Integrating Semantic Web Services and Matchmaking into ebXML Registry

Stefan Schulte, Melanie Siebenhaar, and Ralf Steinmetz

Multimedia Communications Lab (KOM)
Technische Universität Darmstadt, Germany
schulte@kom.tu-darmstadt.de

Abstract. While the "Universal Description, Discovery and Integration" (UDDI) service registry standard has drawn great attention by the research community, it has not been widely adopted by the software industry. Objections towards UDDI include technical as well as conceptional arguments. Being an official ISO standard and providing a number of features UDDI is missing, "Electronic Business using Extensible Markup Language" (ebXML) Registry could be an adequate alternative for the implementation of service registries and/or repositories. However, little work has been done regarding the integration of Semantic Web Services (SWS) into ebXML Registry.
In this paper, we present a solution extending the ebXML Registry by capabilities to handle and provide SWS. This includes a concept for the integration of SWS into ebXML Registry as well as a prototypical implementation using SAWSDL and the open source framework freebXML.

1 Motivation

One of the primary application areas of SWS is service discovery, which has been a major topic from the very beginning of SWS research [19], [22]. In most scenarios, services are registered at some kind of service catalogue, which can be searched by (potential) service consumers. Apart from proprietary solutions, the SWS research community has mostly deployed UDDI as service registry standard and a multitude of solutions to include SWS into such registries have been proposed (cp. Section 2).

Even though UDDI is still deemed to be one of the key building blocks of service-oriented computing, it suffers from some major drawbacks. While conducting research on query formulation, we had to learn that it is difficult to use UDDI as a starting point for an advanced query formalism applied in SWS discovery [25].

A comparison of some major features provided by ebXML Registry and UDDI, which are important for the selection of a registry standard, are shown in Table 1: First, UDDI provides by default only a registry, where metadata about artifacts is stored. The actual artifacts (e.g., service descriptions making use of the "Web Service Description Language" (WSDL)) are not stored in UDDI. Instead, references to these artifacts are published in the registry. This

Table 1. Comparison of Registry Standards ebXML and UDDI (cp. [3], [28])

Category/Feature	ebXML Registry 3.0	UDDI 3.0
Service description standards	WSDL 1.1	WSDL 1.1
Registry	YES	YES
Repository	YES	NO
Object-oriented information model and API	YES	NO
Extensible information model	YES	NO
User-defined queries	YES	NO
SQL query syntax	YES	NO
XML query syntax	YES	YES
JAXR API	YES	YES

aspect of UDDI has been deemed as a major drawback by the software industry, as it makes it difficult to establish service life cycle management for Service-oriented Architectures[1] (SOA) and therefore, service and SOA governance [28]. In contrast to UDDI, the ebXML Registry provides both, a registry and a corresponding repository. Hence, besides the metadata, also the artifacts themselves are published in ebXML Registry.

Second, UDDI makes use of a relatively flat data model, which cannot be extended, whereas ebXML Registry offers an object-oriented and extensible information model and Application Programming Interface (API). Finally, while both registry standards can be used by utilizing the "Java API for XML Registries" (JAXR), which provides a uniform way for communicating with a registry, search facilities differ as ebXML offers enhanced querying capabilities by providing SQL support and user-defined queries in comparison to UDDI, which is only able to process XML-based queries.

Regarding the integration of an advanced query formalism into a service registry (cp. [25]), this is a major point and has been our main motivation to restrain from making use of UDDI. As there has been little work on the application of ebXML Registry in a generic SWS discovery framework, we have developed our own solution which is presented in this paper.

The remainder of this paper is structured as follows: In Section 2, we will give an overview of SWS integration approaches into registries. Afterwards, we give an overview on ebXML Registry. In Section 4, we introduce a solution to integrate SWS into ebXML Registry and a first implementation including an interface for matchmakers. The paper closes with a conclusion and an outlook on our future work.

[1] `http://www.zdnet.com/blog/service-oriented/ibm-acknowledges-bypassing-uddi-calls-for-new-soa-registry-standard/864`, `http://www.computing.co.uk/vnunet/news/2188598/ibm-calls-soa-discovery`, accessed at 2010-09-04

2 Related Work

The integration of SWS descriptions in service registries has been examined in a multitude of approaches, mostly making use of UDDI as service registry standard. In their seminal work on SWS, Paolucci et al. present the integration of "DARPA agent markup language for services" (DAML-S) profiles in UDDI [22]. The authors propose the mapping of a service profile to UDDI records. Besides the DAML-S/UDDI mapping, an external matchmaker architecture is suggested by the authors, which uses DAML ontologies publicly available on the Web for semantic capability matching. In this approach, it is possible to search for services using UDDI keyword-based search and a capability matching engine, if requests are specified in the DAML-S format. Several authors have proposed enhancements of the work by Paolucci et al., e.g., regarding the usage of UDDI-internal matchmakers [1], the application of the "Web Ontology Language for Web Services" (OWL-S) or semantically enhanced WSDL instead of DAML-S [26], [27], or the integration of functionalities enabling the usage of semantic search in UDDI on the client-side instead of altering the UDDI implementation [17].

