A Platform to Evaluate the Technology for Service Discovery in the Semantic Web

Cecile Aberg and **Johan Aberg** and **Patrick Lambrix** and **Nahid Shahmehri**
Department of Computer and Information Science
Linköpings universitet, Sweden
{cecab, johab, patla, nahsh}@ida.liu.se

Abstract

Since the description of the Semantic Web paradigm in 2001, technology has been proposed to allow its deployment and use. However, there is not yet any large and widely deployed set of semantically annotated Web resources available. As a result, it is not possible to evaluate the use of the technology in a real environment, and several assumptions about how the Semantic Web should work are emerging. In order to further investigate these assumptions and the related technology, we propose a simulation and evaluation platform. The platform provides tools to create Semantic Web simulations using different technologies for different purposes, and to evaluate their performance. In this paper we introduce the model of the platform and describe the current implementation. The implementation facilitates the integration of technology for an essential operation on the Semantic Web, namely Semantic Web service discovery. We illustrate the use of the platform in a case study by implementing a Semantic Web where the Jade multi-agent platform provides the framework to describe the agents, and a number of existing Semantic Web technologies are embedded in agent behavior.

Introduction

The Web is a very popular source of information and commercial services. However, as documented by (Nielsen 2006), finding a specific piece of information or service using current search engines is still a complex and time consuming task. The performance of current Web technology, such as search engines, is limited by the fact that it is not possible for a computer to fully understand the content of Web resources. This is due to the fact that the Web resources' content is typically written by humans in languages that humans can understand. To allow computers to unambiguously understand the content of Web resources, Tim Berners-Lee formulated the vision of the Semantic Web. Specifically, "the Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" (Berners-Lee, Hendler, & Lassila 2001). With a Semantic Web, a lot of the manual effort done today to find and use Web resources can be automated, at least partially, by having the user delegating tasks such as resource retrieval to software agents (Hendler 2001).

The need for a Semantic Web is underscored by the recent appearance of numerous new applications that refer to the Semantic Web vision and/or rely heavily on the technology being developed for the Semantic Web. Inter-organizational workflows (Patil *et al.* 2004; Dan *et al.* 2004), information retrieval (Huynh, Mazzocchi, & Karger 2005), e-commerce (Trastour, Bartolini, & Preist 2002) and bio-informatics (Lambrix 2005) are examples of the research domains where such new applications are proposed.

Each new proposed application tends to make its own assumptions regarding three aspects of the Semantic Web: 1) the use case, i.e. who are the users and resource providers, what motivates them to use and provide resources, and what are the social and business rules that govern their interaction, 2) the available resources together with their semantic annotations, and 3) the technologies available (e.g. description logic reasoners, rule engines, ontology managers, etc) and how they are used. For approaches to service discovery, such assumptions specify implicit requirements on the *scalability* (e.g. in terms of response time, bandwidth use, or cost) and the *quality of the result* (e.g. measured as precision and recall). However, since there is not yet any large and widely deployed set of semantically annotated Web resources available for experimentation, it is difficult to know when, and if, these requirements are satisfied.

For this reason we propose a common simulation and evaluation platform for service discovery. In this platform current and future Semantic Web technology can be integrated and evaluated with suitable use cases and resource sets. The integration of technology is facilitated by means of an API for common components together with default implementations of these components. The evaluation is facilitated by means of a monitoring tool where event listeners can be employed for creating performance reports on simulation runs.

This paper describes the model and implementation of this simulation and evaluation platform. The use of the platform is illustrated in a case study where service discovery technology is integrated in a specific use case together with a specific set of resources. The evaluation of the technology is illustrated in terms of scalability and quality of result measures, and we discuss lessons learned from the use of the platform.

The rest of the paper is organized as follows. In the next

section we provide more background information about the current Semantic Web technology for service discovery. We then describe the model and implementation of the simulation and evaluation platform. We continue by illustrating the use of the platform and discuss the lessons learned from the case study. We then compare our approach to related work. Finally, we conclude and discuss directions for future work.

