On Using WS-Policy, Ontology, and Rule Reasoning to Discover Web Services

Natenapa Sriharee[1], Twittie Senivongse[2], Kunal Verma[3], Amit Sheth[4]

[{1,2}] Department of Computer Engineering, Chulalongkorn University
Phyathai Road, Pathumwan, Bangkok 10330 Thailand
natenapa.s@student.chula.ac.th, twittie.s@chula.ac.th
[{3,4}] Large Scale Distributed Information System (LSDIS) Laboratory
Department of Computer Science, University of Georgia, Athens, GA 30602
(verma, amit}@cs.uga.edu

Abstract. This paper proposes an approach to behaviour-based discovery of Web Services by which business rules that govern service behaviour are described as a policy. The policy is represented in the form of ontological information and is based on actions relating to the service and conditions for performing them. The standard WS-Policy is used to associate such a policy to the Web Service. With a framework that extends standard discovery by UDDI, service consumers can query for Web Services by specifying business constraints. The policy of the Web Service will be evaluated against the consumer's query by using OWL ontology querying engine and a rule-based reasoning module. By considering business rules in addition to the conventional attribute-based search by UDDI, the approach will enable more satisfactory discovery results that better fit service consumers' requirements.

1 Introduction

Current standard UDDI registry for Web Services [1] defines fundamental attributes that characterise businesses and services they provide. Search with UDDI is hence restricted to matching of the attribute values in service consumers' queries to those published by service providers (e.g. search is by business name or business service category). This may give a not-so-accurate search result since search constraints cannot filter irrelevant information well and do not reflect semantics and behaviour of the businesses and their services. This paper proposes an approach to discover Web Services by using a policy that enforces service behaviour. The policy will describe rules that define choices of the behaviour of a Web Service. A retailer may, for instance, specify a policy on delivery time for the appliances, which are sold to the customers, based on the appliance types and areas of destination. Correspondingly, a customer may want to buy an appliance from a retailer that can deliver goods within a

[1] On visiting LSDIS Lab, The University of Georgia, Dec 2003 – May 2004.

specified time. Therefore a mechanism to evaluate the retailer's policy against the customer's requirement will be provided.

The conditions under which the retailer's Web Service above provides its service can be defined using a policy in WS-Policy framework [2]. A policy consists of a collection of one or more policy assertions that can be bound to Web Services entities (e.g. operation, message, part) in order to enforce assertion rules on them. The framework allows policies to be created for various purposes (e.g. the currently available WS-Security policy standard [3]) and places no restriction on the language used to represent policy expressions. This paper introduces a business rules policy that is based on WS-Policy framework and concerns rules on business functions. The policy assertions will be modelled by the actions relating to the service and the conditions for performing them. An ontology language (OWL [4] in this case) is used to express the policy. To check the policy of a Web Service against a service consumer's request, an OWL ontology querying engine will be queried and a rule-based reasoning module will evaluate the policy assertions to determine actual service behaviour in response to the request.

The rest of this paper starts with Section 2 that discusses related work. Section 3 outlines the motivation for using policy for service discovery through a supply chain problem domain that will be discussed throughout the paper. Section 4 explains the ontology that conceptualises the business rules policy together with an example of an ontology-based policy. The policy is deployed onto UDDI in Section 5 and used in a discovery framework in Section 6. A discussion about our approach and a conclusion are in Section 7.

2 Related Work

Most of research work that uses policy for services focuses on administrative policies such as security and resource control policies. In [5], policy-based agent management is proposed for Grid services where policies for authorisation, encryption, and resource control are represented using DAML ontology language. In [6], authorisation and privacy for their Web Services are controlled by a language named Rei [7] which is based on RDF ontology language, and policy expressions in Rei will eventually be transformed to Prolog expression for reasoning. Their use of policy allows services to be described as requiring encrypted input or servicing with data privacy, and therefore service consumers can query for services that exhibit these policy-aware aspects. The work in [8] is closer to our work in that the policy will be defined using OWL and RuleML. Policy compatibility is considered in service composition but they do not consider deploying their mechanism onto standard UDDI or WS-policy framework. Our approach expresses rule-based policy from business functions aspect, and the rules will be used in service discovery to constrain the queries and be evaluated at query time. The policy will be deployed onto the standard UDDI and WS-Policy framework. To our knowledge, no other work has proposed a policy based on this framework to support service discovery.

