Aligning WSMO and WS-Policy [⋆]

Dumitru Roman, Jacek Kopecký, Ioan Toma, and Dieter Fensel

Digital Enterprise Research Institute, Innsbruck, Austria
{firstname.lastname}@deri.org

Abstract. Service-Oriented Architectures (SOAs) suggest that IT systems should be developed from coarse-grained, loosely coupled, business-aligned components, so called services. One way towards loose coupling is to refrain from hard-coding policies in the system and to represent them explicitly. Semantic Web Services (SWS) add semantics to Web services (main realization of SOAs). However, SWS research currently ignores most of the work done on Web service policies, therefore in this paper we present a proposal for combining WSMO, a major SWS framework, and WS-Policy Framework, a set of specifications with heavy industrial backing. The resulting combination is aimed at serving as the basis of applying the logical reasoning capabilities that have been developed (or are being developed) for WSMO to WS-Policy, and the basis for integrating WSMO in the WS-Policy framework.

1 Introduction

Semantic Web based approaches to policies have been recently advocated (in e.g. [2, 16, 11, 7, 8, 15]) in order to cope with some of the limitations of traditional policy handling systems (e.g. policy systems existing before the era of Web services) where the policies are hard coded into a system according to the functional requirements, language features, and design decisions, without separating policy specification from policy implementation, and thus making difficult and expensive to change and enforce policies. Policy *specification*, *enforcement* and *revision* are three basic tasks a policy-handling system must provide. In this paper we focus on issues related to policy specification in semantically enriched service-oriented environments. Policy specification languages enable policies to be captured independent from a concrete system implementation. Such languages are to be interpreted by a policy engine at runtime, which makes dynamic policy changes possible; they formalize the intent of the designer into a form that can be analyzed and interpreted by a policy-aware system.

In this context, we are interested in identifying the potential relations between the policy languages and the so-called Semantic Web Services [10, 6]. Semantic Web Services have emerged as a combination of Semantic Web and Web services technologies with the aim of automating different service-related tasks, such as Web service discovery, composition, mediation, or execution. Several proposals have been put forward in this area.[1] In this paper we choose the WSMO approach [13] for analysis of its relation to policy domain because, in contrast to other approaches, it provides a unifying

[⋆] The work is funded by the European Commission under the projects ASG, DIP and SWING.

[1] We refer the reader to [12] for a detailed discussion on the SWS approaches.

framework combining a conceptual model (Web Service Modeling Ontology), a formal syntax and language (Web Service Modeling Language), and an execution environment (Web Service Execution Environment).

Common denominators have been identified between policies that may be leveraged to improve policy uniformity and streamline service-oriented enterprise-wide policy implementation. In particular, a set of new enforcement policies were proposed in the industry using novel policy concepts together with building blocks from e.g. XACML, WS-Policy and SAML. For the purpose of this paper, we have chosen to use the Web Services Policy Framework (WS-Policy) [5], a general purpose model and a syntax to describe the policies of a Web Service. Our choice is motivated by the fact that WS-Policy has significant industry backing. Interested parties are already creating policy assertions for various domains and the major commercial Web services infrastructure stacks already have or are building support for policy-based Web service invocations.

The aim of this paper is to investigate the potential relations between the WSMO approach to SWS and the WS-Policy framework, and to offer a basis for further research on the use of policies in the context of semantically enriched service environments. Issues like policy conformance checking, policy-based semi-automatic negotiations between users and services, or policy-based matchmaking etc., are important in our context, however they are out of scope of this paper. The scope of this paper is limited to identifying the commonalities and differences between the conceptual models of WSMO and WS-Policy, and proposing a concrete way for combining them.[2]

Despite the differences in terminology, we can say that both WSMO and WS-Policy describe the capabilities and constraints of Web services. On one side, WSMO uses very clearly defined terms like *Web service*, *Capability*, *Interface* etc. to capture all the relevant properties of Web services. On the other side, WS-Policy talks about generic assertions, focusing on their combinations in whole policies, and then attaching these policies to various subjects, among which Web service endpoints are especially relevant for our work.

