Book Section:
Kourtesis, Dimitrios and Paraskakis, Iraklis (2009) Supporting Semantically Enhanced Web Service Discovery for Enterprise Application Integration. In: Mentzas, Gregoris and Friesen, Andreas, (eds.) Semantic Enterprise Application Integration for Business Processes: Service-Oriented Frameworks. IGI Global, Hershey, New York, pp. 105-130. ISBN: 9781605668048.

https://doi.org/10.4018/978-1-60566-804-8.ch006

Chapter 6
Supporting Semantically Enhanced Web Service Discovery for Enterprise Application Integration

Dimitrios Kourtesis
South East European Research Centre (SEERC), Research Centre of the University of Sheffield and CITY College, Greece

Iraklis Paraskakis
South East European Research Centre (SEERC), Research Centre of the University of Sheffield and CITY College, Greece

ABSTRACT

The availability of sophisticated Web service discovery mechanisms is an essential prerequisite for increasing the levels of efficiency and automation in EAI. In this chapter, we present an approach for developing service registries building on the UDDI standard and offering semantically-enhanced publication and discovery capabilities in order to overcome some of the known limitations of conventional service registries. The approach aspires to promote efficiency in EAI in a number of ways, but primarily by automating the task of evaluating service integrability on the basis of the input and output messages that are defined in the Web service's interface. The presented solution combines the use of three technology standards to meet its objectives: OWL-DL, for modelling service characteristics and performing fine-grained service matchmaking via DL reasoning, SAWSDL, for creating semantically annotated descriptions of service interfaces, and UDDI, for storing and retrieving syntactic and semantic information about services and service providers.

INTRODUCTION

Service-oriented computing is emerging as the dominant paradigm for enterprise computing and is changing the way business software applications are architected, developed, delivered, and consumed. The model of Service Oriented Architecture (SOA) and its manifestation through Web service technology standards promise to alleviate many of the barriers that stand on the path to Enterprise Ap-

DOI: 10.4018/978-1-60566-804-8.ch006

plication Integration (EAI) and become enablers for business agility in the modern enterprise.

In a service-oriented landscape where contemporary technologies are employed, the integration of a set of enterprise applications (such as ERP, CRM, or WMS), is typically performed by composing the reusable Web services that are exposed by the individual applications into service orchestrations which are encoded in the popular WS-BPEL language -Web Services Business Process Execution Language- (Alves, et al., 2007). A BPEL orchestration is essentially an executable program that specifies how a set of services exposed by different applications should be coordinated in order to realise a specific business process, such as order fulfilment or stock replenishment. By deploying the service orchestration on a BPEL execution engine, the fulfilled business process is externalised as a normal Web service on the corporate network, which means that it can be consumed by client applications or re-composed in new Web service orchestrations.

Web Service Discovery for Enterprise Application Integration

During the phases of construction and maintenance of a service orchestration, the business process expert needs to search and discover Web services that are suitable for carrying out each of the key activities/functions in the workflow of the envisaged business process. The Web services that will finally be selected and included in the orchestration, among the tens or hundreds of services that may potentially be available on the corporate network, have to match a number of requirements. Depending on the application domain and the type of business process that the orchestration seeks to realise, these requirements may involve functional or non-functional aspects of service operation.

In every occasion, however, an essential requirement that needs to be satisfied is the integrability of the Web service on the basis of the input and output messages that are defined in the service's interface. The ability of a Web service to be integrated in a service orchestration depends on whether proper data flow and thus proper communication can be established among the two. More specifically, proper data flow can be achieved only if the amount of data which the BPEL orchestration provides as input when it invokes a service are sufficient with regard to the amount of data that the service expects to receive, and at the same time, the amount of data that the service produces as output are sufficient with regard to the amount of data that the orchestration expects to obtain. If this condition holds, integration can be made possible even if the schema definitions of the business objects to be exchanged by the two parties along input and output messages are not identical (the heterogeneity can be overcome by applying some data mediation/transformation process).

Undeniably, in a fully SOA-enabled business application ecosystem with tens or hundreds of deployed Web services, the task of manually searching and identifying services that satisfy the above requirements for integrability can become extremely resource-intensive and error prone. This is why the existence of intelligent automated Web service discovery mechanisms that can address these needs is considered a core challenge for increasing the levels of efficiency and automation in EAI.

Web Service Discovery with UDDI

The need for efficient search and discovery of services was the original motivation behind the development of the Universal Description, Discovery and Integration (UDDI) specification as a standardised way to catalogue and discover reusable Web services (Clement, Hately, von Riegen, & Rogers, 2004). The UDDI specification was the result of an industry-driven standardisation effort led by the OASIS consortium, and its scope was not limited to providing support for EAI alone,

but for a much wider range of use cases. Primarily due to the active promotion of the standard by the enterprise software industry, UDDI quickly became one of the core standards in the Web service technology stack and an integral part of every major SOA vendor's technology strategy (see IBM WebSphere UDDI Registry, Oracle Service Registry, SAP Enterprise Services Registry, Microsoft Windows Server 2003 Enterprise UDDI Services, etc).

The UDDI specification standardises an XML-based data model for storing descriptive information about Web services and their providers, and a Web service-based application programmatic interface for publishing this information to the registry and performing discovery queries. Web service advertisements are represented as records in the registry. In order to describe the functionality of some service, its respective record contains references to external descriptions of technical specifications or to classification schemes which are developed and maintained by either third-party actors (e.g. standardisation bodies), or by service providers themselves. Numerous such references can be used for representing different aspects of a Web service's functional and non-functional properties. For the purpose of being generic, the UDDI standard does not prescribe any specific method, formal or informal, for creating these specifications and classification schemes. Overall, services advertised in UDDI registries can be searched by prospective service consumers based on one of the following criteria: i) the service's declared conformance to some technical specification, where matching is evaluated against a provided specification identifier, ii) the service's attributed categorisation within a classification system, where matching is evaluated against a provided category title, and iii) the service's name, where matching is evaluated against a provided keyword search term.

The fundamental problem with the UDDI description and discovery mechanism outlined above is that despite the fact that the available service descriptions are machine-processable, they lack the formal rigour and machine-understandable semantics that would make them amenable to logic-based reasoning and automated processing. As a result, UDDI registries cannot offer the kind of fine-grained service matchmaking functionality that would be required for supporting automated integrability-oriented service discovery in the context of EAI. With today's state of practice, a developer in a typical EAI scenario still needs to retrieve the service-related artefacts referenced by a UDDI service advertisement (and most importantly the WSDL document) and inspect them manually, in order to decide if the advertised service can be interoperable with other services assembled in a service orchestration.

Semantically-Enhanced Web Service Discovery

In order to increase the levels of automation in EAI and overcome the problem of ambiguity that currently hinders automated service discovery, service characteristics need to be described in a formal, machine-understandable manner that is amenable to processing within semantically-enhanced service registries. The use of Semantic Web technologies to represent service properties and the introduction of semantic matchmaking functionality in service registries (primarily UDDI) has been the focus of numerous works in recent years, generally within the field of Semantic Web Services (SWS) research. The vision in SWS research (Martin, Domingue, Brodie, & Leymann, 2007; Martin, Domingue, Sheth, Battle, Sycara, & Fensel, 2007) is to bring semantics into the realm of Web service specifications in order to not only enable fully automated service discovery, but facilitate the automation of a broad array of design-time and run-time activities in service-oriented computing.

In this chapter we present a new approach for developing service registries that build on the UDDI standard and offer semantically-enhanced

Web service publication and discovery capabilities. The approach aspires to promote efficiency in EAI in a number of ways, but primarily by automating the task of evaluating service integrability on the basis of the input and output messages that are defined in the Web service's interface. Overall, the semantically-enhanced service registry combines three existing standards from the domains of Web service technologies and Semantic Web technologies to address its objectives: OWL-DL (McGuinness & van Harmelen, 2004), for modelling service characteristics and performing fine-grained service matchmaking via Description Logic reasoning, SAWSDL (Farrell & Lausen, 2007), for creating semantically annotated descriptions of service interfaces, and UDDI (Clement, Hately, von Riegen, & Rogers, 2004), for storing and retrieving syntactic and semantic information about services and service providers. The approach that we put forward has been applied and validated during the development of the FUSION Semantic Registry[1], a semantically-enhanced service registry that has been utilised in research project FUSION[2] and is released as open source software.

The organisation of the chapter is as follows. Section 2 introduces the background to the discussed topic, outlines a set of requirements for Semantic Web Service discovery in the context of EAI, and provides a detailed review of related research works that focus on semantic enhancements to UDDI registries. Section 3 presents our approach for describing service characteristics in order to support integrability-oriented service discovery with the FUSION Semantic Registry. Section 4 presents an overview of the FUSION Semantic Registry architecture and its application programming interfaces. Section 5 provides a walkthrough of the core activities performed during service publication, while section 6 provides a walkthrough of the activities performed during service discovery. Lastly, section 7 summarises the key points presented in this chapter, presents an overview of how our work compares with other related works, and provides an outlook to future research directions.

BACKGROUND AND RELATED WORK

In this section we briefly introduce Semantic Web Services (SWS) as the background to the discussed topic and outline some fundamental requirements for Semantic Web Service discovery in the context of EAI. We also provide a detailed review of related research works which employ SWS technologies in order to provide enhancements for UDDI-based service registries, and contrast each of these works with the requirements set for discovery in the context of EAI. Note that a detailed discussion on how the related works that are presented here compare to our own solution and to the overall requirements is not provided here, but placed in appropriate sections throughout the chapter and finally summarised in the end of the chapter.