3 Business Rules Policy for Supply Chain Domain

To demonstrate the need for involving business rules policies with the service selection process, a scenario of the parts-suppliers problem in a supply chain is as follows.

1. A retailer issues a purchase order for electronics parts to a distributor, specifying parts details including the number to order and the time duration for the parts to be delivered. For example, the order may specify 100 pieces of a particular part to be delivered within 5 days of purchase.
2. The distributor selects candidate suppliers from a UDDI service registry by considering some characteristics, e.g. the suppliers with whom the distributor has a contract or those who can supply the specified parts. In [9], the distributor has the parts from these candidate suppliers analysed to check if they are compatible.
3. The candidate suppliers may additionally publish a policy that relates to their business function. A candidate supplier may publish a policy as in Fig. 1 which specifies the action to be taken (i.e. either deliver parts or check inventory) based on the number of parts to be delivered. The service discovery framework hence should allow the distributor to check whether any candidate supplier has a delivery policy that satisfies the retailer's delivery time constraint based on the number of ordered parts given by the retailer. In this example, this candidate supplier will match by its first rule since 100 pieces of the part can be delivered within 3 days. The paper will focus in this step.
4. Finally, matched suppliers will be returned to the retailer.

```
Perform DeliverParts(days ≤  3)
       IF (numberOfParts  ≤ 100)
Perform DeliverParts(days ≤  10)
       IF (100  <  numberOfParts  ≤ 500)
Perform ContactInventoryBeforeConfirm
       IF (numberOfParts  > 500)
```

Fig. 1. Business rules policy for parts delivery

The business rules policy shown in Fig. 1 will be associated with a particular Web Service. Although many times business rules are tied to the level of business service, (i.e. group of related Web Services), rather than to the level of individual Web Services, it is out of scope of this paper. The focus here is on incorporating business rules policy for individual Web Services.

4 Expressing Business Rules Policy by Ontology

We define common concepts as well as relations between those concepts to represent our business rules policy. A policy is specified as a collection of rules where each rule is an IF_conditions_THEN_action statement. If conditions are evaluated to True, the action is set or performed. The upper ontology for policy consists of the follow-

ing classes of concepts (Fig. 2) which can be referred back to the example of the supply chain problem:

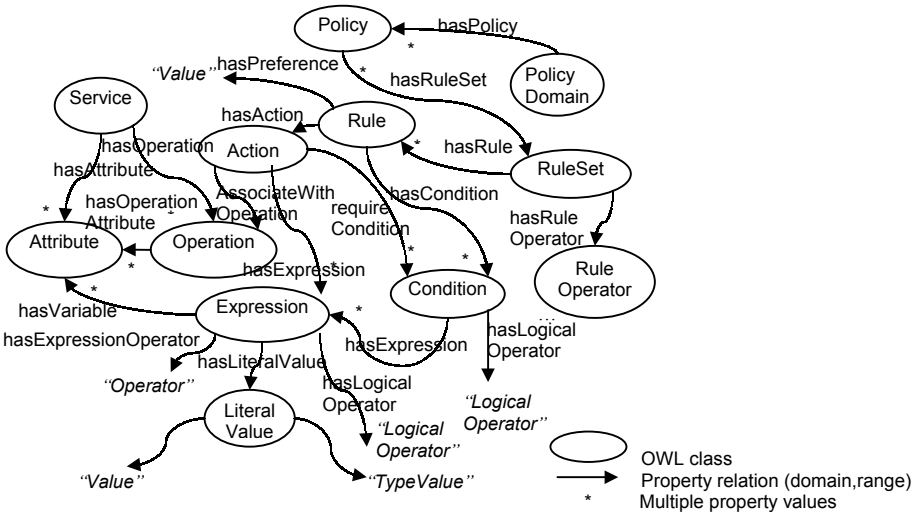

Fig. 2. Upper ontology for business rules policy.

