IDEA GROUP PUBLISHING **ITJ3286**

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

This paper appears in the publication, International Journal of Intelligent Information Systems, Volume 2, Issue 3
edited by Vijayan Sugumaran © 2006, Idea Group Inc.

Applying Advisory Agents on the Semantic Web for E-Learning

Ralf Bruns, University of Applied Sciences and Arts Hannover, Germany

Jürgen Dunkel, University of Applied Sciences and Arts Hannover, Germany

Sascha Ossowski, Universidad Rey Juan Carlos Madrid, Spain

ABSTRACT

In this article, we present the software architecture of a new generation of advisory systems using Intelligent Agent and Semantic Web technologies. Multi-agent systems provide a well-suited paradigm to implement negotiation processes in a consultancy situation. Software agents act as clients and advisors, using their knowledge to assist human users. In the presented architecture, the domain knowledge is modeled semantically by means of XML-based ontology languages such as OWL. Using an inference engine, the agents reason, based on their knowledge to make decisions or proposals. The agent knowledge consists of different types of data: on the one hand, private data, which has to be protected against unauthorized access; and on the other hand, publicly accessible knowledge spread over different Web sites. As in a real consultancy, an agent only reveals sensitive private data, if they are indispensable for finding a solution. In addition, depending on the actual consultancy situation, each agent dynamically expands its knowledge base by accessing OWL knowledge sources from the Internet. Due to the standardization of OWL, knowledge models easily can be shared and accessed via the Internet. The usefulness of our approach is proved by the implementation of an advisory system in the Semantic E-learning Agent (SEA) project, whose objective is to develop virtual student advisers that render support to university students in order to successfully organize and perform their studies.

Keywords: agents; e-learning; ontology; Semantic Web application

INTRODUCTION

E-learning has started to play a major role in the learning and teaching activities at institutions of higher education worldwide (Hamdi, 2006). Students perform significant parts of their study activities decentralized via the Internet. The main focus of current e-learning systems is to provide an appropriate technical infrastructure for content engineering and information exchange.

The emerging trends in individual ways of study are location- and time-independent, consequently requiring a permanently available and direct support in order to answer questions and give advice. A recent comparison of modern e-learning environments (CCTT, 2004) revealed that intelligent advisory agents are not applied so far in e-learning systems.

The objective of the Semantic E-learning Agent (SEA) project (Dunkel, Bruns, & Ossowski, 2004) is to develop virtual student advisers that render support to university students, assisting them to successfully organize and perform their studies. The experiences of human course advisers show that most students have similar problems and questions. The advisory agents should help to resolve these problems. Typical questions concern the regulations of study (e.g., does a student possess all requirements to participate in an examination or a course?) or organizing student mobility.

To achieve these goals, we propose a software architecture of an advisory system in which virtual student advisers are developed with novel concepts from the Semantic Web (Berners-Lee, Hendler, & Lassila, 2001; Horrocks & Hendler, 2002) and Intelligent Agent (Wooldridge & Jennings, 1995) technologies.

The Semantic Web can be defined as an "extension of the current web in which information is given well-defined meaning" (Berners-Lee et al., 2001). The basic idea is to represent domain data and their structure in a well-defined and machine-interpretable way. For this purpose, ontology languages based on XML and RDF/RDF Schema (W3C-RDF, 2004) are defined. The W3C consortium announced the standard ontology language OWL (Web Ontology Language) (W3C-OWL, 2004). Ontology languages allow the explicit formal specification of the entities in a domain and the relations among them. They can encode the knowledge accessible on different Web sites making it understandable for computer programs.

One essential aspect of our proposed software architecture is to model the structure of the e-learning domain by means of ontologies and to represent it by XML-based ontology languages. Software agents apply the knowledge represented in the ontologies during their intelligent decision-making process. We claim that this is a promising approach, because e-learning systems that successfully support students in organizing their studies are still to come. This article reports on the experiences gained from the development of an advisory system architecture that effectively integrates both Semantic Web and intelligent agent technologies.

The first use case that has been implemented reflects the counseling situation in which a student intends to study a semester abroad within the European Socrates/Erasmus exchange program. Together with the international coordinator, the student has to choose the foreign university and the foreign study program that best matches his or her personal interests and his or her situation of study. Subsequently, a study plan for the semester at the host university must be determined that corresponds to the home university syllabus. This study plan constitutes the so-called Socrates Learning Agreement.

