A Semantic Service Matching Middleware for Mobile Devices Discovering Grid Services

Tao Guan, Ed Zaluska, David De Roure

School of Electronics and Computer Science, University of Southampton, Southampton, UK,
{tg04r, ejz, dder}@ecs.soton.ac.uk

Abstract. The combination of mobile and Grid computing enables high performance Grid access through resource-limited mobile devices. Many challenges require to be addressed before vision of building the bridge between Grid and mobile computing field is realized. One of the essential challenges is that mobile devices need to locate and select appropriate Grid services in an automatic and flexible way. However, at the current stage, both Grid service description and discovery mechanisms are still at an immature stage. This paper presents research into building a service matching middleware with Semantic Web technologies. Semantic Web technologies permit an efficient discovery of various Grid services for mobile devices by adding the machine-processable explicit knowledge into the interaction between mobile devices and Grid services. The middleware has been implemented successfully and interacts correctly with other service-oriented mobile Grid middleware, thus facilitating an enhanced Grid access for mobile devices.

1 Introduction

The purpose of Grid computing is to coordinate resource sharing in dynamic and multi-institutional virtual organizations, enabling heterogeneous resources to be aggregated to facilitate new functionalities and capabilities [1]. As a service-oriented approach to Grid computing is increasingly adopted, many systems able to discover Grid resources on-the-fly and access them dynamically become possible. Mobile computing is an extension of the traditional distributed and desktop computing, seamlessly integrating various computing devices (e.g. mobile phones, PDAs) into our daily life. Although new-generation mobile devices have improved their absolute capabilities, there is no doubt that creating complex applications on them is still a challenging objective because such devices are inevitably resource-constrained.

One possible solution is for mobile devices to make use of Grid services so that mobile users are able to access distributed resources automatically on demand with appropriate quality of service delivery. Various Grid services enhance the capabilities of mobile devices, with the potential that complicated tasks can be completed through user handheld devices. However, the combination of Grid and mobile computing models has the potential to realize a very significant

development in the adoption of high-performance Grid access through resource-limited mobile devices. It is quite evident that a great number of challenges must be solved before this vision of building the bridge between Grid and mobile computing is realized. In a previous paper [2], we discussed a system architecture to integrate mobile devices into the service-oriented Grid environment in an open and flexible way. Another important challenge is that at present, Grid service discovery mechanisms are still at an immature stage. Semantic web technologies, providing a very considerable degree of automatic processing, interoperation and integration, will help in the implementation of an effective Grid service matching mechanism.

With the proliferation of Grid services, semantic specifications of Grid services are gradually becoming a necessary requirement for the automatic flexible service provision and utilization necessary for Grid clients to perform various tasks. It is not straightforward for a requesting client to locate required services in order to execute a task in a service-rich Grid environment. Semantics of Grid services abstract top-level concepts and relationships between concepts so that both service discovery and automatic conversion of interaction formats between heterogeneous services can be realized. Furthermore, a semantic definition mechanism provides a comprehensive representation of a variety of Grid service aspects, building an essential foundation for possible automatic behaviors throughout the whole Grid service development lifecycle.

In this paper, we have presented a service matching mechanism by using Semantic Web technologies to support mobile devices accessing Grid services. The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 presents the general methodology for semantic service description and matching. Section 4 describes the attribute definition of Grid services. Service discovery mechanism is discussed in section 5, with its implementation and experimental results in section 6. Section 7 concludes our current research work and discusses further directions.

2 Related Work

Service discovery protocols are adopted to simplify the interaction between service consumers and service providers. A number of service discovery protocols have been introduced during the past several years. In the field of mobile computing, examples of description and discovery solutions are the Bluetooth Service Discovery Protocol [3], Jini [4], and Universal Plug and Play (UPnP) [5]. In the field of Web Services, Universal Description Discovery and Integration (UDDI) is a platform-independent and XML-based registry which enables businesses to publish service listings and discover each other. However, none of these service discovery mechanisms support flexible matching between service advertisements and requests, and users therefore cannot locate services automatically on the basis of the capabilities that these services provide.

Semantic Web technologies allow web services to provide a rich semantic specification to enable flexible automation of service provision and utilization.