There are further approaches to integrate SWS into service registries in general SWS frameworks, with METEOR-S [29] and the "Web Service Modeling eXecution Environment" (WSMX) being prominent examples. However, in both approaches the actual registry is more a means to an end than in the focus of the work. In an early "Web Service Modeling Ontology" (WSMO) Registry Working Draft, UDDI was intended to provide registry functionalities [12]. However, for WSMX, which is the reference implementation of WSMO, no further information is given if a particular registry standard has been applied or not. In fact, WSMX's *Resource Manager* is an internal registry [11]; furthermore, it is stated that an ebXML- or UDDI-based registry *could* be used for WSMX data persistence [5]. More recently, Kourtesis et al. have proposed a combination of SAWSDL, OWL DL, and UDDI (Version 2.0) for semantically enhanced Web service discovery in the FUSION Semantic Registry [16]. While this framework does not rely on any specific SWS standard, the reference implementation presented is based on SAWSDL. Neither the UDDI server nor its specification API are altered, but are wrapped in the semantic registry.

A different approach to the integration of semantic information in service registries has been implemented in PYRAMID-S [23]. Actually, PYRAMID-S is an overlay to service registries which uses a hybrid peer-to-peer topology to manage heterogeneous service registries. The aim of the framework is to allow unified Web service publication and discovery, which does not adhere to a particular service registry standard. As PYRAMID-S facilitates the usage of different service registry standards, it is necessary to define mediators for the designated standards. Mediators for UDDI (based on [6]) and ebXML (based on [4]) have already been defined. There are several differences between PYRAMID-S and the work at hand: Most importantly, in PYRAMID-S, only those matchmakers provided by ebXML Registry are explicitly regarded. The authors give no information on how to extend a registry's matchmaking capabilities. As ebXML

Registry does not provide any semantic matchmaker, matching is limited to syntax-based query statements and "non-fuzzy" semantic matchmaking, i.e., a semantic annotation in a service advertisement has to be *exactly* the same as specified in a service request. Furthermore, Pilioura and Tsalgatidou make use of their own WSDL variant, namely PS-WSDL and do not regard, e.g., SAWSDL or OWL-S.

Dogac et al. introduce another approach, which incorporates the integration of "Web Ontology Language" (OWL) ontologies into ebXML registries in order to enhance service discovery [8]. The work by Dogac et al. has been committed as an OASIS Committee Draft for an *ebXML Registry Profile for Web Ontology Language* [7], which could be used to integrate OWL-S services into ebXML. Notably, this work is limited to OWL Lite, while OWL-S ontologies are written in OWL DL [2]; there is no information given on how this contradiction is handled. The authors define a mapping of OWL elements to ebXML class hierarchies, which can be performed automatically from a given OWL ontology. Concerning the suggested mapping, OWL classes are represented through classification nodes in ebXML, while RDF properties are modeled using ebXML associations. This allows to represent whole OWL class hierarchies through ebXML elements. Finally, stored procedures are defined in order to handle the OWL semantics, e.g., to obtain all the super- or subclasses of a given class. These stored procedures can then be utilized by users in order to retrieve appropriate services that are classified using the OWL classification nodes from the ebXML registry [8]. This way, this solution is very inflexible, as it does not account for inferred semantic relationships and relies on querying predefined semantic hierarchies.

In their work on making use of SPARQL as means to define preconditions and effects in SWS descriptions, Iqbal et al. also use ebXML Registry [13]. The registry is used to store SAWSDL-based service descriptions, while SPARQL-based conditions are stored separately in the repository infrastructure. The authors state that their ebXML-based service repository does not (yet) allow to query for the integrated semantic metadata. The authors suggest to store the semantically enhanced service descriptions within the ebXML infrastructure and indicate a mechanism to reference additional semantic information in form of SPARQL-based conditions. Unfortunately, the details of this approach are not stated explicitly.

As it can be seen, existing solutions to integrate SWS into ebXML Registry are either constricted to the elements needed in a particular matchmaking approach [8], [13] and/or rely on the existing matchmakers provided by ebXML Registry [23]. In contrast, the solution at hand has been designed in order to provide a generic framework for SWS discovery using ebXML Registry.

3 ebXML Registry – Overview

In 1999, ebXML[2] has been initiated by OASIS and the United Nations/ECE agency CEFACT. In general, it provides a modular suite of specifications for en-

