WSTO: A Classification-Based Ontology for Managing Trust in Semantic Web Services

Stefania Galizia

Knowledge Media Institute & Centre for Research in Computing
The Open University, Milton Keynes, UK
S.Galizia@open.ac.uk

Abstract. The aim of this paper is to provide a general ontology that allows the specification of trust requirements in the Semantic Web Services environment. Both client and Web Service can semantically describe their trust policies in two directions: first, each can expose their own guarantees to the environment, such as, security certification, execution parameters etc.; secondly, each can declare their trust preferences about other communication partners, by selecting (or creating) '*trust match criteria*'. A reasoning module can evaluate trust promises and chosen criteria, in order to select a set of Web Services that fit with all trust requirements. We see the trust-based selection problem of Semantic Web Services as a classification task. The class of selected Semantic Web Services (SWSs) will represent the set of all SWSs that fit both client and Web Service exposed trust requirements. We strongly believe that trust perception changes in different contexts, and strictly depends on the goal that the requester would like to achieve. For this reason, in our ontology we emphasize first class entities "goal", "Web Service" *and* "user", and the relations occurring among them. Our approach implies a centralized trust-based broker, *i.e.* an agent able to reason on trust requirements and to mediate between goal and Web Service semantic descriptions. We adopt IRS-III as our prototypical trust-based broker.

1 Introduction

With the widespread proliferation of Web Services, trustworthiness will become a determining factor of any given service's success. Conversely, trust-based automatic discovery and selection will become a significant requirement from a requester's point of view.

In the literature, the notion of "trust" is defined in different ways according to the application domain. We draw on two major approaches: trust based on *ability* and trust based on *reliability*. The former enacts the requirements based on quality of service profiles (data accuracy and precision, timeliness, *etc.*…); for instance, a requester may trust more a service that takes acceptable time to perform a given task . The latter considers mainly the service credibility, which can be measured by a Trusted Third Party.

In e-commerce, security services – such as authentication, data integrity, confidentiality *etc.* – are deployed in order to realize the reliability-based aspect of trust. Secu-

rity services are usually implemented in terms of security mechanisms based on Trusted Third Party (TTP) concepts and Public Key Cryptography.

There are other approaches concerned also with reliability-based trust. In some environments, it seems appropriate to calculate the trustworthiness by reasoning only on security issues. At the other extreme, pure reputation-based algorithms have been implemented especially in those fields where all involved parties can express their opinions, as the social networks.

We concentrate on aspects of trust that we claim are fundamental in the Semantic Web Service context. In the open dynamic environment where the Semantic Web Services lie, trust-based discovery and selection are crucial issues in order to avoid invocation of malicious or unreliable services.

Until now, there are no defined protocols by which Semantic Web Services may expose their trust characteristics. Web Service technology provides only syntactic statements. The interface definition language WSDL specifies only the syntactic signature for a Web Service, but does not specify any semantics or non-functional characteristics.

Adding semantic descriptions to services should allow also reliability specifications and support different notions of trust from both requester and provider perspectives.

Our first assumption is that different users have different demands on trust parameters. Moreover, we believe that in different contexts trust assumes different meanings. Essentially the trust judgement of a service requester will strictly depend on the goal she intends to achieve.

In this paper, we provide a framework that fulfils given requirements for the description of trust properties of Semantic Web Services and enables their selection based on these properties. We represent the notion of trust via an ontology, named *WSTO* (Web Service Trust-management Ontology), through which both requester and Web Service provider can instantiate their individual trust policies. Then, a reasoning module will activate Web Service selection taking into account trust-related properties.

We characterise trust analysis as a classification process, within which valid solutions are those Web Services that match given classification criteria. Consequently, we have found beneficial and enlightening to apply an existing classification ontology [13] to this scheme, creating constructs that adapt the framework for our specific purposes. On the other hand, we preferred to keep our model as general as possible in order to accommodate this extension to a variety of requesters' preferences and requirements for trust-related matters.

We chose WSMO [16] as underlying ontology to state the basic concepts of Semantic Web Services. One of the common principles to our ontology and WSMO is the ontological role separation of client, Web Service and goal.

Our approach implies the concept of delegation to a centralized trusted evaluator, in order to reason about the criteria expressed by the participants. We adopt IRS-III [2] as our prototypical trust-based broker. IRS-III is a framework and implemented platform, which acts as a broker mediating between the goals of a user or client and available deployed Web Services. Moreover, IRS-III uses WSMO as its basic ontology and follows the WSMO design principles.