Semantic Web Service Description Frameworks

The domain of Semantic Web Services is positioned at the intersection of Semantic Web technologies and Web service technologies and has been a distinct research theme since 2001 (McIlraith, Son, & Zeng, 2001). The vision in SWS research is to bring formal logic-based semantics into Web service technology standards such that service characteristics can be explicated in an unambiguous, computer-interpretable manner that facilitates the automation of a broad range of activities, primarily discovery, composition, execution and mediation. The core idea is that by using formal representation schemes to describe Web service characteristics, service-related artefacts can be automatically processed by specialised tools through logic-based inference and automated reasoning.

Evidently, the degree of automation that can be achieved depends on the expressiveness and overall capabilities of the semantic representation formalism that is employed for this purpose. Recent years have seen the development of numerous such formalisms for representing service characteristics, termed SWS description frameworks. The most prominent proposals towards a standardised SWS framework have been OWL-S (Martin, et al., 2004), WSMO (Bruijn, et al., 2005), and WSDL-S (Akkiraju, et al., 2005). The latter provided the foundation for the development of SAWSDL (Farrell & Lausen, 2007) which was eventually ratified by the W3C in 2007 and is currently the only standard in the area of SWS.

Requirements for Semantic Web Service Discovery in the Context of EAI

The application of Semantic Web Service technologies for enhancing various aspects of Enterprise Application Integration has been investigated in numerous works (Bussler, 2003); (Haller, Gomez, & Bussler, 2005); (Preist, Esplugas-Cuadrado, Battle, Grimm, & Williams, 2005); (Anicic, Ivezic, & Jones, 2006); (Izza, Vincent, & Burlat, 2006). One of the most recent research efforts in this direction was that of project FUSION, an EU-funded collaborative research project undertaken by a consortium of industrial and academic partners that was coordinated by SAP. FUSION focused on improving the efficiency of business process integration within and across enterprises by leveraging SWS technologies for achieving interoperability among service-oriented business applications (Alazeib, et al., 2007). The project delivered a complete reference framework and a methodology for semantics-based EAI, a reference implementation of the proposed framework, and a validation of the overall approach through three pilot studies on intra- and inter-organisational integration.

The introduction of semantics to Web service discovery is an essential requirement for realising the Semantic EAI approach that is put forward by FUSION. In general, the development of a semantically-enhanced service registry is an undertaking that encompasses the following research challenges.

- Firstly, devising means for describing service advertisements and service requests in a formal, semantically-rich and machine-understandable form that captures their salient properties and allows for comparing them in an automated way through logic-based inferencing.
- Secondly, developing a service registry that augments the typical functions of UDDI registries by introducing a reasoning mechanism that can process the semantic service descriptions and carry out automated matchmaking among service advertisements and requests.

As a general rule, it would also be desirable to address these requirements in a way that promotes the use of open standards and open source software, such as in the languages to be used for encoding the semantic descriptions of services and in the technologies to be used for the development of the registry.

Beyond the above definition of research challenges which is broad and application-independent, the context of Enterprise Application Integration gives rise to some more specific requirements that must be overcome for effective service discovery, as the FUSION project has demonstrated.

Firstly, concerning the description of service advertisements and requests, the context of EAI imposes some requirements with regard to the type of service properties that need to be described, and consequently, imposes requirements with regard to the ontology language and the ontology structure that is employed for capturing them.

More specifically, a fundamental criterion that must be considered in Web service discovery for EAI, as already mentioned in the introduction, is the integrability of a service on the basis of the input and output messages that are defined in its interface. During matchmaking we need to be able to evaluate if the amount of data that the service consumer (i.e. the BPEL orchestration) can provide as input to a service are sufficient with regard to the amount of data that the service expects to receive, and vice versa for the outputs. Therefore, the input and output data parameters of a service that are defined in WSDL (Christensen, Curbera, Meredith, & Weerawarana, 2001) using XML Schema Definitions (XSD) are regarded as salient properties of that service that need to be semantically represented. Consequently, a critical requirement that is placed on the ontology language in which the schemata of input and output parameters are to be represented, is that it should be expressive enough to allow the preservation of the semantics of arbitrarily complex XML Schema Definitions.

Secondly, concerning the design and implementation of the service registry, the context of EAI places some important requirements with regard to the matchmaking function and the capabilities of the underlying reasoning mechanism. To enable automated discovery, the registry must employ logic-based inferencing for the purpose of matchmaking among service requests and advertisements, on the basis of the ontological representations of their I/O data schemata. For that reason, it is a requirement that the registry's inference engine can perform sound and complete reasoning at a level of expressiveness that is equivalent to that of the ontology in which the I/O data schemata are represented. In addition, since the I/O-based matchmaking function evaluates service suitability on the basis of the service's interface, i.e. only from a technical point of view, it would be desirable for the registry to provide an auxiliary semantic matchmaking function that assesses the suitability of a Web service for some given process task from a business point of view. As demonstrated in the FUSION project, but also in other related works that are presented next, an intuitive way in which this could be achieved is through category-based indexing and searching, whereby each service is assigned a category from some taxonomy of business areas/activities which designates the intended functionality of that service. This auxiliary matchmaking function can significantly improve the results of service discovery by filtering out advertised services that happen to have integrable interfaces because their inputs and outputs match the specifications of the request, but are nevertheless performing business tasks irrelevant to the needs of the requestor (e.g. consider the functionality of CreateOrder vs. CancelOrder).

Note that the above discussion of requirements for the description of service properties and the design and implementation of the service registry is only a brief outline. A more detailed analysis of the motivation behind these requirements and how they are addressed in our approach and implementation is provided later in the chapter.

Related Work on UDDI-Based Semantic Service Registries

The use of SWS frameworks for representing discovery-related service properties and facilitating semantically-enhanced matchmaking in Web service registries has been investigated in numerous research works. In recognition of the fact that UDDI is a widely endorsed Web service technology standard with extensive support by the industry, the vast majority of these works has focused on combining these SWS frameworks with UDDI-based service registries, rather than proprietary registry back-ends. The rationale behind this decision is that the best way to promote the adoption of Semantic Web technologies is by enhancing today's widely-endorsed technology standards with semantics whenever appropriate and where feasible, instead of trying to introduce new

standards. In this review we confine ourselves to works that seek to promote semantically-enhanced service matchmaking specifically in relation to the open standard of UDDI, and in addition, works that are not only theoretic but come with a proof-of-concept system implementation.

Paolucci, Kawamura, Payne, & Sycara (2002) from Carnegie Mellon University were the first to propose that discovery in UDDI registries can be significantly enhanced by introducing semantic matchmaking among service descriptions. The paper presents a matchmaking algorithm able to recognise various degrees of matching among a request and an advertisement that are described with DAML-S (the precursor of OWL-S), by applying subsumption reasoning on the ontological representations of their inputs and outputs. The authors also propose to integrate a matchmaking engine that realises this approach inside the UDDI registry and provide a mapping between DAML-S Profiles and the UDDI data model. Subsequent work by the same group (Srinivasan, Paolucci, & Sycara, 2005) proposes a revised mapping between OWL-S Profiles and the UDDI data model, and an improved version of the matchmaking algorithm from Paolucci et al (2002). Since the SWS framework that is adopted in this work is OWL-S, the ontology language in which input and output parameters are to be represented is OWL. As will be shown later in the chapter, the OWL language includes the dialect of OWL-DL which appears to be sufficiently expressive for representing XSD structures, so the requirement for ontological expressivity that we described earlier could be satisfied. Moreover, in the implementation of their semantic service registry the authors employ an inference mechanism that relies on standard Description Logic reasoners like Pellet and Racer which are known to perform sound and complete reasoning over knowledge-bases encoded in OWL-DL.

The divergence of this work with regard to the requirements that we outlined in the previous section is very small and can be found in the following. Firstly, the introduction of the OWL-S matchmaker in the UDDI registry necessitates the modification of the UDDI server's API which is a practice that conflicts with the standard. Secondly, the approach described in the papers lacks an auxiliary semantic matchmaking method such as category-based matchmaking for complementing the I/O-based matchmaking (although the implemented OWL-S/UDDI matchmaker tool apparently supports classification-based search). Thirdly, the implementation of the OWL-S/UDDI matchmaker is freely available in binary form[3] but the source code is not released in order to be adapted and extended with regard to our set requirements.

A research work by a different group at IBM that expands the approach introduced by Paolucci et al. (2002) is presented in Akkiraju, Goodwin, Doshi, & Roeder (2003). The authors present a method to improve the effectiveness of service discovery in UDDI based on a two-stage service discovery process which combines syntactic category-based search via the standard UDDI search mechanism, and semantic I-O-based search via logic-based inferencing. They also propose extensions to the specification of the UDDI inquiry and publish API in order to support automatic service composition based on DAML-S service descriptions. The main idea is that if no single matching service can be found for a submitted service request, the registry could attempt to construct a sequential composition of Web services that fulfils the request by chaining the output of one service to the inputs of another. The authors report that they have implemented and tested a registry that realises this approach using DAML-S v0.7 for the service descriptions, DAML+OIL for the representation of the domain ontology in which inputs and outputs are defined, DAMLJESSKB for performing inferencing, and IBM's implementation of UDDI version 2.0 for the registry back-end.

