Conceptual Model of Web Service Reputation*

E. Michael Maximilien
IBM and NCSU
maxim@us.ibm.com

Munindar P. Singh
North Carolina State University
singh@ncsu.edu

ABSTRACT

Current Web services standards enable publishing service descriptions and finding services on a match based on criteria such as method signatures or service category. However, current approaches provide no basis for selecting a good service or for comparing ratings of services. We describe a conceptual model for reputation using which reputation information can be organized and shared and service selection can be facilitated and automated.

Keywords

Web services, reputation, endorsement, trust

1. MOTIVATION

Web services promise the dynamic creation of loosely coupled information systems. However, current approaches are logically centralized and lack key functionality, especially to locate, select, and bind services meeting certain criteria of quality. Recently, we developed an architecture that uses software agents who serve as proxies for clients and interact with one or more agencies through which service reputations and endorsements are disseminated. However, this and other service architectures leave open some key semantic questions. Specifically, a proxy should be able to discover and understand new service attributes from their descriptions, especially as they evolve over time, and an agency should be able to aggregate the right information about service quality and present it suitably formally described that they can be understood by proxies.

We address these semantic questions by developing a conceptual model of a service provider's reputation for delivering quality services. The conceptual model has a generic component (e.g., attribute types and common attributes such as price, on-time delivery, and so on) and can be enhanced with domain-specific components (e.g., closeness of itinerary to desired times, which makes most sense for services in the travel domain). By combining service considerations with semantic web representations, this work fits into the recent activity on semantic Web services.

*We are indebted to Amit Chopra, Ashok Mallya, Amit Sheth, Raghu Sreenath, and Pınar Yolum for useful comments. This work was supported by IBM and the National Science Foundation under grant ITR-0081742.

1.1 Background

For our purposes, a Web service encapsulates functionality that can be described, located, and invoked over the Web. [Curbera et al., 2002] present a tutorial on current Web services standards and their typical usage. Here is some essential background for this paper. To facilitate interoperability by making Web services neutral as to vendor and platform, a service is described using an XML application standard called Web Service Description Language (WSDL). Service descriptions are cataloged using a standard registry definition called Universal Discovery, Description and Integration (UDDI).

UDDI registries contain white pages for each registered provider. Each provider can publish service descriptions, which describe how to bind to and invoke a service. References to concrete service implementations are available for binding as part of the *tModel*s—or template models—of the service. A service can possibly be provided by many businesses; in fact, a business may provide many implementations of the same service. Typically, clients know the desired service interface *a priori*. They find a desired implementation by querying registries for services meeting certain criteria and implementing known service interfaces.

1.2 Current Approaches

A key limitation of current approaches is that they lack dynamic discovery and binding processes. We need an architecture that allows clients of services to automatically select and bind with desired services, and with minimal intervention by the designers and implementers or the end users. Further, current approaches support no memory of service bindings and interactions, except whatever a specific application programmer may include in an ad hoc manner. Thus clients cannot learn from their past interactions to improve future decisions. Also, there is no systematic support for learning from the past interactions of others.

Although much data handling and underlying protocols are automated through current standards, key aspects that support flexible decision-making with respect to the composition and selection of services are still treated in an ad hoc manner. Making these aspects systematic is the main motivation for our approach.

1.3 Organization

The rest of this paper is organized as follows. Section 2 introduces our technical framework and the challenges in adding reputation for Web services. Section 3 describes our proposed conceptual model for reputation, and includes an extensive example. Section 4 presents some related literature and further directions.

2. FRAMEWORK

We now introduce our basic framework and the challenges it raises.

2.1 Architecture

Any conceptual model must rely upon an execution architecture. For concreteness, we describe our recently introduced proxy-based architecture [Maximilien and Singh, 2002]. Our conceptual model is compatible with other architectures as well, but we lack the space to describe additional candidates here.