- Policy Domain is a service domain on which the policy is applied, e.g. the supply chain domain. A Policy Domain may have multiple Policies, e.g. the supply chain domain may have one policy about parts delivery and another one about payment.
- Policy defines constraints on a particular task within the service domain. A Policy may have multiple Rule Sets, e.g. the parts delivery policy may have one rule set about delivery time and another one about delivery means.
- Rule Set defines a set of Rules that relate to a particular aspect.
- Rule Operator is defined for a Rule Set to determine the level of rule enforcement within the Rule Set. Rule Operator can either be "All" to indicate that all of that all Rules within the Rule Set will apply, "ExactlyOne" to indicate that only one Rule will apply at a time, or "OneOrMore" to indicate that at least one Rule will apply. For the parts delivery policy in Fig. 1, "ExactlyOne" will apply.
- Rule is a conditions-and-action statement that says what action will be set when particular conditions are satisfied. In Fig. 1, three Rules are defined. A Rule may be tagged with a preference that indicates the degree of satisfaction when the Rule is matched. The preference can help in ranking the policies with matched rules.
- Action is an abstract term that may be associated with an Operation of a Service and Expressions that can be bound to logical operators. For example, the delivery parts action can be associated with the purchase parts operation of the parts supplier Web Service, and in the first Rule of Fig. 1., it is associated with the Expression `(days ≤ 3)`.

- Condition is an abstract term that can be associated with Expressions that can be bound to logical operators.
- Expression specifies an expression that consists of Attribute, Expression Operator, and Literal Value. In the first Rule in Fig. 1, (`numberOfParts ≤ 100`) is an Expression for the condition of the rule where `numberOfParts` is Attribute, ≤ is Expression Operator, and `100` is Literal Value. Another Expression `(days ≤ 3)` in the first Rule is associated with the delivery parts action. An Expression Operator can either be relational operator or functional operator like min, max, and set (e.g. as in `(numberOfParts min 100)` for delivery parts action to perform, `(numberOfParts max 1000)` for credit purchase action to perform, or (cardholder(set{visa, amex, diners}) for credit purchase action to perform. A Literal Value consists of a value and the type of the value.

It is possible to use this upper ontology on its own, or as an enhancement to OWL-S service model [10] to allow rule-based policy evaluation for service composition. From the upper ontology, a domain expert may derive a domain-specific policy ontology that describes the vocabularies for policies, actions, conditions, attributes, and operations used commonly within the domain. Service providers in the domain can create their own policies by deriving from the domain-specific policy ontology, or simply by using a domain-specific policy template, filling in their own policy detail, and having an OWL-based policy specification automatically generated. Below is part of the policy specification called `DeliverPartsToDistributorSupplierA` of a supplier named `supplierA`. The information, represented in OWL, corresponds to the first rule in Fig. 1. Note that the upper ontology is referred to by the namespace `po:` and the domain-specific policy ontology by the namespace `sp:`.

```
xmlns:sp= "http://supplychain.com/policy.owl#"
xmlns:po= "http://samplepolicy.com/policy.owl#"
xmlns= "http://supplierA.com/policy.owl#

<!-----Policy -→
<sp:DeliverPartsToDistributorPolicy
             rdf:ID="DeliverPartsToDistributorSupplierA">
   <po:hasRuleSet rdf:ID="RuleSet1">
            <po:hasRuleOperator rdf:resource="po:ExactlyOne"/>
        <po:hasRule>
         <po:Rule rdf:ID="Rule1">
           <hasAction rdf:resource="#DeliverPartsRule1"/>
           <hasCondition rdf:resource="#CheckQuantity1"/>
         </po:Rule>
        </po:hasRule>
        …
   </po:hasRuleSet>
</sp:DeliverPartsToDistributorPolicy>

<sp:CheckQuantity rdf:ID="CheckQuantity1">
  <po:hasExpression>
    <po:Expression rdf:ID="ExpressionCondition1">
      <po:hasVariable>
        <po:Attribute rdf:resource="sp:NumberOfParts"/>
      </po:hasVariable>
      <po:hasExpressionOperator rdf:resource="po:isLessThanOrEqual"/>
      <po:hasLiteralValue>
```

```
          <po:LiteralValue rdf:ID="LiteralValue2">
            <po:hasValue
               rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
                                                    100</hasValue>
            <po:hasType rdf:resource="po:Integer"/>
          </po:LiteralValue>
        </po:hasLiteralValue>
      </po:Expression>
    </po:hasExpression>
  </sp:CheckQuantity>

   <sp:DeliverParts rdf:ID="DeliverPartsRule1">
     <po:associateWithOperation>
       <sp:PurchaseParts rdf:ID="PurchasePartsSupplierA"/>
     </po:associateWithOperation>
     <po:hasExpression rdf:ID="ExpressionAction1">
     <po:hasVariable>
       <po:Attribute rdf:ID="sp:DeliveryDay"/>
     </po:hasVariable>
     <po:hasExpressionOperator rdf:resource="po:isLessThanOrEqual"/>
     <po:hasLiteralValue rdf:ID="LiteralValue1">
           <po:hasValue
               rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                                                    >3</po:hasValue>
           <po:hasType rdf:resource="po:Integer"/>
     </po:hasLiteralValue>
    </po:hasExpression>
    <po:requireCondition rdf:resource="#CheckQuantity1"/>
  </sp:DeliverParts>
```