The remainder of this article is structured as follows. In the next section the knowledge representation techniques and the knowledge models developed are presented. The next section shows how automated inference can be carried out on the knowledge models. Subsequently, the

software architecture of the agent-based advisory system is outlined. Finally, the last section summarizes the most significant features of the project and provides a brief outlook of future lines of research.

KNOWLEDGE MODELING

The key concept of a semantic advisory system is the semantic modeling of the domain knowledge (e.g., university organization, degree requirements, course descriptions, examination regulations) as well as an individual user model, which reflects the current situation of study (e.g., passed exams, registered courses). The fundamental structures of the available domain knowledge as well as the basic facts (e.g., offered courses) are defined in appropriate models.

In our architecture, the structural part of the knowledge base is modeled by means of ontologies, which formally define domain entities and the relations among them. For this purpose, we apply Semantic Web technology using the W3C standard ontology language OWL to model the knowledge required in the advisory system.

Software agents employ this information as the basis for their reasoning and negotiation. Due to the standardization of these technologies, knowledge models easily can be shared and reused via the Internet. Thus, the developed ontologies can serve as standardized and open interfaces for the interoperability of different advisory systems.

Ontologies

In order to implement the counseling situation of the Socrates/Erasmus exchange program, information is necessary about the possible exchange universities and their offered degree programs. In addition, further information about the living conditions of a particular university city and its urban infrastructure may influence the decision.

Several interrelated ontologies have been developed for our advisory agents: Two central ontologies describe the organizational structure of a university and the offered courses in a semester. In order to facilitate the comparison of different study places and course contents, two subontologies are used. Furthermore, the individual study situation of a specific student is represented by a separate ontology.

Dividing the knowledge base of the advisory architecture into several ontologies is crucial in order to yield a coherent scope for each ontology and to facilitate the reuse of existing ontologies (Noy & McGuiness, 2001). In the following, we describe the responsibilities of the employed ontologies in more detail.

- **University Ontology** The university ontology is the core knowledge base of the case study. It models the essential parts of the organizational structure of a particular university and the departments with different programs of study. Its main domain concepts are university, department, degree program, and offered degrees. The following example shows an excerpt of an instance of the university ontology.

```
<uni:DegreeProgram
        rdf:ID="FHH_Master_CS">
  ...

    <uni:numberOfStudents  rdf:
    datatype=
   "http://...XMLSchema#int" >
  547
```

```
</uni:numberOfStudents>

   <uni:hasContent rdf:re-
    source=
   "http://../subject.
    owl#softwareEng"/>

   <uni:hasContent rdf:re-
    source=
   "http://../subject.
    owl#compGraph"/>

 ...
</uni:DegreeProgram>
```

At first, a degree program instance with `ID FHH_Master_CS` is created. The property `numberOfStudents` specifies how many students are enrolled and has the XML schema data type int. The property `hasContent` describes the content of the degree program and refers to a computer science instance of the subject area ontology specified by the URI.

- **Course Ontology** The course ontology models the courses per semester for a degree program. This information changes each semester and only can be provided by the responsible department. Several properties describe an individual course (e.g., course name, teaching language, number of credit points, keywords describing the course content, and the semester when the course takes place). This knowledge will be used in the second step of our sample use case when open courses of the home syllabus are matched with courses at the exchange university.

Each university participating in the Socrates program should build its own instances of these ontologies. Furthermore, for our counseling scenario, we need further information that is provided by two additional ontologies.

- **Regional Ontology** The regional ontology models the relevant properties of a study place, e.g. in which country, state, and region it is located, its number of inhabitants, which infrastructure is available (e.g. airport, station, theatre). Each study place is represented by an instance of this ontology, thus allowing a comparison due to the students living preferences. It is expected that for many cities this information will be available on the Semantic Web in the near future.
- **Subject Area Ontology** In order to find an appropriate study plan at the exchange university, home and foreign courses must be compared, based on their contents. A simplified taxonomy is modeled in the subject area ontology (e.g., one instance for computer science, one instance for mechanical engineering, etc.).