[2] `http://www.ebxml.org/geninfo.htm`, access at 2010-09-04

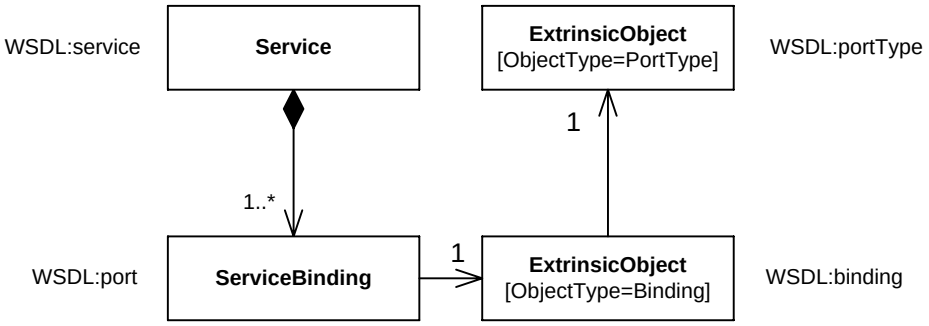

Fig. 1. Mapping WSDL Information to the ebXML RIM

terprises to perform business over the Internet (e.g., message exchange, registration of services), from which the specifications for registries and repositories are relevant within the work at hand. In this context, two documents are currently available as approved OASIS standards: the *ebXML Registry Information Model* (RIM) [9] and the *ebXML Registry Services and Protocols* (RS) [10]. ebXML RIM and ebXML RS have been standardized as ISO 15000, Parts 3 and 4, respectively. The former specifies the underlying data model of the registry, i.e., the metadata classes, whose instances are used to describe the objects stored in the repository, and the latter describes the functionalities provided by the registry and the protocols used for interacting with the registry.

An ebXML registry may further implement different profiles, each defining a processing standard as well as extensions and restrictions of the core ebXML features for a specific type of content. The *ebXML Registry Profile for Web Services* (WS) defines the publication, management and discovery of Web service artifacts [21]. The RIM classes which are relevant for the registration of Web services in an ebXML registry implementing the Web Service profile are depicted in Figure 1.

A service is represented by an instance of the `Service` class, which itself contains one or more instances of the `ServiceBinding` class providing technical information (e.g., the access URI) on how to access a concrete service instance in a specific way. The class `ExtrinsicObject` represents the primary metadata class for items stored within the repository [9]. To specify the type of content submitted to the repository represented by an instance of the `ExtrinsicObject` class, classification schemes are used. In standard ebXML implementations, WSDL files are stored as `ExtrinsicObject` instances and classified with the WSDL classification node. When submitting a WSDL document to the registry, a corresponding *Cataloging Service* is invoked which performs a mapping of the WSDL components to the ebXML RIM [21]. This is due to the fact, that the default service information model as part of the RIM also supports the registration of other types of services than Web services [9], i.e., represents a generic service model. Consequently, the components which are specific to a certain kind of service have to be stored as extrinsic objects and classified using custom-built classification

schemes. So far, ebXML WS has only been defined for WSDL 1.1-based service descriptions.

4 Solution Approach and Prototypical Implementation

In order to integrate SWS into ebXML and provide appropriate service discovery facilities, it is necessary to provide solutions for the following issues:

- Integration of SWS descriptions
- Integration of query formulations
- Integration of matchmaking capabilities

Regarding the integration of query formulations, we refer to our former work presented in [25]. In the following, we will focus on the integration of SWS and the provision of a matchmaking interface in ebXML; query formulation is only regarded if necessary to complete a particular consideration. Afterwards, we present a prototypical implementation using *freebXML 3.1*[3].

4.1 Integration of SWS Descriptions

In order to integrate SWS descriptions into ebXML Registry, it is necessary to enhance the ebXML RIM by a new classification node, e.g., called `SWS`. Using the newly created object type `SWS`, it is possible to classify SWS objects and to distinguish between non-semantic and semantic Web services. The handling of these new objects has to be implemented in the corresponding ebXML Registry realization (cp. Section 4.4). We determine SWS descriptions to be published using a subclass of the new classification node `SWS`; the corresponding WSDL information is published by the standard publication mechanism. In doing so, we assume SWS descriptions to make use of a WSDL grounding, as provided by, e.g., WSMO and OWL-S [14], [15], [18].

SAWSDL services can be published without any modifications using the standard WSDL cataloging service of ebXML registrations. However, it is still necessary to publish the semantic information described in SAWSDL using a separate node in order to make a differentiation for syntax- and semantic-based service discovery.

4.2 Integration of Matchmaking Capabilities

Per se, ebXML Registry offers syntax-based matchmaking capabilities based on service queries defined using SQL or so-called ebXML filters [10]. In order to enable semantic-based service discovery, it is necessary to provide new matchmaking facilities.

Therefore, within the work at hand, an exemplary matchmaker is directly integrated into a service registry as proof-of-concept. A direct integration mechanism for matchmakers into registries demands a generic concept, i.e., the creation

[3] `http://ebxmlrr.sourceforge.net/`

and provision of interfaces. In doing so, further matchmakers can be introduced without changing existing classes, but only through the implementation of additional classes.

Since the registry should also provide semantic matching capabilities, the management of ontologies has to be addressed. Regarding service (information) life cycle management as well as service and SOA governance, (information about) ontologies need to be managed by a service registry itself. Ontologies needed in matchmaking depend on the services published in a registry. Thus, a flexible mechanism for the management of ontologies is required, so that new ontologies can be added at any time. For this, a semantic reasoning engine can be integrated into a service registry in conjunction with an ontology knowledge base, where arbitrary ontologies can be registered. For example, a service provider could register the necessary ontologies together with the service offers at publication time. If a query is enhanced with semantic information, the syntax-based part can be directed to the standard search facilities provided by a registry and the additional semantic information can be directed to semantic matchmakers to allow for real reasoning support.