The above described work does not match all of the previously outlined requirements, because of

the following reasons. Firstly, the category-based matchmaking method is not a semantic one, and as already explained this has several limitations. Secondly, it is unclear whether the expressivity of DAML+OIL would be sufficient for representing arbitrarily complex XSD schemata of service inputs and outputs, and moreover, it is unclear whether the ontology expressiveness supported by the DAMLJessKB inference engine would suffice for reasoning over such representations. Thirdly, similarly to the approach of Paolucci, Kawamura, Payne & Sycara (2002), this work proposes the modification of the UDDI server's API with non-standard functions. Lastly, the reported implementation of the semantically-enhanced UDDI registry has not been made publicly available, although some of the ideas and functionality seem to have been incorporated in the subsequent release of IBM alphaworks Semantic Tools for Web Services[4], which is a set of Eclipse plug-ins (closed source) for semantic matching and composition of Web services that does not rely on UDDI as the registry back-end.

Another approach for developing OWL-S-based semantically-extended UDDI registries is presented in Luo, Montrose, Kim, Khashnobish, & Kang (2006). The key feature of the proposed solution is that relationships among ontology concepts which are encoded in OWL are resolved at the time of publication and indexed in UDDI in a way that enables purely syntactic querying at the time of discovery using the standard UDDI API. An OWL2UDDI transformation method is presented for analysing ontologies encoded in OWL and representing associations among equivalent concepts, parent concepts, and child concepts into the UDDI data model, such that queries for some concept would also return related concepts that have been determined through reasoning at the time of indexing. The modules for publishing and query processing are placed on the client-side and as a result no modifications to the UDDI server implementation or interface are mandated.

This work diverges from our stated requirements because of the following reasons. Firstly, as explained by the authors, the approach covers only a portion of the vocabulary in the OWL language, and thus has a rather limited expressivity capacity that would not suffice for preserving the semantics of arbitrarily complex XML Schema Definitions. For example, it cannot cope with property restrictions within definitions of OWL classes. Secondly, the approach does not address I/O-based matchmaking specifically, but rather, it is said to support a generic matchmaking process that compares OWL-S Profiles of service advertisements and service requests as whole entities, using one-to-one semantic property annotation matching. As a result, it is unclear whether the system that the authors have implemented takes the principle of subsumption asymmetry among inputs and outputs into consideration (i.e. that for a match to exist, the output of the advertised service must be a subtype of the output specified in the service request, and the input specified in the service request must be a subtype of the input of the advertised service). Lastly, the paper reports a proof-of-concept implementation of the approach but the authors have not made it publicly available.

An approach by the LSDIS group at the University of Georgia Athens based on the WSDL-S specification is introduced in Sivashanmugam, Verma, Sheth, & Miller (2003) and elaborated in Li, Verma, Mulye, Rabbani, Miller, & Sheth (2006). In the first of these two works the authors present a theoretical approach for publishing WSDL-S service descriptions that have been semantically annotated with references to concepts defined in an ontology. The paper presents a WSDL-S to UDDI mapping for storing the semantic annotations and facilitating subsequent discovery of Web service operations based upon them. A discovery algorithm is defined which first selects the services using ontological concepts representing the functionality of operations (i.e. a

form of categorisation), and then uses inputs and outputs to prune the search. The service requestor can initiate the discovery by creating a semantic request template that specifies the desired functionality (i.e. category), inputs, and outputs, by references to ontological concepts. In the subsequent work of Li et al (2006) the authors describe the way in which Web service descriptions can be annotated, published and discovered using Radiant and Lumina, a pair of graphical tools integrated with the METEOR-S Web Services Discovery Infrastructure (Verma, Sivashanmugam, Sheth, Patil, Oundhakar, & Miller, 2005) which supports scalable publication and discovery in peer-to-peer networks of distributed registries.

The approach by the LSDIS group is very close to the requirements that we have set in the previous section. The only exception concerns the requirement of sufficient ontological expressivity for the representation of service message parameters and for reasoning, which is however an essential requirement for integrability-oriented service discovery. The theoretic approach that is described in the papers is generic and does not prescribe any particular ontology language for creating semantic representations of inputs and outputs or categories of functionality, neither any specific reasoner for reasoning over these representations. However, the implementation of the approach which is available as open source software with METEOR-S[5] assumes the availability of OWL ontologies and implements an OWL reasoner based on the Jena API. The problem with ontology expressivity lies in the processing capabilities of Jena, because according to its documentation[6], Jena rule-based reasoners are able to provide semantic entailments only for OWL ontologies using the vocabulary of the OWL-Lite dialect, and some constructs from the more expressive dialect of OWL-DL. In order to mitigate the effects from this lack of processing power Jena implements the DIG description logic reasoner interface for connecting to external reasoners, but this does not suffice to overcome the issue, since it is known that some OWL-DL constructs cannot be expressed in the DIG "tell" language, and some desirable queries are not possible. Overall, it appears that the ontology expressivity supported by the Jena-based reasoner would not suffice for reasoning over representations of arbitrarily complex definitions of XSD schemata of service input and output message parameters.

A number of service discovery engine prototypes have also been developed in the context of the WSMX Working Group[7] for supporting the three different discovery approaches that are put forward in WSMO, i.e. keyword-based discovery, lightweight semantic discovery based on WSML-Rule and WSML-DL, and heavyweight semantic discovery based on WSML-Flight (Keller, Lara, Polleres, Toma, Kifer, & Fensel, 2004). The specific works however do not offer themselves for direct comparison with the other approaches presented above, as they do not attempt to provide semantic enhancements to UDDI but rather stand as independent WSMX environment components that are not meant to be integrated with UDDI registries.

INTEGRABILITY-ORIENTED DESCRIPTIONS OF SERVICE PROPERTIES

As mentioned in the previous section, semantically-enhanced publication and discovery of services in UDDI-based registries encompasses two main objectives. Firstly, describing service advertisements and service requests in a machine-understandable form that captures their salient characteristics and allows for comparing them in an automated way. Secondly, augmenting the typical functions supported by UDDI registries (i.e. storing syntactic metadata about services and their providers) with the addition of a mechanism for semantic service indexing and matchmaking. This section of the chapter discusses the first objective. More specifically, we first describe what

are the salient service characteristics (functional and non-functional properties) that are modelled in order to support integrability-oriented service discovery with the FUSION Semantic Registry, and subsequently, we analyse how these characteristics are captured in a suitable semantic representation formalism.

Service Properties for Integrability-Oriented Service Matchmaking

The Semantic Web Services research literature features an abundance of different approaches for service matchmaking. Each of them is intended to address a specific set of requirements and therefore focuses on a different set of service properties, functional or non-functional ones. The set of service characteristics that the FUSION Semantic Registry considers during matchmaking is a combination of functional and non-functional properties and represents the minimum amount of information that would be needed for determining if some advertised service is capable of performing some task and at the same time is syntactically and semantically interoperable with the service consumer, i.e. with the BPEL orchestration that invokes the service and consumes its output.

Functional Properties of Web Services: Inputs and Outputs

As already mentioned in the introduction, in integrability-oriented service matchmaking we need to detect if interoperability at the level of data can be guaranteed among an advertised service and its prospective consumer, such that proper data flow and communication can be established among the two. In the context of FUSION, but also in most of the approaches for Semantic Enterprise Application Integration, the service consumer is an executable Web service orchestration encoded in WS-BPEL. The WS-BPEL-encoded orchestration is essentially a controller program that is itself exposed as a Web service and whose purpose is to specify how a set of Web services exposed by different enterprise applications should interoperate to realise a specific business process. What we therefore seek to determine in our integrability-oriented service matchmaking is if some advertised service can be safely integrated in this executable orchestration.

The instance data to be used at run-time by the executable BPEL orchestration for invoking the advertised service may have originated from a previous step in the process (i.e. from some other Web service participating in the orchestration), may have resulted from numeric calculations or string manipulations within the BPEL code, or may have been provided to the controller service from the external environment (i.e. from the system that triggered the execution of the BPEL orchestration). Similarly, the instance data that the BPEL controller service will receive as output from the invoked service may later on be fed into some other Web service taking part in the orchestration, may be used for performing internal calculations that affect control flow, or may be returned by the controller service to the environment. Data-level compatibility among the inputs and outputs of Web services participating in an orchestration and the orchestrator service itself is therefore an essential requirement for guaranteeing communication and composability (Kourtesis & Paraskakis, 2008a; Kourtesis & Paraskakis, 2008b).

In plain terms, in order to assert this notion of data-level compatibility we need to ensure that the data that the controller BPEL service is able to provide upon invocation are sufficient with regard to the input data that the advertised service expects to receive, and conversely, the output data that the advertised service produces are sufficient with regard to the data that the controller service expects to receive. We use the term *sufficient* to denote that the data schemata of the two parties may not necessarily be identical for integration to be possible. Rather, it would suffice to assert that the service consumer can provide *at least* the amount of data that the advertised service expects

to receive, and at the same time, the advertised service can generate *at least* the amount of data that the consumer (i.e. the controller service) expects to obtain. If this can be asserted, then it is safe to assume that a transformation from the more informative data schema to the least informative one can be obtained in a straightforward manner (manually or semi-automatically) and therefore data flow in the business process can be made possible.