The presented ontologies define some transitive properties that are used for inference and reduce the number of facts significantly. An example for transitivity is the property `isLocatedIn` of the regional ontology.

```
<owl:TransitiveProperty
        rdf:ID="isLocatedIn">
   <rdfs:domain   rdf:
    resource="#region"/>
    <rdfs:range   rdf:
    resource="#region"/>
</owl:TransitiveProperty>
```

For example, from the two facts—Hannover is located in Lower Saxony and Lower Saxony is located in Germany—it can be concluded that Hannover is located in Germany. In a similar way, a hierarchy of subtopics is modeled in the subject area ontology.

In contrast to these ontologies, which model public accessible information, the user ontology serves as the knowledge model of a particular user (e.g., student or faculty member) and, consequently, contains confidential information.

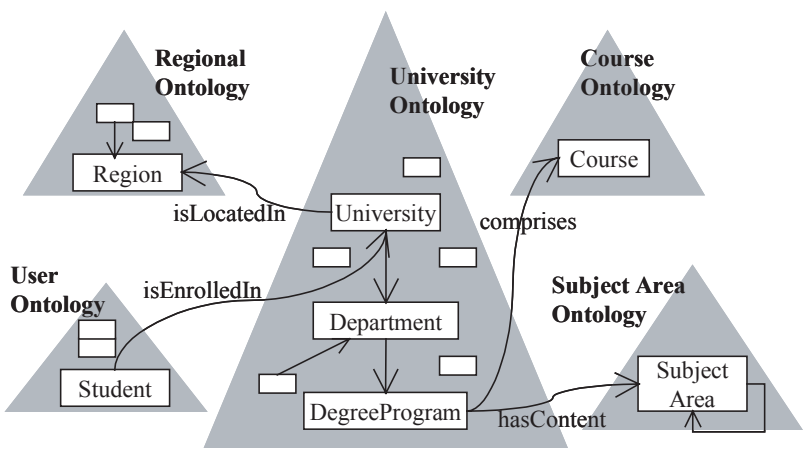
- **User Ontology** The major classes of this ontology are `Student` and `Faculty`. Relevant information of a student is login name, student ID, current semester, passed/failed courses, and so forth. Every student owns his or her instance file of this ontology, reflecting his or her individual progress of study. This information allows the adviser to give personalized advice, considering the individual situation of a student.

Note that the different ontologies are not isolated but are related to each other. So a student instance of the user ontology is related to a course instance of the university ontology via the property `isEnrolledIn`. Figure 1 shows the entire structure of the ontologies with the interrelating properties and some of their classes.

In a Semantic Web infrastructure, the knowledge is spread over the Internet in the form of different OWL files. We can distinguish two types: OWL schemas and OWL instances.

- **OWL schema files** introduce the essential domain concepts and define how they are structured and interrelated. This knowledge is rather data-centric and application-independent. Usually, OWL schemas do not change often and are very stable. Because

Figure 1. Sketch of ontology structure



OWL schema files do not contain any instance data, they are normally publicly accessible.

- **OWL instance files** contain the domain object data. The OWL instances apply to the concepts defined in the OWL schemas and also refer to other objects specified in OWL instances. All the references are determined in the form of URIs, as shown in the examples.

In our advisory system there are five OWL schema files, each containing just one of the described ontologies. To prevent inconsistencies and to ease maintenance, OWL schema files should exist just one per ontology on a central Web server.

However, the OWL instance files are created and maintained locally. It is crucial, that the OWL instances conform to the language specification defined in the OWL schemas and also refer to other instances.

Ontology Development

The previous section described the knowledge base (i.e., the ontologies and their corresponding facts) from a logical point of view. In order to make the knowledge usable for the advisory agents, either internally or via the Internet, they must be defined in a formal ontology language suitable for reasoning. For this purpose, we applied the W3C standard ontology language OWL (Web Ontology Language) (W3C-OWL, 2004) based on XML and RDF/RDF Schema (W3C-RDF, 2004). The expressiveness of OWL-DL was sufficient to model our domain knowledge. Only a few shortcomings of OWL came up, which we resolved within the inference engine, as described in the next section.