4.3 freebXML – An Open Source Reference Implementation of ebXML Registry

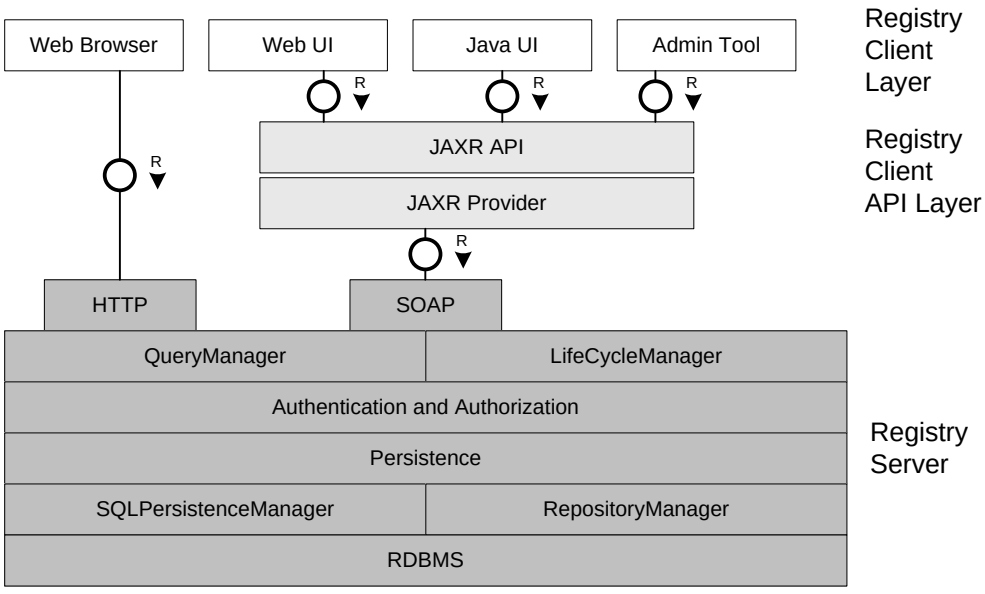

Fig. 2. The freebXML Architecture

For our prototypical implementation, freebXML 3.1 was used and enhanced. freebXML is an open source reference implementation of the OASIS ebXML Registry standards [9], [10]. freebXML is made up from a registry, where metadata about artifacts can be published, and a repository, where the actual artifacts are stored. In general, an ebXML registry may implement different profiles, i.e., provide functional enhancements for a specific type of content. Concerning profiles, freebXML implements, among others, ebXML WS [21]. Basically, the freebXML architecture comprises the three parts depicted in Figure 2: the *Registry Client Layer*, the *Registry Client API Layer* and the *Registry Server*.

The Registry Client Layer depicts possible registry client types, which may be represented by a Web browser with direct HTTP access for querying purposes only, a thin client Web User Interface (UI) running within some Web container, that can be accessed via a Web browser, a fat client Java UI running on a client machine and a command line interface, e.g., in the form of an administration tool. Instances of the last three client types are provided by the freebXML registry.

The Registry Client API Layer is the subsequent layer in the architecture. It is represented by the *JAXR API*, which provides standard Java interfaces to access the registry. The JAXR API requires a JAXR provider, which represents an implementation of the JAXR API. The freebXML registry contains its own *JAXR Provider*, which represents an advanced implementation of the JAXR API.

Concerning the freebXML Registry Server, a HTTP and a SOAP interface are provided. For browsing and discovery capabilities, the HTTP and the SOAP interface offer a binding to the *QueryManager* interface, while the SOAP interface also provides a binding to the *LifeCycleManager* interface for publishing content. Furthermore, modifying access to the registry requires user authentication and authorization. For this, a security layer is part of the registry server. In order to store content and metadata, an abstract persistence layer is defined by the registry. For the actual storage of content and metadata, a relational database management system (*RDBMS*) is used. By default, freebXML makes use of Apache Derby, which runs in the same Java Virtual Machine as the registry server. To manage the persistence of the registry server and the repository items, an SQL persistence manager interface and an repository manager interface are provided, respectively.

A comprehensive presentation of the freebXML registry architecture is given at the project's Web page[4].