5 Deploying Business Rules Policy

A service provider can deploy an OWL-based business rules policy by attaching the policy to WSDL description of its Web Service [11]. Similarly to other kinds of policies that govern a Web Service, the business rules policy may be maintained separately or put into the all-policy specification together with other kinds of policies for easy management. Fig. 3 shows part of the all-policy specification file of supplierA's Web Service, say policy.xml, in which the new business rules policy `DeliverPartsToDistributorPolicy` is added. This policy refers to the OWL-based `DeliverPartsToDistributorSupplierA` policy in Section 4.

```
  base = "http://supplierA.com/policy.xml"
  wspsp = "http://supplychainschema.com/policyspec"
  …
  <wsp:Policy Name="PurchasingProcess">
      <wsp:All>
        <wspsp:DeliverPartsToDistributorPolicy>
  http://supplierA.com/policy.owl#DeliverPartsToDistributorSupplierA
        </wspsp:DeliverPartsToDistributorPolicy>

  <!---Other policy for PurchasingProcess, e.g. WS-Security -→
  …
      </wsp:All>
</wsp:Policy>
```

Fig. 3. Adding business rules policy to the all-policy specification of a Web Service.

The policy.xml file will be associated with the Web Service by attaching it to WSDL [12]. In Fig. 4, the policy file is attached to the `PurchasePartsService` with `PurchasePartsPortType`, provided that the purchasing parts operation is defined in the port type. With this attachment, the policies in the file will be enforced on the Web Service instance.

```
base = "http://supplierA.com/purchaseparts.wsdl"
…
<wsp:PolicyAttachment>
  <wsp:AppliesTo>
    <wsp:EndpointReference>
      <wsp:ServiceName
            Name="PurchasePartsService"/>
         <wsp:PortType Name="PurchasePartsPortType/>
           <wsp:Address URI="http://supplierA.com/policy.xml" />
     </wsp:EndpointReference>
   </wsp:AppliesTo>
  <wsp:PolicyReference Ref="http://supplierA.com/policy.xml"/>
</wsp:PolicyAttachment>
```

Fig. 4. Attaching all-policy specification to WSDL of a Web Service.

When the service provider publishes its Web Service with UDDI and specifies a tModel that references a corresponding WSDL file, the business rules policy is effectively published with UDDI via its attachment to that WSDL. In this way, candidate services can be discovered first by a typical query to standard UDDI (i.e. attribute values matching) or by other behaviour-based query such as the one in [9], [13]. The WSDL files of these candidate services will be retrieved and, as a result, the attached policy files can be referred to. The supporting discovery framework in Section 6 then can accommodate query based on business rules and can evaluate the policies by rule-based reasoning.

6 Policy-Based Discovery Framework

The discovery framework that supports policy-based service selection is depicted in Fig. 5. The main components that extend from standard UDDI are the Policy-Aware Discovery Proxy (or PADP) (a), Policy Server (b), Ontology Querying Module (c), and Rule Reasoning Module (d). PADP is a Web Service that interfaces with service consumers and is responsible for dispatching queries for evaluation and accumulating as well as ranking query results. Policy Server provides policy-defining templates that are associated with domain-specific ontologies. It generates OWL-based policies from the templates and also the corresponding rule-based specifications. It is a Web Service that works with Ontology Querying Module to evaluate semantic aspects of the policies and works with Rule Reasoning Module for rule-based evaluation.