A number of semantic-based web service matching mechanisms have been developed using Semantic Web technologies, such as [6] [7] [8]. They add a semantic layer between users and the web service WSDL description, which makes it possible to compose the web service request with high-level abstract concepts rather than syntactic level terms, addressing several limitations in the traditional web service discovery techniques. The service registry plays an important role for the service discovery. It is responsible for storing the advertisements of web services and finding a match between query requests and service advertisements. UDDI, the traditional Web Service registry protocol, only supports keyword-based search and does not meet the requirement of the capability-based service discovery. To solve this problem, [9] introduce the idea of mapping the OWL-S service profile into the UDDI web service representation and They believe OWL-S and UDDI can complement each other.

Generally speaking, Grid services are stateful Web Services, and most of current service-oriented Grid middleware techniques are based on the Web Service standards. Hence, the semantic approach in the web service discovery provides a potential direction for implementing a semantic-enhanced Grid service discovery mechanism. However, these current solutions have their limitations (e.g. no ranking in the service matching, judging concept similarity with subsumption reasoning only) and another issues are required to be considered when building a semantic framework for the service information centre middleware in the mobile Grid environment. In particular, most of semantic approach for web service discovery perform the service matching with service function attributes (e.g. inputs, outputs, preconditions, effects), and their purpose is the service discovery in the enterprise environment. Our system architecture, on the other hand, combines both Grid and mobile computing. Hence, other service characteristics except service capabilities are required to be considered for the service discovery, for example, service types, service resources and service context information.

3 Methodology for Semantic Matching

A semantic approach for service matching requires a service description complete with the ontology definition and a service search engine with reasoning mechanisms. Ontology approach is usually used to present services with abstract and high-level concepts for the service description. As long as users describe their service requirements with terms from the same ontology model used to build the service descriptions, logical reasoning mechanisms can check the semantic similarity between the service description and the user requirements, and the matching services will be discovered and returned to users.

The service provider represents all characteristics of a service in the service description. The term *description collection* indicates all possible attributes to be described for a service. These attributes may be either concepts or restrictions for existent concepts. For each individual service attribute, an ontology is usually designed to illustrate the definition of the attribute and its relationship with other service attributes. Similar to the service description, a *service request*

often consists of many individual requirements, specifying the service attribute to be expected in a service. These requirements may include service outputs, effects, inputs, function, location or any possible attributes in terms of the different service request. For a specific service request, all of the requirements can be divided into two categories, a group of strict requirements and a set of general requirements. The strict requirement specifies that this kind of requirement is essential for the service request and has to be met precisely in the service matching, while the general requirement means this kind of requirement is not as important as the strict one and only an approximate matching is necessary between the requirement and the related service attribute.

Although we assume that the service request attempts to describe expected requirements with terms from the same ontology model used to build the service description, it is impractical that every service request will obtain the exact same service even though the required services have already been deployed and advertised because one service could have a number of description formats. In fact, the responsibility of the service search engine is to find all of the related services including those that deviate from the request in certain degrees. These deviation services should not be discarded but instead classified in an predefined rule (e.g. by matching degree). The service request determines which service will be selected based on the information returned from the service matching middleware. The service search engine takes a *service request* and available service *description collections* as inputs, and returns matching services as well as their matching degrees.

The service matching engine is responsible for comparing the service request against each service description and judging whether a service should be included in the list of candidate results. The evaluation of semantic similarity between concepts is fundamental for implementing the service matching engine. Most of the previous work adopts subsumption reasoning to determine the semantic distance between concepts in the request and in the description. However, this is not sufficient for building an effective service matching engine. We use the method presented in [10] to check the semantic similarity between two concepts. The attributes in a service description are categorized into three types. Type one includes attributes whose similarity can be judged using the subsumption reasoning. Type two includes attributes whose similarity can not be judged using subsumption reasoning only. These assume that the knowledge of similarities between these concepts can be acquired by using available similarity measurement approaches such as [11] and [12]. Type three means numeric attributes. The similarity between these concept types can be qualified by using the percentage deviation from the requested value or a fuzzy membership function.

4 Attribute Definition for Service Description

4.1 Attribute Definition

Inputs, outputs, preconditions and effects (IOPEs) are important functional attributes for a web/Grid service. Inputs and preconditions define the constrains

required for a service invocation, and outputs and effects indicate the results or the state transformation of a service execution. As a standard web service description language, OWL-S [13] provides an comprehensive ontology definition for describing IOPEs.