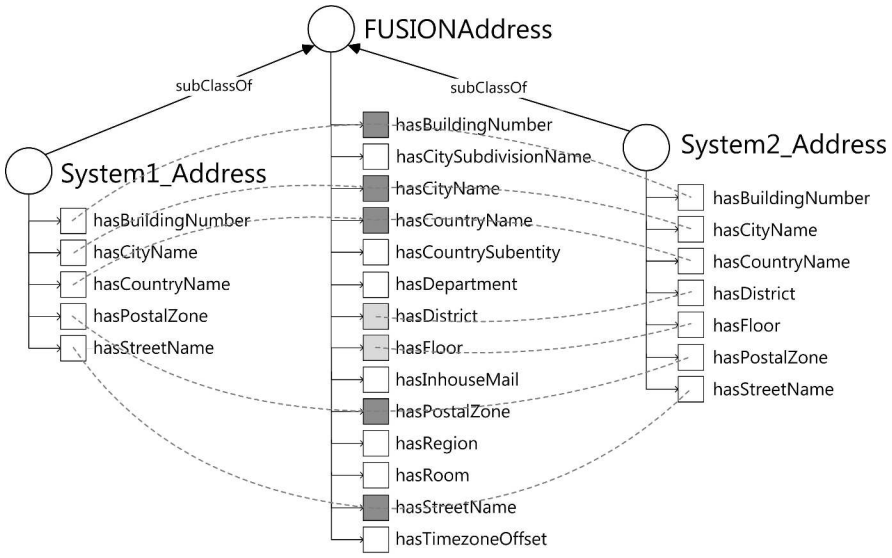
This relates directly to the notions of covariance and contravariance applied in the context of function subtyping and safe substitution, which have been studied in detail within type-theory and object-oriented programming research (Simons, 2002). If we attempt to draw parallels with service-orientation, we could say that in order to substitute a service request with a service advertisement the first must be shown to subsume the latter (i.e. the request must be more generic than the advertisement). In other words, the advertisement must be proven to be a subtype, or special case, of the request. For this subsumption ordering to hold, the subsumption relation among the input types of the request and the input types of the advertisement must be contravariant (i.e. the advertisement input types must subsume the request input types), while the subsumption among their output types must be covariant (i.e. the request output types must subsume the advertisement output types). In practical terms, if a data parameter subsumes another, it means that the one which is subsumed is more specific and thus more informative than the one which subsumes it.

Evaluating this type of compatibility is particularly meaningful in cases where two enterprise applications share a data model specification as a basis for exchanging interoperable business objects or electronic documents, but are not obliged to instantiate or make use of all schema attributes for every entity defined in that model. As a result, the case may arise where the developers of different applications have chosen to instantiate the schema attributes of a base entity in different ways, thus arriving to only partially overlapping and effectively incompatible definitions of data parameters that nevertheless carry the same name. This is also a typical situation when working under the assumption of a shared base ontology that can be specialised and customised for niche application domains through subclassing and applying restrictions on class definitions, as in the case of FUSION (Bouras, Gouvas, & Mentzas, 2008). Different developers may choose to extend a base ontology concept in different ways, thus creating potential interoperability problems. Figure 1 illustrates an example case in which the base concept of FUSIONAddress (depicted in the middle column) has been specialised in two different ways, for modelling the data spaces of two different enterprise applications.

Although System1_Address and System2_Address are subclasses of the same concept (FUSIONAddress), interoperability can be guaranteed only when information flows from System2 to System1, and not the other way around. This is because the schema of System2_Address is more informative than the schema of the latter. To illustrate this, let us assume that we wished a BPEL orchestration controller to consume some service exposed by System2, which required to be provided with address information as input (e.g. in order to calculate the cost of shipping some item). If the controller service had obtained this address information in a previous step from System1 we would have an impedance mismatch problem, because System2 expects to receive data for the hasDistrict and hasFloor attributes that are not part of System1_Address, thus rendering integration impossible. On the contrary, if we wished to feed address-related data retrieved from System2 into System1 then a transformation function (within the BPEL code or externally via XSLT) could be provided to take care of the mapping.

The overall integration-oriented principle of asserting that the consumer is able to provide *at least* the amount of input data expected by the advertised service, and vice-versa for outputs, can

Figure 1. Mismatch at the level of data schema among System1 and System2 due to different ontology class restrictions (adapted from Kourtesis & Paraskakis, 2008b)



also be applied to evaluating compatibility at the service message level. The request and response messages of service operations have their own schema definitions and may be made up of multiple data parameters. For instance, let us assume that some advertised service expects to receive an address, a purchase order, and a product description as part of the request message for invoking one of its operations, but the prospective service consumer (i.e. the BPEL controller service) cannot obtain the product description data from any other participating service or from the external environment. Inevitably, it would be impossible to integrate the specific advertised service into the orchestration.

In order to evaluate the compatibility among inputs and outputs in an automated way and perform integrability-oriented service matchmaking we need to describe the data schema for input and output parameters in an ontological manner. Since the schemata of Web service inputs and outputs are defined using XSD, the ontological formalism to be used for encoding definitions of inputs and outputs should be sufficiently expressive to facilitate modelling of arbitrarily complex XSD schemata as those found in WSDL inputs and outputs, while retaining decidability to enable automated processing.

Based on recent research works on transformations from XML/XSD to OWL (Bohring & Auer, 2005) (Garcia & Gil, 2007) it appears that the minimum level of expressiveness that would be required for representing XSD constructs in OWL while preserving the intended semantics would be that of the OWL-DL dialect. OWL-DL is one of the three dialects of the W3C standard Web Ontology Language (OWL) and is termed "DL" due to its direct correspondence with Description Logics. The other two dialects are OWL-Lite, which is less expressive than OWL-DL due to its restricted vocabulary[8], and OWL-Full, which is more expressive than OWL-DL because it does not restrict the OWL vocabulary, but consequently cannot be used as the basis for inferencing that is sound and complete. In contrast to the other dialects, OWL-DL can be applied in cases where the need for expressiveness is accompanied by the need for computational completeness (guaranteeing that all valid entailments will be computed) and decidability (guaranteeing that all computa-

tions will finish in finite time) for the purposes of automated reasoning (McGuinness & van Harmelen, 2004).

More specifically, the need for OWL-DL arises because the OWL-Lite vocabulary does not suffice for expressing the semantics of some important XSD constructors which are frequently used within WSDL documents for defining the structures of input and output messages. For example:

- The semantics of the xsd:choice compositor (which is equivalent to an XOR) can only be expressed in OWL through boolean combinations of the owl:intersectionOf, owl:unionOf and owl:complementOf constructors. However, the expressivity of OWL-Lite does not suffice because the use of owl:unionOf and owl:complementOf are not allowed. These constructors are allowed only in OWL-DL and OWL-Full.
- The semantics of the xsd:enumeration constraint (which is placed within an xsd:restriction to limit the content of an XML element to a set of acceptable values) can be expressed in OWL using the owl:oneOf constructor. Similarly to the case above, the expressivity of OWL-Lite is not sufficient because owl:oneOf is not allowed in this dialect, in contrast to OWL-DL and OWL-Full.
- The semantics of the xsd:minOccurs and xsd:maxOccurs indicators (which specify the number of times an XML element can be found in a document) can be expressed with the owl:minCardinality and owl:maxCardinality constructors. In contrast to OWL-DL and OWL-Full, the vocabulary of OWL-Lite restricts the use of the owl:maxCardinality and owl:minCardinality constructors to cardinality values of 0 or 1, and therefore does not allow expressing arbitrary numbers for the occurrence of XSD elements.

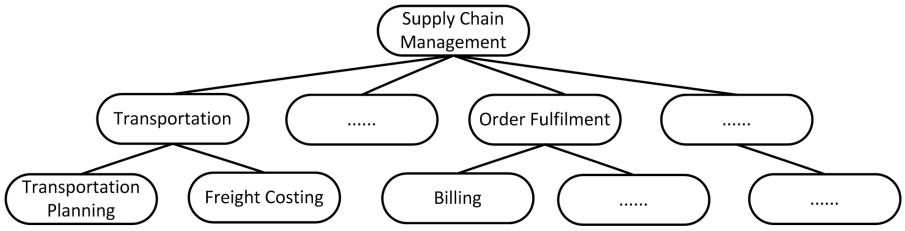
Once an OWL-DL-encoded representation is available for the service inputs and outputs, compatibility among advertisements and requests can be evaluated through standard subsumption reasoning with a Description Logics reasoner. The FUSION Semantic Registry utilises Pellet for this purpose, as will be discussed later in the architecture section. Our matchmaking algorithm, returns a positive match among a service advertisement and a service request if the input concept associated with the advertisement subsumes the input concept of the request (i.e. the first is equivalent or less informative than the second, as happens with System1_Address which subsumes System2_Address in Figure 1), and the output concept associated with the request subsumes the output concept of the advertisement (the latter is equivalent or more informative than the first).

Non-Functional Properties of Web Services: Categorisation

Non-functional properties also play an important role in service discovery, and are increasingly attracting the interest of the Semantic Web Services research community as an important area of study. Non-functional properties may relate to quality of service (QoS), policy compliance, adherence to technical standards or protocols, or categorisation within a classification system. The only type of non-functional property that is taken into account for matchmaking by the FUSION Semantic Registry is the latter, i.e. the categorisation of a service advertisement with regard to some semantically represented classification system, in order to designate the functionality of that service and assist in simple tasks like browsing through advertisements and performing coarse-grained filtering during matchmaking.

Classification systems facilitating this form of categorisation have been used in the industry for a long time. Some of the most known classification systems are the United Nations Standard Products and Services Code (UNSPSC), the North Ameri-

Figure 2. Excerpt from the taxonomy of business functions that is part of the FUSION ontology



can Industry Classification System (NAICS), the MIT Process Handbook (MPH), and the enhanced Telecom Operations Map (eTOM). A number of classification systems have been also built on top of information interchange models such as the Open Travel Alliance (OTA), and the Open Financial Exchange (OFX).