Background - Semantic Web

The Semantic Web can be seen as a set of semantically annotated Web resources. A Web resource may be a text, a picture, a piece of software, a representation of an element of the *real* world such as a person, etc. Semantic annotations describe the semantic of the resources so that software agents can reason about them in order to reach a predefined goal. The goals of the agents vary from application to application, but they all rely on the operation of finding and using the resources necessary to perform the goal. To allow the deployment of the Semantic Web, technology is being developed for representing semantic annotations, for finding them, for reasoning about them and for using the resources that they annotate. The technology provides:

Machine-understandable languages to describe the content of Web resources. RDF and OWL are such languages.

Semantic annotation description languages that provide the set of language constructs for describing the properties, capabilities, and use rules of the Web resources in an unambiguous, machine-understandable manner. Semantic Web services is a category of semantic annotations that comes with a specific management framework as defined in (Fensel & Bussler 2002). Semantic Web services are programmable, machine-understandable interfaces that can be attached to a Semantic Web resource in order to provide the information necessary for software agents to decide if they need to use the specific resource or not. As pointed out in (Lara *et al.* 2003), Semantic Web services are designed to support the operation of resource discovery, composition and interoperability. There are several language propositions for Semantic Web services, such as OWL-S[1], WSMO[2], and OWL-DTP[3].

Semantic-aware tools that use and manage the semantic annotations, as well as the ontologies[4] that the annotations may refer to. Examples of tools that use semantic annotations are Semantically enhanced web browsers like Piggy Bank (Huynh, Mazzocchi, & Karger 2005). Examples of ontology management tools are ontology editors such as Protégé[5], and ontology aligners and mergers such as SAMBO[6]. Examples of management tools for the semantic annotations are automatic generators of semantic annotations such as the one-click publishing process of IRS III illustrated in (Domingue *et al.* 2004). Examples of tools that use ontologies are logic reasoners. Currently the most successful reasoners are using description logics, and one of the most popular such reasoner is Racer (Haarslev, Möller, & Wessel 1999 2006). Reasoners relying on other logics (e.g. F-logic (de Bruijn *et al.* 2005)) are also being proposed.

Semantic Web operations that include resource retrieval, resource use, and Semantic Web management operations such as handling the changes in the resources' content. When the semantic annotations are Semantic Web services, the operation of resource retrieval is called service discovery. These operations use semantic-aware tools. The operations are complex, and solutions are just emerging for applications where semantic annotations are formulated as Semantic Web services. WSMX[7] and IRS III (Domingue *et al.* 2004) apply the recommendation of the Web service modeling framework (Fensel & Bussler 2002) to describe service discovery and service execution. There are some attempts to describe operations that handle changes in the state of resources on the Semantic Web. However, the semantic-aware tools required for such technology are still under development. The work done in the REWERSE network[8] aims at providing such technology.

Platform Model

The simulation and evaluation platform must provide the support to 1) generate simulations of service discovery operations, and 2) generate evaluation reports.

In order to generate a simulation, we need a model of the Semantic Web as well as support to instantiate this model with respect to a set of specific assumptions about the use case, resources, and technology used.

We propose a Semantic Web model with four components: the Web resources, the machine-understandable data, the language specifications, and the operations. The Web resources provide some semantic content presented in a format that may not be machine understandable. The machine-understandable data includes the semantic annotations of the Web resources, the possible queries for resources, and the ontologies available to describe and reason about the annotations and the queries. The language specifications include the machine-understandable languages and the Semantic annotation description languages mentioned in the previous section. The operations are the different approaches to service discovery. Furthermore, the Semantic Web allows for modeling applications as a set of software agents with different capabilities, which collaborate to perform specific goals (Hendler 2001). Operations are such applications. They can thus be represented by a set of *agent*s that embed one or several *semantic-aware tool*(s), and may collaborate with each other by exchanging messages whose content is written in one of the available *languages*.