Service-oriented Grid computing architecture is an extension of current Web Service technologies. In the computing architecture, applications are built on top of a set of common, standard and high-level services, which meet the definition of Open Grid Services Architecture (OGSA). One of the important requirements of OGSA is that the underlying middleware should store information about the service state because Grid application users usually need this kind of information to be maintained from one invocation to another. Because the service resource is the key parameter for the Grid service invocation, we regard it as an important functional attribute in the Grid service description.

In a service-oriented mobile Grid environment, mobile devices form the intersection between the physical world and the digital world. Users execute their tasks by using a variety of Grid services through their mobile handheld devices. Two main styles of application scenario are identified from the viewpoint of users: an information access scenario and a work assistant scenario. In the information access scenario, the main task is to collect required information or knowledge. Mobile devices act as universal operating terminals to access various available Grid services. In the work assistant scenario, users usually need to execute relatively complicated applications (such as data-deluge programs) to achieve specific tasks using their mobile devices. However, due to resource limitations, most complex programs cannot be executed on a handheld device. Users have to offload resource-demanding programs of the task to the Grid, and the Grid provides the executing environment for users to achieve their tasks.

An ontology is defined on the basis of the analysis of two application scenarios. The ontology represents a hierarchy of possible application scenarios and contains a taxonomy of service types which are usable for mobile clients. The top-level concept of the ontology is *Service*, which represents the most generic type. There are two direct subclasses of *Service*: the *InfoAccess* class represents the general service for the information access scenario; the *WorkAssistant* class represents the general service for the work assistant scenario.

Service context attributes are required when describing a Grid service. An ontology is designed to model the context of the service-oriented Grid environment. At present, we consider two context attributes in the Grid service description: the first is the service location, which corresponds to the "Place" class of the environment context ontology; the second is the service access range, based on which a service discovery restricting mechanism is implemented.

Mobile users access Grid services with their portable devices, which may expose their personal information. For example, if the service directory is so "open" that every mobile user can discover and obtain all deployed Grid services, a user location information may be exposed to other users as long as they can locate and try to invoke corresponding location-monitoring services. Protecting personal privacy thus becomes an important issue when designing a

service discovery mechanism. The user personal information decides their accessing level during the authorization process, which is kept in their "User" class. The service provider defines the service range for every service in the service description. When a new mobile user sends a request to search Grid services, the service searching engine will reason and decide whether Grid services can be exposed by comparing the service access range of Grid services and the access level of the mobile user.

4.2 Service Description with Extended OWL-S

OWL-S [13] is a language for describing services, which provides a standard vocabulary that can be used together with other aspects of the OWL description language to create service descriptions. The structure of the OWL-S upper-level ontology is based on the types of knowledge of service description: the "Service Profile" provides the high-level descriptive information of a service, such as the name, input/output of the service, and additional text description; the "Service Model" and "Service Grounding" provide information on how the service is used and how to interact with the service.

We use the OWL-S language to describe Grid services. However, the "Service Profile" does not specify the Grid service attributes required in our mobile Grid computing environment and has to be extended by adding the service parameters discussed above. Figure 1 illustrates the extended service profile class and its properties.

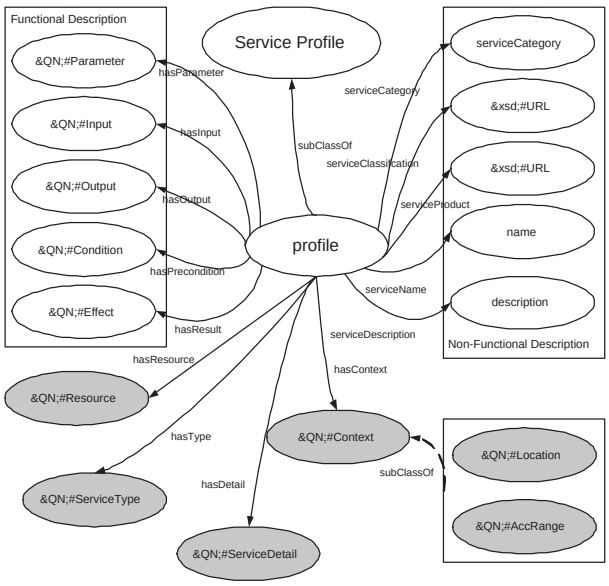

Fig. 1. Extended Service Profile

Grid services are described based on the extended service profile. For a Grid service, its description collection may include the functional attributes (e.g. inputs, outputs, preconditions, effects), the service type (an instance of *InfoAccess* class or *WorkAssistant* class), the service resource, the service detail, and the service context information (the location of the service provider and the service access range).