The paper is organized as follows. First, we provide in Section 2, a rough description of trust assumptions in different contexts. Then we describe in Section 3 our approach via a brief presentation of the underlying classification ontology, and a detailed explanation of WSTO. In Section 4 we introduce the execution layer of our approach, by referring to future implementation within IRS-III. Finally, in Section 5 we conclude by summarizing WSTO benefits and proposing future work.

2 Trust Considerations

The meaning of trust is very difficult to catch. Trust is a social phenomenon inherent to human beings. In that context trust is:

- A means for understanding and adapting to the complexity of the environment;
- A means of providing robustness to independent agents;
- A useful judgement in the light of experience of the behaviour of others
- Applicable to artificial agents.

Trust in an artificial agent is a means of providing an additional tool for the consideration of other agents and the environment in which it exists. The provision of explicit trust into an agent is still rather a research subject. The current approaches to trust are more about how to assume trust (to establish a replacement for trust).

The most of the systems that are at present being designed *assume* trust, i.e., an agent entering into communication with an other agent (believes absolutely or to a certain degree) that good (promised or intended) things will happen. In this context, security is about how to ensure that bad (not intended) things do not happen.

The different existing approaches to trust are about how the trust assumption is made and its enforcement ensured. The most popular approaches are: i) Reputation-based; ii) Trusted Third Party; iii) Contract-based.

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.

Web-based social network models have been one of the first research fields where trust, in terms of reliability, has become a central issue. Every actor in a social network can express his opinion on another one, by means of an available vocabulary. Several algorithms for trust propagation and different metrics have been defined in this field. Some systems use discrete values, for instance “low”, “medium” and “high”, to express the trustworthiness, others make use of real-valued measures, usually expressed in the interval [0,1], especially in those algorithms requiring high precision. At the other extreme, some networks make use of binary rating, either 1 for trustworthy neighbours, or 0 for who are not trustworthy.

Two main trust properties, modelled in many network systems, are transitivity and asymmetry. In general, trust is not symmetric; one actor can trusts another one, belonging to same network, and the latter can no trust the former. The trust can be transitive, but many different meanings, not properly mathematical, have been associated to transitivity [7].

Many new projects based on social networks have arisen in the last few years. The most famous is perhaps Friend-Of-A-Friend (FOAF), a Semantic Web based social network with many users distributed on the Web [6]. One application of FOAF concerns the creation of a trust module is based on users' rating of each other's trustworthiness and expressed on a discrete scale between 1 and 10 [7].

In the last few years, trust has become of crucial importance within peer-to-peer networks. A peer-to-peer (P2P) computer network is a network that relies on the computing power and bandwidth of the participants in the network rather than concentrating it in a relatively few servers. In order to use P2P networks in a useful setting, it is extremely important to provide security and to prevent unwanted elements from participating. Several algorithms are available for peer trust rating; most of them are based on security considerations (e.g., public or private key cryptography) and on reputation [14, 15, 18]. The basic idea is to assign to each peer a trust rating based on its credentials, in case provided by trusted third parties, such as certification authorities, and on its performance in the overlay network and to store it at a suitable repository. The existing trust algorithms consider different aspects, most of them monitor the peer behaviour on the time; other ones emphasize the concept of cooperation. In [18], for instance, the authors present an algorithm where all peers in the network cooperate to compute and store the global trust vector. In general, in peer-to-peer systems, the information propagation and the reputation management are central issues of trust rating.

On the other hand, to evaluate the Semantic Web Services trustworthiness, several different approaches are already proposed. Existing technologies for Web Services only provide descriptions at the syntactic level, making it difficult for requesters and providers to interpret or represent nontrivial statements. Semantic descriptions of Web Services are, in fact, necessary in order to enable their automatic discovery, composition and execution across heterogeneous users and domains. In Semantic Web Service contexts, when the user expresses the goal she would like to achieve, the actual Web Service that matches the goal is dynamically discovered and selected, and so its features are not completely known *a priori*. In this environment, semantic annotation of trust features becomes a considerable parameter during the discovery phase. Most of existing approaches inherit methodologies from the peer-to-peer networks [11, 15], as Semantic Web Services provide P2P interaction between services. Several approaches rely on an external matchmaker that works as repository of service description and policies [8] and calculates the service trustworthiness according with given algorithms. Trust evaluation algorithms for Semantic Web Services consider especially security issues, such as confidentiality, authorization, authentication, as rating statements [8, 9, 10, 11]. Even W3C Web Service architecture [22] recommendations consider trust policies inside security consideration, but the way to disclose their security policies is still not clear. UDDI does not refer to security features for Web Services.

