Extending UDDI with context-aware features based on semantic service descriptions

Stanislav Pokraev
Telematica Instituut
P.O. Box 589
7500 AN Enschede
The Netherlands

Johan Koolwaaij
Telematica Instituut
P.O. Box 589
7500 AN Enschede
The Netherlands

Martin Wibbels
Telematica Instituut
P.O. Box 589
7500 AN Enschede
The Netherlands

Abstract

The web has evolved from the HTML-based repositories of text and images towards the current web services and semantic web developments. Web services are believed to help the integration of diverse applications while the semantic web promises to increase the "intelligence" of the web, enabling richer discovery, data integration, navigation and automation of tasks. A natural next step would be the combination of these developments, resulting in semantic web services. Another trend we observe is the development of next generation networks, such as GPRS and UMTS, which provide a richer environment as they will have access to information about the user's, location, preferences and changing environment. With the aid of this information, service providers will be able to develop services that adapt to the user's context. However, current web services technologies, based on WSDL and UDDI, do not provide a means for context-aware service discovery. In this paper we present the design and implementation of an enhanced UDDI server, capable of storage, matching and retrieval of semantically rich service profiles that contain contextual information. We also present a tool that facilitates the creation, publication and discovery of such service profiles.

Keywords

Web services, Semantic web, UDDI enhancement, Context-awareness, DAML-S

1. Introduction

The concept of web services is gaining momentum in the sense that both the understanding and the popularity of this concept are increasing. However, as it is an emerging technology, we can still identify a number of open issues. In this paper we focus on the issues related to service publication and discovery, and especially the issues that come forth from the use of the web service paradigm in context-aware applications. To determine which services are relevant for an end-user in his own specific context we need semantically rich service discovery, and hence also rich publication. A typical user context can be divided into several context elements: environmental elements, personal elements, task-related elements, social elements, historical elements, spatio-temporal elements, etc. To serve a user in his context with a relevant service offering is a complex task when all these context elements have to be taken into account. As an example we can imagine a tourist in Amsterdam (spatial), that wants to visit a nice museum (task) that suits his interests (personal) with a couple of friends in his holiday (social) because it is a rainy day (environment). Chances are that a system will not have access to all the relevant data. Finding relevant museums with the services they offer requires a registry with powerful semantic searching capabilities. Therefore, we propose an enhanced service registry that alleviates the limitations of the current web services technologies concerning searching and publication, by the creation of a registry with a rich semantic model. On the one hand such a registry enables service providers to describe their services in detail (capabilities, functionality, constraints, context, etc.) and to relate those service descriptions to existing ontologies. On the other hand, it enables the searching parties to perform a much more subtle search that delivers a manageable result set, by using constraints, relations between concepts, approximate matches and semantically rich queries. Last but not least, it allows the registry maintainers means

to improve the quality of the registry by ontology monitoring and advertisement expiration to avoid registry pollution.

2. Current state of technology

A web service can be characterized as a software component exposed on the web via a well-defined interface, and described and invoked via international standards, including (but not limited to) SOAP[1] and WSDL[2]. Web services can be seen as building blocks for creating open distributed systems, which allow companies and individuals to make their digital assets available to the global community at large in a simple and effective manner. The spectrum of possible web services is impressive, and ranges from single functions (e.g. financial calculations or construction analysis) to entire business processes such as airline ticket reservation. Web services can be reused across multiple applications, hence allowing fast and efficient web application development.

In general, three roles can be distinguished in the web services concept: a service provider, a service requestor, and a service registry. The requestor and the provider are able to discover each other via a third party, the registry, which holds up-to-date information about businesses, and the services they offer. This way, a service requestor can discover existing web services, determine their purpose, functionality, and operation instructions, and hence use a service to his benefit.