5 Service Discovery Algorithm

A service request is composed of a number of individual requirements, specifying various attributes to be expected in services. The service matching engine takes a service request and a group of service description collections as inputs, and is responsible for determining whether a Grid service is a matching service for this service request. The comparison between the service request and the service description collection consists of two steps. Initially, the service matching engine will check to judge whether each strict requirement can be matched precisely in the service description. If a service description does not contain the expected attributes, it will be dismissed and the service matching engine will compare the next service description to the service request. If a Grid service satisfies all of the strict requirements, the matching engine will then turn to estimate the general requirements.

As discussed in the Methodology section, the concepts are categorized into three types when checking the semantic similarity. For type one, subsumption reasoning based on the taxonomic relation is used to determine the matching degree between general requirements of the request and related attributes of the service. Three expected matching level for general requirements are defined:

- "Substitute" indicates that the user expects to find a concept in the service description which is equal to or is the direct superclass of the concept in the requirement.
- "Cover" indicates that a concept which subsumes the concept in the service request is expected to be found.
- "Fuzzy" means this requirement is less important for the service matching. As long as a concept in the service description can be found which has the subsumption relationship (either superclass or subclass) with the concept in the requirement, it will be satisfied.

These expected matching levels can be set when the service request is submitted to the service matching engine. The service matching engine will check the similarity between each general requirement in the service request and the related service attribute in the service description. The actual matching level is determined by the semantic relationship in the predefined ontology structure. If all of the expected matching levels are satisfied, this service will be a reasonable candidate for the service request.

For type two, the knowledge of similarities between concepts is assumed to be available, and the service matching engine will take the decision according to

all of close degrees between user requirements and related attributes. For type three, both the attributes and the requirements are numeric. Their similarity can be checked using the percentage deviation from the requested value or a fuzzy membership function, depending on the detailed service implementation and the user requirement.

The service matching engine may find a number of candidate services for a specific service request. Although the service discovery mechanism is not responsible for the service selection, the matching degree information about each candidate service is required to be provided as a result for the service request. We use the term "MatchingScore" to show the matching degree of the candidate service. For a candidate service, its MatchingScore is calculated using the following equation:

$$MatchingScore = \sum_{i=1}^{n} Score_i/n \tag{1}$$

The "$Score_i$" indicates the matching degree of every individual general requirement in the service request against the related service attribute in the service description, which is obtained based on the concept types categorized for checking the semantic similarity. For type one, because the subsumption relation exists between these concepts, the score can be obtained based on the semantic distance $||C_r, C_a||$ between the individual requirement (C_r) and the related service attributes (C_a) in the ontology structure. The following equations are used to calculate the individual score:

$$Score_i = \begin{cases} 1 & if\ C_a\ =\ C_r \\ \frac{1}{2} + \frac{1}{2*(||C_r,C_a||+1)} & if\ C_a\ is\ a\ superclass\ of\ C_r \\ \frac{1}{2*(||C_r,C_a||+1)} & if\ C_r\ is\ a\ superclass\ of\ C_a \end{cases} \tag{2}$$

For type two, the score is assumed to be acquired from an available similarity measurement approach [11] [12]. For type three, similar to the similarity checking, the score can be obtained using the percentage deviation from the requested value or a fuzzy membership function.

The matching score of each candidate service is calculated based on $Score_i$, and it will determine the ranking of candidate services. The higher the score is, the higher ranking the candidate service has.

6 Implementation and Experimental Results

The ontology classes for the service description are defined with the OWL language using the Protege toolkit (an open-source ontology editor and knowledge-based framework). Protege can also be used to create OWL-S services by integrating an OWL-S editor plug-in [14].

The Grid service matching middleware has been implemented with the jUDDI toolkit, the Racer system [15], the MySQL database and other related techniques. Figure 2 shows its internal components. The middleware is written as both a Java Web Service for use by other middleware in the system architecture and an AJAX web application which can be accessed through a standard web interface.