[1] http://www.daml.org/services/owl-s/1.0

[2] http://www.wsmo.org/

[3] http://www.ida.liu.se/~iislab/projects/SWSlanguages/OWL-DTP/20051124/

[4] From (Neches *et al.* 1991): "An *ontology* defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary."

[5] http://protege.stanford.edu/

[6] http://www.ida.liu.se/~iislab/projects/SAMBO/

[7] http://www.wsmx.org

[8] http://rewerse.net

Figure 1 illustrates the use of the platform where the set of assumptions pictured in the ASSUMPTIONS box is used by the components of the platform (represented in the PLATFORM box) to generate a monitored Semantic Web as sketched in the SIMULATION box. The support tools provided by the platform use the assumptions about the resources to 1) instantiate the Web resource component by gathering the resource URIs in a single database, 2) gather the semantic annotation in another database that refers to the database of resource URIs, and 3) generate a set of *service provider* agents in charge of advertising and managing the resources. The instantiation of the language specifications component of the platform requires the identification of the set of languages used in the use case, the technology, and the data. The instantiation of the machine-understandable data component requires the gathering of the semantic annotations, the queries defined by the use case, and the ontologies to which the annotations and the queries refer. As illustrated in the OPERATION box in figure 1, the instantiation of the operation component requires the implementation of the service discovery operations as multi-agent systems where each agent packages specific uses of semantic-aware tools.

Further, to facilitate evaluation, the platform must allow definition of different settings of the same simulation in terms of, for example, the number of resources used or the number of agents available with a specific behavior. This is handled by a specific set of support tools represented by the "Use Case Settings" in the PLATFORM box.

Finally, in order to evaluate a Semantic Web simulation, a monitoring mechanism is required. We propose to adopt an event listening approach where the different components of the simulation can generate events. As a result, and as illustrated with the "Evaluation support" in the PLATFORM box, the platform provides the API for implementation of specific monitoring behaviors that listen to specific events and compute specific evaluation reports, and a monitoring agent in charge of running parallel threads for each of these behaviors.

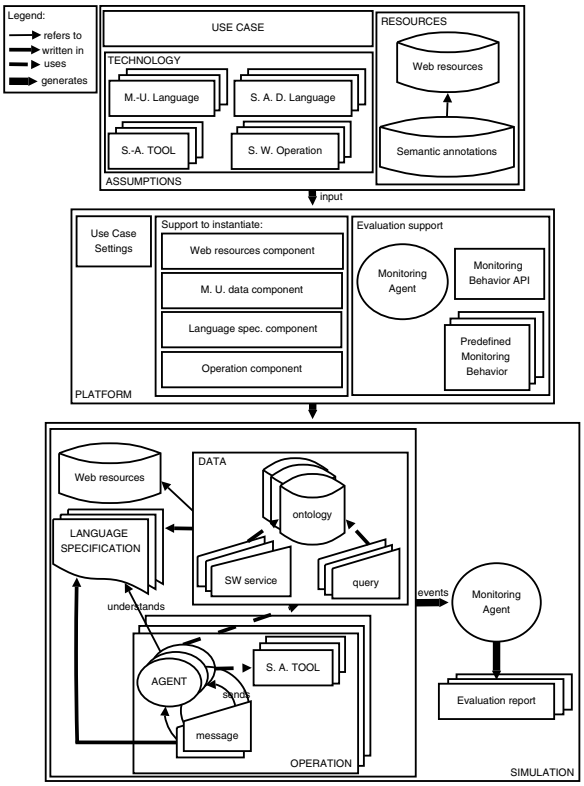

Figure 1: Platform Model

Platform Implementation

As a first step towards a full implementation of the support tools provided by the platform model, we implemented the evaluation support, some support for changing the settings of the simulation, and some support for the operation component. The operation component requires the most complex support. Each operation requires identification of the categories of agents that will participate, the algorithms that each agent will implement, and assurance that the agents establish coherent collaborations. In the current implementation of the platform we provide the following support to create service discovery operations:

- The description of the different categories of agents that typically take part in the operation of service discovery. Concretely, the description consists of:
  - A set of agent categories in natural language (see below).
  - An API description of each agent category in terms of the minimal set of functions that they must provide, and the minimal set of messages that each agent is expected to be able to interpret.
  - An illustrative implementation of one Semantic Web simulation corresponding to the Travel scenario discussed in the case study in the next section.

  With these tools, the potential users of the platform do not have to identify their own agent categories, but can focus on specifying the agent categories that are taking part in the operation that they want to implement, and what mode of collaboration they must adopt.
- One default implementation for each agent category. This is useful for users who do not wish to specify all the agent behaviors, but only the specific ones corresponding to the specific technology that they want to test.
- A mechanism for supporting monitoring and an illustrative monitoring tool that is able to compute the time to get an answer to a specific request message. This allows evaluation data.

When it comes to the actual implementation of these tools we considered two facts. First, the implementation of the operations requires integration of different technologies written in different programming languages, possibly running on different machines. Second, to allow for the comparison of different technologies and different settings, the evaluation platform should provide the means to minimize the effort required by changing one or several technologies used by

the operations. By providing the possibility to describe applications whose architecture is strongly decoupled, multi-agent systems as defined by FIPA, provide an environment that support these needs. We thus implemented the support above in Jade[9], a Java framework that provides the means to describe FIPA multi-agent systems. The API is a set of Java interfaces and the messages exchanged by the agents are ACL messages whose content is written in a Semantic Web language such as OWL. We further adopted the service discovery solution of the multi-agent system introduced in (Aberg, Lambrix, & Shahmehri 2005). As a result, the current implementation of the support for instantiating the operation component provides an API and a default implementation for the following set of agent categories:

A Requester is able to formulate a query for a specific service, and to send it to the agent(s) able to start up the process of service discovery, i.e. the Web service discovery agents described next. A Requester may also be able to enact a service once it is discovered.

A Web service discovery agent is able to find the services that match a given query for services. Web service discovery agents may also be able to discover compositions of services that match the query.

A Web service manager is a directory of Semantic Web services. Web service managers are associated to one or several Semantic Web service description languages such as OWL-S, WSMO or OWL-DTP. A Web service manager is able to answer queries for specific Web services. A Web service manager does not perform composition of services.

A Service provider sends service advertisements to Web service managers. The service advertisements are formulated as Semantic Web services.

An Ontology Agent (OA) is able to reason about a specific domain (e.g. Travel, Car.) Any agent can delegate part of their reasoning to ontology agents. OAs can answer several types of queries such as "is A a subclass of B?" , "what are all the subclasses of class C?" or "what are all the instances of class C?"

For each agent category above, the API specifies the minimal set of behaviors that they must provide as well as the minimal set of messages that they must understand. Each of the agents' behavior can embed one or several Semantic Web technologies. For example, requester agents may embed query editors, which in turn may refer to ontology browsers. Semantic Web managers may typically embed Semantic Web service matchmaking algorithms. Service discovery agents can embed composition algorithms that may refer to work done on choreography and orchestration. Ontology agents embed ontology technologies such as editors, aligning tools, and domain specific matchmakers. Service providers may embed technology such as automatic generators of Semantic Web services (Domingue *et al.* 2004; Ciravegna 2004).

[9]http://jade.cselt.it/

Moreover, the support also provides a java package of the classes implementing the minimal set of messages and providing the methods to parse the content of the messages. In order to allow the users of the platform to focus on the integration and monitoring of their own technology, the implementation of the platform also provides for a default behavior for each agent. Finally, to satisfy the platform model's requirements on a monitoring agent, the current implementation of the platform provides a Jade **Monitor agent** which can run several monitor behaviors in parallel. We also provide one MonitorAnswerTime behavior that observes the time when a message is sent and when an answer is received in order to compute the resulting *answer time*.