4.4 Prototypical Implementation

Based on the defined requirements, we have prototypically enhanced freebXML as a generic SWS discovery framework. As SWS formalism, we make use of SAWSDL and WSDL 1.1. Furthermore, we make use of a "query by example"-approach, i.e., a SAWSDL description needs to be provided by the service requester. In the following, the integration of SAWSDL into ebXML Registry, an

[4] `http://ebxmlrr.sourceforge.net/`

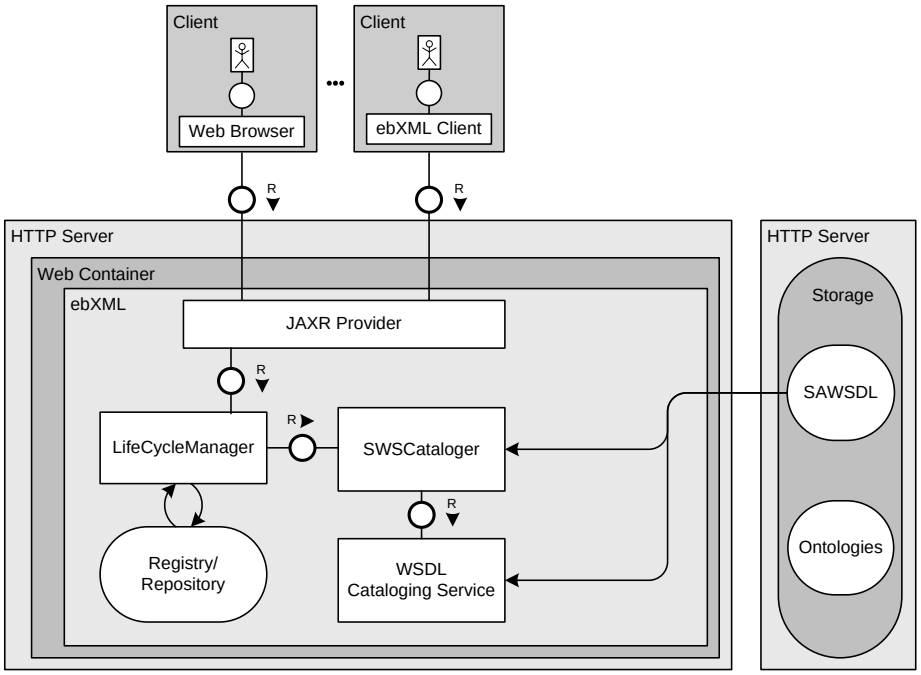

Fig. 3. Enhanced ebXML Architecture (Publishing)

interface for arbitrary matchmakers, and the utilization of an example matchmaker are presented.

Apart from freebXML, the following major software components have been used in our prototypical implementation:

Apache Tomcat 5.0.28 is the Web container freebXML is deployed to,
Java JDK 5.0 Update 22 is deployed for enhancements of freebXML and the development of a registry client, and
Apache HTTP Server 2.2 is used to store Web service descriptions and ontologies.

Integration of SAWSDL into ebXML Registry: For SAWSDL-based service descriptions, a classification node `SAWSDL` has to be created in the registry; as presented in Section 4.1, `SAWSDL` is derived from `SWS`. Generally, depending on the type of content, an associated cataloging service, which extracts the required information from submitted services, is invoked on publication time. The information is then mapped to instances of the ebXML RIM representing the content's metadata, while the actual content is stored within the repository,

In detail, by submitting SAWSDL documents to the registry, they are classified using `SAWSDL`. The *LifeCycleManager* (cp. Figure 3) of the registry makes a call to the appropriate cataloging service associated with the `SAWSDL` object

type. For this, a new cataloging service *SWSCataloger* has been developed. On invocation, the SWSCataloger first makes a call to the standard *WSDL Cataloging Service* of freebXML, which performs a normal publication of the WSDL information associated with the SWS document.

Finally, the published WSDL information has to be associated with the SAWSDL representation of the service. For this, ebXML provides the ability to relate any two objects in the registry using arbitrary relationship types. The resulting registry and repository objects are then passed to the *LifeCycleManager*, which submits the contents to the *Storage* database of freebXML.

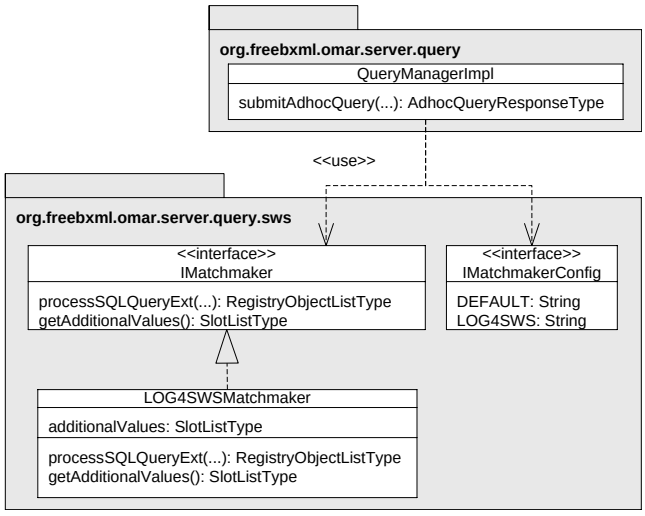

Fig. 4. Registry Enhancements for the Integration of Matchmakers

Matchmaker Interface: For the integration of different matchmakers, the QueryManager implementation of freebXML has been modified and additional classes have been created. An overview of the classes is depicted in Figure 4. In detail, each SQL query that is submitted to the registry is scanned for an SQL extension, i.e., an enhanced SQL query string indicating that a semantic "query by example" has been submitted as service request. If no extension is found, the SQL query is processed in the usual way. Else, the enhanced query string is forwarded to the semantic matchmaker.