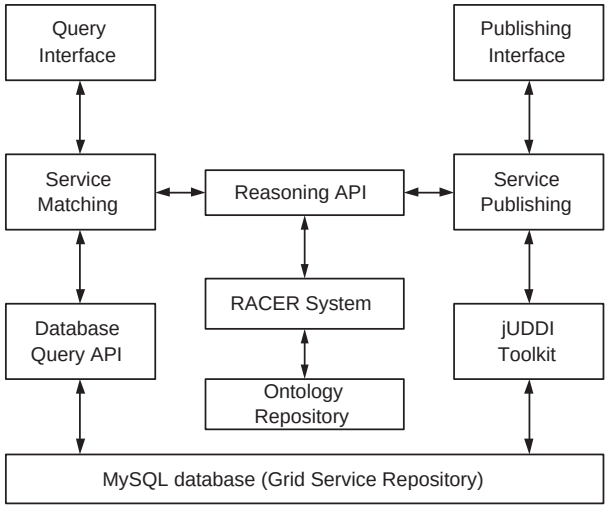

Fig. 2. Internal Components of Service Matching Middleware

We evaluated our service matching middleware in two stages. First, we integrated it into our system architecture to evaluate whether it interfaces well with the other middleware. Second, we measured the performance of our service matching middleware against a number of typical service requests.

In our system architecture, the static distributed Grid resources and mobile devices are interconnected by the Grid gateway. The Grid gateway is a small server available for nearby mobile devices within the covered range through the local wireless network, providing a relatively resource-rich, stable execution environment with enhanced network connectivity for handheld devices. Three kinds of middleware exist in the Grid gateway: the mobile deputy middleware performs a reliable task management mechanism (including task submission, execution, monitoring and result storing on behalf of mobile users) by accepting and packing tasks as the deputy object; the service-based Grid middleware provides access interfaces for Grid service invocation to execute user tasks; the service matching middleware is responsible for the implementation of Grid service registration, discovery and management so that mobile users are able to locate and select required Grid services to achieve their tasks.

In order to evaluate the system performance, we compare our semantic service matching middleware with UDDI, the traditional web service registry. We use the system response time as the performance index and focus on calculating the time required to process a query. The time of publishing a Grid service is not relevant in this study because in our system architecture, mobile users are usually Grid service consumers rather than service providers.

The following experiment was designed in order to obtain the time of querying a Grid service. Both the description information of real Grid services and a large number of pseudo services are published in the service repository. Altogether, more than fifty services can be accessed by the semantic service matching middleware. A UDDI web service registry was built and a number of web services is published onto it. Table 1 shows the average time of querying a service on two different service discovery platform. The time of querying a Grid service with semantic concepts is longer, because the additional computation effort is required to determine the concept subsumption relationships in the logic reasoning system.

	UDDI	Semantic Matching Middleware
Time (ms)	37.4	52.1

Table 1. Time of Querying a Service

Although UDDI has a faster system querying performance than our semantic service matching middleware, it has several shortcomings when used in practice for the service discovery. UDDI does not provide sufficient technical details of the service, does not support any inference based on the concepts, can only support the search based on the string comparison, and cannot identify a match between functionally equivalent services that are described by different key words. Our service matching middleware overcomes these shortcomings by using the semantic service description and discovery mechanism. We believe it is worth obtaining a relatively-significant improvement in system function at the price of a small increase in the service discovery time.

In the above experiment, the time of returning one service only is measured. To test the system scalability, we used five kinds of service repository sizes (10, 20, 50, 100, and 200) and varied the number of matching services to be one, two, four or eight. Figure 3 shows the experimental results. As we expected, the system response time is within an acceptable limit and loosely proportional to the size of the service repository and the number of the matching results.

7 Conclusion

In a service-oriented Grid environment, mobile clients are concerned about three aspects when considering the problem of using Grid services to perform their

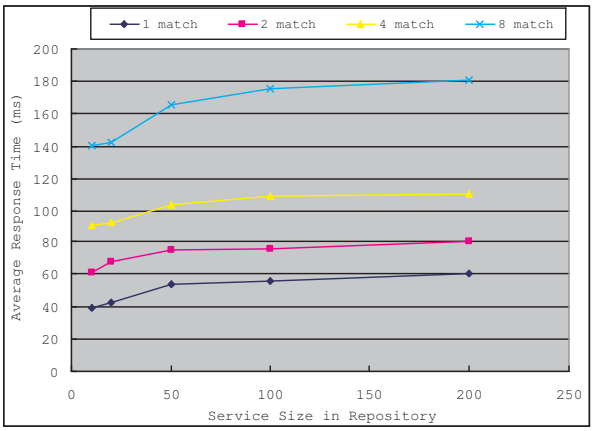

Fig. 3. Average Query Time from Service Matching Middleware

tasks. The first is a method that describes what capabilities Grid services support is required to be developed so that services can be advertised to provide contributions for the task achievement. The second is building a Grid service discovery mechanism so that services can be located by mobile users. The third is the mechanism of Grid service invocation so that the required information or resources can be utilized during the process of the task execution. Essentially, Grid service description, discovery and invocation are interdependent: Grid service description is the prerequisite of Grid service discovery; the mechanism of Grid service discovery determines how a Grid service should be described; the service invocation process depends on the discovery of Grid services.