We propose the addition of a Web Service Agent Proxy (WSAP) to access each service. For our purposes, an agent is a software component that automates some tasks for its principal. Agents communicate with other agents, accept requests from their users, and are typically autonomous. A WSAP is an agent that acts as a proxy for clients of Web services. That is, a client would have a WSAP for each service that it needs. WSAPs are knowledgeable about the various service standards. All activities of the client pertaining to the given service—including requests and responses, and communications with UDDI registries and bindings—occur via the WSAP for that service. In this manner, when the client needs to bind to a service, it instantiates a WSAP, which consults outside registries as well as reputation and endorsement agencies, helps find appropriate providers, records any feedback from the client, learns from the experience, potentially shares its knowledge with the external agencies and other WSAPs, and hopefully helps find better providers the next time around. Figure 1 provides an overview of how clients typically use WSAPs.

2.2 Key Concepts

A distributed trust system consists of a set of *principals*, i.e., the parties involved either as service provider or requester. The principals interact with each other over a set of services. A *rating* of a service is a vector of attribute values. The *reputation* of a service is a general opinion i.e., it aggregates the ratings of the given service by other principals. Typically, a reputation would be built from a history of ratings by various parties. An *endorsement* of a service by a principal is modeled as a boolean scalar and a time limit on the validity of the endorsement.

Although we concentrate primarily on reputation, the underlying conceptual model is quite similar for endorsements as well. This is because an endorsement effectively states that the service being endorsed offers high quality with regard to some selected attributes: price, reliability, and so on.

In our proposed approach, WSAPs would maintain and contribute ratings to others and discover reputations. However, the ratings are based ultimately on feedback received from their clients. Some service qualities such as price and delay may be calculated automatically, whereas others may require human participation. Even the latter kind, although clearly harder to automate, can be accommodated by our architecture. For such cases, the application should be designed so that it is possible to receive feedback from the human user after usage of the service. In this case, the WSAP will exploit this human feedback to learn the user's preferences.

2.3 Challenges

Our architecture opens up some interesting challenges, of which the following are germane to the topic of this paper.

2.3.1 Conceptual model of service attributes

Can we define a generic conceptual model for attributes reusable across domains?

Our agents are configured to capture the wishes of the application user. The agent uses this configuration information to maximize the utility of the user. However, in order for the agent to make intelligent decisions it will need more than just the reputation and endorsement agencies. It will need knowledge of attributes the user cares about, such as the following:

- The threshold of the values for attributes that the user is willing to accept.
- The risk tolerance of the user. For instance, the WSAP could find a reputable service matching the user's preferences but because it is relatively new, selecting that service could be regarded as higher risk than a known more mature service.

Answering the above question will enable our WSAP agents to efficiently and thoroughly capture the preferences of their WSAP users. Further, the service selection will often need to be fast because the user may be waiting for a service to be found. Therefore, the agent must be able to make quick decisions, comparing the user's preferences with information provided by the agencies.

2.3.2 Semantics of service attributes

How can we add semantics to service attributes, thereby allowing a WSAP to dynamically discover new attributes without having to be reconfigured or reprogrammed?

How can the agent acquire the knowledge of new attributes that were not specified by its client? That is, how can the agent relate the attributes specified by its client with attributes from other agents? The W3C's *Semantic Web* initiative [W3C, 2000] is a promising direction in capturing the semantics of service attributes.

2.3.3 Effects of attribute type on reputation

How should reputation be related to history of previous interactions? Should the effect of an interaction decay over time?

The notion of reputation is tightly bound to history and time. The reputations of human services tend to vary with time and recollection. In the digital world, history and memory can be collected easily. Because of this, the notion of reputation for humans and for agents have important differences. Some reputation systems [Zacharia and Maes, 2000] build in this decay effect. One approach to include time in our proposed architecture would be to, for instance, associate timestamps with attribute values, thereby allowing the reputation rating to weight attributes depending on their age. Further, since service quality will tend to change over time, decaying the reputation helps by reducing the effect of interactions over time, effectively increasing the currency of the evaluations.

A similar situation arises with endorsements. The goal is that it should be as easy as how people now look into the local newspaper and select a movie by looking at the number of stars it was awarded. Of course, a movie-goer may be biased towards a movie because of his knowledge of its actors, director, or producers—these intangibles will have conscious and subconscious implications to the movie-goer's decision. This is not completely the case for the software agents. However, endorsements do affect the agent's final decision. An endorsed service can similarly bias an agent towards a particular service regardless of its rating. How should agents weigh reputation ratings with respect to endorsements? What we need is a scheme by which attributes and endorsements can be systematically combined.

