be used for reasoning by the rule engine. The matching engine also performs ranking. Matched services will be ranked and reported in an XML document which will be returned to the consumer (7).

This paper assumes that service providers use a shared ontology for a service domain. To enable large scale discovery, the agent may collaborate with other agents for knowledge reasoning and constraint reasoning.

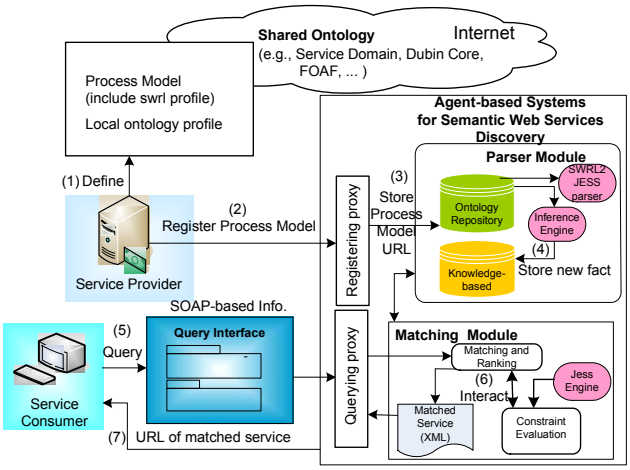

Fig. 4 Agent-based System for Semantic Web Services Discovery

6 Related Work

A number of the related work present semantic-based frameworks based on description logics formalisation and description logic reasoning. The approaches by [13] and [14] propose matchmaking in the e-commerce scenario in a multi-agent system when an advertisement represents the product description. They consider matching by subsumption relationship between the query and each advertisement in the repository. The work in [20] propose the agent-based framework in which the service capability is described by a description language LARKS. The matchmaker consists of a number of filters, each of which performs partial matching on the service descriptions. The work in [21] considers matchmaking by input and output match. Our work is close to the approximate match

based on the similarity through subsumption which is proposed by [21]. However, our matching scheme is more refined since it also considers matching of constraints. Also, ranking is not addressed by any work mentioned above.

7 Conclusion and Future Work

OWL-S Process Model describes dynamic aspects of the Web Service capability and is therefore a candidate as a metadata for use in service discovery. This paper focuses on specifying rule-based expressions to refine the behaviour of the Web Service and considers such rules in the matchmaking process. The proposed matching scheme gives an intuitive ordinal scale which is a basis for ranking query results.

As constraint expressions in the Process Model can be described by many different languages, matching criteria can be extended further according to the expressiveness of the language. Moreover, we expect to consider Process Model more extensively in the matchmaking, i.e. consider matching of process flows. Also, although an evaluation of the practicality of our matchmaking and ranking techniques has been reported in [22], the matching is performed on the functional properties, one by one, and this can be time-consuming as reported in [23]. Hence, the performance of the system will be evaluated in the future work.

References:

[1] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, Web Service Description Language (WSDL) 1.1, 2001. Available: http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[2] uddi.org, UDDI: Universal Description, Discovery, and Integration of Web Services, 2002. Available: http://www.uddi.org

[3] T. Gruber, A Translation Approach to Portable Ontologies, *Knowledge Acquisition*, Vol. 5, No. 2. 1993, pp.199-220.

[4] N. Sriharee, T. Senivongse, Discovering Web Services Using Behavioural Constraint and Ontology, *Proceedings of the 4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2003)*, Paris, France, November 2003, 248-259.

[5] D. Martin et al., OWL-S: Semantic Markup for Web Services, 2004. Available: http://www.w3.org/Submission/OWL-S/.

[6] The DAML Services Coalition, DAML-S: Web Service Description for the Semantic Web, *Proceedings of the 1st International Semantic Web Conference (ISWC 2002)*, Sardinia, Italy. 2002.

[7] WSMO, Web Services Modeling Ontology, 2004. Available: http://www.wsmo.org

[8] D.J. Bruijn, H. Lausen, A. Polleres, D. Fensel, The Web Service Modeling Language WSML: An Overview, *DERI Technical Report 2005-06-16*, June. 2005.

[9] N. Sriharee, T. Senivongse, K. Verma, A. Sheth, On Using WS-Policy, Ontology and Rule Reasoning to Discover Web Services, *Proceedings of the IFIP International Conference on Intelligence in Communication Systems*, 23-26 November 2004, pp. 246-255.

[10] N. Sriharee, T. Senivongse, Enriching UDDI Information Model with an Integrated Service Profile, *Proceedings of the 5th IFIP International Conference on Distributed Applications and Interoperable Systems*, Athens, Greece, 15-17 June 2005.

[11] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, SWRL: A Semantic Web Rule Language combining OWL and RuleML, 2003. Available: http://www.daml.org/2003/11/swrl/

[12] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P.F. Patel-Schneider, *The Description Logic Handbook : Theory, Implementation, and Applications*, Cambridge University Press, 2003.

[13] D. Trastour, C. Bartolini, J. Gonzalez-Castillo, A Semantic Web Approach to Service Description for Matchmaking of Services, *Proceedings of the International Semantic Web Working Symposium,* 2001.

[14] L. Li, I. Horrocks, A Software Framework for Matchmaking Based on Semantic Web Technology, *Proceedings of the 12th International World Wide Web Conference*, 2003.

[15] P. Resnik, Using Information Content to Evaluate Semantic Similarity in a Taxonomy, *Proceedings of International Joint Conference on Artificial Intelligence*, November 1995.

[16] Andreasen, T., Bulskov, H., Knappe, R., On Ontology-Based Querying, *Proceedings of Workshop on Ontologies and Distributed Systems, 18th International Joint Conference on Artificial Intelligence*, August 2003.

[17] I. Constantinescu, B. Faltings, Efficient Matchmaking and Directory Services, *IEEE/WIC International Conference on Web Intelligence*, 13-17 October 2003.

[18] Jena: A Semantic Web Framework for Java. Available: http://jena.sourceforge.net/index.html.

[19] Jess the Rule Engine for the Java™ Platform, Version 6.1, April 2003. Available: http://herzberg.ca.sandia.gov/jess/

[20] K. Sycara, S. Widoff, M. Klusch, J. Lu., LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace, *Autonomous Agents and Multi-Agent Systems*, Volume 5, Issue 2, June 2002, pp. 173 – 203.

[21] M. Paolucci et al., Semantic Matching of Web Services Capabilities, *Proceedings of the 1st International Semantic Web Conference*, 2002.

[22] N. Sriharee and T. Senivongse, Towards Semantic Discovery of Web Services Using an Integrated Service Profile, *Technical report*, Dept. of Computer Engineering, Chulalongkorn University, 2005.

[23] N. Srinivasan, M. Paolucci, K. Sycara, An Efficient Algorithm for OWL-S based Semantic Search in UDDI, *Proceedings of 1st International Workshop on Semantic Web Services and Web Process Composition*, San Diego, CA, USA, July 6, 2004.