In order to develop complex ontologies, an adequate tool support is indispensable. OWL, as any XML code, is intended for the usage of software programs and cumbersome for humans, as the short OWL example in the previous section illustrates. In our project, we used the well-known Protégé Version 2.1 with the OWL Plugin (Protégé, 2004) for ontology development. Except some smaller technical problems, we had good experiences with this tool. It allowed us to specify ontologies with a graphical user interface and to generate the corresponding OWL files, avoiding a potentially error-prone manual OWL coding. Furthermore, facts in the form of OWL instances were created based on these ontologies.

INFERENCE

The semantic advisory agents should act similarly to human advisers according to their knowledge modeled in the ontologies. This is achieved by using the rule-based inference engine JESS (Java Expert System Shell) (Friedman-Hill, 2004) in order to carry out the automated inferences entailed by the semantics of OWL. JESS provides a convenient way to integrate reasoning capabilities into Java programs.

OWL Transformation

JESS initially was developed as a Java version of CLIPS (C Language Integrated Productions System). With JESS, complex rules and queries can be specified.

In order to make use of the knowledge modeled in an ontology, the OWL semantics must be mapped into facts and rules of an inference engine. Because JESS does not provide an interface to import an OWL ontology, we employed an appropriate tool named OWL Engine to load OWL

ontologies and OWL instances into a JESS knowledge base (OWL Inference Engine, 2004), which provides an XSLT-based transformation process.

The OWL inference engine consists of three parts. One file contains JESS rules describing the OWL meta model (i.e., the OWL built-in rules). Two XSLT stylesheets transform files with OWL schemata or with OWL instances into JESS assertions.

A major advantage of the XSLT stylesheets approach is that the stylesheets can be adjusted easily to individual requirements. In our project, we extended the transformation rules for the `owl:transitiveProperty` and the `owl:UnionOf` OWL constructs.

Ontology Reasoning

Mainly, the advisory agents reason on the basis of the OWL knowledge model loaded into the JESS knowledge base. In our case study, the semantic expressiveness of OWL is nearly sufficient. But to express more sophisticated expert knowledge (e.g., complex examination regulations), domain-specific rules must be developed. Inference engines such as JESS provide their own languages to specify complex rules for developing rule-based systems. A simple example of a domain-specific rule out of the scope of OWL is a JESS rule that categorizes cities according to their size.

The data modeled in OWL is usually domain-specific but independent of a certain application. The OWL properties define rules that represent the general structure of the knowledge. They are mainly data-oriented and, therefore, usage-independent and applicable to different applications. The rules additionally specified in an inference engine are process-oriented; they specify the reasoning capabilities of an advisory system and are tailored to a specific use case.

AGENT ARCHITECTURE

To develop the software architecture of an advisory system, it is helpful to analyze real-life consultancy situations, since the main components of the software architecture must implement the capabilities and responsibilities of the human participants.

Advisory situations are characterized by two participants: a client and an advisor, who exchange information and proposals in order to solve a client's problem. Both participants need different kinds of information; some information is private (e.g., the study situation of a student looking for an exchange semester), while other information is publicly available on the Internet (e.g., the course descriptions of an exchange university). Depending on the actual status of the consultancy situation, clients and advisors use their private knowledge but also collect knowledge from publicly available sources. Clients as well as advisors reason on their knowledge to make proposals or decisions. The participants negotiate to reach an agreement, which depends on their intents and how the consultancy situation develops.

The multi-agent software paradigm offers a direct way to implement negotiation processes between the consultancy participants and to express their personal intents (Jennings, Parsons, Sierra, & Faratin, 2000). Figure 2 shows the main components of an agent-based advisory system, corresponding directly to the described consultancy situation. Multi-agent systems provide an architectural pattern that fits well to the described situation (Wooldridge, 2002). The advisory system can be viewed in

terms of autonomous agents of two different types: student agents and international coordinator agents. The two agent types are conceptually identical. These agents interact to find an exchange university and a suitable study plan. Multi-agent technology provides the right level of abstraction to model a negotiation process between independent partners (Jennings et al., 2000; Kraus, 1997) and, consequently, is well-suited for our purposes.

The software architecture of an advisory system should reflect the situation of a real counseling interview. In our use case, a student intends to study abroad for one semester and consults the international coordinator of the department to get advice. Together they first look for an appropriate exchange university and then for a study plan that fits best with the course program at the home institute.