3. CONCEPTUAL MODEL

A Web service represents a set of functions addressing a particular domain. For instance, a travel service might include functions to return a list of trips for a particular airline on a specified date, time,

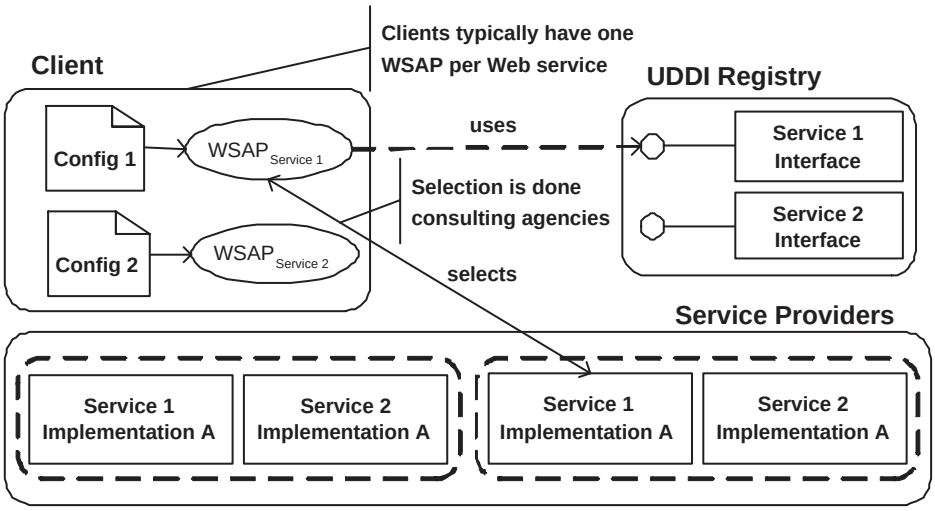

Figure 1: WSAP operation summary

origin and destination airport. For each service we can extract a series of generic attributes and domain-specific attributes that apply to the service. For instance, for the travel service we mentioned, a generic attribute could be the speed at which a search produces its results and a domain-specific attribute could be the accuracy of the return results—e.g., whether it includes up-to-the-minute trip information. From this example and what we discussed before, it is clear that a conceptual model for the reputation of a service must include the different categories of attributes that apply to a service.

The reputation of the service is a function of the various attributes that matters to a specific agent. For instance, an agent that cares more about accuracy of the trips returned would take into account this attribute more than the relative price attribute, which might be of concern to another user. In essence, the relative weight given to an attribute affects the overall reputation of the service and depends on the user of the service. This is analogous to how various human real world services can appear to be rated differently depending on the end user who is providing the rating. A car rental service that charges more but is flexible by giving convenient access at major airport may be of higher value to a business user than a person looking for a rental car for vacation purposes. So within a specific domain the reputation of the service depends on the subjective view of the user of the service on the various attributes that matters to the current agent that is proxying this service. Other factors that affect the reputation of a service include the following.

- The relative weights given to the attributes. Each agent will have preferences that biases it towards certain attributes and therefore make these attributes weigh more than others for a specific domain.

- The attribute aggregation algorithm. A simple weighted majority can be the normed algorithm but using different algorithms will affect the resulting reputation value.

- The set of endorsers of the service as well as the list of trusted endorsers for the agent. Again, matching endorsers will bias the reputation value.

- The history of the service. The reputation of older services will be affected more by previous usages.

- Any type of damping for the ratings as in [Zacharia and Maes, 2000] will affect the reputation value. Such damping is necessary to allow for service's reputation to be regulated. For example, a service that acquired a bad reputation but then become better (and started receiving good ratings) can have its reputation improved since older ratings matter less than the newer ones.

The conceptual model shown in Figure 3 represents a UML static model for the different components that make up the reputation of a service. First, a *Service* associates with one *Reputation* which can have many *Rating*s. The reputation value is determined with a *ReputationAlgorithm* that aggregates the various *Attributes* that the agent determining the reputation chooses to consider. The reputation is also affected by a *History* that keeps previous ratings for the particular service. The rating for a service is determined by the *Principal* in question and calculated using the *RatingAlgorithm*. Any number of principals can endorse a service which might affect the calculation of the reputation since, for instance, an endorsement by a trusted third party can be considered of higher value than certain attribute values. As mentioned above, each attribute has a value and range, and associates with one or more *Domain*s. Domains act as collections of attributes for specific types of services.