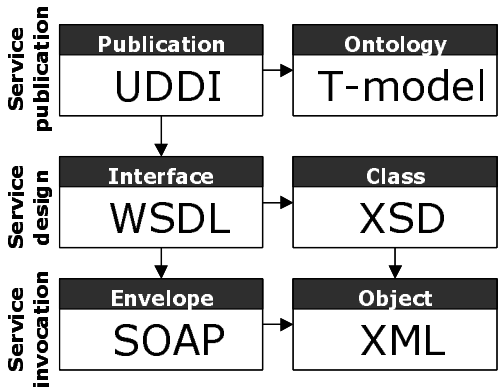


Figure 1

Applications typically communicate with web services via SOAP messaging. SOAP messages encapsulate the XML payload that is transported to a SOAP server, which deserializes the XML, and invokes the target web service. The interaction pattern with the web service is described in the interface definition specified in WSDL. The WSDL refers to an XML schema (XSD) that describes the structure of a specific class of XML messages. These WSDL descriptions can then be referenced from the service advertisements in a UDDI[3] registry. The registry uses simple ontologies based on so called t-models, but these models can only be used to identify or classify businesses and their services, and do not relate to the classes of objects that determine the input and output of a particular service.

The registry plays a central role in connecting a service requestor with relevant providers. Such a registry can be company-internal, branch or region oriented, but the most common notion is that of the universal business registries. From its creation in 2000, UDDI has had the intention of providing a directory service where service providers and service requestors come together to satisfy their needs. It is a central concept in the web services architecture[4]. However, the concept still has a number of flaws that prevent UDDI from reaching the critical mass required for success:

- Lack of moderation in the central registries causes registry pollution and decreasing data quality.
- Inadequate quality of service guarantees for the published services in a registry.
- Lack of semantic description mechanisms in WSDL and UDDI (lack of semantic interoperability, explicit semantic models to understand the queries and reason about the knowledge).
- Lack of process specification – e.g. the current web service descriptions do not allow for specification of the right order of operation invocations, and constraints that follow from the high-level process.
- Lack of mechanisms for fuzzy or inexact match e.g. a request for a “transportation service” cannot discover “shipping services”.
- Lack of ontology support. Service providers and service requestors have completely different knowledge about a service. Service descriptions and service requests have to be understood and agreed upon between the two parties by means external to the registry. A common ontology is a must in order to facilitate an effective discovery process.
- General (Interoperability between standards, service execution, security, quality-of-service, etc)

Concepts from the semantic web[5] vision can be used to alleviate the limitations on the semantic expressiveness of UDDI. The semantic web is a W3C

initiative that aims at making the current web machine-understandable. It is "an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation"[6]. The main principles of the semantic web are that all resources in the web (and indirectly, the resources from the real world) are identified by Universal Resource Identifiers (URIs) and have well-specified types. Using URI's makes it easy to reuse other parties ontologies and resources. Machines can use the type of a resource to classify it or to adapt their behavior when using it. The type therefore encodes (part of) the resources semantics. The semantic web tolerates partial information – one can create links to other's resources, without having to worry about reversal links. Semantic web does not require an absolute truth – it does not solve the issue of trustworthiness of resources, but it does have mechanisms to express who (or what) stated a "fact" in its knowledge-base. Finally, evolution is supported - different people can create similar concepts - the semantic web will use descriptive conventions to allow for effective combination of the independent work.

Ontologies will play a crucial part in the realization of the semantic web vision, facilitating the sharing of information between communities, both people and software agents. To support the use of ontologies, a number of representational formats have been proposed, including RDF Schema[7], the Ontology Interchange Language (OIL)[8] and the DARPA Agent Markup Language (DAML)[9]. These last two have been brought together to form DAML+OIL[10], which is the basis for the ontology web language (OWL)[11], a language now being proposed as a W3C standard for ontology and metadata representation.

DAML+OIL has been designed to describe the structure of a particular domain of interest. It takes an object-oriented approach by describing the structure in terms of classes and properties. An ontology described by DAML+OIL consists of set of axioms that asserts relationships between classes or properties. From a formal point of view, DAML+OIL can be seen as an expressive description logic enabling sound, complete and efficient reasoning services. DAML+OIL classes can be names or expressions. The expressions can be built by means of variety of constructors like intersectionOf, unionOf, complementOf, etc.

