Open Research Online

A Trust Based Methodology for Web Service Selection

Stefania Galizia, Alessio Gugliotta and John Domingue
Knowledge Media Institute
Open University
Milton Keynes, MK7 6AA, UK
{s.galizia, a.gugliotta, j.b.domingue}@open.ac.uk

Abstract

In this paper, we propose a methodology for addressing trust in Semantic Web Services (SWS) - based applications. The aim is to enhance the capability-driven selection provided by current SWS frameworks with the introduction of trust-based selection criteria. We present an ontology - Web Services Trust Ontology (WSTO) – that models the context of a trust-based interaction and enables the participants to describe semantically their trust requirements and guarantees. WSTO makes use of WSMO as reference ontology for representing Web Services and embodies the problem of finding the most "trusted" Web service as a classification problem. To test our methodology, we implemented a specific module within IRS-III – a WSMO-based SWS broker – and deployed a prototype application based on a use case scenario.

1. Introduction

Semantic Web Service (SWS) technology combines the flexibility, reusability, and universal access that typically characterize Web Services (WS), with the expressivity of semantic mark-up and reasoning of semantic Web [3]. SWS introduce semantic descriptions of the capabilities of Web services to enable their automatic discovery and selection. Whenever a user expresses a goal that she wants to achieve, the most appropriate Web service is dynamically discovered and selected on the basis of the available descriptions. Since the selected Web service is not known a priori by the user, the notion of trust becomes an important aspect of WS selection. Nevertheless representing trust is not a trivial task. Though in the well known "layer cake"[1] for semantic Web architecture "trust" is the higher layer, few

---

[1] http://www.w3.org/2002/Talks/04-sweb/

approaches in semantic Web really provide a methodology to represent it. In particular, the most common approaches for describing semantic Web services, such as WSMO [7] or OWL-S [16], do not provide exhaustive means for trust annotation.

We believe that the main difficulty of representing trust lies in its context-based nature: the same user may have different trust policies in different contexts. For instance, a user tends to trust a Web service with strong security certifications, whenever she has to provide credit card information. Otherwise, she may trust Web services with high data accuracy in contexts where required data are crucial, such as biomedical services. Moreover, different users may privilege distinct trust parameters in the same context; their priority may depend on their personal preferences.

In literature, a number of trust establishments can be found [1], [11], [15], [17], [20]. Some of them are very complex and elaborate, but no one is suitable for all contexts. For this reason, our approach is intentionally general. We do not provide any new trust definition; we developed an ontology – Web Services Trust Ontology (WSTO) - that is able to represent generic trust specifications within SWS-based interaction context. Differently to other approaches, we embodied the Web service selection in a classification problem: given a set of user and Web service policies and established a classification criterion, our goal is to identify the solution, i.e. the class of Web services matching with trust policies of involved interaction participants. To accomplish this, we based WSTO on a general purpose classification library developed within the European project IBROW [8], [13]. Furthermore, WSMO is our reference model for describing semantic Web services; WSTO extends it by supplying the trust management mechanism introduced above.

IRS-III [4], the Internet Reasoning Service, is a suitable tool that we use as execution environment, for actualizing our methodology. IRS-III is a broker that is able to perform capability-driven selection of WSMO

compliant semantic Web services. We improved the selection mechanism of IRS-III reasoning on the concepts defined in WSTO. As a result, whenever several Web services with the same capability can satisfy a user's goal, the class of Web services that exposes trust policies matching with the user policies is selected.

An earlier version of WSTO has been described in [9]. In the present paper, we outline the background information at the basis of our approach (Section 2) and propose an improved version of WSTO (Section 3). Moreover, we describe a prototype application based on a use case scenario that adopts the trust-based version of IRS-III (Section 4), and outline the related work in section 5. Section 6 concludes this paper.

2. Background

In this section, we provide some information to place our approach among existing trust approaches as well as to introduce the two ontologies that are at the basis of our methodology: WSMO and Classification Library.