As an example, consider the taxonomy illustrated in Figure 2, which is an excerpt from the taxonomy of business functions that is part of the FUSION Ontology. Let us assume that a service request is classified under Supply Chain Management, and that some advertisement is classified under Freight Costing. As seen from the diagram, Freight Costing is a subcategory of Transportation that is itself classified under Supply Chain Management. A semantic representation of this taxonomy and a suitable matchmaking mechanism allows detecting that the service advertisement matches the request, since the category of Supply Chain Management services is more generic than the Freight Costing services category.

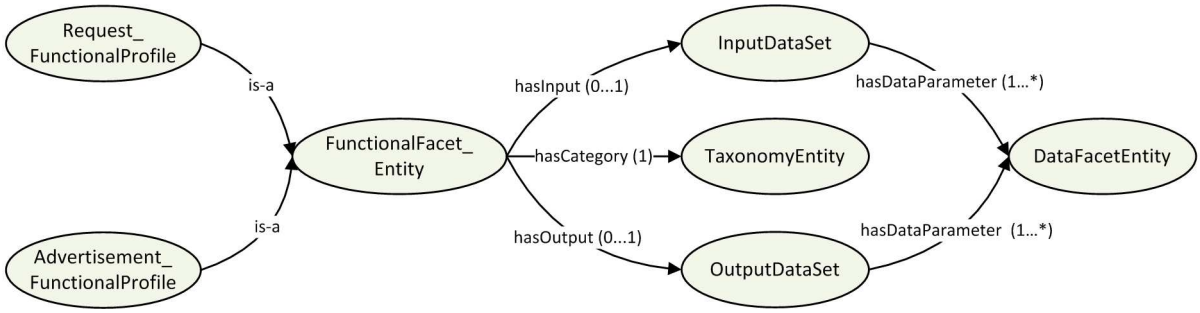
Intuitively, the end goal in categorisation-level matching within the FUSION Semantic Registry is to determine if the semantic categorisation class attributed to some service request is equivalent, more specific, or more generic than the one specified in some service advertisement. In OWL-DL terms, in order to have a positive match, the categorisation class associated with a request must subsume the categorisation class of an advertisement (i.e. the first must be equivalent or more generic than the second).

Semantic Representation of Service Characteristics in FUSION

By using a semantic representation formalism to express the above presented characteristics of Web services, providers and requestors create definitions of service capabilities that are automatically processable through reasoning and logic-based inference. In turn, this facilitates fine-grained service matchmaking for supporting integrability-oriented service discovery, and effectively, for increasing the levels of automation in EAI. As already said in the background section, the extent to which this can be achieved depends on the semantic representation formalism that is adopted for this purpose.

Although the FUSION reference framework is abstract and does not prescribe the use of any specific Semantic Web Service description framework, the tools that comprise the reference implementation of the FUSION System, including the FUSION Semantic Registry, utilise SAWSDL. In contrast to developing Web service descriptions at a high conceptual level and then linking these specifications to concrete Web service interfaces that are described in WSDL (as proposed in OWL-S and WSMO), the approach that SAWSDL puts forward is bottom-up: the WSDL documents are to be enriched with annotations that capture machine processable semantics by pointing to concepts defined in externally maintained semantic models. This approach has numerous advantages, but the most important one is that SAWSDL can be agnostic to the knowledge representation formalism one

Figure 3. Fragment of FUSION ontology used for modeling service requests and advertisements



adopts for modelling service characteristics.

The semantic model that serves as the basis for creating, storing, and reasoning upon representations of service characteristics in the FUSION project is the FUSION Ontology (Bouras, Gouvas, & Mentzas, 2007), which has been encoded in OWL-DL. Its multi-faceted structure reflects different types of concepts necessary for modelling a service: the data structures a service exchanges through input and output messages (data semantics), the functionality categorisation of a service with regard to a taxonomy of business functions (classification semantics), and the behaviour it may expose within a complex and stateful process execution (behavioural semantics). As we already mentioned the latter is not employed in the context of service discovery within FUSION.

In order to represent the functional and non-functional service properties that are of interest for matchmaking in the FUSION Semantic Registry, one needs to create a so-called Functional Profile, and define its key attributes in terms of references to the abovementioned FUSION Ontology. As presented in Kourtesis and Paraskakis (2008b) and also illustrated in Figure 3, a Functional Profile is expressed as a named OWL class that is attributed a set of three different OWL object properties:

- **hasCategory:** associates a FunctionalProfile with exactly one TaxonomyEntity concept from the service classification taxonomy that is part of the FUSION Ontology, to represent the service's categorisation.
- **hasInput:** associates a FunctionalProfile with an InputDataSet concept, in order to represent the set of data parameters that a service expects to receive and consume. The cardinality of this property is zero in the case of an *out-only* Message Exchange Pattern (MEP), or one, in the case of an *in-out* MEP.
- **hasOutput:** associates a FunctionalProfile with an OutputDataSet concept, in order to represent the set of data parameters that a service will produce if invoked. The cardinality of this property is zero in the case of an *in-only* MEP, or one, in the case of an *in-out* MEP.

Finally, each InputDataSet and OutputDataSet concept is associated with one or more DataFacetEntity concept(s) through a hasDataParameter object property, in order to represent the individual data parameters which are exchanged as part of the whole set of inputs or outputs (e.g. address, purchase order, product description, etc).

Depending on the perspective from which the Functional Profile is viewed, the provider's or the requestor's, we can make a distinction among Advertisement Functional Profiles (AFPs) and Request Functional Profiles (RFPs). The first are created automatically by the FUSION Semantic

registry at the time of service publication, while the latter are created by the service requestor at the time of discovery (or even at an earlier stage to be used as service request templates).

To allow for the automated construction of Advertisement Functional Profiles (AFPs) in the FUSION Semantic Registry, service providers need to augment the WSDL interfaces of their provided services with semantic annotations, as per the SAWSDL specification. According to the SAWSDL annotation conventions that apply in FUSION, the semantics of a Web service's input and output data should be captured by adding modelReference annotations to the appropriate <xs:element> entities under <wsdl:types>, while functionality categorisation semantics should be captured via modelReference annotations on <wsdl:portType> entities.

ARCHITECTURE OF THE FUSION SEMANTIC REGISTRY

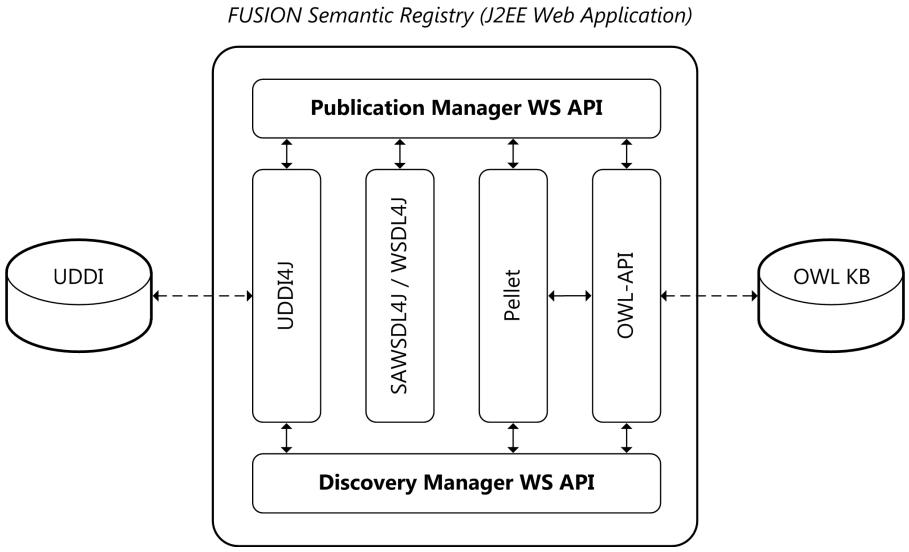
In the previous section we described the salient service characteristics (functional and non-functional properties) that should be modelled to support integrability-oriented service discovery, and analysed how these characteristics are captured in a suitable semantic representation formalism. This section of the chapter discusses the technical aspects of our approach for augmenting UDDI-based service registries with semantic matchmaking extensions. We provide an overview of the architecture that we employed in the development of the FUSION Semantic Registry and an outline of the programmatic interfaces that it exposes.

A distinctive characteristic of the FUSION Semantic Registry architecture is that it can augment the search facilities of a UDDI registry without mandating any modifications to the standardised UDDI registry API as required by the approach of Akkiraju et al (2003) and without requiring to tamper with the implementation of the UDDI registry at source code or configuration level in order to integrate the matchmaking mechanism as required by the approach of Akkiraju et al (2003), Paolluci et al (2002), and Srinivasan et al (2005). This is considered an important advantage compared to other approaches, as it allows adopters of this solution to use their existing or preferred UDDI server implementation (e.g. IBM WebSphere UDDI Registry, Oracle Service Registry, SAP Enterprise Services Registry, etc) without performing any changes, thus encouraging uptake of such technology by end users.

As illustrated in Figure 4, we propose an architecture where the UDDI server stands independently to the semantically-enabled service registry modules and works as a back-end. The FUSION Semantic Registry exposes two specialised Web service APIs to the client for publication and discovery functions, and is responsible for performing the associated SAWSDL parsing, OWL ontology processing, and DL reasoning operations. Approaches based on this principle of accommodating semantic processing functions without imposing any changes to the UDDI server implementation or interface have been also proposed in other works (Pokraev, Koolwaaij, & Wibbels, 2003; Colgrave, Akkiraju, & Goodwin, 2004; Luo, Montrose, Kim, Khashnobish, & Kang, 2006).

