

A Framework for Selecting Trusted Semantic Web Services*

Stefania Galizia, Alessio Gugliotta

Knowledge Media Institute
The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK
(S.Galizia, A.Gugliotta)@open.ac.uk

Abstract. Trusted semantic Web services might play a key role in the Future Internet. In this paper, we describe WSTO our comprehensive trust based framework supporting the selection and invocation of semantic Web services. Our framework combines the Web Services Modelling Ontology (WSMO) with a classification framework developed in the IBROW project. Our approach is generic enough to be able to account for a wide variety of trust models including those based on security, policy and end-user recommendations. Expanding on earlier work within this paper, we describe the model in detail.

Keywords: Semantic Web services, Trust, Classification

1 Introduction

In our vision, the Future Internet will provide an effective and trustworthy environment where people and organizations can easily accomplish their daily tasks, by exploiting thousands of independent computing devices and services. We particularly advocate that the notion of trust will play a fundamental role in the actualization of such a vision. In fact, users accessing Internet will not know a priori most of the available services and computing devices, and thus appropriate mechanisms assuring certain level of trustworthiness in service provision have to be adopted.

Semantic Web services (SWS) technology represents a promising approach for providing users with the most appropriate services. On the basis of formal descriptions with well-defined semantics, SWS aim indeed at automating the selection, composition and mediation of available services to achieve a given goal.

In the literature, a number of approaches enhancing SWS with trusts can be found. Many of them [11, 15] adopt policies involving security statements, such as confidentiality, authorization, and authentication. W3C Web service architecture recommendations [18] also propose security statements to address trust, even though the way to disclose security policies is still not completely clear.

Some policy-based models rely on a Trusted Third Party (TTP) [21]. A TTP works as a repository of service descriptions and policies as well as an external matchmaker, which evaluates service trustworthiness according to given algorithms. Further mod-

* This work was supported by LHDL (Living Human Digital Library) project (FP6 – 026932).

els of trust are based on reputation [7, 16]; they make use of ratings coming from other agents or a central engine and based on heuristic evaluations of Quality of Service (QoS). Some of the models quoted above are very complex and elaborate, but no one is flexible enough to deal with any trust understanding. Moreover, the most common approaches for describing semantic Web services, such as WSMO [3] or OWL-S [12], do not provide exhaustive means for trust annotation.

We believe that the main issue of representing trust lies in its context-based nature – the same user may have different trust preferences in different contexts.

In this paper, we propose a general approach for managing trust in SWS, which accommodates all of the possible approaches described above. We have developed an ontology – Web Services Trust-management Ontology (WSTO) - which is able to represent trust specifications within a SWS-based interaction context. In WSTO, we characterize trust-based Web service selection as a classification problem – i.e., given a set of user and Web service trust requirements and guarantees, our goal is to identify the class of Web services that match the trust statements of involved interaction participants, according to an established classification criterion. To accomplish this, we have based WSTO on a general purpose classification library, and adopted WSMO as our reference model for describing semantic Web services. An earlier version of WSTO was described in a previous paper [5]. Whereas in [5] we outlined the idea of characterizing trust as a classification process, in the present work we propose a more complex framework, which is able to accommodate monitoring the history of Web services behavior and a reputation module.

It is worth to highlight that the main contribution of our approach lies in its generality; i.e. our model enables participants to represent their needs and guarantees with a high level of flexibility. For example, a user may trust a WS with a highly rated security certificate whenever she has to provide her credit card details. Conversely, in other environments, the opinion of other users that have already experienced an interaction with a given class of Web services is crucial. In this case, reputation evaluation is a priority in the definition of trust requirements. Moreover, different users may privilege distinct trust parameters in the same context; their priority may depend on their personal preferences. In contrast with other approaches, WSTO allows the interacting participants to model and utilize their own conception of trust – both service requester and provider are able to explicitly model their trust guarantees and requirements. This assumption enables our framework to take context into account and alleviates a number of current issues of selecting a “trusted” Web service in a dynamic environment, such as the Future Internet. In the rest of the paper, we outline the basis of our approach (Section 2), describe the classification library that we use (Section 3), and provide details of our methodology in Sections 4. Section 5 contains a comparison with related work. Finally, Section 6 concludes the paper and outlines our future work.

2 Approach Overview

The Web Service Trust-management Ontology (WSTO) is a novel approach for managing trust in semantic Web services. In our model, user preferences are the main

elements on which Web service selection depends. Essentially, the user can decide which variables should be taken into consideration, in order to determine which class of Web services is trusted.