2.1. Trust Approaches

As trust can have different meaning in different contexts, several specifications can be found in literature. We can classify existing models into the following three main approaches:

- *Policy-based*. Policies are a set of rules that specify the conditions to disclose own resources.
- *Reputation-based*. Reputation based approaches make use of rating coming from other agents or a central engine, by heuristic evaluations.
- *Trusted Third Party-based (TTP)*. Trusted Third Party based models use an external, trusted, entity that evaluates trust.

These general approaches can be refined and/or combined in order to build a concrete trust establishment solution that can be deployed in a real system.

Many models [15], [11] formulate trust policies in SWS by security statements, such as confidentiality, authorization, authentication. W3C Web service architecture [21] recommendations base trust policies on security consideration, even if the way to disclose their security policies is still not clear.

Some policy-based models rely on a TTP, which works as a repository of service description and policies [15] and meanwhile as an external matchmaker that evaluates service trustworthiness according to given algorithms.

Reputation based models reuse concepts and approaches taken from Web-based social networks [10]. In SWS as well as in social networks, trust is a central issue. In both the cases, interactions take place whenever there is trustworthiness.

The idea is that involved participants express their opinion of trust, by means of a shared vocabulary. Several algorithms for trust propagation and different metrics have been defined. In reputation-based approaches most of trust algorithms are more generically Quality of service based [17], [20] by making the service ability the main trust statement. Quality of service (QoS) is defined by a set of properties related to the service performance. Precision and accuracy of data, timeliness in executing a task, are the main features; also security can be considered part of QoS.

Our approach can be classified as *Policy/TTP-based*, since the interacting participants express their trust policies in their – semantically described - profiles. The SWS broker – in our case IRS-III – will behave as a TTP by storing participant profiles and reasoning on them.

2.2. WSMO

The Web Service Modelling Ontology (WSMO) [7] is a formal ontology for describing the various aspects of services in order to enable the automation of Web service discovery, composition, mediation and invocation. The meta-model of WSMO defines the following four top level elements. *(i) Ontologies* provide the foundation for describing domains semantically. They are used by the three other WSMO elements. *(ii) Goals* define the tasks that a service requester expects a Web service to fulfil. In this sense they express the requester's intent. *(iii) Web Service* descriptions represent the functional behavior of an existing deployed web service. The description also outlines how web services communicate (*choreography*) and how they are composed (*orchestration*). *(iv) Mediators* handle data and process interoperability issues that arise when handling heterogeneous systems.

2.3. Classification library

Classification can be seen as the problem of finding the solution (class) which best explains a certain set of known facts (observables) about an unknown object, according to some criterion.

The classification framework - that we use and extend for our purposes - is a library of generic, reusable components whose purpose is to support the

specification of classification problem solvers. The library was developed within the European project IBROW [8], [13] and its basic structure is centered around UPML [5], a framework for libraries of reusable knowledge level components, founded on tasks, problem solving methods and domain models. Furthermore, the library was specified using the OCML modelling language [12]. We use the term "observables" to refer to the known facts we have about the object (or event, or phenomenon) that we want to classify. Each observable is characterized as a pair of the form *(f, v)*, where f is a feature of the unknown object and v is its value. Here, we take a very generic viewpoint on the notion of feature. By feature, we mean anything which can be used to characterize an object, such that its value can be directly observed, or derived by inference. As is common when characterizing classification problems [22], we assume that each feature of an observable can only have one value. This assumption is only for convenience and does not restrict the scope of the model. The solution space specifies a set of predefined classes (solutions) under which an unknown object may fall.

A solution itself can be described as a finite set of feature specifications, which is a pair of the form *(f, c)*, where f is a feature and c specifies a condition on the values that the feature can take. Then, we can say that an observable *(f, v)* matches a feature specification *(f, c)* if *v* satisfies the condition *c*.

