Applying Semantic Web Technologies to Matchmaking and Web Service Descriptions

Amer Al Shaban and Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University, Montreal, Canada

1 Introduction

The recent expansion of representing web services using agents is causing difficulties in finding specific types of web services. The main reason for these problems is the employed matchmaking techniques. Most of the existing techniques are based on search using string comparison, so, if service providers neglect to provide sufficient or appropriate terms for the matchmaking process, the search techniques will return incomplete results. This paper addresses the problem of matching requested services to proper provider agents by making use of OWL (Ontology Web Language) ontologies and the OWL reasoner RACER. In the following we first describe the used tools, and then introduce an implemented prototype, where an agent (MatchMaker) was added to an existing agent framework (DECAF), where the new matchmaker employs OWL-S for matching requests to available services.

2 Tools

2.1 DECAF

DECAF (Distributed, Environment-Centered Agent Framework) is a toolkit which allows a well-defined software engineering approach to build multi *agent* systems. The toolkit provides a stable platform to design, rapidly develop, and execute intelligent agents to achieve solutions in complex software systems. DECAF provides the necessary architectural services of a large-grained intelligent agent communication, planning, scheduling, execution monitoring, coordination, and eventually learning and self-diagnosis [1].

DECAF has been used in the provided prototype as the framework for agents, which takes care of the communication part and the assurance of populating agents without any conflicts.

2.2 RACER

RACER (Renamed ABox and Concept Expression Reasoner) is a description logic reasoner for OWL DL that implements a highly optimized tableau calculus for very expressive description logics. It also offers reasoning services for multiple TBoxes and for multiple ABoxes encoded as OWL DL knowledge bases [2].

RACER is used as the reasoner for provided OWL ontologies (which will be explained in the next section), so the seeker agents would have the ability of querying the ontologies about provided web services.

2.3 Ontologies

The definition of an ontology differs from one context to the next. Generally an ontology can be considered as a formal description of ***classes*** in a domain of discourse (also called ***concepts***), where classes describe common characteristics of individuals, properties of individuals specify common features and attributes of a concept ***slots*** (also called ***slots*** or ***roles***), and restrictions on slots called ***facets*** (also called ***role restrictions***). Ontologies together with a set of assertions about individuals or ***instances*** of classes usually define a ***knowledge base***. In reality, there is a fine line where an ontology ends and a knowledge base begins [3]. In general we can say that an ontology is an explicit specification of a conceptualization where it makes concepts more specific by using other concepts and roles [4].

Ontologies are used to specify offered web services. If a certain agent is providing a service, it will specify the service using an ontology format, such as OWL (Ontology Web Language), when it is registering this service. In this way, the ontology specification will be the reference for the service when it is being retrieved.

2.4 OWL-S

OWL-S is a OWL-based web service ontology, which supplies web service providers with a core set of markup language constructs for describing the properties and capabilities of their web services in unambiguous, computer-interpretable form. The OWL-S markup of web services will facilitate the automation of web service tasks, including automated web service discovery, execution, composition and interoperation [5].

3 Suggested Solution

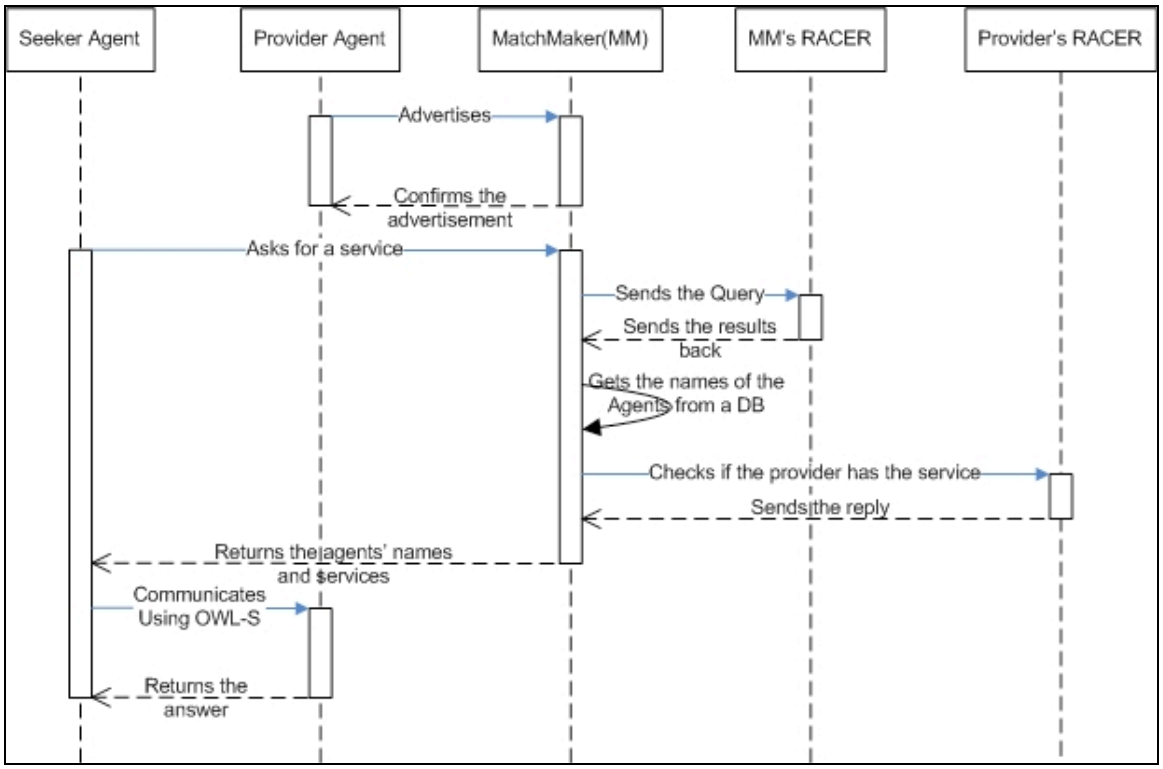
After briefly mentioning the used tools and the main problem, the suggested solution (provided in the sequence diagram, see below) is discussed. In general, the presented solution can be divided into three main steps. An agent providing web services (*Provider Agent*), an agent requesting web services (*Seeker Agent*), and an agent which takes care of matching the seeker agent to correct provider agents (*MatchMaker*).