We can identify two generic concepts from the WS-Policy framework that could encompass elements from WSMO: Policy Assertion and Policy Subject. In other words, parts of a WSMO description can be mentioned as assertions in some policy, or a policy could be attached to parts of a WSMO description. We describe these two directions and the syntax realizing the actual connections in the following two sections (Section 2 and Section 3). In Section 4 we briefly highlight related works, and in Section 5 we conclude this paper and point out potential directions for future work.

2 Attaching Policies to WSMO

To analyze how policies can be attached to WSMO, we need first to introduce a distinction between functional and non-functional properties.[3] In any application, the *func-*

[2] Because of the limited space we do not provide an overview of the concepts that WSMO and WS-Policy introduce, however, we refer the reader to [13] and [5] for a detailed introduction to WSMO and WS-Policy, respectively.

[3] A more detailed discussion on functional vs. non-functional properties can be found in the position paper for the W3C Workshop on Constraints and Capabilities for Web Services [1].

tional part of any data contains crucial information necessary for the application to do its job. *Non-functional* properties (NFPs), on the other hand, contain such additional data that may help the application do a better job, or to refine its functionality.

For example, one of the aims of WSMO is Web service discovery (we can see WSMO as an application for service discovery), and to enable discovery, WSMO describes client *Goals* and the *Capabilities* of the available service. *Goals* and *Capabilities* are the the necessary inputs to a matching algorithm, therefore they are functional aspects of WSMO descriptions. While pure Web service discovery only requires the *Capabilities* and *Goals*, the match maker can also take into account preferences and constraints over parameters like the price, availability or quality of a service. Because such parameters are not critical for the match maker, they are modeled as non-functional properties.

The distinction between functional and non-functional parameters depends highly on the application that uses the particular parameter — a functional parameter of one application can be non-functional in another. For instance, Semantic Web Services are an application that automates the use of Web services, and the price of a service is generally modeled as a non-functional property; however a shopping agent application will have price as one of its main functional parameters.

In WSMO, the distinction between functional and non-functional properties is made very clear: WSMO enumerates all the relevant functional properties (for example *Web service* has *Capability* and *Interface* as its functional properties) and it allows an extensible bag of NFPs everywhere. WS-Policy does not have any such distinction, so it can be used to express both functional and non-functional policy assertions, depending on the application that employs policies. Since WSMO enumerates in its conceptual model all the parameters that are functional for the aim of WSMO, policies can be treated as non-functional data, therefore when a WS-Policy is attached to a WSMO element, it is abstractly added to the non-functional properties of that element.

Fig. 1. Attaching Policies to WSMO Elements

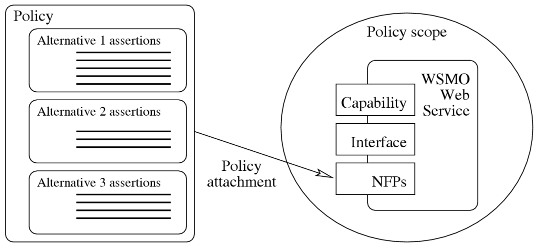


Figure 1 shows how WSMO elements can be used as policy assertions in a policy that is then attached to a particular policy scope. In particular, a policy is attached here to a WSMO Web service description, and it is treated as a non-functional property of that service. Due to the way policy attachment works syntactically, the policy is in fact embedded or referenced from within the non-functional properties block of the Web service description.

To summarize, WSMO elements can serve as WS-Policy Policy Subjects, and any policies attached to those WSMO elements are treated as non-functional properties. In the following, we present the syntax for including policies as non-functional properties in WSMO descriptions. Using this mechanism, the existing and future policy assertions can usefully complement the Dublin Core NFPs [18] currently used by WSMO.

To attach policies generically to XML elements such as WSDL descriptions, the WS-PolicyAttachment specification [4] defines an XML attribute called `PolicyURIs` and an XML element called `PolicyReference`. Both the attribute and the element point to external policies using URIs. WS-PolicyAttachment introduces both of them because some XML languages restrict attribute or element-based extensibility.