It is possible to envisage different solution criteria. For example, we may accept any solution which satisfies some condition and is not inconsistent with any other condition. This criterion is called positive coverage [18]. Alternatively, we may require a complete coverage - i.e., a solution is acceptable if and only if it satisfies all conditions. Thus, the specification of a particular classification task needs to include a solution (admissibility) criterion. This in turn relies on a match criterion, i.e., a way of measuring the degree of matching between candidate solution and a set of observables. By default, this library provides a match criterion based on the aforementioned model. More details are available in [13].

3. WSTO

The Web Service Trust Ontology (WSTO) introduces a novel approach for managing trust in SWS-based applications. The underlying idea is considering trust strictly depending on the context, without providing a further trust definition. To accomplish this, we embed the trust-based selection of Web services into a classification problem: Web services are classified according to the specific user as

well as policies. A policy is expressed in terms of either trust requirements or guarantees.

Briefly, both user and Web service expose their trust guarantees, which are represented as observables of the Classification Library (Section 2.3); conversely, trust requirements are conditions represented within the candidate solutions. Given observables and candidate solutions, a classification criterion is necessary to first classify Web services and then identify the most appropriate class that address both the user and Web service requirements and guarantees. The selection of one - or a set - of Web services corresponds to the task of finding the solutions in a classification problem.

The following section details the main elements of WSTO that address the process introduced above.

3.1. Classifying Semantic Web Services

Figure 1 summarizes how our classification framework is applied within SWS trusted discovery.
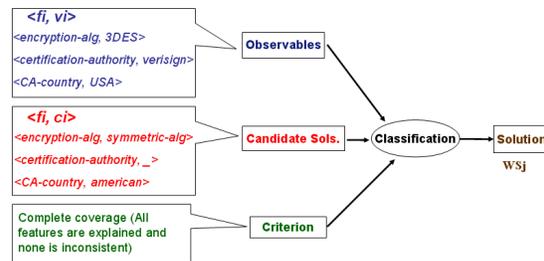

Figure 1 Classifying Web Services

Candidate solutions - i.e. user requirements - are defined as pairs (feature, condition). In the example user requirements are represented by a pair feature-condition (f_i,c_i), such as *(encryption-algorithm, { encryption-algorithm= any symmetric algorithm}), (CA-country, {country-origin= any country in the continent America})*.

The observables are pairs (feature, value) representing WS guarantees. The values of the specific encryption algorithm adopted, certification authority (CA) issuing security tokens and CA origin country are provided by Web services as guarantees. According to our methodology, they are represented as pairs feature-value (f_i,v_i). The chosen solution admissibility criterion, in this example, is complete coverage. Our classification goal is to identify the class of Web services that fit with user trust requirements, given a set of WS trust guarantees.

Figure 2 depicts the main concepts of WSTO. *Goal*, *WS* and *User* are the key concepts. Both *User* and *WS*

are subclasses of the class p*articipant*. Every participant can expose their own trust profiles.

A trust profile represents the policy that a participant declares to the execution environment – IRS-III in our case - in order to have a trust based interaction. It is expressed in terms of trust guarantees and/or requirements. For instance a client, that has to provide a credit card number in order to obtain a service, will trust Web services that provide "good" security guarantees.
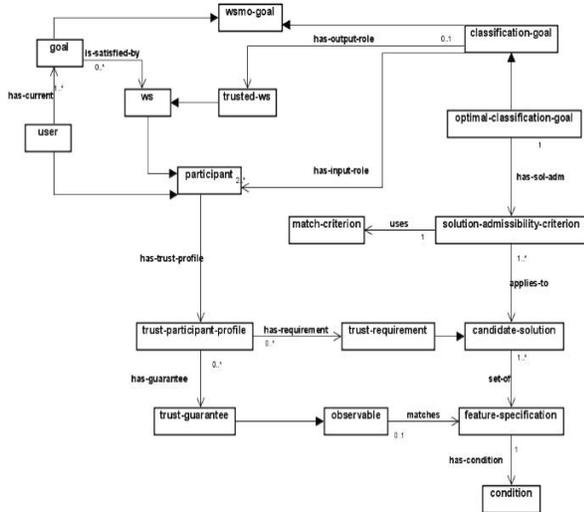

Figure 2 Diagram partially representing WSTO

In WSTO, *Trust-guarantees* are represented as observables: pairs of feature and corresponding value *(f, v)*; *Trust-requirements* are candidate solutions, which are represented by pairs of feature, condition *(f, c)*. The classification match criterion that we use in our prototype is that one described in Section 2.3. However, WSTO can be easily extended and further classification criteria will be introduced.