Illustration

To illustrate the use of the platform we show how service discovery technology was integrated and evaluated for a specific use case and with a specific set of Web resources. Lessons learned from the case study with respect to the platform's ease of use are discussed in the next section.

Assumptions

Our initial assumptions with respect to the use case, the service discovery technology, and the available Web resources are as follows. For the use case, we assume the Semantic Web to be an open world where requesters and service providers can specify the kind of transaction that they agree to participate in (e.g. buying, lending, acquiring for free). The service providers provide travel itineraries and requesters query for specific travel itineraries, and expect to get answers at least as quickly as when they consult a database of travel itineraries.

As for the service discovery technology, we assume that the underlying architecture corresponds to the agent architecture introduced in (Aberg, Lambrix, & Shahmehri 2005) where the Semantic Web service language is OWL-DTP and the Web Service manager agent integrates the OWL-DTP matchmaking algorithm introduced in (Aberg 2005). This algorithm requires the use of description logic reasoning for which the Racer system is used. The Jena-DIG interface is used as a Java interface to Racer. The matchmaking algorithm is implemented in a straightforward way that requires each query to be matched against each service description. Each operation of matching a query and a service, requires a set of reasoning operations including some subsumption operations. To implement the full service discovery operation, we use the default agents provided by the platform and package our matchmaking algorithm as a Web Service manager and a Travel ontology agent.

With respect to the Web resources available we consider a set of 183 services providing travel itineraries. These services correspond to those provided by the different Web sites of the travel providers that the employees of our department are authorized to use when planning work-related trips. We also consider a set of 14 queries corresponding to real travel queries expressed by some employees when planning their trips. There are two categories of queries. Queries that will require a composition of services (e.g. "Give me an itinerary

to go from Stanford University, Palo Alto, CA, USA, to the Conference center in Chiba city, Japan"), and queries for which service composition is not always necessary (e.g. "Give me all the flights to go from Heathrow Airport, London, UK, to Kastrup Airport, Copenhagen, Denmark").

Evaluation

Scalability We measure the scalability of the service discovery approach with respect to the number of services and the technical capabilities of the machines running the agents, by measuring the average response time to the queries. To do that we use the MonitorAnswerTime behavior provided by the platform. Additionally, all the agents trigger an event when they send or receive a message. We run the simulation in different settings where the agents run on different machines.

This first set of evaluation runs teaches us that the triplet Jena-DIG-Racer cannot run the required knowledge base on a machine with too little CPU and RAM. Concretely, the reasoner freezes if it uses the Travel ontology agent knowledge base on a pc with the x96 Family 6 Model 1 stepping 9 processor (ca. 199 MHz) and 64 MB of RAM. Further, the reasoner that uses both the knowledge base for the Web service manager and the Travel ontology agent[10] and runs on a pc with an Athlon processor (1.33 GHz) and 512 MB of RAM, freezes after treating a random number of queries (ten, eleven or even forty). We identified one machine setting that works well for our technology use: the reasoner that uses the Web service manager knowledge base runs on a pc with an Athlon processor (1.33 GHz) and 512 MB of RAM, and another reasoner that uses the Travel ontology agent knowledge base runs on a pc with an Intel Pentium M processor (ca. 1400 MHz) and 512 MB of RAM. Additionally, in the machine settings providing for the best average time for the set of 183 services, we obtain an average response time to the queries of approximately 14 minutes. This is clearly not an acceptable performance with respect to the use case. Upon more detailed inspection we find that the reason for this great delay in response time is that the current matchmaking approach performs approximately 300 subsumption operations per query. Most of these operations are required to match the travel itineraries.