The UDDI module that is depicted in Figure 4 can be any UDDI server implementation that complies with the UDDI v2 or v3 specification, although the FUSION Semantic Registry has been developed and tested using Apache jUDDI[9]. The OWL KB module is a typical OWL ontology with RDF/XML serialisation that the Semantic Registry uses for storing the Advertisement Functional Profiles it generates at the time of service publication, as will be explained in the next section of the chapter. In the centre of the figure is the actual FUSION Semantic Registry, a J2EE Web Application that complies with the Java Servlet 2.4 specification and can be deployed on any compatible container implementation, such as Apache Tomcat.

Figure 4. Semantic registry architecture (adapted from Kourtesis & Paraskakis, 2008a)



The Publication Manager module of the FUSION Semantic Registry provides a Web service API to the user for adding, removing, or updating Web service advertisements, as well as adding, removing, or updating descriptions of service providers. A list of the Web service operations exposed by the Publication Manager and the parameters of the respective request and response messages is provided in Table 1.

The Discovery Manager module provides a Web service API for retrieving a specific service advertisement or service provider record via its key, discovering a set of services or service providers through keyword-based for terms contained in their names, and most importantly, discovering a set of services based on a Request Functional Profile. A list of the Web service operations exposed by the Discovery Manager and the parameters of the respective request and response messages is provided in Table 2.

The dependencies that the Publication Manager and Discovery Manager modules have on the third-party components depicted in the centre of Figure 4 are examined in the following sections, along with the overviews of the semantic service publication and discovery processes.

SERVICE PUBLICATION PROCEDURE

As detailed above, the Publication Manager Module provides a Web service API to the user for adding, removing, or updating descriptions of Web services, as well as adding, removing, or updating descriptions of service providers. This section of the chapter focuses on the most important of these functions, the process of publishing a semantically-enhanced service description (addService).

Apart from the authentication token, the publication query that initiates the publication process includes the following parameters: (i) the service provider ID (every service advertisement is associated to exactly one service provider that is identified by a UUID key), (ii) a URL pointing to the SAWSDL document that describes the service, (iii) an optional service name, and (iv) an optional free text description. The process that follows based on this input comprises a number of phases that are presented in the following subsections.

Table 1. Publication manager Web service API

Publication Manager Web Service Operation	Request Message Parameters	Response Message Parameters
initiatePublicationSession	username, password	authenticationToken
terminatePublicationSession	authenticationToken	terminationSuccess
addService	authenticationToken, serviceName, serviceFreeTextDescription, serviceProviderUUID, sawsdlURL	serviceUUID
addServiceWithoutSAWSDL	authenticationToken, serviceName, serviceFreeTextDescription, serviceProviderUUID, sawsdlURL, hasCategoryAnnotationURI, hasInputAnnotationURIList, hasOutputAnnotationURIList	serviceUUID
removeService	authenticationToken, serviceUUID	serviceRemovalSuccess
modifyService	authenticationToken, serviceUUID, serviceName, serviceFreeTextDescription, serviceProviderUUID	serviceModificationSuccess
addServiceProvider	authenticationToken, serviceProviderName, serviceProviderFreeTextDescription	serviceProviderUUID
removeServiceProvider	authenticationToken, serviceProviderUUID	serviceProviderRemovalSuccess
modifyServiceProvider	authenticationToken, serviceProviderUUID, serviceProviderName, serviceProviderFreeTextDescription	serviceProviderModificationSuccess

Table 2. Discovery manager Web service API

Discovery Manager Web service operation	Request message parameters	Response message parameters
getAllServiceProviderUUIDs	-	List of all service provider keys (UUIDs)
doKeywordSearchForServiceProviders	keyword	List of all service provider keys (UUIDs)
getServiceProviderDetails	serviceProviderUUID	serviceProviderName, serviceProviderFreeTextDescription, listOfProvidedServiceUUIDs
getAllServiceUUIDs	-	List of all service keys (UUIDs)
doKeywordSearchForServices	Keyword	List of all service keys (UUIDs)
doSemanticSearchForServices	requestFunctionalProfileURI, serviceProviderUUID	List of all service keys (UUIDs)
getServiceDetails	serviceUUID	serviceName serviceFreeTextDescription, locationOfSAWSDLDocument, serviceProviderUUID, categoryAnnotationURI, listOfInputAnnotationURIs, listOfOutputAnnotationURIs, listOfMatchingRFPURIs

Phase 1: Parsing of the Service SAWSDL Document

The first step that the Publication Manager performs is to retrieve the SAWSDL document from the specified URL and parse it to extract the semantic annotations it contains. As discussed in section 2, WSDL interfaces are augmented with potentially multiple modelReference annotations on <xs:element> entities, in order to capture the

data semantics of the service (consumed inputs or produced outputs), and a single modelReference annotation on <wsdl:portType> entities to capture its functionality categorisation semantics. At the time of this writing the current implementation of the Semantic Registry SAWSDL parser relies on the WSDL4J[10] and SAWSDL4J[11] libraries to create an in-memory representation of the SAWSDL document and extract the URIs of the ontological concepts being referenced by the modelReference annotations.

Phase 2: Construction of a UDDI Advertisement

The next step in the publication process is to map the information that was provided as part of the publication query (i.e. the service name, free text description, and service provider's UUID) and the information that was extracted by parsing the SAWSDL document (i.e. input, output, and category annotation URIs), into a UDDI service advertisement. Communication between the FUSION Semantic Registry and the UDDI server for this purpose is facilitated by UDDI4J[12]. As illustrated in Figure 5, this mapping requires creating a uddi:businessService entity and instantiating the values of its uddi:name, uddi:description, and uddi:businessKey attributes, as well as a uddi:categoryBag that includes one uddi:keyedReference entity for every extracted annotation URI.

In order to support the representation of syntactic properties and binary relations among WSDL entities in UDDI, Colgrave & Januszewski (2004) introduced a number of Canonical tModels that should be registered in a UDDI server installation before publication and discovery of WSDL documents (i.e. during the UDDI server's deployment). The FUSION Semantic Registry extends this idea and makes use of pre-registered canonical tModels (see Table 3) for representing the different types of semantic annotations that can be placed on SAWSDL documents (input, output, or category annotations). Depending on the type of semantic information being modelled, each uddi:keyedReference entity should point to the appropriate canonical tModel (Input Annotation tModel, Output Annotation tModel, or Category Annotation tModel). As depicted in

Figure 5. SAWSDL to UDDI mapping (adapted from Kourtesis & Paraskakis, 2008a)

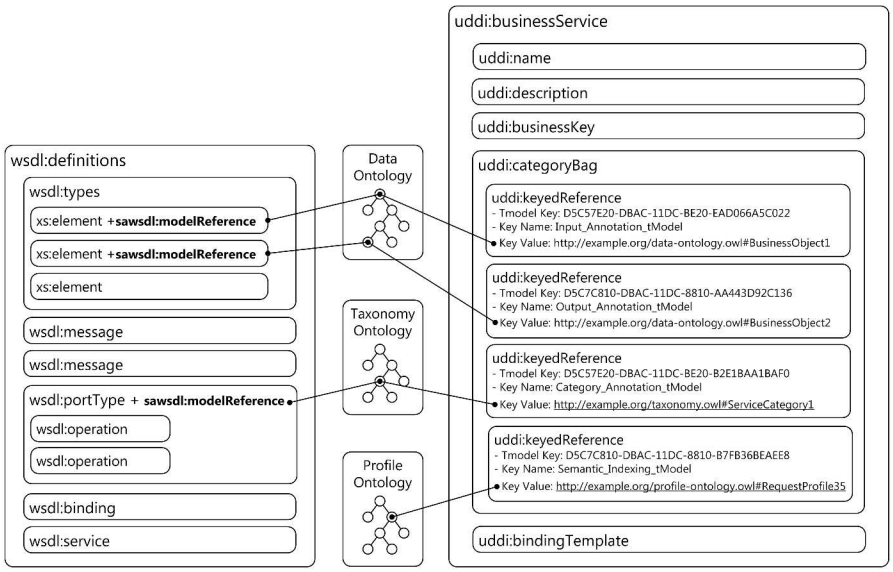


Table 3. Sample pre-registered canonical tModels for facilitating indexing in the registry

tModel Key	Name
uuid:7CB6D040-0F32-11DD-9040-B5988DE060A3	Category_Annotation_tModel
uuid:7CB94140-0F32-11DD-8140-8AB199A03241	Input_Annotation_tModel
uuid:7CBB8B30-0F32-11DD-8B30-A33C65E2A5DF	Output_Annotation_tModel
uuid:7CBB8B30-0F32-11DD-8B30-D549BB31EB3E	Semantic_Indexing_tModel

Figure 5, an additional canonical tModel is used for indexing service advertisements with respect to the Request Functional Profiles that they can readily satisfy (Semantic Indexing tModel), but the uddi:keyedReference entities that point to this tModel are created at a later stage in the publication process.

Phase 3: Generation of Advertisement Functional Profile and Matchmaking

The next step in the publication process is to create an Advertisement Functional Profile (AFP) based on the extracted semantic annotations and add it to the registry's internal OWL Knowledge Base (KB) with the help of the OWL API library[13]. The construction of the AFP follows the modelling conventions analysed in section 3. Once the AFP has been constructed, the Pellet DL reasoner[14] is used for performing an "eager" semantic classification of the new AFP against all known Request Functional Profiles (RFPs). The purpose of this classification procedure is to identify RFPs representing service requests that the newly added service advertisement can readily satisfy.