We apply the complete coverage as solution admissibility criterion, because we look for a solution that satisfies all conditions and is not inconsistent with any data. In other words, our constraint is that all user trust requirements have to be satisfied.

The Classification Library implements two different classification methods: *single-solution-classification* and *optimal-classification*. The former implements hill climbing algorithm with backtracking to find a suitable solution, the latter executes an exhaustive search for an optimal solution. In turn, the two implemented classification tasks are: *optimal-classification-task* and *single-solution-classification-task*, respectively solved by the two methods described above, according to UPML framework [8]. WSMO is based on WSMF [6] which is itself partly descendent from UPML, thus,

there is an existing strong relationship between the two frameworks. In particular, we can simply link WSMO goals and UPML tasks, as they both represent the user's objective.

We make use of the *optimal-classification-task* and redefine it as WSMO *goal - optimal-classification-goal*; *participant-profiles* represent the goal pre-conditions and *trusted-ws* could be an example of goal post-conditions.

4. Prototype Application

To prove the feasibility and applicability of our methodology, we deployed a prototype application based on an improved version of IRS-III. In particular, we implemented a new IRS-III module that introduces a trust-based selection mechanism. In this way, IRS-III is now able to behave as TTP (Section 2.1) by storing participant profiles and reasoning on them.

4.1. IRS-III Overview

IRS-III is a platform and a broker for developing and executing semantic Web services. By definition, a broker is an entity which mediates between two parties, and IRS-III mediates between a service requester and one or more service providers. To achieve this, IRS-III adopts a semantic Web based approach and thus it is founded on ontological descriptions. In particular, IRS-III has incorporated and extended WSMO as the core epistemological framework. A core design principle for IRS-III is to support goal-centric and capability-based invocation mechanism. An IRS-III user simply asks for a goal to be solved. Using a set of SWS descriptions, IRS-III will: a) discover potentially relevant Web services; b) select the set of Web services which best fit the incoming request; c) mediate any mismatches at the data, ontological or business process level; and d) invoke the selected Web services whilst adhering to any data, control flow and Web service invocation constraints. Additionally, IRS-III supports the SWS developer at design time by providing a set of tools for defining, editing and managing a library of semantic descriptions and also for grounding the descriptions to either a standard Web service with a WSDL description, a Web application available through an HTTP GET request, or code written in a standard programming language (currently Java and Common Lisp).

4.2. Prototype Implementation

Web service selection in IRS-III - up to now restricted to a capability-based model - became trust-

based thanks to WSTO. Given several Web services, semantically described in IRS-III, all with the same capability, but different trust profiles, the class of Web service selected will be the one that matches closest with the user trust profile.

In this section, we present a prototype, an implemented example of a virtual travel agent service. The prototype is implemented in OCML [12], the modelling language underlying IRS-III. The goal is to find the train timetable, at any date, between two European cities. Origin and destination cities have to belong to the same country (European countries involved in our prototype are: Germany, Austria, France and England). The client that uses this application in IRS-III publishes her trust-profile, with trust requirements and/or trust guarantees. In our prototype, we provide three different user profiles and three different Web services, able to satisfy the user goal. Specifically, in this basic example, User profiles are expressed solely through trust requirements, without trust guarantees. In principle, our approach is symmetric: users as well as Web services can provide trust requirements and guarantees.