All students are characterized by their personal situations and intents; the international coordinators give their advice based on their knowledge of the study regulations and the various exchange programs.

Each agent type uses an inference engine to reason on its knowledge base that consists of private and public information. An agent can dynamically expand its knowledge base by collecting further knowledge from various Internet sources. Both agents exchange information in a problem-specific negotiation process in order to resolve a specific problem. Each agent acts according to his or her particular behavior, which reflects his or her intents and desires.

Figure 2. Structure of the agent-based advisory system with two types of agents

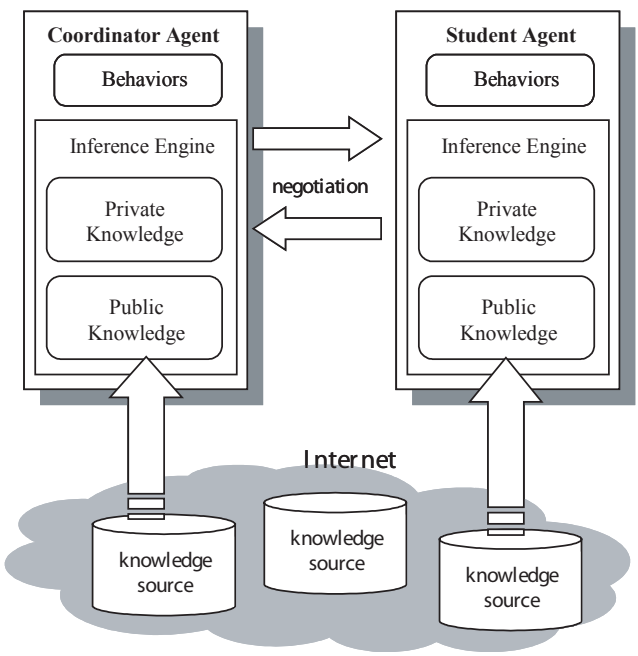
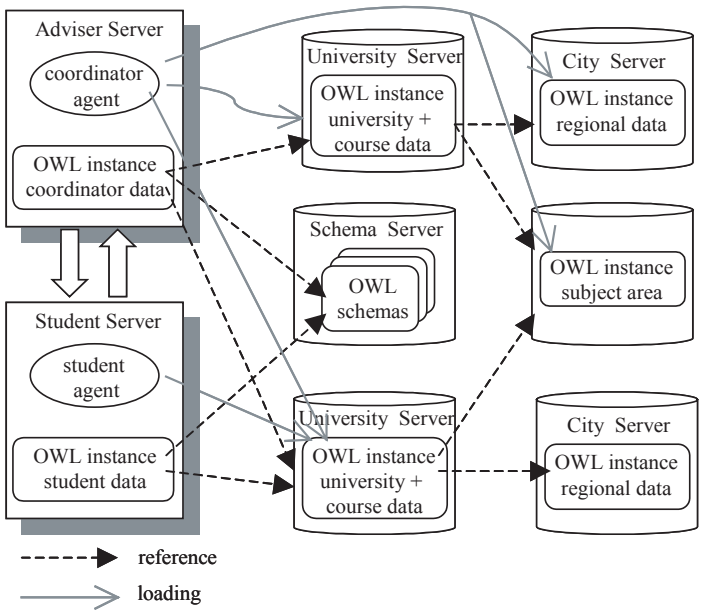


Figure 3. Distributed knowledge sources in the Semantic Web



Semantic Web Architecture

In a Semantic Web architecture, all information is spread over the Internet (Horrocks & Hendler, 2002) in the form of OWL schemas and instance files.

It is crucial that we distinguish different protection levels for instance data.

- Private data has to be protected against unauthorized access. For example, a student agent protects the confidential information of its human owner. As in a real consultancy, an agent only reveals sensitive private data, if it is indispensable for finding a solution.
- On the other hand, there is some publicly accessible knowledge spread over various Web sites. In our example application, each university participating in the Socrates/Erasmus exchange program should build its own instances of the ontologies and make them available on its Web site.

Of course, further protection levels are possible. Typically, there is information not indented for the public but for trusted partners. In this case, agents are acting as trustworthy partners exchanging confidential data.