Possible matchmakers can be registered within the `IMatchmakerConfig` class in the form of a string-based constant that is associated with the fully qualified name of the main class of a matchmaker. Using the Java Reflection API[5], the

[5] `http://java.sun.com/docs/books/tutorial/reflect/index.html`

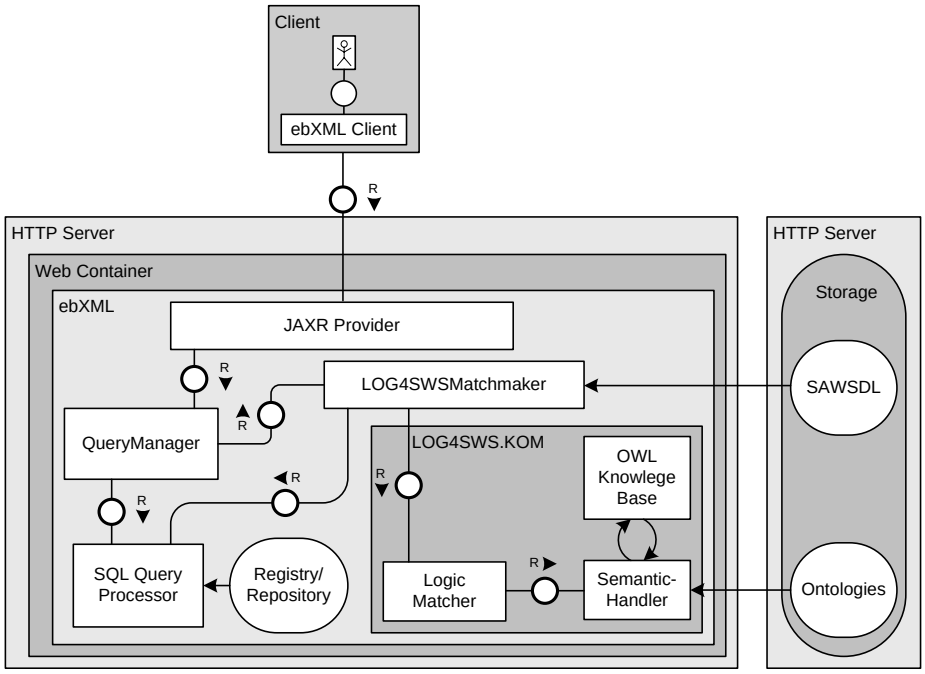

Fig. 5. Enhanced ebXML Architecture (Discovery)

QueryManager implementation is able to redirect a query to the desired matchmaker. For the integration of a new matchmaker, the `IMatchmaker` interface can be implemented by the respective class. In doing so, two methods have to be realized by the new matchmaker; one for processing the SQL query extensions and a second method to return an optional response slot list, which can be used to send back additional information (e.g., a similarity value) of a matching service to the client. The optional response slot list is provided by default by the freebXML registry implementation as part of the response to an SQL query. For the proof of concept implementation, the `LOG4SWSMatchmaker` class has been created (see below), which establishes the connection to the matching facilities provided by our matchmaker LOG4SWS.KOM [24]. In order to retrieve the set of available service descriptions from the registry, the LOG4SWSMatchmaker class makes use of the querying facilities provided by the *SQL Query Processor* of freebXML. Each service object that matches the query is added to the final result set, which is then sent back to the respective client.

The process for the retrieval of SWS is illustrated in Figure 5. First of all, a "query by example"-enhanced SQL query string is created by a service requester using the ebXML client. Afterwards, the matchmaker processes the SAWSDL service descriptions according to its specific matchmaking algorithm. Finally, the result is sent back to the ebXML client.

Example Matchmaker: For the prototypical implementation, an exemplary matchmaker for SAWSDL, namely LOG4SWS.KOM, is utilized, which has been presented in [24] and is also participating in this year's "Annual International Contest S3 on Semantic Service Selection Retrieval Performance Evaluation of Matchmakers for Semantic Web Services" (S3 Contest)[6]. LOG4SWS.KOM takes a "query by example" as service request and is provided with a set of service offers by the ebXML Registry.

By default, LOG4SWS.KOM aims at performing a logical subsumption matching, which accounts for the semantic annotations on the different component levels of a SAWSDL-based service description. These components comprise interfaces, operations and parameters (i.e., inputs and outputs). For each component level, an individual similarity value is computed during the matching process, which is then aggregated to a global similarity value for the whole service based on predefined weights for each level. In doing so, the resulting, individual similarity values representing classical Degrees of Match [22] are transformed into numerical representations between 0 and 1 for their combination. Since semantic annotations may not be present on all service component levels or required ontologies may be missing within the OWL knowledge base of a registry or may even fail to load, LOG4SWS.KOM also provides a fallback strategy, which is applied in these situations. The fallback strategy then processes the remaining information, i.e., the names of the components in the first case or the names of the semantic concepts in the other cases. The similarity measure is then determined using the WordNet ontology [20], which represents a semantic net of words for the English language. Based on the distance of a pair of words in WordNet, the similarity value is computed.