Another very important feature of DAML+OIL is the set of axioms that the language defines. The axioms make possible to describe subsumption and equivalence with respect to classes and properties or unions or disjoints of classes. The following example gives an idea of how an ontology could be created by means of just a few starting concepts and relations (Human, Male, parent (Human, Human)):

$\text{Female} \subseteq \neg\ \text{Male}$

$\text{Man} \subseteq \text{Human} \wedge \text{Male}$

$\text{Woman} \subseteq \text{Human} \wedge \text{Female}$

$\text{father} \subseteq \text{parent (Human, Man)}$

$\text{mother} \subseteq \text{parent (Human, Woman)}$

$\text{child} \subseteq \text{parent}^{-}$

$\text{son} \subseteq \text{child(Human, Man)}$

$\text{daughter} \subseteq \text{child(Human, Woman)}$

The DAML-S coalition[12] is in development of semantic markup for service descriptions[13] based on DAML+OIL. DAML-S complements WSDL and aims at enabling automated discovery, invocation, composition and execution monitoring of the web services. It has well-defined semantics that enables the creation of a web services vocabulary by means of objects and complex relationships between them (e.g. classes, subClass relations, cardinality restrictions, etc.). Figure 2 presents the DAML-S upper ontology for services.

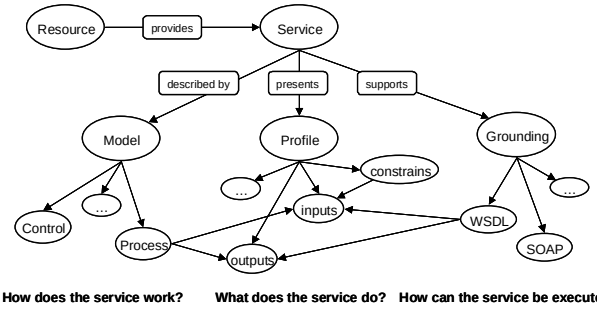


Figure 2

The top of the DAML-S ontology is the class Service. At this level all defined properties are very general. The idea is to provide a conceptual basis for structuring the services taxonomy, but it is expected that the taxonomy itself will be created according to the functional and domain specific needs.

The Service Profile is a high level description of the service and its provider. It describes the service in a human readable way, specifies the functionalities provided by the service and its functional attributes (e.g. requirements and capabilities). The Service Profile is used for populating service registries, automated service discovery and matchmaking.

The Service Model describes what the service does. It facilitates automated services invocation, composition, interoperation and execution monitoring. It works as follows: Each service is conceived as a simple or composite process. Associated with each service is a set of inputs, outputs, preconditions and effects. Composite processes are compositions of simple or other composite processes in terms of constructs such as sequence, choice, if-then-else, etc. The Service Model provides means for describing the data flow and the control flow for each composite service.

The Service Grounding provides a specification of service access information. It specifies the communication protocols, transport mechanisms, etc.

3. Our approach

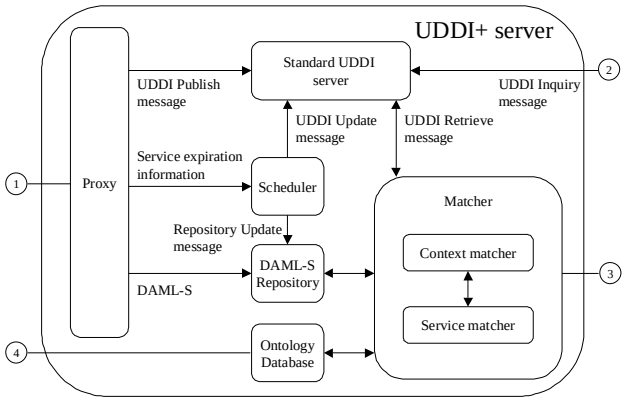
Our enhanced UDDI server (UDDI+) is an attempt to improve the existing discovery mechanisms in current UDDI servers. The main goals of UDDI+ are (i) to enable service discovery based on semantic information associated with the services and (ii) provide the user with services that are relevant to his context (e.g. location, weather conditions, task, etc.).