In Semantic Web Services context, some trust algorithms are more generically Quality of service based [1, 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, but even security is a part of QoS.

We deem that the key to enable a trust based discovery for Semantic Web Services lies in a common ontological representation, where Web Service and client perform

their trust requirements. Some QoS taxonomy [17] or service policy ontology [9] already exist, nevertheless an exploration of how to provide a common means for runtime monitoring the services trustworthiness is only beginning.

3 Our Approach

In this paper, we present an ontology, WSTO (Web Service Trust-management Ontology), that enables both client and Web Service to express their trust requirements in a Semantic Web Services environment. Our starting point is to identify the goal that the requester wishes to achieve, and to describe the trust requirements associated with this goal.

We have shown that in different contexts trust assumes different meanings. On the one hand a requester tends to trust a travel agency that proposes to fly by airlines deemed safe, on the other hand, a banking service is trusted as it provides confidentiality or authentication certifications that promise data privacy.

The more fruitful way to express the actual trust requirements is providing a shared and common framework that allows both requester and provider to express their policies, but even enables them to extend the framework according to their needs. Our ontology pursues this purpose.

We adopt WSMO [16] as basic vision that provides ontological specifications for the core elements of Semantic Web Services.

The primary concepts in our ontology are “web-service”, “user” and “goal”. The ontologically specified concept of “user” is not stated in WSMO. This specification facilitates the description of the individual policies by a service requester. The user instantiates a goal during the goal-based invocation process. The expected outcome is the selection of a class of Web Service that fits the goal and trust requirements of all transaction participants. It is worth to emphasize that, while the goal specified by the user is an instance of the class “goal” represented in WSMO, the selected Web Services are the instances of the class of Web Services that meet all policies declared by trading partners involved in the transaction. For this reason, we have characterised the analysis of trust-related properties and requirements as a classification process, within which valid solutions are those Web Services that match given criteria.

Selecting one, or a set of Web Services that match a given criterion corresponds to the task of finding the solutions in a classification problem. The solution will be the class of Web Services that fit criteria established by requester and provider. The match criteria represent the trust requirements. This vision, intentionally general, allows also natural application to other fields, not strictly related with trust. In our framework, in fact, participants (user and Web Services) can easily express any kind of policies, in particular, their trust policies.

WSTO builds on the classification library, created within the IBROW project [5, 13], then it makes use of the classification mechanisms already defined in that task. We opportunely extend and adapt it to the Semantic Web Services field, emphasizing the role of service selection as an important part of goal invocation.

In the following of this section, we expose the main features of the underlying classification task and then we describe in details WSTO.

3.1 Classification

The classification problem is an important issue in several fields [19]. For example, identifying a class of symptoms is crucial in the investigation of diseases; or, classifying goals and requirements is the starting point of a planning process. In general, reasoning about classes is simpler, especially in the presence of a large set of instances.

In order to find the proper class for a set of objects, it is necessary for an agent to reason about differences among a given set of features. For instance, we can classify living beings as separate classes *plants* and *animals* – and continue to further subclassify the animals as *carnivores* and *herbivores* – by identifying different observable characteristics. This problem can be expressed in terms of search within a solution space, by applying a criterion over a given set of facts.

The classification ontology we extended is a 'task ontology' [3, 12, 13], which specifies the general classification problem. The classification library, as a whole, is very extensive, being composed by a huge number of classes, relations and functions; it also provides heuristic evaluations and refinement methods. We extend only a subset of the classification task, useful for our trust requirements. Figure 1 illustrates the classification framework by means of a UML Class Diagram; the large open-headed arrows relate classes in *is-a* relations, the simple arrows represent normal relations between classes.

