ENHANCED DISCOVERY OF WEB SERVICES
Using Semantic Context Descriptions

Simone A. Ludwig
School of Computer Science, Cardiff University, Cardiff, UK

S.M.S. Reyhani
Department of Information Systems and Computing, Brunel University, Uxbridge, UK

Keywords: Context information, Semantics, Ontology, Web Service Discovery

Abstract: Automatic discovery of services is a crucial task for the e-Science and e-Business communities. Finding a suitable way to address this issue has become one of the key points to convert the Web into a distributed source of computation, as they enable the location of distributed services to perform a required functionality. To provide such an automatic location, the discovery process should be based on a semantic match between a declarative description of a service being sought and a description being offered. This problem requires not only an algorithm to match these descriptions, but also a language to declaratively express the capabilities of services. This paper presents a context-aware ontology selection framework which allows an increase in precision of the retrieved results by taking contextual information into account.

1 INTRODUCTION

Recently, more and more organisations are implementing IT systems across different departments. The challenge is to find a solution that is extensible, flexible and fits well with existing legacy systems. Replacing legacy systems to cope with the new architecture is not only costly but also introduces a risk to fail. In this context, the traditional software architectures prove ineffective in providing the right level of cost effective and extensible Information systems across the organisation boundaries. Service Oriented Architecture (SOA) (McGovern, 2003) provides a relatively cheap and more cost-effective solution addressing these problems and challenges.

Dynamic discovery is an important component of SOA. At a high level, SOA is composed of three core components: service providers, service consumers and the directory service. The directory service is an intermediary between providers and consumers. Providers register with the directory service and consumers query the directory service to find service providers. Most directory services typically organise services based on criteria and categorise them. Consumers can then use the directory services' search capabilities to find providers. Embedding a directory service within SOA accomplishes the following:

- Scalability of services
- Decoupling consumers from providers
- Allowing updates of services
- Providing a look-up service for consumers
- Allowing consumers to choose between providers at runtime rather than hard-coding a single provider.

Although the concepts behind SOA were established long before web services came along, web services play a major role in SOA. This is because web services are built on top of well-known and platform-independent protocols (HTTP (Hypertext Transfer Protocol) (HTTP, 2004), XML (Extensible Markup Language) (XML, 2004), UDDI (Universal Description, Discovery and Integration) (UDDI, 2000), WSDL (Web Service Description Language) (WSDL, 2004) and SOAP (Simple Object Access Protocol) (SOAP, 2004)). It is the combination of these protocols that make web services so attractive. Moreover, it is these protocols that fulfil the key requirements of a SOA. That is, a SOA requires that a service be dynamically

A. Ludwig S. and Reyhani S. (2005).
ENHANCED DISCOVERY OF WEB SERVICES - Using Semantic Context Descriptions.
In *Proceedings of the Second International Conference on e-Business and Telecommunication Networks*, pages 193-196
DOI: 10.5220/0001407501930196

discoverable and invokeable. This requirement is fulfilled by UDDI, WSDL and SOAP.

However, SOA in its current form only performs service discovery based on particular keyword queries from the user. This, in majority of the cases leads to low recall and low precision of the retrieved services. The reason might be that the query keywords are semantically similar but syntactically different from the terms in service descriptions. Another reason is that the query keywords might be syntactically equivalent but semantically different from the terms in the service description. Another problem with keyword-based service discovery approaches is that they cannot completely capture the semantics of a user's query because they do not consider the relations between the keywords. One possible solution for this problem is to use ontology-based retrieval.

In this paper, ontologies are used for classification of services based on their properties. This enables retrieval based on service types rather than keywords. This approach uses context information to discover services using context and services descriptions defined in ontologies.

2 FRAMEWORK

Related work has shown the need for more expressiveness of service descriptions revealing the limitation of a syntactic approach to service discovery. To follow this direction proposed by related work towards a semantic based approach for service discovery the context-aware ontology selection framework is proposed. Additional requirements have driven this framework towards a context-aware ontology selection framework described and are summarised as follows:

1. *High Degree of Flexibility and Expressiveness*
2. *Support for Subsumption*
3. *Support for Data Types*
4. *Matching Process should be Efficient*
5. *Flexible and Modular Structure*
6. *Lookup of Matched Services*

The architecture shown in Figure 1 comprises of clients, matchmaker, context and service ontologies, registries, and web servers hosting the web services.

The components are now explained in more detail:

- *Clients* provide an interface for the users to describe their service requests. The client also lists the matches and provides the facility to call the web services retrieved.
- *Registries* contain the service information. Service descriptions are in the form of service name, service attributes (inputs and outputs) and service description.
- *Web Servers* host the web services.
- *Matchmaker* consists of the matching module including the matching algorithm and a reasoner for the ontology matching process. The matching algorithm is explained in further detail in the following section.
- *Ontologies* (context and services) describe the domain knowledge such as book shop services and provide a shared understanding of the concepts used to describe services. Contextual information is crucial to ensure a high quality service discovery process (Gruber, 1992).