All user requirements are performed in terms of security parameters: *encryption-algorithm*, *certification-authority* and *certification-authority-country*. Every user expresses a qualitative level of preference for every parameter.

```
USER4

(def-class trust-profile-USER4 (trust-
                                profile)
 ((has-trust-guarantee :type guarantee-
                                USER4)
  (has-trust-requirement :type requirement-
                                USER4)))


(def-class requirement-USER4 (security-
                              requirement)
 ((encryption-algorithm :value high)
  (certification-authority :value medium)
  (certification-authority-country :value
                                medium)))

USER5
........

(def-class requirement-USER5 (security-
                              requirement)
 ((encryption-algorithm :value medium)
  (certification-authority :value low)
  (certification-authority-country :value
                                low)))

USER6
........

(def-class requirement-USER6 (security-
                              requirement)
 ((encryption-algorithm :value low)
  (certification-authority :value high)
  (certification-authority-country :value
```

```
                                     high)))
```
Listing 1 User Profiles

For instance, the class *user-4* would like to interact with a Web service that provides a high security level in term of encryption algorithm, but she accepts medium value for certification authority and certification authority country. Representing user requirements in a qualitative way seems to be more user-friendly. Heuristics are necessary for express quantitative representations in qualitative. The listing below is an example of heuristic.

```
ENCRYPTION-ALGORITHM HEURISTIC

(def-instance encryption-algorithm-
                 abstractor abstractor
   ((has-body '(lambda (?obs)
              (in-environment
                ((?v . (observables-feature-
         value ?obs 'encryption-algorithm)))
              (cond ((== ?v DES)
                       (list-of 'encryption-
                          algorithm 'high

(list-of (list-of 'encryption-algorithm
                           ?v))))
((== ?v AES)
 (list-of 'encryption-algorithm 'medium
 (list-of (list-of 'encryption-algorithm
                           ?v))))
 ((== ?v RSA)
  (list-of 'encryption-algorithm 'low
  (list-of (list-of 'encryption-algorithm
                           ?v))))))))))

(applicability-check (kappa (?obs)
     (member  'encryption-algorithm
     (all-features-in-observables ?obs))))))))
```
Listing 2 Encryption Algorithm Heuristic

The heuristic *encryption-algorithm-abstractor* establishes that whenever the encryption algorithm adopted by a Web service provider is like *DES*, then its security level is considered high. Whenever both *User* and *Web service* describe their profiles, they implicitly agree with the qualitative evaluation expressed the heuristic. If a Web service provides an encryption algorithm *3DES*, for instance, it is considered secure, as *3DES* adopts *DES*, how it specified in the Listing 3.

```
3DES SUBCLASS OF DES ALGORITHM

  (def-class DES-type () ?x
    :iff-def (or (= ?x DES)
                 (subclass-of ?x DES)))

  (def-class DES (algorithm))
  (def-class 3DES (DES))
```
Listing 3 3DES subclass of DES Algorithm

In turn, whenever the Web service provider makes use of an algorithm like *AES*, according to the heuristic in Listing 2, its encryption ability is deemed medium,

otherwise, if the adopted algorithm is like *RSA*, the security level is low. Other heuristics provide qualitative evaluations of Certification Authorities (CA), and CA countries. For instance, security level of *globalsign-austria* is retained high, conversely German CAs are considered medium-secure.

The user can apply these heuristics, or define her own, sharing her expertise and knowledge with other users. Alternatively, the user can even express her requirements in precise/quantitative way, by specifying the exact values expected from Web service guarantees, for example, the certification authority issuing security token has to be *VeriSign*.