In WSMO, we use the namespace-qualified name `wsp:PolicyReference` for an NFP; its value is one or more URIs of policies attached to the owner WSMO element:

```
service ACMEService
  nonFunctionalProperties
    wsp#PolicyReference hasValue
      {_"http://fabrikam123.example.com/policies/DSIG",
       _"http://fabrikam123.example.com/policies/SECTOK"}
  endNonFunctionalProperties
```

In some cases it can be useful for manageability reasons to include the whole policy in the WSMO description, especially when the policy is fairly small. For this purpose we reuse the namespace-qualified name `wsp:Policy` as the name of a non-functional property whose content is the XML serialization of the whole `Policy` element. This is illustrated by the following example:

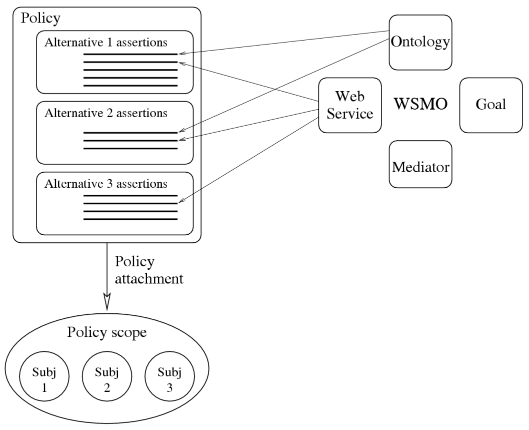
```
service ACMEService
   nonFunctionalProperties
     wsp#Policy hasValue
       "<wsp:Policy>
          <wsp:ExactlyOne>
            <wsse:SecurityToken>
              <wsse:TokenType>wsse:Kerberosv5TGT
              </wsse:TokenType>
            </wsse:SecurityToken>
            <wsse:SecurityToken>
              <wsse:TokenType>wsse:X509v3
              </wsse:TokenType>
            </wsse:SecurityToken>
          </wsp:ExactlyOne>
        </wsp:Policy>"
   endNonFunctionalProperties
```

To summarize, we allow attaching both external and embedded policies to WSMO elements, treating the attached policies as non-functional properties. For this, we reuse the WS-Policy element names (`wsp:PolicyReference` and `wsp:Policy`) as NFP identifiers in WSMO.

3 WSMO as Policy Assertions

WS-Policy is a mechanism of combining domain-specific policy assertions and attaching them to various policy subjects. WSMO descriptions can be viewed as policy assertions and combined with others in policy alternatives. Figure 2 shows how WSMO elements can be used as policy assertions in a policy that is then attached to a particular policy scope, for example a Web service endpoint. Such a policy would thus attach the WSMO Web service description to that endpoint.

Fig. 2. Using WSMO Elements as Policy Assertions



We can envision, for example, a policy that ties the capabilities of a Web service with various security settings. A service may offer some basic functionality with strong authentication but weak communication channel encryption, and more advanced functionality can be available provided that strong encryption is employed.

WSMO currently does not have any specific mechanism for expressing that alternative WSMO descriptions are in effect in conjunction with various non-functional properties,[4] and WS-Policy seems to be a widely-adopted mechanism for expressing exactly such alternatives, therefore even though WSMO descriptions would not normally be treated as policy assertions, such an approach may prove beneficial.

Because policy assertions must be XML elements, we can reuse WSML/XML serialization format (see [3]) for representing WSMO descriptions as policy assertions in WS-Policy. To attach a whole WSMO description as a single policy assertion, we use the element `wsml:wsml`.[5] For finer granularity (e.g. only asserting a single capability description) we can reuse the appropriate elements like `wsml:capability`.

In some situations it may be beneficial only to refer to a WSMO description (as opposed to including it inline as a policy assertion). For referring to a whole WSML file we introduce the element `wsml:descriptionReference` that refers to a WSML document, and similarly we can introduce specific elements for referring to specific elements of WSMO, for instance to refer to a capability or an interface we can use elements `wsml:capabilityReference` and `wsml:interfaceReference` that would refer to the identifiers of the WSMO capability or interface descriptions.