We refer to this classification procedure as "eager" since it takes place at publication-time. In contrast, a "lazy" classification procedure would not have taken place before the actual need for matchmaking arises during discovery-time. This approach is placing an inevitable overhead on the time required to complete the publication of a service advertisement, but it substantially reduces the time required to perform matchmaking at discovery-time, so it is considered particularly beneficial.

In order to claim that the new service advertisement (AFP) can satisfy a pre-registered service request (RFP), three conditions must be checked independently and be asserted:

1. the InputDataSet concept associated with the RFP must be subsumed by the InputDataSet of the AFP,
2. the OutputDataSet of the RFP must subsume the OutputDataSet of the AFP,
3. the TaxonomyEntity concept associated with the RFP must subsume the TaxonomyEntity of the AFP.

Phase 4: Indexing of Semantic Matching Results in the UDDI Registry

The last step in the publication process is to map the semantic matchmaking information that resulted from the publication-time matchmaking algorithm described above into the UDDI service advertisement. This requires retrieving the advertised uddi:businessService entity and its associated uddi:categoryBag from the UDDI server, and creating one uddi:keyedReference for every RFP that the service matches with. What this essentially achieves is indexing the service advertisement with respect to all service requests it can readily satisfy. As depicted in Figure 5, uddi:keyedReference entities should be made to point to the canonical tModel used for this purpose (the Semantic Indexing tModel), and the URI of

each RFP should be specified as the Key Value of the uddi:keyedReference. When this step is completed, a new semantic service advertisement has been created, registered with the UDDI registry, and is available for discovery.

SERVICE DISCOVERY PROCEDURE

As presented previously, the Discovery Manager module provides a Web service API for retrieving service advertisements or service provider records via their unique keys, discovering sets of services or service provider records through keyword-based search, and most importantly, discovering sets of services based on a Request Functional Profile that represents the requirements of the service consumer. This latter type of semantic matchmaking functionality is the focus of this section.

The discovery query that initiates the semantic matchmaking process comprises two elements: (i) a URI pointing to some Request Functional Profile (RFP), and (ii) an optional UUID designating the preferred service provider, i.e. the company, business unit, or specific business application that should expose the service. The RFP that the URI points to may be defined within an ontology that is shared by service providers and service requestors alike (i.e. be a reusable RFP defined in the FUSION Ontology), or within some third-party ontology that imports and extends the shared ontology (i.e. be a custom-built and non-shared RFP). Depending on which of the two cases holds, the algorithm would follow a different discovery path. Resolving the location of the ontology in which the RFP is identified is therefore the first step in the discovery process.

If the RFP is defined in the shared FUSION Ontology the Discovery Manager will look for service advertisements indexed in UDDI with a reference to that RFP. This means looking for services with AFPs that have matched the requested RFP during the "eager" publication-time classification. To retrieve such advertisements the Discovery Manager places a simple syntactic matchmaking query to the UDDI server, looking for uddi:businessService entities having a uddi:categoryBag that contains a uddi:keyedReference which points to the Semantic Indexing tModel, and moreover, has a Key Value that is equal to the URI of the RFP.

Since the matchmaking and indexing process is repeated every time a new RFP is created and added to the shared ontology, the UDDI server's semantic matching index is bound to always be accurate and up to date. This means that if some service advertisement matches some RFP which is defined in the shared ontology, the registry is guaranteed to have this association indexed in the UDDI server, and be able to instantly retrieve the advertised service.

Due to the shared ontology assumption that is made in the context of FUSION, this is the most typical type of discovery querying envisaged for the FUSION Semantic Registry, and is also the simplest and fastest type of matchmaking possible. Since the time-consuming process of subsumption reasoning and hierarchy classification has been already performed at publication-time, the computational complexity of discovery-time matchmaking for RFPs defined in a shared ontology is essentially as low as that of a conventional UDDI server. In other words, the use of semantics does not impose any noteworthy overhead compared to syntactic matchmaking.

If the RFP is defined in a non-shared ontology the Discovery Manager would need to load that ontology into memory and perform a complete semantic matchmaking process among the specified RFP and all AFPs stored in the OWL-KB. The conditions that need to be checked in order to assert that a service advertisement can satisfy the request are the same as the ones defined for publication-time matchmaking.

The result of the discovery process, regardless of the ontology in which the RFP is defined, is a list of UUID keys corresponding to advertisements of services that comply with the matchmaking

criteria modelled in the RFP. If a service provider UUID has been also specified in the discovery query, the UDDI server will restrict the result set to only those services offered by the specified provider.

CONCLUSIONS AND OUTLOOK

The availability of sophisticated Web service discovery mechanisms is an essential prerequisite for increasing the levels of efficiency and automation in Enterprise Application Integration. In a contemporary service-oriented business application ecosystem, the integration of a set of different applications is typically realised by creating executable specifications of how the Web services that these applications expose should be orchestrated in order to fulfil a particular business process. The outcome of the integration procedure is a set of executable business processes, each of which invokes a number of Web services in the order dictated by the underlying business logic, assigning the output of one service into the inputs of others, and where necessary, applying transformations from the data representation of one service provider to that of another. Therefore, an essential criterion for selecting services that are suitable for composition, among the tens or hundreds of Web services potentially available, is the integrability of a service on the basis of the input and output messages that are defined in its interface. The description and discovery mechanism of contemporary UDDI-compliant service registries is not sufficiently sophisticated and fine-grained to address the above criterion for service selection, and thus cannot support automated service discovery in the context of EAI. The fundamental problem is that the service descriptions available in UDDI lack the machine-understandable semantics that would make them amenable to automated processing.

In this chapter we presented an approach for developing service registries which build on UDDI and offer semantically-enhanced Web service publication and discovery capabilities by employing Semantic Web Service technologies. Our approach aspires to promote efficiency in EAI in a number of ways, but primarily by automating the task of evaluating Web service integrability on the basis of the input and output messages that are defined in a service's interface. The approach that we put forward has been applied and validated during the development of the FUSION Semantic Registry, a semantically-enhanced service registry that has been utilised in research project FUSION and is released as open source software. Our solution places emphasis on the use of open standards and has been realised by combining three prominent standards from the area of Web Services and the Semantic Web: OWL-DL, for modelling salient service characteristics and performing fine-grained service matchmaking via Description Logic reasoning, SAWSDL, for creating semantically annotated descriptions of service interfaces, and UDDI, for storing and retrieving syntactic and semantic information about services and service providers. To the best of our knowledge the work presented in this chapter represents the first attempt to combine these three standards into a comprehensive and openly available solution.

Our approach has been specifically tailored to support Semantic Web Service discovery in the context of EAI according to the requirements that we outlined in Section 2 and explained in detail in Section 3. The following table provides a comparison among our work and other related works that we have reviewed in this chapter, on the basis of some features that are central to our work and stem from the above mentioned requirements. As already stated, we confine ourselves to evaluating works that seek to promote semantically-enhanced service matchmaking specifically in relation to the open standard of UDDI, and in addition, works that are not only theoretic but come with a proof-of-concept system implementation.

As can be seen from the table, all of the related works address the problem of matchmaking based

Table 4. Comparison with related works

	FUSION Semantic Registry	(Sivashanmugam et al, 2003) (Li et al, 2006)	Akkiraju et al, 2003	(Paolluci et al, 2002) (Srinivasan et al, 2005)	(Luo et al, 2006)
Matchmaking based on inputs and outputs?	+	+	+	+	+/-
Ontological expressiveness for I/O at OWL-DL level?	+	-	-	+	-
Sound and complete reasoning at OWL-DL level?	+	-	-	+	-
Matchmaking based on service categorisation?	+	+	+	+	+/-
Loose coupling with UDDI registry?	+	+	-	-	+
Support for SAWSDL standard?	+	-	-	-	-
Semantic regstry rleased as open source software?	+	+	-	-	-

on service inputs and outputs, and most of them also cater for categorisation-based matchmaking. Nevertheless, it appears that only our work and the work described in Paolucci et al (2002) and Srinivasan et al (2005) meet the requirement for ontology language expressiveness that would be sufficient for representing arbitrarily complex XSD schemata of service inputs and outputs, in conjunction with the ability to perform sound and complete reasoning at the same level of expressiveness. Moreover, in our attempt to promote the use of open standards our work is one of the few that have been designed for loose-coupling with the UDDI registry, and thus do not necessitate any modifications to the UDDI server's API or to its internal logic. As already mentioned this is considered an advantage compared to other approaches, as it allows adopters to use their existing UDDI server implementation without performing any changes, thus encouraging uptake of SWS technology by end users. Lastly, the semantically-enhanced service registry that was developed by the LSDIS group (Sivashanmugam et al, 2003; Li et al, 2006) and the FUSION Semantic Registry are currently the only implemented systems that are made publicly available as open source software, and our registry is at the time of this writing the only available service registry that supports the newly ratified SAWSDL specification, which is the only standard in the SWS area.

Using the presented approach and registry implementation as the foundation for our future work, we plan to expand into Web service discovery based on behavioural service descriptions, considering service preconditions and effects, and discovery based on non-functional properties of services, considering aspects such as compliance to policies and business rules and adherence to Service Level Agreements. The scope of the registry can be expanded by the addition of repository functions for handling semantic metadata, and its functionality can be augmented to include the validation of services through registry-based functional testing (Kourtesis, Ramollari, Dranidis, & Paraskakis, 2008). These extensions would be steps towards investigating the application of semantic technologies in a wider context of Service Lifecycle Management and towards the development of a theoretical and technological approach for supporting SOA Governance through the realisation of semantically-enhanced registry and repository solutions.