The Web services able to satisfy the user goal, implement their profile only in terms of guarantees. As shown in Listing 4, three different Web services show their trust guarantees by the same parameters provided by the users in their requirements, that are *encryption-algorithm*, *certification-authority* and *certification-authority-country*.

```
TRUSTED WEB SERVICE WS1

(def-class get-train-timetable-service-T1
                           (trust-web-service)
 ((has-capability :value get-train-
                     timetable-capability-T1)
  (has-interface :value get-train-
               timetable-service-interface-T1)
  (has-trust-profile :type get-train-
        timetable-service-trust-profile-T1)))

(def-class get-train-timetable-service-
             trust-profile-T1 (trust-profile)
 ((has-trust-guarantee :type get-train-
               timetable-service-guarantee-T1)
  (has-trust-requirement :type get-train-
          timetable-service-requirement-T1)))

(def-class get-train-timetable-service-
  guarantee-T1 (Trust-non-functional-
                                 properties)
 ((encryption-algorithm :type 3DES)
  (certification-authority :value verisign)
  (certification-authority-country :value
              north-american-country)))

TRUSTED WEB SERVICE WS2
  ........
(def-class get-train-timetable-service-
  guarantee-T2 (Trust-non-functional-
                                 properties)
 ((encryption-algorithm :type RSA)
  (certification-authority :value
                        globalsign-austria)
  (certification-authority-country :value
                              austria)))

TRUSTED WEB SERVICE WS3
  ........
(def-class get-train-timetable-service-
  guarantee-T3 (Trust-non-functional-
                                 properties)
 ((encryption-algorithm :type AES)
  (certification-authority :value tc-trust-
                                      center)
  (certification-authority-country :value
```

```
                                germany)))
```

Listing 4 Web Services Trust Profiles

Our example only includes users and Web services with trust policies expressed by same parameters. As we apply the complete coverage admissibility solution criterion, we look only for an exact match, thus every user requirement has to be satisfied by at least one Web service trust guarantee; Web services that do not satisfy all conditions are not selected.

Listing 4 shows that the Web service *get-train-timetable-service-T1* warrants its security level by declaring that encryption algorithm it adopts is *3DES*, the CA issued the security token is *VeriSign* and its head office is in USA. The Web service *get-train-timetable-service-T2* publishes values *RSA*, *globalsign-Austria*, *Austria*, for the same guarantee statements. The Web service *get-train-timetable-service-T3* warrants secure interaction by adopting *AES* as encryption algorithm, and declaring the German CA *tc-trust-center* as security token issuer.

We developed a user-friendly Web application to test our implementation, which is available at http://irs-test.open.ac.uk/trusted_vta/.

The snapshot in Figure 3 shows the Web application interface. The user who would like to know train timetable between two European cities enters the desired city names and date. The user owns a trust profile associated to her name: *dinar* is instance of *user4* trust profile, *vanessa* of *user5*, *stefania* of *user6*.

Whenever the application starts, IRS-III recognizes, from the user name, the trust user profile. In the prototype, the requirements expressed by the user are treated as candidate solutions within the classification goal. The class of Web services whose trust guarantees best match with user requirements is selected. As we applied the complete coverage criterion, the match is strict, that means every user requirement is explained (matches with a Web service trust guarantee) and none is inconsistent.

The user *dinar*, instance of user profile *user4*, likes to travel by train between *Berlin* and *Frankfurt* on *9th December 2006*. She would like to interact only with the Web services with high security level encryption algorithm, medium security level CA and CA country. Suitable heuristics consider highly secure encryption algorithms like *DES*, medium level secure the CA Americans, particularly *Verisign*.

The class of Web services that satisfies all *dinar* trust requirements is *get-train-timetable-service-T1*. Actually, *get-train-timetable-service-T1* adopts the encryption algorithm *3DES*, in turn, the heuristic in Listing 2 shows *DES*, as highly secure. As *3DES* is a subclass of *DES* algorithm, then it matches with *dinar* trust requirements.

Figure 3 WSTO Web Application

The application returns the list of Web services able to satisfy the user goal, and that one invoked, which matches with *dinar* trust requirements. It follows the Web service output, the requested timetable. Easily the application can be tested with the other user instances implemented, *vanessa* and *stefania*. It can be noticed that *vanessa* trust profile matches with Web service class *get-train-timetable-service-T3*, while *stefania* with *get-train-timetable-service-T2*.