The listing in Figure 3 is a policy that claims the policy subject is described by the WSMO services http://example.org/services/ticketService and http://example.org/services/billingService. Lines 2–7 show a WSML/XML element `wsml:webService` that, in this context, means a policy assertion assigning a WSMO service description defined inline. Similarly, lines 13–15 show a reference to such a description, using the

[4] See Section 2 for discussion of why WSMO treats WS-Policy as non-functional properties.

[5] The first `wsml` is a namespace prefix and the second is the name of the XML element container for WSML/XML syntax.

Fig. 3. Example policy with WSMO webService as policy assertion

```
01  <wsp:Policy>
02    <wsml:webService
03         name="http://example.org/services/ticketService">
04      <wsml:capability name="ticketing">
05        ...
06      </wsml:capability>
07    </wsml:webService>
08    <wsml:ontology
09         name="http://example.org/ontologies/ticketing/">
10      <wsml:concept name="Ticket"/>
11      ...
12    </wsml:ontology>
13    <wsml:serviceReference>
14      http://example.org/services/billingService
15    </wsml:serviceReference>
16  </wsp:Policy>
```

new element `wsml:serviceReference`. Finally, lines 8–12 contain an ontology which is used by the ticketService definition.

The conclusion is that to represent a policy assertion “the policy subject has the following WSMO description” we can use any global WSML/XML element as appropriate, and to represent an assertion only referencing a WSMO description we have to create specific elements for each type of WSMO entity that we want to reference.

4 Related Works

To the best of our knowledge, the approach presented in this paper is the first attempt to combine the WSMO approach to SWS with policies. However, it is worth mentioning other works that deal with combining policies and Semantic Web technologies in general. Such approaches differ from our work in the sense that they are focused on Semantic Web technologies other than WSMO, and some of them are aimed at solving a specific policy-related task (e.g. policy conformance check, matchmaking of Web services, etc.), whereas we are mainly focused in this paper on combining WSMO and WS-Policy, without emphasizing at this stage any particular policy-related task.

We can classify relevant related works as those that deal directly with WS-Policy, and those which take a more general approach to policies, and thus not committing to WS-Policy as a framework for representing policies.

Among works that deal directly with WS-Policy, [9] provides a mapping of WS-Policy to OWL-DL in order to use OWL-DL reasoners to check policy conformance, [17] uses OWL ontologies for creating policy assertions with semantics in WS-Policy in order to enable matching the non-functional properties of Web Services represented using WS-policy, and [14] proposes an approach to behaviour-based discovery of Web Services by which business rules that govern service behaviour are described as a policy, in the form of ontological information; here WS-Policy is used to associate such a policy to the Web Service.

Works that do not deal directly with WS-Policy (but which take into account Semantic Web based approaches to policies) focus more on administrative policies such as security and resource control policies. In this category it is worth mentioning KAoS [16] as one of the first efforts to represent policies using OWL, [11] that discusses how to represent policy inheritance and composition based on credential ontologies and formalizes these representations in Frame-Logic, [15] that proposes a hybrid approach for policy specifications which exploits the expressive capabilities of Description Logic languages and Logic Programming approaches; and Rein [8],[6] which is a framework for representing and reasoning over policies in the Semantic Web.

5 Conclusions and Outlook

This paper is a first step to combine WS-Policy (a policy framework with significant industry backing) and the WSMO approach to Semantic Web Services (one of the most important proposals for Semantic Web Services to date). We identified potential ways of combining them, and provided the necessary syntax. The new syntax elements are summarized in Table 1.

Table 1. Syntax for combining WSMO and WS-Policy

Name	**Type**	**Description**
wsp#PolicyReference	NFP identifier	non-functional property referring to an external policy file by URI
wsp#Policy	NFP identifier	non-functional property containing an XML serialization of a policy
wsml:wsml	XML Element	the root element of WSML/XML syntax, reused as a policy assertion “the embedded WSML description applies (to the policy subject)”
wsml:webService wsml:goal ...	XML Element	other WSML/XML elements reused as a policy assertion “the embedded WSML description applies (to the policy subject)”
wsml:descriptionReference wsml:webServiceReference wsml:goalReference ...	XML Element	policy assertions that refer to WSMO definitions by their identifier URIs

The proposals presented in the paper represent the basis for enabling the use of policies from SWS environments for tasks such as policy-based semi-automatic negotiations between users and services, policy-based matchmaking between users requests and services, scheduling of service compositions under certain constraints, etc; but also for enabling the use of semantic descriptions as policy assertions.