3.1 Attributes model

For each domain, the attributes in that domain are important inputs to the overall rating and therefore the reputation of a service. Some attributes are common across domains and some are specific to domains. Each attribute has the following aspects.

- The value set for that attribute (and its allowed range or enumeration). For instance, an attribute such as failure rate or availability for a service can be expressed as a percentage. The speed of service function execution could be instead a simple bounded integer.

- The domains that this attribute belongs to. For instance, is this a cross-domain attribute or an attribute specific for a domain? And within each domain, some attributes will be of greater importance than others—this can depend on some standard definition of attributes for a domain.

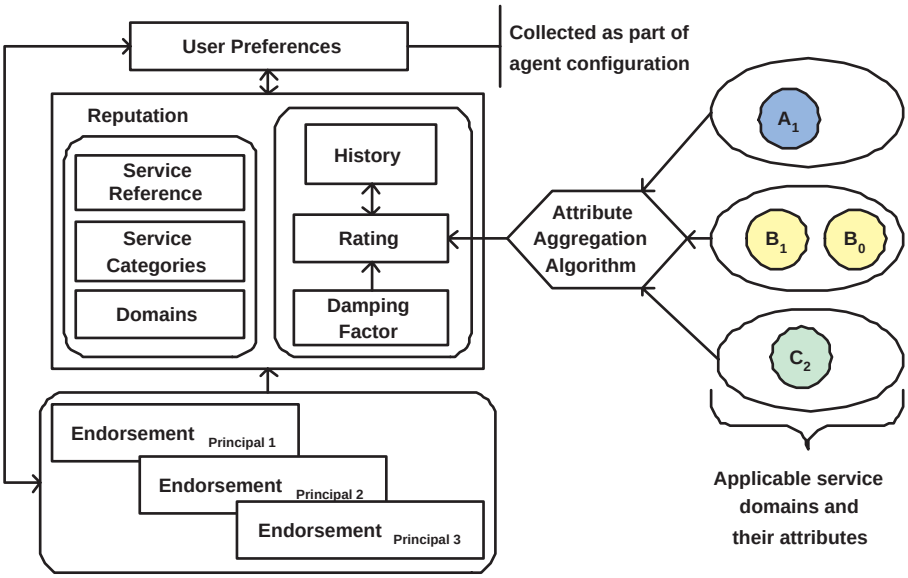

Figure 2: Generic framework model of a service reputation

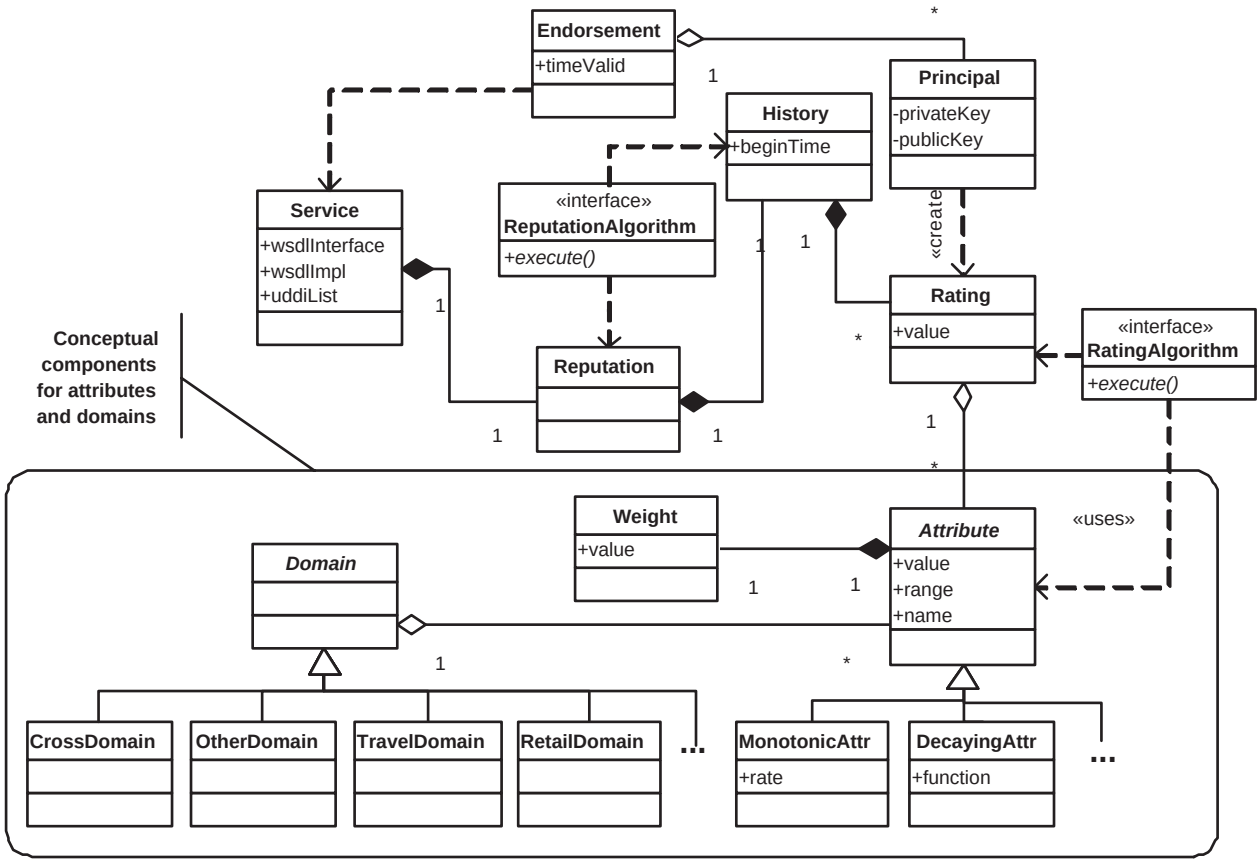

Figure 3: UML conceptual model for service reputation

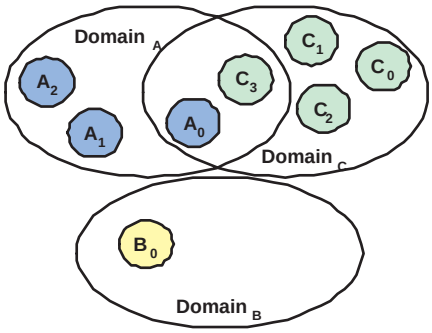

Figure 4: Domain model for service attributes

- The weight of this attribute relative to its domain and the user preferences. This would determine the impact of this attribute on the final decision regarding a provider.

- The characteristic of the function from attribute values to ratings. For instance, some attributes such as price are monotonic, at least in typical business scenarios. That is, the more the price decreases the better. Generally all agents will consider price in the reputation calculation and have a preference for lower prices. Of course, if the price were to decrease in conjunction with reductions in the values for other