In order to be able to determine subsumption relationships between the concepts specified within a service query and the concepts specified within a service offer, a reasoning engine has to be integrated into the registry. For this purpose, a *SemanticHandler* is part of LOG4SWS.KOM, which provides an interface to access a semantic reasoner (here: Pellet 2.0) and an OWL knowledge base. Since the initialization of the semantic reasoner and the OWL knowledge is a very time consuming task, it cannot be performed once again on every incoming service query. Therefore, an instance of the SemanticHandler has been integrated into the *RepositoryManagerFactory*, which represents a permanent and unique instance within the freebXML registry, so that the SemanticHandler and the corresponding OWL knowledge base can be initialized once upon registry startup. Within the work at hand, it is further assumed, that the required ontologies for discovery already exist within the OWL knowledge base, when a query is submitted to the registry. For this purpose, a list with the required ontologies is provided to the SemanticHandler, so that the necessary ontologies are loaded into the OWL knowledge base at registry startup.

[6] `http://www-ags.dfki.uni-sb.de/~klusch/s3/html/2010.html`

5 Conclusion

Even though ebXML Registry provides a valid alternative to the usually applied UDDI-based service registries, surprisingly little work has been done regarding the application of ebXML Registry in SWS discovery frameworks. In the paper at hand, we have presented a corresponding approach, which has been prototypically implemented deploying SAWSDL as SWS formalism and freebXML as ebXML implementation.

Also our prototypical implementation is fully operational, we consider it primarily as a foundation for more sophisticated solutions. Among other thing, we want to address the following questions in our future work:

In this paper, we have proposed a quite lightweight interface for matchmakers. The interface is based on the assumption that service requests are formulated using a "query by example"-approach. However, as we have stated in our previous work [25], it might be helpful to provide more sophisticated and fine-grained query formalisms for SWS. Thus, the matchmaker interface might be replaced by a more heavyweight one; the features such an interface should provide are subject to a discussion in the research community.

Second, the prototypical implementation is restricted to SAWSDL-based service descriptions. In our opinion, the heterogeneity of SWS formalisms is a major obstacle especially regarding service discovery as, e.g., it is not possible to find an OWL-S service profile based on a SAWSDL-based service request. In our future work, we want to address this issue by providing a sophisticated, unified query language in ebXML Registry.

Finally, as presented in Section 2, matchmakers can either be integrated into registries or service queries can be intercepted to allow for a redirect to specific external matchmakers. Both approaches require a modification of the registry's source code. However, using the first approach, a repetitive modification of the source code is necessary, when new matchmakers are to be integrated. In the second approach, a generic redirection mechanism can be implemented, so that a modification of the registry's source code has to be only performed once. Although, a generic redirection mechanism is preferable, it is far more complex. Thus, we used the first-mentioned approach in the work at hand. In our future work, we will examine the integration of a generic redirection mechanism for service requests to external matchmakers.

Acknowledgements. This work is supported in part by the E-Finance Lab e. V., Frankfurt am Main, Germany (`www.efinancelab.de`).

References

1. Akkiraju, R., Goodwin, R., Doshi, P., Roeder, S.: A Method for Semantically Enhancing the Service Discovery Capabilities of UDDI. In: Workshop on Information Integration on the Web (IIWeb-03) at Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03). pp. 87–92 (2003)