In future work, we plan to investigate concrete applications of the combination of WSMO and WS-Policy presented in this paper, especially applying existing reasoning techniques developed in the context of WSMO to policy specification, as well as developing new techniques in order to provide automated support for Web service discovery, negotiation, selection, composition, invocation and monitoring with policy awareness.

[6] `http://dig.csail.mit.edu/2006/06/rein/`

References

1. S. Arroyo, C. Bussler, J. Kopecký, R. Lara, A. Polleres, and M. Zaremba. Web Service Capabilities and Constraints in WSMO. In *W3C Workshop on Constraints and Capabilities for Web Services*, September 2004.
2. P. A. Bonatti and D. Olmedilla. Semantic Web Policies: Where are we and What is still Missing? A tutorial at ESWC'06, 2006. Available at `http://www.l3s.de/~olmedilla/events/2006/ESWC06/ESWC06_Tutorial.html`.
3. J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language WSML. Available via `http://www.wsmo.org`, October 2005.
4. C. Sharp (editor) et al. Web Services Policy 1.2 – Attachment (WS-PolicyAttachment). Technical note, April 2006. Available via `http://www.w3.org/`.
5. J. Schlimmer (editor) et al. Web Services Policy 1.2 – Framework (WS-Policy). Technical note, April 2006. Available via `http://www.w3.org/`.
6. D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2), 2002.
7. L. Kagal, T. Berners-Lee, D. Connolly, and D. Weitzner. Self-Describing Delegation Networks for the Web. In *Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, pages 205–214, 2006.
8. L. Kagal, T. Berners-Lee, D. Connolly, and D. J. Weitzner. Using Semantic Web Technologies for Policy Management on the Web. In *AAAI*, 2006.
9. V. Kolovski, B. Parsia, Y. Katz, and J. A. Hendler. Representing Web Service Policies in OWL-DL. In *International Semantic Web Conference*, pages 461–475, 2005.
10. S. McIlraith, T. Son, and H. Zeng. Semantic Web Services. In *IEEE Intelligent Systems (Special Issue on the Semantic Web)*, 2001.
11. W. Nejdl, D. Olmedilla, M. Winslett, and C. C. Zhang. Ontology-Based Policy Specification and Management. In *2nd European Semantic Web Conference (ESWC)*, pages 290–302, 2005.
12. D. Roman, J. de Bruijn, A. Mocan, I. Toma, H. Lausen, J. Kopecky, D. Fensel, J. Domingue, S. Galizia, and L. Cabral. Semantic Web Services – Approaches and Perspectives. In J. Davies, P. Warren, and R. Studer, editors, *Semantic Web Technologies: Trends and Research in Ontology-based Systems*, pages 191–236. John Wiley & Sons, 2006.
13. D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology. *Applied Ontology*, 1(1):77–106, 2005.
14. N. Sriharee, T. Senivongse, K. Verma, and A. P. Sheth. On Using WS-Policy, Ontology, and Rule Reasoning to Discover Web Services. In *INTELLCOMM*, pages 246–255, 2004.
15. A. Toninelli, J. Bradshaw, L. Kagal, and R. Montanari. Rule-based and Ontology-based Policies: Toward a Hybrid Approach to Control Agents in Pervasive Environments. In *Proceedings of the Semantic Web and Policy Workshop*, November 2005.
16. A. Uszok, J. M. Bradshaw, R. Jeffers, A. Tate, and J. Dalton. Applying KAoS Services to Ensure Policy Compliance for Semantic Web Services Workflow Composition and Enactment. In *International Semantic Web Conference*, 2004.
17. K. Verma, R. Akkiraju, and R. Goodwin. Semantic Matching of Web Service Policies. In *SDWP Workshop*, 2005.
18. S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. Dublin Core Metadata for Resource Discovery. RFC 2413, IETF, 1998.