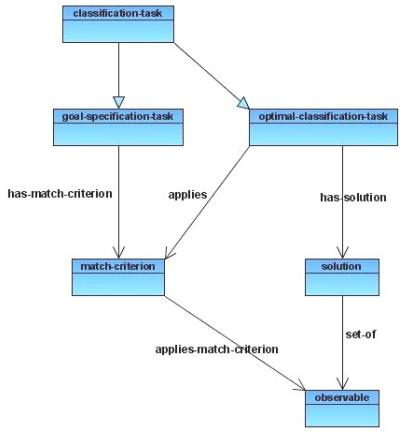

Figure 1. The classification task ontology

The `classification-task` class is a subclass of the general `goal-specification-task`. The `optimal-classification-task` is a reasoning module, it applies match criteria in order to derive the best solutions by evaluating the facts, stored in the class `observable`. The observables are a finite set of facts represented by pairs like *(f, v)*, where *f* are features and *v* their associated values. The

solution space is defined by a set of predefined classes (solutions) under which an unknown object may fall. The `match-criterion` specifies the methods to find a solution, according to a chosen classification task.

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 say that an observable *(f, v)* matches a feature specification *(f, c)* if *v* satisfies the condition *c*.

Several definitions of classification tasks can be provided. In some cases, only an admissible solution is required, in other cases optimal solutions may be requested. In Figure 1 we show only optimal-classification-task, which requires a solution to be optimal with respect to a given match criterion.

3.2 WSTO: Web Services Trust-based selection Ontology

WSTO is composed of two logical levels: a static layer that provides our vision of Semantic Web Services invocation scenario, and a dynamic level, composed by a reasoning module, where every requestor can specify its own trust requirements.

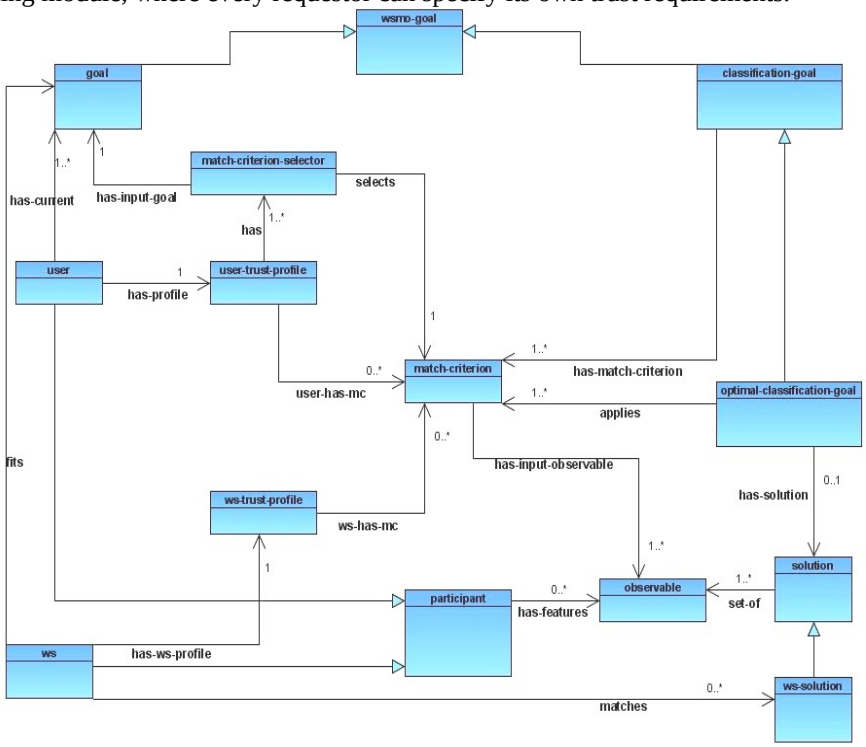

Figure 2. The Trust Ontology

The former identifies three main components during a service invocation: user, goal, ws (Web Service); the latter describes how dynamically is established a solving method to select the Web Service according with all trust requirements.

The *user* is the client, which can be a human actor or in turn another Web Service. The class *ws* represents the Web Service. A goal specifies the objectives that a client may have when consulting a Web Service, describing aspects related to user desires with respect to the requested functionality and behaviour. Our goal definition can be a WSMO goal [16].

In the WSMO vision, a goal specifies the objectives that a client may have when consulting a Web Service, describing aspects related to user desires with respect to the requested functionality and behaviour. Ontologies are used as the semantically defined terminology for goal specification. Goals model the user view in the Web Service usage process and therefore are a separate top level entity in WSMO.

As shown in figure 2, *ws* and *user* are subclasses of *participant*. We include the class *participant* as superclass of *user* and *ws*, to compactly specify common relations involving both user and Web Service entities. User and Web Service should be able to express their trust requirements and publish their own guarantees. For instance, they could expose promised execution parameters or security certifications, as non-functional properties. Several certification authorities, such as the well-known Verisign [21], may provide either requester or Web Service with security certification.