WSTO is founded on two ontologies: WSMO [3] and a classification task ontology developed within the European project IBROW [9]. WSMO defines our basic vision of Semantic Web Services and their ontological specification; the classification ontology provides the overall methodology that we have adopted for managing trust. Essentially, we embed trust-based SWS selection invocation into a classification framework.

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.

For our purposes, we classify Web services according to both the user and Web services trust requirements and guarantees. In WSTO, both user and Web service disclose their trust guarantees by means of observables, displayed as feature-value pairs. Additional guarantees concerning historical and reputation evaluations can be provided by a behaviour-monitoring engine and a reputation module, respectively. Conversely, participant trust requirements express conditions on the values that the feature can take. Given observables and conditions, a classification criterion is necessary to classify Web services and find the appropriate class that addresses both user and Web service requirements and guarantees.

The execution environment we use is IRS-III [2], a platform for developing and executing semantic Web services. 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. A key feature of IRS-III is that Web service invocation is capability driven. An IRS-III user simply requires a goal she wishes to achieve, and the IRS-III broker locates an appropriate Web service semantic description and then invokes the underlying deployed Web service.

Web service selection in IRS-III - up to now restricted to a capability-based model - has become trust-based thanks to WSTO. Given several Web services, semantically described in IRS-III, all with the same capability, but different trust guarantees, the class of Web services selected will be the one that matches closest with the user trust requirements.

There are reasons for using IRS-III as our execution environment. Firstly, this framework has been designed and built within our institution. Secondly, the Web Services Modelling Ontology (WSMO), our underlying SWS ontological model, has been incorporated and extended as the core IRS-III epistemological framework. Moreover, the classification library we use and extend is represented in OCML [8], the ontological representation language adopted by IRS-III.

3 A Classification library

The classification framework that we use and extend for our work is a library of generic, reusable components whose purpose is to support the specification of classification problem solvers. The basic structure is the UPML framework [4], on which

WSMO is based. The library has been specified in the OCML modelling language [8], and implemented in IRS-III [2].

Within the classification framework, 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 as a feature which can be directly observed, or derived by inference. As is common when characterizing classification problems - see, e.g., [19], 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. Thus, we can say that an observable (f, v) matches a feature specification (f, c) if v satisfies the condition c .

As we have seen, generally speaking, classification can be characterized as the problem of explaining observables in terms of pre-defined solutions. To assess the explanation power of a solution with respect to a set of observables we need to match the specification of the observables with that of a solution. Given a solution, $sol: ((f_{sol1}, c_1), \dots, (f_{solm}, c_m))$, and a set of observables, $obs: ((f_{ob1}, v_1), \dots, (f_{obn}, v_n))$, four cases are possible when trying to match them:

- A feature, say f_j , is inconsistent if $(f_j, v_j) \in obs$, $(f_j, c_j) \in sol$ and v_j does not satisfy c_j ;
- A feature, say f_j , is explained if $(f_j, v_j) \in obs$, $(f_j, c_j) \in sol$ and v_j satisfies c_j ;
- A feature, say f_j , is unexplained if $(f_j, v_j) \in obs$ but f_j is not a feature of sol ;
- A feature, say f_j , is missing if $(f_j, c_j) \in sol$ but f_j is not a feature of obs .

Given these four cases, it is possible to envisage different solution criteria. For instance, we may accept any solution, which explains some data and is not inconsistent with any data. This criterion is called positive coverage [14]. Alternatively, we may require a complete coverage - i.e., a solution is acceptable if and only if it explains all data and is not inconsistent with any data. 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 solutions and a set of observables. By default, our library provides a match criterion based on the aforementioned model. In short words, our analysis characterizes classification tasks in terms of the following concepts: observables, solutions, match criteria, and solution criteria. Please refer to [9] for more technical details concerning the classification library we adopt.

4 Embedding SWS Trust Management in a Classification Problem

Figure 1 summarizes how our classification framework is applied within an SWS context. We have characterized trust analysis as a classification process, within which valid solutions are those Web services that match with user requirements, given clas-

sification criteria. Candidate solutions - i.e. user requirements - are defined as pairs (feature, condition). In the example showed in Figure 1, they express conditions on QoS, data-freshness and execution-time, and reliability.

The observables are pairs (feature, value) representing Web service guarantees. The possible available guarantees associated with a WS concern data-freshness, execution-time, confidentiality, and WS reliability.