attributes, the overall reputation might not improve. For instance, for the trip service we considered, if the price attributes of trips were to decrease while the promptness (on arrival and departure times) attribute were to become worse, then this decrease in price might not help the overall reputation of the trip service.

  The characteristics of attributes can be quite rich and need to be further categorized. Initially, we consider monotonically increasing and decreasing, S-shaped characteristics (where there is a substantial benefit to ratings when the given attribute improves, but only if it is above a tolerance threshold and not above a saturation threshold).

- The temporal characteristic of the attribute value. A possible temporal characteristic for attributes is decaying values where the decay function can vary from exponential to a step function. For instance, an attribute such as accuracy in the travel domain might be acceptable to allow a range of minutes up to a certain point. That is, the trip that is scheduled for 3:00 PM does not cause major harm if the actual departure time is 3:20 PM. However, a departure of 4:00 PM might become unacceptable for a user that depends on an on time arrival with a gap of 30 minutes in order to catch a connecting flight. Other attributes might have values that decay more progressively rather than in a step-wise fashion.

3.2 Generic and domain-specific attributes

In general each particular domain that a service belongs to will have its own set of attributes that apply to all services of that domain. However, certain attributes will be cross-domain attributes. Figure 4 depicts this idea showing three domains with two of them overlapping for some attributes.