We do not intend to provide technical security consideration in this paper, nevertheless, we show how easily the participants can extend WSTO in order to disclose their security guarantees.

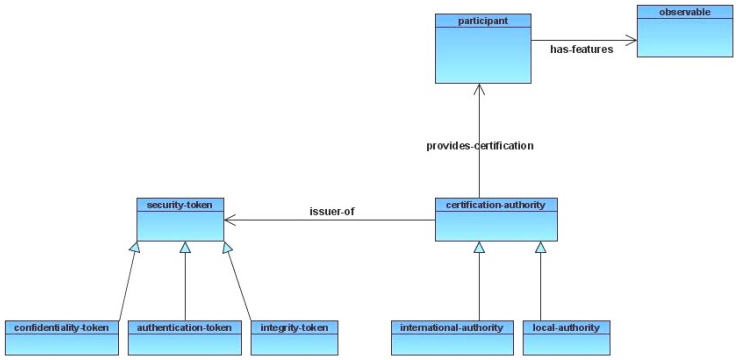

Fig. 3. Security Ontology

In the figure 3 we show a possible ontology extension. *Certification-authority* class represents an entity that provides security certification, for instance, the aforementioned Verisign. There exist different kinds of authorities, international, national, university, etc.. Usually the certificates (like the certificates exploited in the well-known SSL protocol, X509), provide different classes of security. Authentication verifies whether a potential partner in a conversation is capable of representing a person or an organization. Integrity assures that the data must be identically maintained during any operation. Confidentiality serves to keep the message secret by using encryption.

A user requesting a service on the Web usually demands authentication and encryptions services (confidentiality). Our general framework allows the participants to express their individual requirements in a flexible way.

The certification authority may provide other guarantees not mentioned in our framework (non-repudiation, legislative requirements, etc.), and, moreover, we consider explicitly the case in which security requirements could change in the future, due, for instance to legislative requirements. For this reason both actors, WS and user, are enabled to dynamically extend our ontology in accordance with their own needs. Trust preferences of a requester may also relate to Web Service execution properties. Timeliness, precision and accuracy are all judged with respect to execution data, although often represented as objective and invariant QoS properties. While security is certificated by trusted authorities, evaluation of QoS execution properties is inherently more complex. In essence, the provider usually describes its own quality of service, and the requestor selection is based on the promised parameters.

In this context, an objective third party performing selection would have to take into account the historical behaviour of the Web Service, and compare the promised QoS statements with the properties of actual executions. This mechanism will be one subject of our future work.

Security or execution parameters can be represented as *(f,v)*, pairs of features and relative values, as per *observable* in the general classification task ontology. Thus the relation has-feature, between the classes *participants* and *observable*, stores all of the considered Web Service's non-functional trust properties (see Figures 2 and 3).

A goal matches with a number of Web Service; we express the goal in terms of WSMO notion. Moreover, a classification goal specifies the general goal in the previously described classification task, and may itself be expressed as a WSMO goal.

In our scenario, the user asks for a goal and establishes its criteria to be applied in the trust-based selection. The *user-trust-profile* represents the set of criteria associated to the user requirements. All criteria are stored in the class match-criterion, derived from the classification ontology. This class is the core of the dynamic level of our framework, in the sense that a user can populate it by defining new methods according to their own particular trust requirements. A user, for instance, can state that authentication has a greater weight than confidentiality certification and she can establish furthermore the score for the type of certification authorities. Furthermore, she can designate a particular given Web Service as trusted, without relating her choice with any QoS or Security parameters; in this case, she will instantiate a new criterion in the class match-criterion.

Given a *user-profile* instance, a selector engine (*match-criterion-selector* in figure 2) will select the right criterion associated to requested goal. Only one match criterion will be executed for a given goal invocation and a given corresponding user trust profile.

On the other hand, the Web Service owns its trust policies and can decide what to disclose. The class ws-trust-profile represents trust policy of the Web Service. While the user selects both the match criterion and the goal that wishes to achieve, instead, the Web Service is associated to a goal by its capability, and it will select (or define) only the preferred criterion.