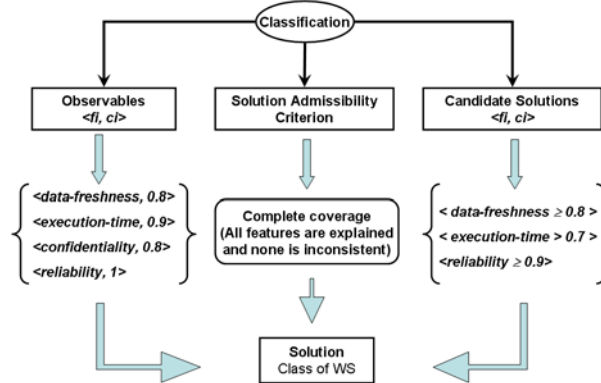


Figure 1 Classifying Web Services

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 users’ trust requirements, given a set of WS trust guarantees. Trust for us is a binary evaluation of trustworthiness: “trust” or “distrust”. Whenever conditions for trustworthiness are established, the interaction between participants occurs; otherwise not. Trust as perceived by the user *u*, can be either strong or weak. It is strong when is adopted complete coverage for classifying Web services. In turn, when the criterion selected is positive coverage, trust is regarded as weak.

Participant profiles. In our ontology (Figure 2), the classes *user*, *ws* and *goal* are key concepts. Both *user* and *ws* are subclasses of *participant*, where *user* denotes the service requester, *ws* is the service provider. In principle, a participant is any actor involved within the interaction. Nevertheless, we distinguish between *user* and *ws* for emphasizing the different perspectives that requester and provider have. Following the basic WSMO notions a *goal* represents the service requester’s desire or intention. The *user* usually expresses different trust requirements in achieving different goals. For example, she can be interested in data accuracy when retrieving timetable information, and security issues when disclosing her bank accounts. On the other hand, the *ws* aims to provide a set of trustworthy statements, in order to reassure the requester as well as to appear as attractive as possible.

The participants are associated with trust profiles, represented in the ontology by the class *trust-participant-profile*. A profile is composed of a set of trust requirements and guarantees. *Trust-guarantees* are observables (pairs of feature and corresponding value (*f*, *v*)), while *trust-requirements* are candidate solutions (pairs of feature and condition (*f*, *c*)). We distinguish three logical elements in trust requirements: (i) a set of candidate solutions for expressing conditions on guarantees promised by the rele-

vant parties; (ii) a candidate solution for requesting their reliability; and, (iii) a candidate solution for requesting their reputation evaluation. In a participant profile, the three elements are optional; choice depends strictly on the participant preferences in matter of trust. In turn, the participant trust-guarantees have three components: (i) a set of observables for representing the promised trust guarantees; (ii) an observable corresponding to the evaluation of the participant reliability; and, (iii) an observable for representing the reputation level of the participant.

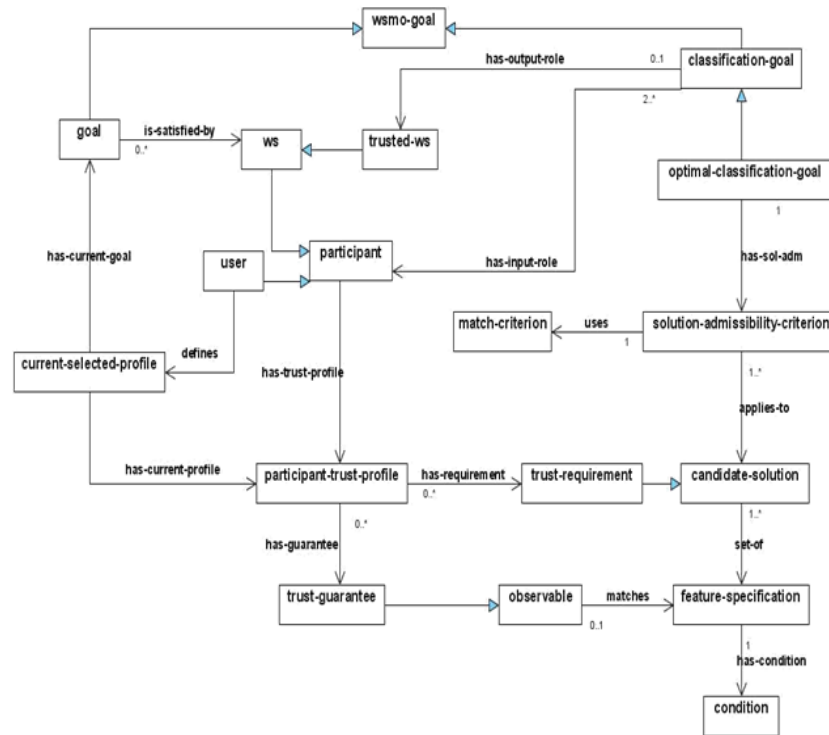


Figure 2 Trust as Classification: Partial Representation of WSTO

The example displayed in Figure 1 does not include reputation, as optional statement; in turn, the graphical representation in Figure 2 does not show the above-described logical groups of both requirements and guaranties, for improving readability. Whereas the promised trust guarantees are a set of promised values stated by the participant - such as (*execution-time*, 0.9) and (*data-freshness*, 0.8) - reliability and reputation guarantees are computed on-the-fly within IRS-III.