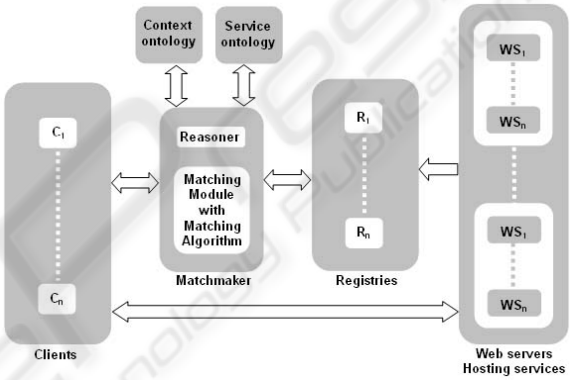

Figure 1. Matching Architecture.

The interactions of a service request are the following: The user contacts the matchmaker where the matching algorithm is stored. The matchmaker contacts the context ontology and reasons depending on a set of rules defined. The same is carried out for the services ontology. Having additional match values the registry is then queried to retrieve services descriptions which match the request and returns the service details to the user via the matchmaker. The parameters stored in the registry are service name, service attributes, service description and contact details. Having the URL of the service the user can then call the web service and interact with it.

The matching algorithm reads the service request parameters (context attributes and service attributes) from the client first. Then the context ontology is parsed and rules are applied to match the context keyword by providing the context attributes. Having the context keyword and the service attributes allows to query the services ontology which in turn returns the service matches. This list is then forwarded to the registry module where the lookup is performed retrieving the necessary contact details for each service.

3 APPLICATION SCENARIO

An application scenario was chosen to demonstrate the usability of the approach. It is assumed that many e-shopping web services are available on the Web. These can be any kind of services e.g. Amazon, eBay, etc., wrapped as web services offering different goods to buy such as *Books*, *Bikes* and *CDs*. It is furthermore assumed that in most cases a client searches for a service not knowing the service name. The user only specifies a service request with a few keywords describing the service needs. For this scenario a context ontology was created supplying the categories of services for e-shopping. The context ontology contains categories representing *Food*, *Clothes*, *Bikes*, *Cars*, *Shoes*, *Books* and *CDs*. The underlying classes contain many associative relations to each of the categories. Each of the classes belonging to one of the categories contains attributes describing the class further. E.g. class *Business* (belonging to context *Books*) contains the attributes *computer*, *reading*, etc. For a special application domain two identical attributes in more than one class could be eliminated. However, if context ontologies would be reused from other sources this ambiguity can not be disqualified. The prototype implementation solves this problem by taking the additional context parameters into account to eliminate the "wrong" context. If the user only specifies one context parameter which matches two categories then the prototype returns a mismatch statement.

The context ontology (Context Ontology, 2005) is written in OWL (W3C Working Draft, 2004) description containing class and subclass relationships. The structure of the e-shopping services ontology is the following: The first level contains the corresponding categories of the context ontology. The second level represents the actual service implementation with the attributes below. For example, one service specification outlines the *Books* web service. Different service implementations are *BookBuy*, *Bookshop*, *BuyBooks*, *Books* and *BookSale*.

In the services ontology (Services Ontology, 2005) not only class and subclass relationships are declared but also data type property relationships describing the attributes of the service.

In order to demonstrate how the process from service request to service response works is shown next. The user issues a service request consisting of context and service attributes. The context attributes (e.g. *computer* and *reading*) are taken first and the context ontology is queried using these search attributes resulting in the context keyword *Books* which is used for the service search part. The services ontology is then reasoned using the context keyword and the service attributes specified in the service request query. The retrieved services are *BookBuy*, *Bookshop*, *BuyBooks*, *Books* and *BookSale*. After these services are matched the service details are retrieved from the registry and returned to the user.

4 EVALUATION

The evaluation is done by calculating precision and recall rates. Precision is the fraction of advertised services which are relevant, i.e. the highest number is returned when only relevant services are retrieved. Recall is the fraction of relevant services which have been retrieved, i.e. the highest number is returned when all relevant services are retrieved.

For the evaluation of precision and recall values a comparison of a keyword-based approach with the prototype approach was conducted. The focus for this evaluation was on book services.

Table 1: Relevant Services.

	service1	service2	service3	service4	service5
context attributes	**computer reading**				
service attributes	**title**	**heading**	**name**	**writing**	**title**
	author	**writer**	**authors**	**maker**	**composer**
	number	**issue**	**no**	**product**	**id**
	category	**class**	**family**	**concept**	**category**
	price	**cost**	**amount**	**worth**	**value**
	publisher	**owner**	**proprietor**	**publisher**	**owner**
	pages	**page number**	**page**	**pages**	**pages**

Table 1 shows the relevant services. All attributes shown in the table are the service attribute parameters used for this evaluation. Matches are indicated in bold.