2. Ankolekar, A., Martin, D., McGuinness, D., McIlraith, S., Paolucci, M., Parsia, B.: OWL-S' Relationship to Selected Other Technologies. W3C Member Submission (November 2004), http://www.w3.org/Submission/OWL-S-related/, access at 2010-08-12
3. Châtel, P.: Service Registries Study. Thales Study (June 2006), http://www.chatelp.org/work/LUCAS_registry_study.pdf, last access at 2010-04-13
4. Chiusano, J.M., Najmi, F.: Registering Web Services in an ebXML Registry, Version 1.0. OASIS Technical Note (March 2003), http://www.oasis-open.org/committees/download.php/11907/regrep-webservices-tn-10.pdf, access at 2010-01-24
5. Cimpian, E., Zaremba, M. (eds.): Web Service Execution Environment (WSMX). W3C Member Submission (June 2005), http://www.w3.org/Submission/WSMX/, access at 2010-04-03
6. Colgrave, J., Januszewski, K.: Using WSDL in a UDDI Registry, Version 2.0.2. OASIS Technical Note (June 2004), http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm, access at 2010-01-24
7. Dogac, A. (ed.): ebXML Registry Profile for Web Ontology Language (OWL). OASIS Committee Draft (September 2006), http://docs.oasis-open.org/regrep/v3.0/profiles/owl/regrep-owl-profile-v1.5.pdf, access at 2010-09-05
8. Dogac, A., Kabak, Y., Laleci, G., Mattocks, C., Najmi, F., Pollock, J.: Enhancing ebXML Registries to Make them OWL Aware. Distributed and Parallel Databases 18(1), 9–36 (2005)
9. Fuger, S., Najmi, F., Stojanovic, N. (eds.): ebXML Registry Information Model Version 3.0. OASIS Standard (May 2005), http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.pdf, access at 2010-02-09
10. Fuger, S., Najmi, F., Stojanovic, N. (eds.): ebXML Registry Services and Protocols Version 3.0. OASIS Standard (May 2005), http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rs-3.0-os.pdf, access at 2010-02-09
11. Haller, A., Cimpian, E., Mocan, A., Oren, E., Bussler, C.: WSMX – A Semantic Service-Oriented Architecture. In: 2005 IEEE International Conference on Web Services (ICWS 2005). pp. 321–328. IEEE Computer Society, Washington, DC, USA (2005)
12. Herzog, R., Lausen, H., Roman, D., Zugmann, P. (eds.): D10 v0.1 WSMO Registry. WSMO Working Draft (April 2004), http://www.wsmo.org/2004/d10/v0.1/20040426/, access at 2010-04-03
13. Iqbal, K., Sbodio, M.L., Peristeras, V., Giuliani, G.: Semantic Service Discovery using SAWSDL and SPARQL. In: Fourth International Conference on Semantics, Knowledge and Grid (SKG 2008). pp. 205–212. IEEE Computer Society (2008)
14. Kopecký, J., Moran, M., Vitvar, T., Roman, D., Mocan, A. (eds.): D24.2 v0.1 WSMO Grounding. WSMO Working Draft (April 2007), http://www.wsmo.org/TR/d24/d24.2/v0.1/#grounding_wsdl, access at 2010-09-03
15. Kopecký, J., Roman, D., Moran, M., Fensel, D.: Semantic Web Services Grounding. In: Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW 2006). IEEE Computer Society, Washington, DC, USA (2006)
16. Kourtesis, D., Paraskakis, I.: Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery. In: 5th European Semantic Web Conference (ESWC 2008). LNCS, vol. 5021, pp. 614–628. Springer (2008)
17. Luo, J., Montrose, B., Kim, A., Khashnobish, A., Kang, M.: Adding OWL-S Support to the Existing UDDI Infrastructure. In: 2006 IEEE International Conference

on Web Services (ICWS 2006). pp. 153–162. IEEE Computer Society, Washington, DC, USA (2006)
18. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic Markup for Web Services. W3C Member Submission (November 2004), http://www.w3.org/Submission/OWL-S/, access at 2009-06-12
19. McIlraith, S.A., Son, T.C., Zeng, H.: Semantic Web Services. IEEE Intelligent Systems 16(2), 46–53 (2001)
20. Miller, G.A.: WordNet: a lexical database for English. Communications of the ACM 38(11), 39–41 (1995)
21. Najmi, F., Chiusano, J. (eds.): ebXML Registry profile for Web Services. Version 1.0 Draft 3. Draft OASIS Profile (September 2005), http://www.oasis-open.org/committees/download.php/14756/regrep-ws-profile-1.0-draft3.pdf, access at 2010-03-05
22. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.P.: Importing the Semantic Web in UDDI. In: International Workshop on Web Services, E-Business, and the Semantic Web (WES 2002) in connection with The 14th Conference on Advanced Information Systems Engineering (CAiSE 2002). Lecture Notes in Computer Science, vol. 2512, pp. 225–236. Springer, Berlin Heidelberg (2002)
23. Pilioura, T., Tsalgatidou, A.: Unified publication and discovery of semantic Web services. ACM Transactions on The Web 3(3), 1–44 (2009)
24. Schulte, S., Lampe, U., Eckert, J., Steinmetz, R.: LOG4SWS.KOM: Self-Adapting Semantic Web Service Discovery for SAWSDL. In: IEEE 2010 Fourth International Workshop of Software Engineering for Adaptive Service-Oriented Systems (SEASS '10) at 2010 IEEE 6th World Congress on Services (SERVICES 2010). pp. 511–518. IEEE Computer Society, Washington, DC, USA (2010)
25. Schulte, S., Siebenhaar, M., Eckert, J., Steinmetz, R.: Query languages for semantic web services. In: Informatik 2010 (FORTHCOMING). Gesellschaft für Informatik (2010)
26. Sivashanmugam, K., Verma, K., Sheth, A.P., Miller, J.A.: Adding Semantics to Web Services Standards. In: International Conference on Web Services (ICWS 2003). pp. 395–401. CSREA Press (2003)
27. Srinivasan, N., Paolucci, M., Sycara, K.P.: An Efficient Algorithm for OWL-S Based Semantic Search in UDDI. In: First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), Revised Selected Papers. Lecture Notes in Computer Science, vol. 3387, pp. 96–110. Springer, Berlin Heidelberg (2004)
28. Sun Microsystems, Inc.: Effective SOA Deployment using an SOA Registry Repository. A Practical Guide (September 2005), http://www.sun.com/products/soa/registry/soa_registry_wp.pdf, access at 2010-04-14
29. Verma, K., Sivashanmugam, K., Sheth, A.P., Patil, A., Oundhakar, S., Miller, J.: METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services. Journal of Information Technology and Management 6(1), 17–39 (2005)