Figure 3 shows a possible situation for our advisory system. There are five OWL schema files on a central OWL schema server. The OWL instances describing the department structure and the offered courses are located at each university's Web site, where they also are maintained.

The client and adviser agents reside on different servers, where their private knowledge is stored in corresponding OWL instance files. The student agent owns an OWL file describing the student's personal

study situation (e.g., the passed exams), and the adviser agent has some private knowledge about all exchange agreements or the utilization of the departmental courses.

Human participants in a counseling interview use their personal knowledge but also acquire publicly available knowledge, depending on the actual consultancy situation. The client and adviser agents should act in a similar way. As in reality, the knowledge of the agents is not static but increases during the counseling interview. Depending on the state of the interview, agents dynamically acquire useful knowledge from different sources on the Semantic Web and integrate it into their personal knowledge base.

In order to build up its knowledge base, each agent has to process the following steps:

1. According to the status of the interview, the agent determines the required information for the actual counseling context.
2. If the information is publicly available, the agent locates the corresponding OWL files on the Internet.
3. It downloads the OWL instances, transforms them, and imports them into JESS using the OWL Engine.

The international coordinator agent requires information about all exchange universities and their course contents. In order to gain this knowledge, it can dynamically expand its knowledge base by accessing the locally stored OWL files of the universities registered in the advisory system. Beyond the information that the coordinator agent collects from the Internet, it can hold some private knowledge. For example, it may know about all exchange agreements of its university or the utilization of the courses in its department.

The student agent is characterized by its individual study situation, which can be described by the study year, the attended lectures, and the exams passed. Of course, this information is confidential, and therefore, it is represented in a personal OWL instance file, which is protected against unauthorized access.

Furthermore, each agent can have more sophisticated reasoning capabilities expressed by some further JESS rules, as explained earlier.

Agent Interaction and Negotiation

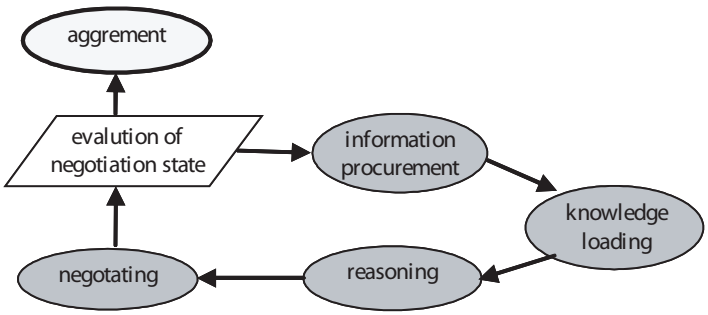
In a real counseling situation, a problem is resolved by a communication and negotiation process, which is characterized by information exchange between the dialog partners. In a multi-agent system, the communication between the agents reflects this negotiation process between clients and advisers. The agent behaviors implement the negotiation protocol, determining the rules that govern the interaction (Jennings et al., 2000; Ossowski & Omicini, 2002).

The negotiation protocol is specific to the application domain and the problem type. However, in general, the agents behave during the negotiation process according to the stages shown in Figure 4.

Each negotiation state is analyzed and evaluated by the agents. The negotiation process stops if a final agreement between all agents (see step 5) can be achieved. Otherwise, the agents iterate through the following steps:

1. **Information Procurement.** The agent determines the information required for the next negotiation step.

Figure 4. Agent behavior



In principal an agent has to know which kind of information it needs but not necessarily the physical location of the information. It can be advantageous to delegate the administration or brokering of the information sources to a separate so-called middle agent, as described in the next section.

2. **Knowledge Loading.** The agent accesses the knowledge stored in OWL schema and instance files and dynamically loads it into the knowledge base of its inference engine. If necessary, the agent uses a tool to transform the OWL descriptions into facts and rules of the inference engine (e.g., with the tool OWL Engine).
3. **Reasoning.** Using the inference engine, the agents perform reasoning based on their knowledge to make decisions or proposals. Especially the adviser agents usually own application-specific rules to provide more sophisticated reasoning. The expressiveness of the languages incorporated in the inference engines is normally much more powerful than of OWL.
4. **Negotiation.** The information exchanged among agents is application-specific and determined by the negotiation state and their interpretation of the situation (Jennings et al., 2001). After passing general information, more details are necessary when a solution is narrowing. Each agent has his or her own desires and, eventually, conflicting interests. In our example, the student agent is interested in certain subjects or prefers studying at a specific location. Also, the advisors may have some intentions (e.g., they could be interested in sending students to particular universities or accepting only students with suitable skills and precognitions).
5. **Agreement.** Depending on the issues over which an agreement has to be reached, all participating agents must accept a certain negotiation state. An agreement can be achieved by the agents or delegated to their human owners.