Table 2: Irrelevant Services.

	service6	service7	service8	service9	service10
context attributes	graph picture				
service attributes	**title**	**issue**	**name**	**product**	**composer**
	number	**owner**	**proprietor**	**pages**	**id**
	price	isbn	issn	book	**value**
	pages	drink	shop	meal	**pages**
	book	pixel	colour	point	book
	shop	font	paragraph	space	food
	colour	space	bold	font	colour

Table 2 shows the irrelevant services. The attributes indicated in bold match with the extended context ontology taken for this experiment; however the context parameters do not match the *Book* category. The number of service attributes is the same for relevant and irrelevant services.

The context parameters define the category of the service which results in the two tables (Table 1 and 2) being relevant services and irrelevant services. The user wants to find *Book* shop services and specifies a service request 1 (context parameters: *computer*, *reading*; service parameters: *title*, *author*, *number*, *category*, *price*, *publisher*, *pages*) with the parameters specified for service 1 in Table 1. Service request 2 is specified with the parameters of service 2 (Table 1) and so on. The context parameters of the service request are always *computer* and *reading*.

Figure 2 shows the results of the precision and recall values. The precision and recall results of the keyword-based approach range between 20% and 70%, whereby the prototype approach achieved a precision and retrieval rate of 100% in this experimental setup. As the recall and precision rates from the prototype show higher values than the rates from the keyword-based approach, it shows that the user receives a better subset of services that are relevant and in addition, the user receives no services that are irrelevant.

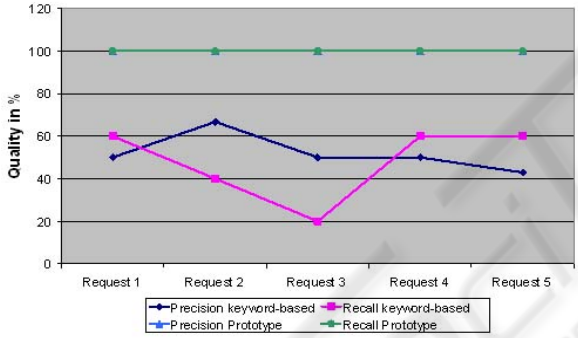

Figure 2: Evaluation of Precision and Recall Values.

Due to the fact that this research is conducted in a limited application domain, the set of advertised services, query and ontology are highly adapted and therefore a result of 100% is retrieved. In a real-world application scenario this correlation might not always be that high, especially if a context ontology from third-parties is used.

The accomplished result of service matches does not state that in every application scenario always values of 100% are achieved but it indicates the improvement in quality of service discovery results by using this semantic approach. Precision and recall measures showed the increase of quality of service matches, which was achieved by the customisation of the context and services ontologies.

5 CONCLUSION

The use of contextual information results in a better service discovery process due to an increased precision of the matched services. The contextual information enhances the expressiveness of the matching process, i.e. by adding semantic information to services, and also serves as an implicit input to a service that is not explicitly provided by the user. The prototype approach facilitates interoperability as the context and service properties are defined and specified in associated ontologies. Re-writing of code or interface wrapping does not need to be done in order to make systems interoperable. The development and maintenance is much easier due to the modular structure and encapsulation of context matching, service matching and registry selection. A drawback of this approach is that users registering services need to know the category their services belong to. Cases where a service falls into more than one category need to be registricted in order to allow an automatic and precise discovery and selection of service matches.

REFERENCES

McGovern, J., Tyagi, S., Stevens, M., Mathew, S., 2003. *The Java Series Books - Java Web Services Architecture*. Chapter 2, Service Oriented Architecture.

HTTP - Hypertext Transfer Protocol, 2004. W3C, http://www.w3.org/Protocols/.

Extensible Markup Language (XML), 2004. W3C. http://www.w3.org/XML/.

UDDI Technical White Paper, 2000. http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf.

Web Services Description Language (WSDL), 2004. Version 1.1, W3C. http://www.w3.org/TR/wsdl.

SOAP Version 1.2, 2004. W3C. http://www.w3.org/TR/soap/.

Gruber, T.R., 1992. ONTOLINGUA: A Mechanism to Support Portable Ontologies, Version 3.0, Technical Report KSL 91-66, Department of Computer Science, Stanford University.

W3C Working Draft, 2004. “Requirements for a Web Ontology Language”. http://www.w3.org/TR/webont-req/.

Context Ontology, 2005. http://users.cs.cf.ac.uk/Simone.Ludwig/ontologies/webservices/ContextOntology.owl.

Services Ontology, 2005. http://users.cs.cf.ac.uk/Simone.Ludwig/ontologies/webservices/ServicesOntology.owl.