3.1 MatchMaker

In order for an agent to find a certain service from another agent, there should be a third party which takes care of that specific issue. That third party is called the Matchmaker. The matchmaker is middle-ware agent that comes with DECAF, which is in charge of matching requested services to proper provider agents. But the problem is that the matchmaker's searching technique is based on string comparison which, as mentioned before, can cause performance problems and produce incomplete results for seeker agents. So the main part of the provided solution is to re-implement the matchmaker by replacing string comparison with reasoning based on ontologies.

The new matchmaker was re-implemented with several new features. The main feature is having an upper ontology that includes everything related to the different agent domains. That upper ontology acts as glue which connects all the agents' domains. Thus, when a seeker searches for a service, the upper ontology is used as a grounding for

communication and by getting only the related agents. Another feature is the use of a database to store information about agents (e.g., their names and offered services) instead of storing it in a text file. The database is used to keeps track of the requests while one is searching for matching agents.



Sequence diagram illustrating the agent communication.

There are three basic services implemented in the matchmaker. *Advertisement*, where it expects two parameters. The first one is a set of **concepts**, which includes all the related concepts for that service which occur as leaves in the upper ontology (e.g., if an agent is providing apartments, and the upper ontology includes housing as a leaf, then the agent will provide housing as a concept for that service), and the URI of an **Ontology** which will be used in the protocol implementing the agent communication. The second service is *asking*, which basically addresses the search part in the matchmaker. This service is called by the seeker agents. It expects two parameters, **ConceptBasedQuery** which is a general query that will be used as a filter on the upper ontology at the matchmaker's side to get all the related provider agents**,** and **RQL** (Racer Query Language) which describes a more specific query that will be used at the provider agent's side to check whether the filtered agents truly provide the requested service. The third service is *deeper*; this service in particular will never be called by any agent but the matchmaker itself. It takes the list of filtered agents form the *asking* service, and sends the RQL query to each of them and gets the answer from them. Eventually it returns the matched agents' names with their offered services to the seeker agent.

3.2 Provider Agent

The provider agent represents a web service, and makes it possible for any seeker agent to use that web service. It was implemented with two main features, *deeperSearch* and *activateService*. The deeperSearch service is the one which deals with deeper service from the matchmaker side. It takes the RQL query, poses it to RACER by referring to the corresponding ontology. It replies back with the service name if available, or with none if no matching services could be found on the basis of this ontology. The activateService is used by the seeker agent after receiving the results from the matchmaker. This service makes use of OWL-S by representing the existing web service in an ontology format, so that a seeker agent can retrieve the parameter profile of the service and executes the web service call. Using OWL-S makes the representation of web services more flexible and dynamic because the interface of an agent might change at almost any time. The service description is compatible with WSDL and agents can adapt to changed services at runtime without the need to code of any of the involved agents. The OWL-S description specifies a communication protocol between the seeker agent and provider agent for executing the web services successfully.

3.3 Seeker Agent

The seeker agent basically communicates first with the matchmaker, retrieves the names of the matching provider agents, and then communicates with the provider agents to execute the web service using OWL-S.

4 Conclusion

So far there does not exist a convenient language or representation for building agents that depend on ontologies. All of the previous work done in those fields show that there might be a bright future for such an approach. In this short paper, a brief description of the integrated tools was presented. Moreover, a new way to integrate ontologies with agents in DECAF has been suggested with the ability of reasoning the ontologies using RACER. The described prototype has been implemented in Java using the mentioned tools. We tested it using a selling/buying scenario with various configurations of agents. Hopefully this prototype can be a good start point for switching to the emerging semantic web and using ontologies with mobile agents in a wider application area.

References

[1] *John R. Graham, Keith S. Decker, Michael Mersic*, DECAF - A Flexible Multi Agent System Architecture, *Autonomous Agents and Multi-Agent Systems*. 2003.

[2] *Volker Haarslev, Ralf Möller*, Description of the RACER System and its Applications, Proceedings of International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3. August 2001.

[3] *Thomas R. Gruber*, A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 1993.

[4] *Holger Knublauch, Mark A. Musen, Natasha F. Noy*. Creating Semantic Web (OWL) Ontologies with Protégé. International Semantic Web Conference, Sanibel Island, Florida, USA, October 20-23, 2003.

[5] OWL-S (OWL Web Service) 1.0 Release http://www.daml.org/services/owl-s/1.0/