As a specific example, a service such as car rental service will involve attributes belonging to multiple domains, such as the travel and retail domains. Attributes such as price belong to both domains, but an attribute such as flexibility of reservation changes has a specific meaning in the travel domain—clearly, allowing flexible changes to a travel reservation is a particular characteristic of a travel service that might allow important service differentiation for a particular client and the client's WSAP.

Determining the attributes that apply to a particular domain is nontrivial, but needs be decided by the community of users and providers as they settle upon ways to distinguish and evaluate different offerings. This typically occurs when markets form, where different parties position their offerings according to what they believe are their key qualities: speed, price, taste, and so on. A technical challenge here is to distribute the attributes among various domains so that an agency does not necessarily have to capture all possible attributes and a WSAP can search for services without consulting an excessive number of agencies.

3.3 Adding and disseminating attributes

Another important aspect that the proposed conceptual model allows is the creation and dissemination of new attributes for specific domains. A consequence of the attributes having a common model and the specific attributes being subclasses of the abstract attribute model is that new attributes with unknown characteristics can be added to the system. New attributes are disseminated via a domain definition that could contain references to common attributes as well as to domain-specific attributes. Our conceptual model can be readily mapped into a common schema in a standard notational framework such as the resource description framework (RDF). The existence of such a schema would enable the description and widespread dissemination of attributes. Notice that the interesting component would be the conceptual model itself, not the notation, although agreeing on the notation is also essential.

3.4 Example

As a comprehensive example showing how the agent proxying a service can use the conceptual model we describe, imagine a travel reservation Web application. This application is used by agents to set up business and personal travel arrangements. Part of this application is a facility to search, select, and reserve a car rental that will be included as part of the overall travel arrangement. Since there exist many car rental companies, each advertising its services on the Web, it is easy to imagine that a common car rental Web service interface could be created. Each car rental company would provide an implementation of the service thus allowing its business to collaborate with others and thus integrate into coarse-grained services, such as a travel planning service. How is the WSAP proxying the car rental service able to pick the best service for a particular client?

Using our conceptual model, the WSAP could pick the correct service implementation by looking at the reputation of the various services. According to our model, reputation is a function of the historical ratings provided by previous users filtered to take into account the attributes that matter most to the current WSAP's principal. So, for instance, if the current user weighs price as an important cross-domain attribute then services giving lower prices might be selected over those giving higher prices. Of course, the reputation for the service will depend on several attributes. For instance, some attributes such as comfort and reliability of the cars rented might matter more to a traveler who intends to use the vehicle for long subtrips than a business person on a tight budget. As another example, though the car rental service domain attributes overlap with a car selling service, certain attributes such as color choice will clearly have more significance for a buyer as opposed to a renter. Our model takes these subtleties into account, because the WSAP is configured with attributes that apply to the domain that the service belongs to. Of course, the choice of which domain a service

belongs to is an important precondition that must be satisfied prior to agent configuration. However, we are assuming that this is done as part of classifying the services prior to them being introduced for wide availability.

Our model is generic enough to allow the introduction of new attributes. For instance, if the domain definition for car rental was updated to include a new attribute such as safety, then any new agent that was configured with this attribute could take that attribute into account for its ratings calculations. The attribute's values could possibly be captured as part of a user survey or automatically by collecting information on accidents from the car types that the car rental company rented for certain periods and the relative safety outcome of the rentals.

A lot of engineering work goes into making a large application. The above example is no exception. The contribution of our approach is in streamlining the attributes so that their treatment is standardized up to a point and, where it is not standardized, placed explicitly under the control of individual WSAPs and their principals. New attributes can added on the fly. More importantly, the different agencies can upgrade to the new attributes or add an existing attribute that they had previously ignored. Likewise, the conceptual model enables the WSAPs to share information directly with each other, which extends their power further.

4. DISCUSSION

In principle, service standards enable applications to connect to any suitable and accessible service provider. However, a key challenge is to select and locate a service provider who offers the best implementation of a particular service. It is this challenge that our approach addresses.