Given these observations we design a new matchmaking algorithm such that the Web Service manager decomposes the OWL-DTP representation in three components, and indexes them at service advertisement time. The indexing of the components referring to travel itineraries is performed by the Travel ontology agent, which stores the generated indexes in a database. The indexes are then used at query time. We change the behavior of the Web Service manager and Travel ontology agent to integrate the new algorithm. The new algorithm requires two subsumption operations and one SQL query to match a query with all the available services. Running the simulation now provides an answer in 10 seconds on average. This is a result that better fits the use case requirements with respect to time, even if there is still room for improvement.

The monitoring also provides the time to advertise the services. With the straightforward algorithm, it takes ca. 28 seconds to advertise 183 services in one Web service manager. With the second version of the algorithm, it takes ca. 183 seconds to advertise 183 services in one Web service manager. The preprocessing done at advertisement time takes its toll. However, it is still a reasonable processing time for advertisements since they need to be done only once per service in this use case.

Result quality In order to measure the quality of the result we measure precision and recall for each query. This is done by implementing a monitoring behavior that compares the set of services returned for each query, with a precompiled ideal set of services. The results show that we obtain 100% precision and recall for the 3 queries that request one specific travel leg (i.e. they correspond to one or several existing services), showing that the service description language is suitable for the corresponding information needs. For the other 11 queries that requested travel itineraries composed of several legs, and thus requiring service composition, we got 0% precision and recall. This result provides us with a clear next step for the development of a complete service discovery operation, namely to package a service composition algorithm as a Web Service discovery agent behavior and evaluate how that would influence the precision and recall of the corresponding queries.

Summary

We have illustrated how the platform was used in a case study. We showed how service discovery technology was evaluated and analyzed, in terms of scalability and result quality, and refined, based on assumptions in the use case. This analysis helped us narrow down the main performance bottleneck of the technology. After fine-tuning the matchmaking algorithm the platform also facilitated the comparison with the previous version, while indicating the unwanted side effect of increased advertisement time that the new algorithm implied. All in all, the platform helped us maintain a high-level view of the service discovery problem, while facilitating our work on the details.

Lessons Learned

When implementing the service discovery approaches described above, we noticed three clear advantages of using the platform. The first advantage concerned time gain at design time. When pondering how to implement the assumptions of our case study in a service discovery operation, the platform provided us with a clear model of the operation. We immediately identified the need for a requester, a set of service provider and a Web service manager agent. The platform also made us consider the decomposition of the matchmaking algorithm so that the travel-related part of the reasoning would be delegated to a specific ontology agent. This is a good design choice if we consider that we will later want to extend the scope of services.

The second advantage concerned both debugging and the integration of the second version of the matchmaking algo-

[10]Jena-DIG related note: both knowledge bases are defined in their own model.

rithm. In both cases, because of the strongly decoupled architecture of the implementation, including the different behaviors implemented by each agent, the code rewriting could be done locally, requiring very little, if any, rewriting of the code of other behaviors.

The third advantage is also connected to ease and rapidity of implementation: the predefined package of messages allowed us to very quickly set up the communication between agents.

All this allowed us to concentrate on the one task that was really important to us: integrating the matchmaking algorithm and evaluating its performance.

Related Work

The similarity of the paradigms of the Semantic Web and multi-agent systems has been acknowledged by others. However, most other work concentrates on providing an interface between multi-agent systems and the Semantic Web. Jade does go in the direction of supporting the integration of the Web paradigm in multi-agent systems by providing the possibility to use the HTTP protocol as the communication protocol between agents. However, more advanced features such as the management of Semantic Web resources are not taken into account by any other agent approach that we know of. IRS III (Domingue *et al.* 2004) and WSMX do provide platforms to manage the life cycle of Semantic Web services in terms of service discovery. However they force the use of one Semantic Web service representation (i.e. WSMO), which may not fit all Semantic Web use cases. Further, they do not provide any means to evaluate and compare different approaches.