As mentioned earlier, a participant profile is composed of requirements as well as guaranties. For example, a Web service may expose high *data-freshness* and strong *confidentiality* as guaranties. Moreover, the same Web service may define security requirements, such as conditions under which a service requester can access it. However, in the rest of the paper we assume that requirements are only part of the user profile, while a WS profile is composed solely by guaranties. The reason for this choice lies in the fact that we classify objects according to observables we know, and our aim is to find the class of Web services that best explains a set of guaranties - i.e.

observables – according to the user requirements. This assumption simplifies the classification process, makes it user centric - since we consider only user trust requirements - and it is not restrictive. Whenever both user and Web service trust profiles are composed of guarantees and requirements, we apply the classification process twice. The second classification process verifies the user guarantees with the class of those WS considered trusted during the first classification process.

Listing 1 shows two user profiles, represented in OCML, within IRS-III - *trust-profile-USER1-A* and *trust-profile-USER1-B*. Both the profiles are associated with the user *USER1* and only her trust requirements are specified. The class *requirement-USER1-A* concerns security needs, in particular, the encryption algorithms used by the WS and the certification authority, which issued the WS security certificate.

```
(def-class trust-profile-USER1-A (trust-profile)
  ((has-trust-guarantee :type guarantee-USER1-A)
   (has-trust-requirement :type requirement-USER1-A)))

(def-class trust-profile-USER1-B (trust-profile)
  ((has-trust-guarantee :type guarantee-USER1-B)
   (has-trust-requirement :type requirement-USER1-B)))

(def-class requirement-USER1-A (trust-requirement) ?req
  ((encryption-algorithm :type algorithm)
   (certification-authority :type ca)
   :iff-def (and (exists ?x (encryption-algorithm ?req ?x)
                  (> ?x 0.9))
                 (exists ?x (certification-authority ?req ?x)
                  (> ?x 0.8))))

(def-class requirement-USER1-B (trust-requirement)
  ((datafreshness :value high)
   (execution-time :value low)
   (reputation :value high)))
```

Listing 1 User Profiles

The class *requirement-USER1-B* expresses the user needs around general WS performance and WS reputation. Note that whereas the former requirement class expresses conditions through real normalized values, the latter performs qualitative constraints. Representing user requirements in a qualitative way facilitates end-user comprehension. In IRS-III, implemented heuristics act as translators from a quantitative to a qualitative representation. The user can adopt the heuristics, or define new ones, sharing her expertise and knowledge with other users. Optionally, she can express her requirements in a precise, quantitative way, by specifying the exact values expected from Web service guarantees.

Profile Selection. The diagram in Figure 2 shows that, as participants, both WS and user have a trust profile. However, whereas a WS discloses, at design time, only one profile, the user publishes in IRS-III one or more profiles, which may be adapted from previously published profiles. Indeed, a user can associate distinct profiles to different goals by defining multiple definitions of *current-selected-profile*. This choice allows our model to include context.

Classification criteria. The classification match criterion we apply is the one described in Section 3, although other classification criteria can be represented in WSTO. As solution admissibility criteria, we can apply *complete coverage* and *positive coverage*. The former demands that all requirements of the interaction have to be satisfied; the latter accepts that some requirements are fulfilled and no inconsistencies

exist. Our classification library implements two different classification methods: *single-solution-classification*, and *optimal-classification*. The former implements a hill climbing algorithm with backtracking to find a suitable solution, the latter executes an exhaustive search for an optimal solution. In turn, the two classification tasks implemented in IRS-III are: *optimal-classification-task* and *single-solution-classification-task*, respectively solved by the two methods described above, according to the UPML framework [4].

We make use of the *optimal-classification-task* and redefine it as WSMO goal[†], *optimal-classification-goal* (see Figure 2), whose *participant-profiles* and *trusted-ws* represent the pre-conditions and post-conditions of the goal, respectively.