The *optimal-classification-goal* class, inherited from the general classification ontology, contains a set of problem solving methods, applied to the class *match-criterion*. Essentially, the reasoning module identifies a class of Web Services that satisfy the requested goal, according with both user and Web Service trust requirements. The solutions will essentially be a set of pairs *(f, c)*, according to the classification task, where *f* expresses the trustworthiness features and *c* the conditions established in the match criterion. For example, {(*certification-authority, verisign), (key-length,128)}* is a possible solution. In our ontology, the class *solution* represents general solutions in the classification task, but we specialize, in ws-profile, the solutions of our interest. The relation *matches* between *ws* and *ws-profile* identifies all Web Service descriptions compatible with the solutions.

4 Execution

In this section, we provide more details about WSTO dynamic layer, that is, how actually the WSTO reasoner works. For example, we outline a scenario where the user looks for a secure loan Web Service with some security certifications. We assume that there exist several services fitting with goal and user trust needs. In turn, every loan service has its trust policies. For instance, concerning financial guarantees we may specify that the user has to have a bank account, a credit card, a permanent job, *etc.*.

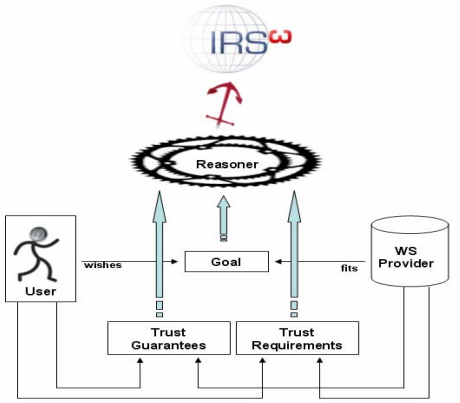

Figure 4 The anchor to IRS-III

The user could consent to show only bank account and credit card number, but withhold information regarding his job. WSTO target is to find the class of loan Web Services conformant with both user and Web Services trust policies. Figure 4 shows the basic idea: both user and WS disclose their policies at two levels, by providing trust guarantees and requirements. The trust guarantees are stored in the observables, as discussed above; the requirements are expressed in terms of match criteria. We now turn our attention to the role of the reasoner that applies the match criteria, according to each party's trust policies, in order to find the correct set of Web Services.

Our approach implies a centralized trust-based matchmaker. WSTO has to use the services of an external broker, to carry out the reasoning. In P2P and Semantic Web Services community [15, 8], several approaches adopt this centralized matchmaker idea, especially because the delegation to a trusted third party becomes essential when more than one entity is involved while taking a decision.

We believe that the centralized approach carries many advantages. First, a broker can store information and apply reputation-based algorithms that learn from involved parties' historical behaviours. The second big advantage is the simplicity of interaction, being a one-shot access of the broker.

We plan to use IRS-III (see Figure 4), as trust-based matchmaker for WSTO. In the following subsections we provide an IRS-III overview and a sketch of a possible execution example by using IRS-III.

4.1 IRS-III Overview

IRS-III is a tool and an implemented framework with the overall aim of supporting the automated or semi-automated construction of semantically enhanced systems over the Internet. The IRS uses WSMO as its basic ontology and follows the WSMO design principles [16].

IRS-III has three main classes of features, which distinguish it from other work on Semantic Web Services: Firstly, it supports *one-click publishing* of 'standard' program code. In other words, it automatically transforms programming code (currently we support Java and Lisp environments) into a Web Service, by automatically creating an appropriate wrapper. Hence, it is very easy to make existing standalone software available on the Internet, as Web Services. Secondly, by extending the WSMO goal and Web Service concepts, clients of IRS-III can directly invoke Web Services via goals - that is IRS-III supports *capability-driven* service invocation. Finally, IRS-III services are Web Service compatible – standard Web Services can be trivially published through the IRS-III.

The main components of the IRS-III architecture are the IRS-III Server, the IRS-III Publisher and the IRS-III Client, which communicate through the SOAP protocol.

IRS-III was designed for ease of use, in fact a key feature of IRS-III is that Web Service invocation is capability driven. The IRS-III Client supports this by providing a goal-centric invocation mechanism. An IRS-III user simply asks for a goal to be solved and the IRS-III broker locates an appropriate Web Service semantic description and then invokes the underlying deployed Web Service. We plan to implement WSTO in IRS-III, in order to make the client invocation, now capability-based, further trust-based. We believe that IRS-III is particularly suitable for our purpose because it is already a broker between goal and semantically described Web Service. Moreover, the classification library, to which we refer, is already implemented in IRS-III.