A non-trusted based version of the application is available at http://irs-test.open.ac.uk/not_trusted_vta/ for comparison purposes. In the non trusted version, if more then one Web service with same capability are able to satisfy a user goal, the selection happens completely random. The output of this Web application returns only the train timetable requested, without any trust based comparison.

5. Related Work

There is a growing corpus of literature on trust, and different approaches focus on how trust assumptions are made and enforced. A number of current approaches model social aspects of trust [10], while some recent efforts in the last few years concern service-oriented views of trust [2]. However, few approaches provide methodologies for managing trust in a SWS, and none comprehensively incorporate all possible approaches of trust (policy, reputation TTP), as we do in WSTO.

The work proposed by Vu and his research group [20], who use WSMX[2] [23] as an execution environment, is closely related to the work reported here. Vu et al. [20] propose a methodology for enabling a QoS-based SWS discovery and selection, with the application of a trust and reputation management method. Their approach yields high-quality results, even under behaviour which involves cheating. With respect to their work, our methodology does not propose any algorithm for service behaviour prediction. However, their algorithm is wholly founded on reputation mechanisms, and is therefore not suitable for managing policy-based trust assumptions. Currently, policy-based trust mainly considers access control decisions via digital credentials. Our framework, by enabling participants to declare general ontological statements for guarantees and requirements, is also able to accommodate a policy-based trust framework.

Olmedilla et al. [15] propose a methodology for trust negotiation in SWS. They employ PeerTrust [14], a policy and trust negotiation language, for establishing if trust exists between a service requester and provider. The main issue, which distinguishes their methodology from ours, is that they assume that trust is solely based on policy. Similar to our approach, they use WSMO as the underlying epistemology. Moreover, they assume delegation to a centralized trust matchmaker, where the participants disclose policies. Similarly, in our approach, we assume that IRS-III plays the role of trust matchmaker. Furthermore, they also address negotiation, which is an important issue in SWS interaction. We do not propose a formal negotiation mechanism here, but, as both requester and provider disclose their guarantees, as credentials within IRS-III, we are able to automatically enable an implicit negotiation process.

There are other approaches for managing trust in SWS which are less closely related to ours such as KAoS [19]. Even though KAoS presents a dynamic framework, and recognize trust management as a challenge for policy management, the framework is not specifically tailored to trust management in SWS.

6. Conclusion

In this paper, we have presented a prototype of a trust-based selection in IRS-III, based on WSTO, an ontology for managing trust in SWS. Furthermore, our framework makes use of a classification library, developed within the European project IBROW [8]. We have envisaged the Web service selection as a classification problem, where the solution is the class of Web services matching with participant trust profiles. Trust profiles are represented in terms of requirements and guarantees. Whenever participant trust policies match, a trusted interaction can occur.

This work does not provide any new definition of trust, because we strongly believe that trust holds different meanings in different contexts.

---

[2] http://www.wsmo.org/TR/d13/d13.0/v0.2/

One novel feature of our approach is that the framework allows trust to be modelled using a specific set of concepts that best capture the particular context.

WSTO is a general ontology, that can be easily extended to include different trust approaches, such as the ones that make use of QoS statements, or reputation based mechanisms.

10. References

[1] Almendra, V., and Schwabe, D. (2006). Trust Policies for Semantic Web Repositories. In proceedings of 2nd International Semantic Web Policy Workshop (SWPW'06), at the 5th International Semantic Web Conference (ISWC), Athens, GA, USA, Nov. 5-9, 2006.

[2] Anderson, S., et al. (2004). Web Services Trust Language (WS-Trust), version 1.1. May 2004. http://msdn.microsoft.com/ws/2004/04/ws-trust/.

[3] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(4):34-43, (2001).

[4] Cabral, L., Domingue, J., Galizia, S., Gugliotta, A., Norton, B., Tanasescu, V., Pedrinaci, C. (2006). IRS-III: A Broker for Semantic Web Services based Applications. In Proceedings of the 5th International Semantic Web Conference (ISWC2006), Athens, USA, November 2006.