However, information about the user's context is distributed across a number of heterogeneous resources. For instance, the location of the user might be obtained from the future 3G mobile platform, user's activities from his calendar (located on the his PDA or company server), the user's colleagues/friends list from his instant messaging server, etc. Moreover, sometimes we might need to evaluate a certain context parameter dynamically given information about another context parameters. For instance, if we want to provide the user with a service that is "nearby" we need to know what "nearby" means for this user. If user walks then "nearby" could probably mean the area with 1 km radius from the current location of the user, whereas if the user drives, "nearby" could mean area with radius 20 km connected with roads. We could probably infer whether user walks or drives given the speed of the user and the geographical properties of his location (city street, highway, sea or river, etc) or simply from the fact that his phone is attached to a car kit. The speed of the user's car for instance, can be calculated given the user's previous and current position and the time elapsed between the measurements of these two positions, or from the odometer if the car supports exposing that information. But to really make use of this information, collected from different sources or be able to calculate it on demand, we need a common, agreed upon context model. We believe that such model can be specified by means of ontologies.

In our approach the service requestor creates a description of a virtual, desired service – a service that completely matches his requirements. Furthermore, the service requestor provides his contextual information, e.g. location, calendar, buddy list, etc. The matching then involves comparison of the requested service description with the registered ones by using the knowledge in the common service and context ontologies. An implementation of this approach is described in the next section.

4. Design

Figure 3 depicts the design of the UDDI+ server:



1. Standard UDDI Publish interface (DAML-S is embedded in the UDDI Publish message
2. Standard UDDI Inquiry interface
3. Inquiry+ interface
4. Ontology management interface

Figure 3

When the UDDI+ server receives a publish message for a service, the proxy first forwards it to the standard UDDI server. This will handle the message and return a Universal Unique Identifier (UUID) of the new or updated service. The proxy checks if the pub-

lished businessService contains a value for a predefined t-Model that defines its value as a DAML-S description. If so this description is stored in the DAML-S repository using the UUID assigned by the UDDI server for identification. The same technique is used to extract service expiration information. A service can hereby specify that it is only available for a limited period of time. This information is forwarded to the scheduler which is responsible for removing services that have expired from the UDDI registry and the DAML-S repository.

For simple inquiry tasks the standard UDDI inquiry interface can still be used. For more advanced queries that can use the DAML-S data, the Inquiry+ interface has been added. By use of this interface the service requestor is able to pose an advanced query (DAML-S description of a virtual, desired web service) and provide information about his context. The matcher component then uses the knowledge in the ontology database to discover all services that match the user request and filters out the services that are not relevant to the user based on his context. Depending on the return type of the invoked Inquire+ method the matcher can then retrieve the UDDI service data from the standard UDDI server using the UUID stored with the DAML-S data. This makes it possible to feed the results to existing UDDI aware tools.

Finally, an interface has been added for managing the ontology database. It currently contains methods for reading and modifying the ontologies. This way, external tools can discover supported concepts, use them and/or add new concepts and relate them to existing ones.

We adopt and extend the matching algorithm proposed in[14]. The main rationale behind this algorithm is that an advertisement matches a request when the service provided by the advertiser can be of some use to the service requestor. A match between an advertisement and a request consists of the match of all the outputs of the request against the outputs of the advertisement and all the inputs of the advertisement against the inputs of the request. The degree of match between two inputs or two outputs depends on the relation between the concepts associated with these inputs and outputs. The algorithm proposes 4 degrees of match - exact match, plug-in match, subsumes match and fail. The detailed algorithm is described in[14].