In our example, both agents must agree on a certain exchange university and a corresponding study program,

which constitutes the Socrates Learning Agreement. In the sample implementation, the adviser agent does not have any intentions (i.e., leaves all decisions about the exchange program to the client agent, who again delegates them to the human user).

There are a lot of approaches for negotiation protocols and decision-making models, which usually are highly dependent on the application domain. A good overview can be found in Kraus (1997).

Agent Behavior in the Case Study

The following steps illustrate in more detail how the agents interact in our chosen exemplary e-learning use case:

1. A student starts his or her personal student agent (SA) to search for a suitable exchange semester and logs in.
2. The SA loads the OWL user ontology and the OWL instance data representing the student's specific study situation into its JESS knowledge base.
3. The student can enter some preferences regarding the exchange university (e.g., the subject of study, the teaching language, desired location). The SA extracts the specified parameters and queries further personal data (e.g., the aimed degree) from the knowledge base. Then the SA sends a request to the international coordinator agent (ICA).
4. The ICA collects instance data about all universities registered in the system as well as about the study places and loads them in its knowledge base. During its initialization, the ICA already has loaded all ontology schema files.

Figure 5. Study plan proposal of the coordinator agent

5. Then the ICA reasons on the knowledge base, aggregates the results, and sends a ranked list of appropriate foreign degree programs to the SA.
6. The SA receives the result and presents it to the student, who chooses his or her favorite exchange university and degree program. The student's decision and further user instance data are sent to the ICA (e.g., the study program based on the passed exams).
7. The ICA accesses the OWL course instance data of the selected foreign degree program via the Internet and loads it into its knowledge base. Usually, the course's information is maintained separately in each exchange university. On the basis of the expanded knowledge base, the ICA suggests the foreign courses that best fit the study program of the home university (see Figure 5).
8. The SA receives the results from the ICA, and the student manually chooses the desired course plan from the suggested options. Finally, the SA

generates a formal document, called Socrates Learning Agreement, determining the personalized exchange study plan.

The student agent protects the confidential information of its human owner. Similar to a real consultancy situation, it only reveals sensitive private data if they are necessary in order to find a solution. The knowledge of the student agent is rather restricted; it mainly knows the personal situation of its owner. The international coordinator agent has a much broader knowledge, which it expands dynamically in the Semantic Web.

Of course, the behavior of both agents could implement personal desires and intentions. For example, the coordinator agent could present only a selection of possible exchange universities, depending on the exchange agreements or the number of applicants.

Middle Agents

During the negotiation process, the agents have to know where the publicly accessible OWL schema and instance information is located. Of course, the contents and their location can change dynamically. In our university example, new exchange universities or new study programs should be integrated continuously into the informational base of the system, especially new network architectures, as peer-to-peer systems foster spontaneous ad hoc networks where nodes continually join and leave the network. In order to cope with this situation, an adequate mechanism for the notification of new information sources on the Internet is needed (Carzaniga, Rosenblum, & Wolf, 2000). In agent systems, so-called middle agents assist in locating and connecting information providers with information or service requesters. Middle agents can be implemented according to different designs, as discussed in-depth in Wong & Sycara (2000).

In advisory systems, each adviser agent can register with the middle agent, which provides a decentralized service for information procurement. Furthermore, middle agents can act as mediators in order to set up consultancies between client and adviser agents.

Implementation Issues

Powerful agent development frameworks facilitate the development of multi-agent systems. The semantic advisory agents are developed with JADE (Java Agent Development Framework) (Bellifemine, Giovanni, Trucco, & Rimassa, 2002), which complies with the FIPA (Foundation of Intelligent Physical Agents) standards (FIPA, 2003). JADE includes two main components: a FIPA-compliant agent platform and a framework to develop Java agents. The core part of the FIPA architecture is a standard for agent communication (i.e., its ACL, Agent Communication Language). The interaction between the student and the international coordinator agent is based on the exchange of ACL messages.