Let us revisit to the parts-suppliers problem. The distributor is looking for suppliers that can deliver 100 pieces of the parts within 5 days. Part of the request that is related to the policy and submitted via a simple XML template (Fig. 6) is invoked on PADP (1).

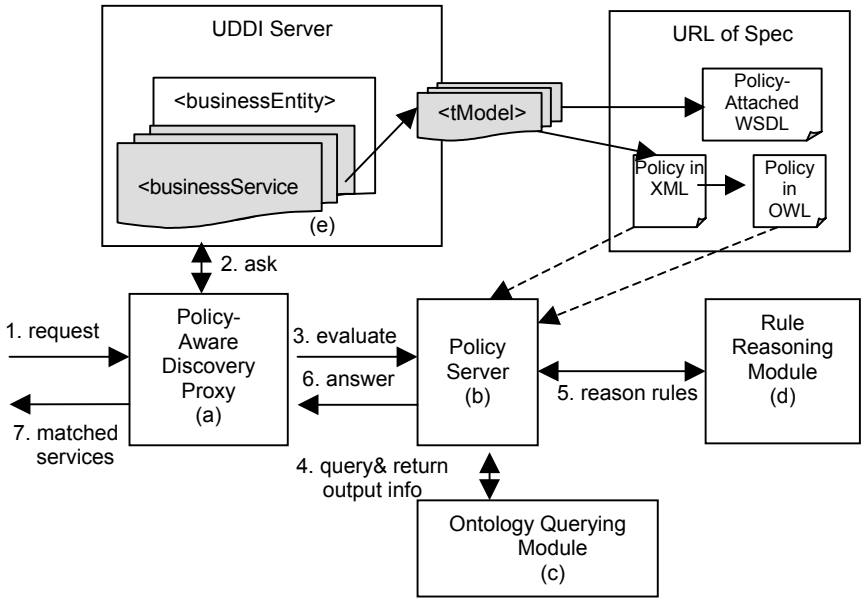

Fig. 5. Policy-based discovery framework.

```
<findPolicy>
      <domain>SupplyChain</domain>
      <policy>DeliverPartsToDistributorPolicy</Policy>
      <action name="DeliverParts">
            <expression attribute="DeliveryDay">
                <Value>5</Value>
                <expressionOperator>isLessThanOrEqual
                                          </expressionOperator>
            </expression>
      </action>
      <condition name="CheckPartsQuantity">
            <expression attribute="NumberOfParts">
                 <Value>100</Value>
                 <expressionOperator>isEqual</expressionOperator>
            </expresion>
      </condition>
</findPolicy>
```

Fig. 6. Policy-based request in XML template.

Assume that some candidate services that satisfy some characteristics of the request are already obtained by the method discussed at the end of Section 5. PADP will consult UDDI server for the tModels that refer to WSDL files of all candidate services (2). From WSDL, PADP can get the reference to the all-policy file for each Web Service and sends the reference to the file to the Policy Server (3). On receiving the reference, the Policy Server then loads the corresponding all-policy file, extracts the reference to the OWL-based file, and at the end retrieve the OWL-based business rules policy. The Policy Server will interact with the Ontology Querying

Module to match ontological concepts of the policy specification and the request (4) such as matching of domain and policy. The Policy Server further parses the OWL-based policy into a rule-based specification and contacts the Rule Reasoning Module to evaluate the specification (5). Finally, the Policy Server reports matched results back to PADP (6) which may in turn rank the results before returning to the distributor (7).

On technical details, we use StAX [14] to generate OWL-based policy and the corresponding rule-based specification in ABLE Rule Language [15]. SNoBASE [16] is used as the ontology querying tool and ABLE [17] is the rule reasoning engine.

7 Conclusion

In this paper we have proposed a new approach to defining business rules policies for Web Services and using such policies in Web Services discovery. An OWL ontology language is used to represent rules for business functions while rule-based reasoning is used for evaluation of rules in service matching. The deployment of the policies is adhered to the standard WS-Policy framework.