In our approach we use an upper context-ontology that facilitates the context matching. Figure 4 gives a high-level view of this ontology. The ontology incorporates external domain specific ontologies as much as possible. For example device capabilities are taken from the CC/PP (composite capabilities, personal preferences)[15] RDF ontology, where as the buddy lists uses the vCard[16] ontology. This makes it much easier to integrate information from external sources. For example the CC/PP profile from mobile devices are typically provided by the manufacturer of the device.

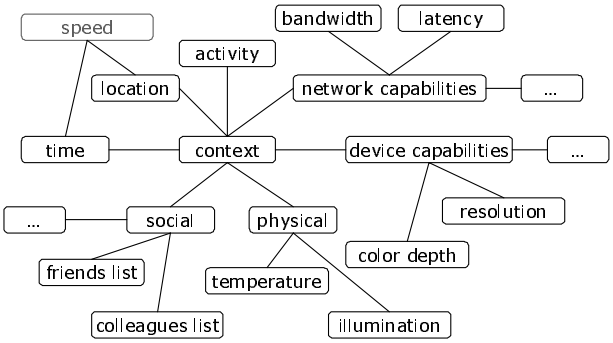


Figure 4

Once services are discovered by the service matcher component, based on their types, capabilities and models, the context matcher component filters out the services that are not relevant for the user. To perform this action, it uses the context ontology and domain-specific rules.

Currently, we are developing a tool to support the use of our UDDI+ server. The tool enables easy creation and management of domain-specific DAML-S and context ontologies, service profiles and user requests. It also automates the publishing and inquiry of service profiles.

An ontology is about shared understanding of some specific domain of interest. It provides interoperability and knowledge sharing and reuse and that is why its creation requires a domain-specific expertise. However, we do not expect domain experts to be also DAML-S experts. Therefore, one of the main roles of our tool is to provide an intuitive, easy to use, visual interface for domain experts to create domain-specific service and context ontologies.

Furthermore, we do not expect service providers and service requestors to have any knowledge about DAML-S, the domain-specific ontology and the UDDI+ publication/inquiry process. For that reason,

another important role of our tool is to hide the complexity of the ontologies as well as the complexity of the UDDI+ publication/inquiry process.

Finally, we need a tool to monitor and manage our UDDI+ server. This functionality is crucial because it increases the quality of the data stored in our UDDI+ server.

The tool consists of two main parts – a DAML-S API that allows for programmatic manipulation of DAML-S domain-specific ontologies and their instances, and a dynamically generated user interface that allows for visual manipulation of these ontologies and instances.

The tool is aware of the DAML+OIL class and property restrictions and can automatically validate the user input. Another important feature of the tool is the WSDL-to-DAML mapper (figure 5) – a component that allows service providers to visually assign meaning to their WSDL operations and parameters.

Figure 5

5. Related Work

The DAML-S Coalition[12] (BBN Technologies, Carnegie Mellon University, Nokia Research Center, SRI International, Stanford University and Yale University) presents an approach for translating DAML-S descriptions to UDDI records[17]. In this work the description of the provider of the service is mapped onto an instance of the UDDI BusinessEntity that is used as a representation of the Business that delivers the service. The mapping of the other attributes requires the specification of a set of UDDI t-models, one for each attribute from the DAML-S Profile representation. In this way services can be found and retrieved using the standard UDDI keyword search. In addition, DAML-S descriptions are stored in an advertisement database, enabling semantic matching by using a DAML+OIL reasoner.

The University of Carnegie Mellon is developing an agent capability description language called LARKS (Language for Advertisement and Request for Knowledge Sharing)[18]. Middle or matchmaking agents use this language to pair service-requesting with service-providing agents, such that the requesting agents' requirements are met. LARKS is expressive and capable of supporting inference. It also incorporates application domain knowledge in agent advertisements and requests. Domain-specific knowledge is specified as local ontologies in the concept language ITL.