Conclusions and Future Work

We have highlighted the need for a platform to support the integration of Semantic Web technology to build service discovery operations and evaluate them with respect to scalability and quality of the results generated. We have provided the model and an implementation of such a platform, and illustrated its use on a service discovery operation in the travel domain. The platform allowed us to integrate service discovery technology, identify their weaknesses, and limit the effort to change parts of the implementation. Our work is a first step towards providing a full-fledged platform for simulating and evaluating the Semantic Web. As for the future, we plan to complete the current support for describing service discovery, and provide support for the other operations of service use and Semantic Web management as well.

References

Aberg, C.; Lambrix, P.; and Shahmehri, N. 2005. An Agent-based Framework for Integrating Workflows and Web Services. In *IEEE WETICE workshop on Agent-based Computing for Enterprise Collaboration*, 27–32.

Aberg, C. 2005. *Integration of Organizational Workflows and the Semantic Web*. Licentiate thesis, Linköpings universitet.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The Semantic Web. *Scientific American*.

Ciravegna, F. 2004. Amilcare - adaptive IE tool. http://nlp.shef.ac.uk/amilcare/ (Accessed 2006-02-13).

Dan, A.; Davis, D.; Kearney, R.; Keller, A.; King, R.; Kuebler, D.; Ludwig, H.; Polan, M.; Spreitzer, M.; and Youssef, A. 2004. Web services on demand: WSLA-driven automated management. *IBM Systems Journal* 43(1):136–158.

de Bruijn, J.; Polleres, A.; Lara, R.; and D.Fensel. 2005. OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web. In *the 14th International World Wide Web Conference*, 623–632.

Domingue, J.; Cabral, L.; Hakimpour, F.; Sell, D.; and Motta, E. 2004. IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services. In *Workshop on WSMO Implementations*.

Fensel, D., and Bussler, C. 2002. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications* 1(2):113–137.

Haarslev, V.; Möller, R.; and Wessel, M. 1999-2006. Racer: Semantic Middleware for Industrial Projects Based on RDF/OWL, a W3C Standard. http://www.sts.tu-harburg.de/ r.f.moeller/racer/ (Accessed 2004-12-08).

Hendler, J. 2001. Agents and the Semantic Web. *IEEE Intelligent Systems* 16(2):30–37.

Huynh, D.; Mazzocchi, S.; and Karger, D. 2005. Piggy Bank: Experience the Semantic Web Inside Your Web Browser. In *International Semantic Web Conference*, 413–430.

Lambrix, P. 2005. Towards a Semantic Web for Bioinformatics using Ontology-based Annotation. In *14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, 3–7. Invited Talk.

Lara, R.; Lausen, H.; Arroyo, S.; de Bruijn, J.; and Fensel:, D. 2003. Semantic Web Services: description requirements and current technologies. In *International Workshop on Electronic Commerce, Agents, and Semantic Web Services, In conjunction with the Fifth International Conference on Electronic Commerce (ICEC 2003)*.

Neches, R.; Fikes, R.; Finin, T.; Gruber, T.; Patil, R.; Senator, T.; and Swartout, W. 1991. Enabling Technology for Knowledge Sharing. *AI Magazine* 12(3):26–56.

Nielsen, J. 2006. Jakob Nielsen's Alertbox, February 6, 2006: Users Interleave Sites and Genres. http://www.useit.com/alertbox/cross_site_behavior.html (Accessed 2006-02-08).

Patil, A.; Oundhakar, S.; Sheth, A.; and Verma, K. 2004. METEOR-S Web service Annotation Framework. In *International World Wide Web Conference*, 553–562.

Trastour, D.; Bartolini, C.; and Preist, C. 2002. Semantic Web Support for the Business-to-Business E-Commerce Lifecycle. In *International World Wide Web Conference*, 89–98.