4.2 Execution Example in IRS-III

We now propose the outlined scenario in the beginning of this section and detail how IRS-III manages the trust-based Web Service selection.

The client is a construction company that, through IRS-III, asks for a loan service, specifying its trust policies. In turn, various loan services disclose their trust requirements and guarantees. The only data the client intends to disclose is its bank account, but only to the services that promise given security guarantees. In particular, its constraints are that Web Service provider uses encryption algorithm type *DES*, or one based on this, and that it owns an authentication certificate released by Verisign or any American certification authority. This last requirement is a weak constraint: in the case that are no available Web Services with American authentication certificates, the client considers certification from German or Italian authorities to be acceptable, in that order of preference. IRS-III maintains a user trust profile for regular clients, which contains personal preferences; and it updates their profiles every time those clients specify new trust policies.

Several Web Services with capabilities functionally fitting the goal are semantically described in IRS-III, but only a sub class of them will match with the client's trust requirements. For instance, those loan services that need the company's credit card number as guarantee are automatically excluded, because the client discloses only its bank account.

Both Web Service and client can extend WSTO. The security extension example, shown in the section 3.1 is a typical extension that can occur in this case study. In fact, in order to disclose security guarantees, it makes sense to add to WSTO classes that store the main certification authorities, or the possible security tokens, as shown in Figure 3.

The loan services populate the class observable with their trust guarantees. We consider three different loan Web Services that instantiate the following pairs:

- WS1: *(certification-authority, verisign);*
  (country-authority, united_states);
  (encryption, AES);
  (certificate-type,X.509);
- WS2: *(certification-authority, globalsign-austria);*
  (country-authority, austria);
  (encryption, 3DES);
- WS3: (*certification-authority, tc-trustCenter);*
  (country-authority, germany);
  (encryption, DES);
  (keyType, RSA);

To simplify the case study we do not instantiate any Web Service trust requirement, and assume that all those three services accept as clients' guarantees only their bank accounts, the only data the construction company wants to disclose. The client, instead, selects an available parametric match criterion, which allows the client to establish weighs for parameters, concerning encryption algorithms and certification authorities' properties.

Our client will provide the following values:

$$\begin{cases} encryption \Rightarrow (value : DES, score : 1) \\ authority \Rightarrow \ (value : veriSign, score : 0.5) \\ nationality \Rightarrow \begin{pmatrix} value : american, score : 0.5 \\ value : german, score : 0.3 \\ value : italian, score : 0.2 \end{pmatrix} \end{cases}$$

This criterion provides a trust value in the real interval [0,1] for every parameter. The score 0 means that there is no trust at all, the score 1 signifies absolute trust, the values between 0 and 1 represent the linear variation in trust. For values that are given no score, a score of 0 automatically applies. For instance, it is implicit that the client does not trust any English certification authority. The criterion returns the final computed trust measure as a real value between 0 and 1, by normalization of some composition of the scores for all provided values. The optimal classification task provides the class of best solutions, by reasoning on all match criteria selected or created. The solutions are a set of valid observables, those represent all features the Web Services must have. This set is stored in the class *ws-solution*. The actual Web Services correspond to the valid features are returned by the relation `matches`, between *ws* and *ws-solution*.

No one among the loan services considered has maximal trustworthiness, i.e. is the subject of absolute trust. In fact, the only Web Service that matches both the goal and client trust requirements is WS3. WS1 does not respect the encryption constraint; and the client does not accept the nationality of the authority that provides WS2 with security certification.

5 Conclusions

In this paper, we have presented an ontology, WSTO, that facilitates trust based invocation and selection in the Semantic Web Services environment. We have considered the trust-based selection as a classification problem. This simplifies the problem's tractability, especially in presence of a lot of instances. This is of particular relevance in our context due to the distributed and open nature of the web.
WSTO presents several important benefits that we summarize as follows:

- **Generality**. Trust has different meanings in different contexts, we differentiated trust on ability and trust reliability and even trust on reputation and trust through third parties. Often the trust evaluation depends on the perceptions of the parties involved in a communication. WSTO allows specifying any trust needs; its general nature makes it adaptable to any scenario.
- **Open**. Our aim is to make WSTO as open as possible. We intend to implement it in IRS-III, which is publicly accessible. More significantly, the constituents of WSTO are Semantic Web Services, so they can be represented a) in term of ontologies e b) in terms of components. All participants can re-

place the main parts of the WSTO, by instantiating new match criteria, or publishing new semantic descriptions of own trust policies.