In this paper, we have presented a semantic service matching middleware for the service-oriented mobile Grid environment. A number of service attributes have been defined to represent service characteristics in the service description. Because of the centralized range of Grid service registry in the system infrastructure, a service search engine is built with the extended OWL-S semantic language and the RACER ontology reasoning system, providing query interfaces for users or other middleware systems to locate required services. The semantic service matching middleware has been integrated into our system architecture and demonstrated to interact correctly with other middleware. We have also measured its performance against a number of service request, and the results show that there is only a small increase in the service discovery time compared to the traditional service discovery mechanism in return for the significant improvement obtained.

In the future, we plan to continue the current research work to allow such a Grid service matching mechanism to be extended so that it can be more suitable for the service-oriented mobile Grid environment. OWL-S supports not only automatic service discovery, but also automatic service invocation, composition and interoperation. At present, we only use the “Profile model” to describe

Grid services. In the future, we will extend the system to use both the "Process model" and the "Grounding model" to build Grid service descriptions, enabling the vision of automatic service discovery, composition, and invocation to be realized.

References

1. Foster, I., Kesselman, C., Teucke, S.: The anatomy of the grid: Enabling scalable virtual organizations. International Journal of Supercomputer Applications **15** (2001)
2. Guan, T., Zaluska, E., Roure, D.D.: Extending pervasive devices with the semantic grid: A service infrastructure approach. In: Sixth IEEE Conference on Computer and Information Technology, Seoul,Korea (2006)
3. Haartsen, J., Allen, W., Inouye, J.: Bluetooth: Vision, goals, and architecture. Mobile Computing and Communications Review **1** (2006) 1–23
4. Waldo, J.: The Jini Specifications. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2000)
5. Allard, J., Chinta, V., Gundala, S., Richard, G.: Jini meets upnp: an architecture for jini/upnp interoperability. In: Proceedings of Symposium on Applications and the Internet. (2003) 268–275
6. Srinivasan, N., Paolucci, M., Sycara, K.: Semantic web service discovery in the owl-s ide. In: Proceedings of the 39th Hawaii International Conference on System Sciences, Hawaii, USA (2006) 109
7. Majithia, S., Ali, A.S., Rana, O.F., Walker, D.W.: Reputation-based semantic service discovery. In: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Modena, Italy (2004) 297–302
8. Li, L., Horrock, I.: A software framework for matchmaking based on semantic web technology. In: Proceedings of 12th International World Wide Web Conference Workshop on E-Service and the Semantic Web, Budapest, HUNGARY (2003) 331–339
9. Martin, D., Paolucci, M., Mcilraith, S., Burstein, M.: Bringing semantics to web services: The owl-s approach. In: First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), San Diego, CA, USA (2004)
10. Bandara, A., Payne, T., Roure, D.D., Lewis., T.: A semantic approach for service matching in pervasive environments. In: Technical Report in IAM group shool of ECS. (2007)
11. Schwering, A.: Hybrid model for semantic similarity measurement. Lecture Notes in Computer Science **3761/2005** (2005) 1449–1465
12. Tverski, A.: Features of similarity. Phychological Review **8** (1977) 327–352
13. Acuna, C., Marcos, E.: Modeling semantic web services: A case study. In: Proceedings of the 6th international conference on Web engineering, Palo Alto, California, USA (2006) 32–39
14. Elenius, D., Denker, G., Martin, D., Gilham, F., Khouri, J., Sadaati, S., Senanayake, R.: The owl-s editor - a development tool for semantic web services. In: Proceedings of the Second European Semantic Web Conference. (2005)
15. Haarslev, V., Moller, R.: Racer: a core inference engine for the semantic web. In: Proceedings of 2nd International Workshop on Evaluation of Ontology-based Tools, Sanibel Island, Florida, USA (2003) 27–36