REFERENCES

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A., et al. (2005). *WSDL-S: Web service semantics. W3C member submission*.

Akkiraju, R., Goodwin, R., Doshi, P., & Roeder, S. (2003). A method for semantically enhancing the service discovery capabilities of UDDI. *Workshop on Information Integration on the Web,* Acapulco, Mexico.

Alazeib, A., Balogh, A., Bauer, M., Bouras, A., Friesen, A., Gouvas, P., et al. (2007). Towards semantically-assisted design of collaborative business processes in EAI scenarios. *5th IEEE International Conference on Industrial Informatics (INDIN 2007)* (pp. 779-784). Vienna, Austria.

Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., et al. (2007). *Web services business process execution language version 2.0.* OASIS Standard.

Anicic, N., Ivezic, N., & Jones, A. (2006). An architecture for semantic enterprise application integration standards. In D. Konstantas, J. P. Bourrières, M. Léonard & N. Boudjlida (Eds.), *Interoperability of enterprise software and applications* (pp. 25-34). London: Springer-Verlag.

Beyer, D., Chakrabarti, A., & Henzinger, T. (2005). Web service interfaces. *14th International Word Wide Web Conference (WWW 2005)* (pp. 148-159), Chiba, Japan.

Bohring, H., & Auer, S. (2005). Mapping XML to OWL ontologies. *13th Leipziger Informatik-Tage Conference (LIT 2005)* (pp. 147-156).

Bouras, A., Gouvas, P., & Mentzas, G. (2007). ENIO: An enterprise application integration ontology. *18th International Conference on Database and Expert Systems Applications* (pp. 419-423), Regensburg, Germany.

Bouras, T., Gouvas, P., & Mentzas, G. (2008). Dynamic data mediation in enterprise application integration. In O. Cunnigham & M. Cunnigham (Eds.), *Collaboration and the knowledge economy: Issues, applications, and case studies, e-challenges e-2008 conference* (pp. 917-924), Stockholm, Sweden.

Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Keller, U., et al. (2005). *Web service modeling ontology (WSMO*). W3C member submission.

Bussler, C. (2003). The role of Semantic Web technology in enterprise application integration. *A Quarterly Bulletin of the Computer Society of the IEEE Technical Committee on Data Engineering*, *26*(4), 62–68.

Christensen, E., Curbera, F., Meredith, G., & Weerawarana, S. (2001). *Web services description language (WSDL) version 1.1. W3C note.*

Clement, L., Hately, A., von Riegen, C., & Rogers, T. (2004). *Universal description, discovery, and integration version 3.0.2.* OASIS Standard.

Colgrave, J., Akkiraju, R., & Goodwin, R. (2004). External matching in UDDI. *2004 IEEE International Conference on Web Services (ICWS'04)* (pp. 226 - 233), San Diego, CA.

Colgrave, J., & Januszewski, K. (2004). *Using WSDL in a UDDI registry version 2.0.2. OASIS UDDI specification TC technical note*.

Farrell, J., & Lausen, H. (2007). *Semantic annotations for WSDL and XML schema. W3C recommendation.*

Garcia, R., & Gil, R. (2007). Facilitating business interoperability from the Semantic Web. In W. Abramowicz (Ed.), *BIS 2007.* (LNCS 4439, pp. 220-232). Berlin/Heidelberg: Springer.

Haller, A., Gomez, J., & Bussler, C. (2005). Exposing Semantic Web service principles in SOA to solve EAI scenarios. *14th International World Wide Web Conference (WWW 2005),* Chiba, Japan.

Izza, S., Vincent, L., & Burlat, P. (2006). A framework for Semantic enterprise integration. In D. Konstantas, J. P. Bourrières, M. Léonard, & N. Boudjlida (Eds.), *Interoperability of enterprise software and applications* (pp. 75-86). London: Springer-Verlag.

Keller, U., Lara, R., Polleres, A., Toma, I., Kifer, M., & Fensel, D. (2004). *WSMO D5.1-WSMO Web service discovery (v0.1). WSML working draft.*

Kourtesis, D., & Paraskakis, I. (2008). Combining SAWSDL, OWL-DL, and UDDI for semantically enhanced Web service discovery. In S. Bechhofer, M. Hauswirth, J. Hoffmann & M. Koubarakis (Eds.), *ESWC 2008.* (LNCS 5021, pp. 614-628). Berlin/Heidelberg: Springer.

Kourtesis, D., & Paraskakis, I. (2008). Web service discovery in the FUSION semantic registry. In W. Abramowicz & D. Fensel (Eds.), *BIS 2008.* (LNBIP 7, pp. 285-296). Berlin/Heidelberg: Springer.

Kourtesis, D., Ramollari, E., Dranidis, D., & Paraskakis, I. (2008). Discovery and selection of certified Web services through registry-based testing and verification. In L. Camarinha-Matos & W. Pickard (Eds.), *Pervasive collaborative networks, IFIP 283/2008* (pp. 473-482). Boston: Springer.

Lécue, F., Salibi, S., Bron, P., & Moreau, A. (2008). Semantic and syntactic data flow in Web service composition. *2008 IEEE International Conference on Web Services (ICWS '08)* (pp. 211-218), Beijing, China.

Li, K., Verma, K., Mulye, R., Rabbani, R., Miller, J., & Sheth, A. (2006). Designing Semantic Web processes: The WSDL-S approach. In J. Cardoso & A. Sheth (Eds.), *Semantic Web services, processes, and applications* (pp. 161-193). Springer.

Luo, J., Montrose, B., Kim, A., Khashnobish, A., & Kang, M. (2006). Adding OWL-S support to the existing UDDI infrastructure. *2006 IEEE International Conference on Web Services (ICWS'06),* Chicago, IL.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., et al. (2004). *OWL-S: Semantic markup for Web services. W3C member submission*.

Martin, D., Domingue, J., Brodie, M., & Leymann, F. (2007). Semantic Web services, part 1. *IEEE Intelligent Systems*, *22*(5), 12–17. doi:10.1109/MIS.2007.4338488

Martin, D., Domingue, J., Sheth, A., Battle, S., Sycara, K., & Fensel, D. (2007). Semantic Web services, part 2. *IEEE Intelligent Systems*, *22*(6), 8–15. doi:10.1109/MIS.2007.118

McGuinness, D., & van Harmelen, F. (2004). *OWL Web ontology language overview. W3C recommendation*.

McIlraith, S., Son, T., & Zeng, H. (2001). Semantic Web services. *IEEE Intelligent Systems*, *16*(2), 46–53. doi:10.1109/5254.920599

Paolucci, M., Kawamura, T., Payne, R. T., & Sycara, K. (2002). Semantic matching of Web service capabilities. In I. Horrocks & J. Hendler (Eds.), *The Semantic Web-ISWC 2002.* (LNCS 2342, pp. 333-347). Berlin/Heidelberg: Springer-Verlag.

Pokraev, S., Koolwaaij, J., & Wibbels, W. (2003). Extending UDDI with context aware features based on semantic service descriptions. *2003 International Conference on Web Services (ICWS'03)* (pp. 184-190), Las Vegas, NV.

Preist, C., Esplugas-Cuadrado, J., Battle, S. A., Grimm, S., & Williams, S. K. (2005). Automated business-to-business integration of a logistics supply chain using Semantic Web services technology. In Y. Sure, W. Nejdl, C. Goble, G. Antoniou, P. Haase, S. Staab, et al. (Eds.), *The Semantic Web–ISWC 2005* (pp. 987-1001). Berlin/Heidelberg: Springer.

Simons, A. J. (2002, November-December). The theory of classification, part 4: Object types and subtyping. In R. Wiener (Ed.), *Journal of Object Technology*, 27-35.

Sivashanmugam, K., Verma, K., Sheth, A., & Miller, J. (2003). Adding semantics to Web services standards. *2003 International Conference on Web Services (ICWS'03)* (pp. 395-401), Las Vegas, NV.

Srinivasan, N., Paolucci, M., & Sycara, K. (2005). An efficient algorithm for OWL-S based semantic search in UDDI. In J. Cardoso & A. Sheth (Eds.), *Semantic Web services and Web process composition.* (LNCS 3387, pp. 96-110). Berlin/Heidelberg: Springer-Verlag.

Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S., & Miller, J. (2005). METEOR-S WSDI: A scalable P2P infrastructure of registries for semantic publication and discovery of Web services. *Journal of Information Technology Management*, *6*(1), 17–39. doi:10.1007/s10799-004-7773-4

ENDNOTES

1 http://www.seerc.org/fusion/semanticregistry/
2 http://www.fusion-strep.eu/.
3 http://www.daml.ri.cmu.edu/matchmaker/inst-mm.htm
4 http://www.alphaworks.ibm.com/tech/ws-sem
5 http://lsdis.cs.uga.edu/projects/meteor-s/downloads/Lumina/
6 http://jena.sourceforge.net/how-to/dig-reasoner.html
7 http://www.wsmx.org/
8 http://www.w3.org/TR/2004/REC-owl-features-20040210/#s2
9 http://ws.apache.org/juddi/
10 http://sourceforge.net/projects/wsdl4j
11 http://knoesis.wright.edu/opensource/saws-dl4j/
12 http://uddi4j.sourceforge.net/
13 http://owlapi.sourceforge.net/
14 http://pellet.owldl.com/