By the definition of policy as a set of rules, it is obvious that the effective way to represent a policy is by using a rule-based language since it is the most convenient for policy evaluation with a rule-based engine. Nevertheless, we take the overheads of representing the policy with an ontology language and later transforming it into a rule-based representation. This is due to the added benefit that policy evaluation would gain from ontological inference. For example, a supplier may have a policy `Perform OrderAndDeliverParts(days ≤ 2)` which means the supplier will order parts from another supplier first before making a delivery and all this is done within 2 days. If the distributor is looking for a supplier who can deliver parts within 3 days and there is an ontology that declares `OrderAndDeliverParts` as a subclass of `DeliverParts`, then this supplier will match the requirement.

This paper is merely an initial attempt to integrate the benefit from the world of ontology with the benefit of rule-based reasoning through the use of business rules policies. We plan to explore such integration further and consider the mechanism to determine policy compatibility, the degree of matching, and policy conflict. We also see the possibility to unify the policy ontology in this paper with the behavioural ontology in our previous work [13], which models a Web Service in terms of its operation, input, output, precondition, and effect, so that the behaviour of Web Services is better modelled.

Acknowledgements

This work is partly supported by Thailand-Japan Technology Transfer Project and Chulalongkorn University-Industry Linkage Research Grant Year 2004.

References

1. uddi.org: UDDI: Universal Description, Discovery and Integration of Web Services Version 3 (online). (2002) http://www.uddi.org
2. Box, D. et al.: Web Services Policy Framework (WS-Policy) (online). (2003). http://www-106.ibm.com/developerworks/library/ws-polfram/
3. Della-Libera, G. et al.: Web Services Security Policy (online). (2002) http://www-106.ibm.com/developerworks/webservices/library/ws-secpol/
4. w3.org: OWL Web Ontology Language Overview (online). (2004) http://www.w3.org/TR/2004/REC-owl-features-20040210/
5. Uszok, A. et al.: KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement. In: Proceedings of Policy Workshop, Italy (2003) 93-98
6. Kagal, L., Paolucci, M., Srinivasan, N., Denker, G., Finin, T., Sycara, K.: Authorization and Privacy for Semantic Web Services. In: Proceedings of AAAI 2004 Spring Symposium on Semantic Web Services (2004)
7. Kagal, L.: Rei: A Policy Language for the Me-Centric Project. HP Labs : Tech Report: HPL-2002-270 (2002)
8. Chun, S. A et al.: Policy-Based Web Service Composition. In: Proceedings of 14th International Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications (RIDE'04) (2004)
9. Verma, K. et al.: On Accommodating Inter Service Dependencies in Web Process Flow Composition. In: Proceedings of 1st International Semantic Web Services Symposium (2004)
10. DAML: OWL-S: Semantic Markup for Web Services (online). (2004) http://www.daml.org/services/owl-s/1.0/
11. w3.org: Web Services Description Language (WSDL) 1.1 (online). (2001) http://www.w3.org/TR/wsdl
12. Box, D. et al.: Web Service Policy Attachment (WS-PolicyAttachment) (online). (2003) http://www-106.ibm.com/developerworks/library/ws-polatt/
13. Sriharee, N., Senivongse, T.: Discovering Web Services by Using Behavioural Constraints and Ontology. In Stefani, J-B., Demeure, I., Hagimont, D. (eds.): Proceedings of 4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2003). Lecture Notes in Computer Science, Vol. 2893. Springer-Verlag (2003) 248-259
14. StAX: Streaming API for XML Version 1.0 (online). (2003) http://dev2dev.bea.com/technologies/stax/index.jsp
15. ABLE Rule Language User's Guide and Reference, Version 2.0.1 (online). (2003) http://www.alphaworks.ibm.com/tech/able
16. Lee, J., Goodwin, R.T., Akkiraju, R., Doshi, P., Ye, Y.: SNoBASE: A Semantic Network-Based Ontology Management (online). (2003) http://alphaWorks.ibm.com/tech/snobase
17. ABLE: Agent Building and Learning Environment (online). (2003) http://www.alphaworks.ibm.com/tech/able