- **Trust-based invocation**. The core purpose of our ontology is to enable trust-based invocation. We believe that this approach is useful in an open and distributed environment such as the Semantic Web Services environment.
- **Explicitness**. Policies and their evaluation mechanism are explicitly formally described.

We adopt WSMO [16] as basic vision that provides ontological specifications for the core elements of Semantic Web Services. WSMO specifies a set of and *non-functional properties* that describe information that does not affect the functionality of the element, such as title, authorship, copyrights, etc. Among them "trust" is listed as a recommended property for web service description. Nevertheless, until now, the WSMO effort has not specified any process to enable trust-based discovery and selection. We claim that our approach has a natural fit with the WSMO requirements and so propose to extend WSMO with our ontology.

Further work is underway on implementation of WSTO in IRS-III.

Acknowledgements

This work is supported by DIP (Data, Information and Process Integration with Semantic Web Services) (EU FP6 - 507483) and AKT (Advanced Knowledge Technologies) (UK EPSRC GR/N15764/01) projects.

I would like specially to thank John Domingue for his constructive guidance and feedback. I also have benefited from discussion with Barry Norton and Andreas Friesen.

References

1. Bilgin, A. S., Singh, M. P. (2004). A DAML-based repository for QoS-aware Semantic Web Service selection. In Proceedings of the IEEE International Conference on Web Services (ICWS'04), Washington, DC, USA, 2004.
2. Domingue, J., Cabral, L., Hakimpour, F., Sell, D., Motta, E. (2004). Irs-III: A platform and infrastructure for creating WSMO-based Semantic Web Services. In Proceedings of the Workshop on WSMO Implementations (WIW 2004), Frankfurt, Germany, September 2004.
3. 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.
4. Fensel, D., Bussler C. (2002). *The Web Service Modeling Framework WSMF*. Electronic Commerce Research and Applications, 1(2), 2002.
5. 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 Deliverable 1.1.

6. The Friend-Of-A-Friend (FOAF) Project. (2004). Available at http://www.foaf-project.org/.
7. Golbeck, J. and Hendler, J. (2005). Inferring trust relationships in web-based social networks. *submitted to ACM Transactions on Internet Technology*, 2005.
8. Kagal, L., Paoucci, M., Srinivasan, N., Denker G, Finin, T., Sycara K. (2004). Authorization and privacy for Semantic Web Services. In Proceeding of AAAI 2004 Spring Symposium on Semantic Web Services, Stanford University, Mar. 2004.
9. Kolovski, V., Parsia, B., Katz, Y., Hendler, J. (2005). Representing Web Service Policies in OWL-DL. in Proceedings of 4th International Semantic Web Conference (ISWC 2005), November 6-10, 2005, Galway, Ireland.
10. Mani, A., Nagarajan, A. (2002). Understanding quality of service for Web Services: Improving the performance of your Web Services -IBM-report- 2002. (Available at http://www-128.ibm.com/developerworks/library/ws-quality.html).
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 (ICSOC 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. Ngan, T. W., J, Wallach, D., S., Druschel, P. (2003). Enforcing Fair Sharing of Peer-to-peer Resources. 2nd International Workshop on Peer-to-Peer Systems (IPTPS) (Berkeley, California), February 2003.
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, Jul.. 2004, San Diego, California, USA.
16. Roman, D., Lausen, H. and Keller, U. (Eds) (2005). The Web Service Modeling Ontology WSMO, final version 1.1. WSMO Final Draft D2, 2005.
17. Sabata, B., Chatterjee, S., Davis., M., Sydir, J., Lawrence, T. (1997). Taxonomy for QoS Specifications. In Proceedings of the 3rd Workshop on Object-Oriented Real-Time Dependable Systems (WORDS '97), February 1997.
18. 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.
19. Stefik M. (1995). Introduction to Knowledge Systems. Morgan Kaufmann, San Francisco, CA.
20. Vu L., H., Hauswirth, M. and Aberer, K. (2005). QoS-based Service Selection and Ranking with Trust and Reputation Management. Technical Report IC2005029, Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland, June 2005.
21. VeriSign. (2005). Intelligent Infrastructure Services At Work. Information available at: http://www.verisign.com/.
22. W3C (2004). Web Services Architecture. W3C Working Draft 11 February 2004 (Available at http://www.w3.org/TR/ws-arch/).