DReggie[19] is a Jini[20] based service discovery infrastructure which uses a Prolog reasoner to do a semantically richer service discovery. It is an attempt to enhance the matching mechanisms in Jini and other service discovery systems. The key idea in DReggie is to enable service discovery based on semantic information associated with the services. In the DReggie system, a service discovery request contains the description of an “ideal” service - one whose capabilities match exactly with the requirements. Thus, matching now involves comparison of requirements specified with the capabilities of existing services. Depending on the requirements, a match may occur even if one or more capabilities do not match exactly.

6. Future work

Service composition will play an increasingly important role in the future. The current approaches for automating this process do not consider the context of the web services nor the context of the service requestors. In our future work, we would like to extend the functionality of UDDI+ server such that it not only discover, but also automatically composes new services from existing ones taking into account their context.

DAML-S is not enough to facilitate the desired functionality mentioned above. In our future work we want develop an upper ontology for describing the service’s and user’s context. Moreover, we expect that the future web services will be able to adapt themselves to the context of the user. In order to discover such services or compose new ones, we need a model that can

describe the adaptability features of web services. To achieve this goal we plan to make use of process ontologies that define a web service composition taking into account rules based on the context of both the service and requestor.

7. Conclusions

We have presented a new approach to enhance the publication and discovery mechanisms of the existing UDDI service registries with concepts coming from the semantic web movement. This is necessary because of the increasing requirements from the actual use of the web service paradigm. Especially in context-aware applications, with a highly dynamic user context, a service registry should be able to serve a relevant service offering based on complex and less deterministic queries. These queries can be answered by adding semantics to the registry. During publication, providers, their services and service properties can be related to existing ontologies. This enables more efficient and powerful search in the service registry. A tool will be developed that enables easy creation and management of domain specific ontologies, and semi-automates the publication of new services by relating them to the existing ontologies.

Our developments are part of the WASP project[21]. WASP aims at development of a mobile, service platform, that operates in multi-operator and multi-vendor environments and provides intelligent discovery and integration of broad and dynamic range of services.

References

[1] Simple Object Access Protocol (SOAP)
http://www.w3.org/TR/SOAP/

[2] Web Services Description Language (WSDL)
http://www.w3.org/TR/wsdl

[3] Universal Description, Discovery and Integration (UDDI)
http://www.uddi.org/

[4] Web Services Architecture
http://www.w3.org/TR/2002/WD-ws-arch-20021114/

[5] Semantic web
http://www.w3.org/2001/sw/

[6] The Semantic Web,
Tim Berners-Lee, James Hendler and Ora Lassila
http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

[7] RDF Vocabulary Description Language: RDF Schema
http://www.w3.org/TR/rdf-schema/

[8] Ontology Inference Layer (OIL)
http://www.ontoknowledge.org/oil/

[9] DARPA Agent Markup Language (DAML)
http://www.daml.org/

[10] DAML+OIL (March 2001) Reference Description
http://www.w3.org/TR/daml+oil-reference

[11] Web Ontology Language (OWL)
http://www.w3.org/TR/owl-ref/

[12] DAML-S Coalition, DAML Services,
http://www.daml.org/services/

[13] DAML-S
http://www.daml.org/services/daml-s/0.7/

[14] Semantic Matching of Web Services Capabilities, Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara
http://www.daml.org/services/ISWC2002-Matchmaker.pdf

[15] Composite Capabilities/Preferences Profile
http://www.w3.org/Mobile/CCPP/

[16] Representing vCard Objects in RDF/XML
http://www.w3.org/TR/vcard-rdf

[17] DAML-S Coalition, Importing Semantic Web in UDDI,
http://www-2.cs.cmu.edu/~softagents/papers/Essw.pdf

[18] University of Carnegie Mellon, Language for Advertisement and Request for Knowledge Sharing,
http://www-2.cs.cmu.edu/~softagents/larks.html

[19] University of Maryland , DReggie: Semantic Service Discovery for M-Commerce Applications,
http://daml.umbc.edu/papers/dreggie.pdf

[20] Jini[tm] Network Technology
http://wwws.sun.com/software/jini/

[21] WASP project
http://www.freeband.nl/projecten/wasp/ENindex.html