The Semantic Web has lately become an active area of research. Various projects such as DAML, DAML-S, OIL, OWL, and RDF [van Harmelen et al., 2002] address various research aspects of adding ontology description to the WWW as well as Web services. Current research such as [Mcllraith et al., 2001], takes an agent approach, using DAML and DAML-S in conjunction with Web services to show how services can be dynamically composed. Though similar in spirit, our work is focused instead on trying to define the service attributes such that they can be dynamically discovered and understood by the WSAP without having to force all clients to know the set of attributes ahead of time. Further, we would like to allow the service attributes to evolve over time without necessarily requiring the WSAP to be reprogrammed. This is similar to the overall aim of the Semantic Web where one vision is that annotated semantic information will allow agents to infer meaning of the described things on the WWW without necessarily requiring agents to know *a priori* much information on the things being discovered.

Another approach for design-time service selection is presented in [Cardoso and Sheth, 2002] where Web services are selected by matching the input and output of the given services as well as some quality characteristics with the desired criteria. Our approach differs in two ways: (1) we assume that the Web service clients already chose the interface of the desired services, but need to decide on a service implementation, possibly at run time, and (2) the criteria of selection reflect the reputation of the given services with respect to different attributes.

The main goal of the WSAP when using the attributes attached to the services is to be able to dynamically select and invoke the best matching service. IBM's WSIF [IBM, 2001] has a similar aim in allowing dynamic invocation of services but in the current incarnation of WSIF the service selection is fixed at design time. [Fensel and Bußler, 2002] propose a complete framework to facilitate usage and invocation as well as composition of services called WSMF—Web Services Modeling Framework. However, WSMF does not address the issue of defining attributes that can adorn services thus allowing them to be reasoned about in their domain or across domains, as we are proposing.

[Oram, 2001] presents an overview of existing systems that use a trust model similar to ours. For instance, the Pretty Good Privacy (PGP) *web of trust* system handles automated reputation collection to assist in verifying that public keys of principals are valid. The popular news site Slashdot uses a manual reputation gathering system to screen out and rank news postings, thereby allowing clients to filter out postings that are judged to be of low value.

Reputation mechanisms are used in e-commerce Web sites such as Amazon, eBay, and OnSale as a means of keeping track of ratings [Zacharia and Maes, 2000]. A buyer or seller can capture his experience in specific transactions by rating the other party numerically or writing a text note about it. The marketplace makes this information available to other users so they can factor it in when deciding with whom to deal. Like in the above marketplaces, we model reputation in terms of attributes reflecting a user's experiences with a given service. However, here the reputation data need not originate with humans, but may be provided by agents. More importantly, existing marketplaces have a fixed, often trivial, conceptual model for reputation, where a single scalar is recorded along with text comments, which cannot be used automatically.

We presented a preliminary conceptual model for reputation and endorsement. We are building a prototype implementation that exercises this model. Eventually a model such as this should become core of a new standard for Web service location and selection.

References

Jorge Cardoso and Amit Sheth. Semantic e-workflow composition. Technical report, University of Georgia, July 2002.

Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana. Unraveling the Web services Web. *IEEE Internet Computing*, 6:86–93, 2002.

Dieter Fensel and Christoph Bußler. The Web service modeling framework (WSMF). In *Database and Information Research for Semantic Web and Enterprises*, 2002.

IBM. Web services invocation framework (WSIF), 2001. http://alphaworks.ibm.com/tech/wsif.

E. Michael Maximilien and Munindar P. Singh. Reputation and endorsement for Web services. *SIGecom Exchanges*, 3:24–31, 2002.

Sheila A. Mcllraith, Tran Cao Son, and Honglei Zeng. Semantic Web services. *IEEE Intelligent Systems*, pages 46–53, 2001.

Andy Oram, editor. *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*. O'Reilly, 2001.

Frank van Harmelen, Ian Horrocks, Peter Clark, Peter F. Patel-Schneider, Mike Uschold, Marie-Christine Rousset, James Hendler, and Guus Schreiber. Ontologies' KISSES in standardization. *IEEE Intelligent Systems*, 17:70–79, 2002.

W3C. Semantic Web, 2000. http://www.w3.org/2001/sw/.

Giorgos Zacharia and Pattie Maes. Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14:881–907, 2000.