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 [6], while some recent efforts in the last few years concern service-oriented views of trust [1]. However, few approaches provide methodologies for managing trust in a SWS context, 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 [16, 17], who use WSMX [20] as an execution environment, is closely related to the work reported here. Vu et al. [16, 17] 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, the methodology we propose is less accurate in terms of 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. [11] propose a methodology for trust negotiation in SWS. They employ PeerTrust [10], a policy and trust negotiation language, for establishing if trust exists between a service requester and provider. The main issue that distinguishes their methodology from ours is that they assume that trust is solely based on policy. They do not propose any mechanism for managing reputation or monitoring past service behaviour, as we do. Similar to our approach, though, 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

[†] WSMO goals can be seen as an evolution of UPML tasks.

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 that are less closely related to ours, such as KAoS [15]. Within KAoS a set of platform-independent services which enable the definition of policies ensuring the adequate predictability and controllability of both agents and traditional distributed systems is proposed. Even though they present a dynamic framework, and recognize trust management as a challenge for policy management, the framework is not specifically tailored to trust management in an SWS context.

6 Conclusions and Future Work

In this paper, we have presented a framework for managing trust in SWS and have envisaged Web service selection and invocation as a classification problem, where the solution takes the form of a class of Web services matching participating trust profiles. Embodied within the trust profiles are the participant priorities with respect to trust, which can be related to reputation, credentials, or actual monitored behavior. Our conception of trust is described through a binary measure: whenever participant trust profiles match, a trusted interaction can occur, otherwise trusted interaction is deemed to not be feasible.

Trust has different meanings within different contexts: trust can be based on service ability or on reliability. In other contexts, trust can be related to reputation or delegated to TTP evaluations. The main contribution of our approach is to provide a framework that enables a comprehensive range of trust models to be captured.

In order to showcase the benefits of our approach we have deployed two prototypes available at public URLs: <http://lhdl.open.ac.uk:8080/trusted-travel/untrusted-query> and <http://lhdl.open.ac.uk:8080/trusted-travel/trusted-query>. The former implements a virtual travel agent based on the standard IRS-III goal invocation method. The latter introduces a trusted goal invocation, according to our trust framework, and taking into account three different user profiles.

Future work will extend our implementation to incorporate a comprehensive management suite for WS reputation and reliability. Additionally, we also plan to import a range of sophisticated reputation algorithms, and to improve the monitoring component.

References

1. 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/>.
2. 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, Athens, USA, November, 2006.
3. 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.

4. 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. (1999). The Unified Problem-solving Method Development Language UPML. IBROW3 Project (IST-1999-19005) Deliverable 1.1.
5. 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.
6. Golbeck, J. and Hendler, J. (2006). Inferring trust relationships in web-based social networks. ACM Transactions on Internet Technology, 2006.
7. 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.
8. Motta E. (1999). Reusable Components for Knowledge Models: Principles and Case Studies in Parametric Design Problem Solving. IOS Press.
9. 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.
10. Nejdil, 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.
11. 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.
12. 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/dam1/services/owl-s/1.2/>).
13. Sierra, C., Debenham, J.K. (2005). An Information-Based model for Trust, in proceedings Fourth International Conference on Autonomous Agents and Multi Agent Systems AAMAS 2005, Utrecht, Netherlands, 25-29 July 2005, pp497-504.
14. Stefik M. (1995). Introduction to Knowledge Systems. Morgan Kaufmann, San Francisco, CA, USA.
15. 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).
16. Vu, L. H., Hauswirth, M, Aberer, K. (2005). QoS-based Service Selection and Ranking with Trust and Reputation Management, 13th International Conference on Cooperative Information Systems (CoopIS 2005), Agia Napa, Cyprus, Oct 31 - Nov 4, 2005.
17. Vu L., H., Hauswirth, M., Porto, F. Aberer, K. (2006). A Search Engine for QoS-enabled Discovery of Semantic Web Services. International Journal of Business Process Integration and Management (IJBPIIM), 2006.
18. W3C (2004). Web Services Architecture. W3C Working Draft 11 February 2004 (Available at <http://www.w3.org/TR/ws-arch/>).
19. 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.
20. WSMX working group. (2005). Overview and Scope of WSMX, <http://www.wsmo.org/TR/d13/d13.0/v0.2/>.
21. Zhengping, W., and Weaver, A. C. (2006). Using Web Service Enhancements to Bridge Business Trust Relationships, Fourth International Conference on Privacy, Security, and Trust (PST'06), University of Toronto, Institute of Technology, Markham, Ontario, Canada, October 30-November 1, 2006.