[5] Fensel, D., Benjamins, V. R., Motta, E. and Wielinga, B. J. (1999a). A Framework for knowledge system reuse. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-99). Stockholm, Sweden, July 31 – August 5, 1999.

[6] Fensel, D. and Bussler, C. (2002) The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications 1(2): 113-137.

[7] Fensel, D., Lausen, H., Polleres, A., De Bruijn, J., Stollberg, M., Roman, D., Domingue, J. (2006). Enabling Semantic Web Services: Web Service Modeling Ontology. Springer, 2006.

[8] Fensel, D., Motta, E., Benjamins, V. R., Decker, S., Gaspari, M., Groenboom, R., Grosso, W., Musen, M., Plaza E., Schreiber, G., Studer, R. and Wielinga, B. (1999b). The Unified Problem-solving Method Development Language UPML. IBROW3 Project (IST-1999-19005) Deliverable 1.1.

[9] Galizia, S. (2006). WSTO: A Classification-Based Ontology for Managing Trust in Semantic Web Services, in Proceedings of 3th International Semantic Web Conference (ESWC 2006), Budva, Montenegro, June 11-14 2006.

[10] Golbeck, J. and Hendler, J. (2006). Inferring trust relationships in web-based social net-works. ACM Transactions on Internet Technology, 2006.

[11] Maximilien, E. M., Singh, M. P. (2004). Toward Autonomic Web Services Trust and Selection. In Proceedings of 2nd International Conference on Service Oriented Computing (IC-SOC 2004), New York, November 2004.

[12] Motta E. (1999). Reusable Components for Knowledge Models: Principles and Case Studies in Parametric Design Problem Solving. IOS Press.

[13] Motta, E., Lu, W. (2000). A Library of Components for Classification Problem Solving. In Proceedings of PKAW 2000 - The 2000 Pacific Rim Knowledge Acquisition, Workshop, Sydney, Australia, December 11-13, 2000.

[14] Nejdl, W., Olmedilla, D. Winslett, M.(2004). PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web. Workshop on Secure Data Management in a Connected World (SDM'04), in conjunction with 30th International Conference on Very Large Data Bases, Aug.-Sep. 2004, Toronto, Canada.

[15] Olmedilla, D., Lara, R., Polleres, A., Lausen, H. (2004). Trust Negotiation for Semantic Web Services. 1st International Workshop on Semantic Web Services and Web Process Composition in conjunction with the 2004 IEEE International Conference on Web Services, July 2004, San Diego, California, USA.

[16] OWL-S working group. (2006). OWL-S: Semantic Markup for Web Services. OWL-S 1.2 Pre-Release. (Available at http://www.ai.sri.com/daml/services/owl-s/1.2/).

[17] Sepandar D., K., Schlosser, M. T., Garcia-Molina, H. (2003). The EigenTrust Algorithm for Reputation Management in P2P Networks. In Proceedings of the Twelfth International World Wide Web Conference. Budapest, Hungary, 20-24 May 2003.

[18] Stefik M. (1995). Introduction to Knowledge Systems. Morgan Kaufmann, San Francisco, CA, USA.

[19] Uszok, A., Bradshaw, J. M., Johnson, M., Jeffers, R., Tate, A. Dalton, J., Aitken, J. S. (2004). KAoS Policy Management for Semantic Web Services. IEEE Intelligent Systems 19(4): 32-41 (2004).

[20]Vu, L.-H., Hauswirth, M., Aberer, K. (2005). QoS-based Service Selection and Ranking with Trust and Reputation Management 2005 International Conference on Cooperative Information Systems (CoopIS), Agia Napa, Cyprus, 31 Oct - 4 Nov 2005.

[21] W3C (2004). Web Services Architecture. W3C Working Draft 11 February 2004 (Available at http://www.w3.org/TR/ws-arch/).

[22] Wielinga, B. J., Akkermans, J.K. and Schreiber, G. (1998). A Competence Theory Approach to Problem Solving Method Construction, International Journal of Human-Computer Studies, 49, pp. 315-338.