To avoid a user having to install the student agent on his or her computer, we chose a Web architecture; the user agent resides on a central server and has a Web interface implemented with JavaServer Pages.

CONCLUSION

In this article, we described how Semantic Web and agent technology can be integrated in order to build an intelligent

advisory system for an e-learning environment. Our goal is to create and deploy semantic advisory agents capable of supporting university students in successfully organizing and performing their studies.

The first issue of our architecture is to model the knowledge used in advisory systems by means of the Semantic Web. Ontology languages like OWL can express the domain concepts, its structure, and interrelations. In a Semantic Web architecture, all information is spread over the Internet in the form of OWL instance and schema files. As in reality, the knowledge of the agents is not static but expanding during the counseling interview. Depending on the state of the interview, agents collect their knowledge dynamically from various sources on the Semantic Web and integrate it into their personal knowledge bases of the inference engine. Additionally, each agent possesses private data, which have to be protected against unauthorized access. As in a real consultancy, an agent only reveals sensitive private data if they are crucial for finding a solution.

Furthermore, we showed the principal stages of the negotiation process between the agents. There are many approaches for negotiation protocols and decision-making models, which usually are highly dependent on the application domain. In our future work, we will improve the negotiation process between the agents, especially their decision-making process.

The major difficulty encountered was the integration of the concepts: on the one hand, the knowledge bases written in RDF and OWL, and on the other hand, the inference engine JESS and the agent environment JADE. We implemented a prototype system in which the agents were able to reason upon the knowledge base in the desired manner. Our experiences show that the employed technologies are mature and well-suited for the implementation of advisory systems.

In our future work, we will implement more use cases for the Semantic E-learning Agent project. For example, advisers should be able to announce new opportunities for students who are looking for suitable thesis subjects and to answer questions regarding the regulations of study.

REFERENCES

Bellifemine, F, Giovanni, C., Trucco, T., & Rimassa, G. (2002). *JADE programmer's guide*. Retrieved August 25, 2004, from http://jade.tilab.com/doc/programmersguide.pdf

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American, 284*(5), 34-43.

Carzaniga, A., Rosenblum, D.S., & Wolf, A.L. (2000). Achieving scalability and expressiveness in an Internet-scale notification service. In *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing* (pp. 219-227).

CCTT - Center for Curriculum, Transfer and Technology. (2004). Retrieved August 25, 2004, from http://www.edutools.info/course/compare/all.jsp

Dunkel, J., Bruns, R., & Ossowski, S. (2004). Semantic e-learning agents. In *Proceedings of the Sixth International Conference on Enterprise Information Systems,* Porto, Portugal (pp. 271-278).

FIPA - Foundation of Intelligent Physical Agents. (2003). Retrieved August 25, 2004, from http://www.fipa.org

Friedman-Hill, E. (2000). *JESS, the rule en-*

gine for the Java platform. Retrieved August 25, 2004, from http://herzberg.ca.sandia.gov/jess/

Hamdi, M.S. (2006). MASACAD: A multi-agent system for academic advising. *International Journal of Intelligent Information Technologies*, *2*(1), 1-20.

Horrocks, I., & Hendler, J. (Eds.). (2002). The Semantic Web. In *Proceedings of the First International Semantic Web Conference.*

Jennings, N. et al. (2001). Automated negotiation: Prospects, methods and challenges. *International Journal of Group Decision and Negotiation, 10*(2), 199–215.

Jennings, N.R., Parsons, S., Sierra, C., & Faratin, P. (2000). Automated negotiation. In *Proceedings of the 5th International Conference on Practical Application and Intelligent Agents and Multi-Agent Systems,* Manchester, UK,(pp. 23-30).

Kraus, S. (1997). Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, *94*(1-2), 79-98.

Noy, N.F., & McGuiness, D.L. (2001). *Ontology development 101, a guide to creating your first ontology* (Tech. Rep. No. KSL-01-05). Stanford, CA: Stanford University, Stanford Knowledge Systems Laboratory.

Ossowski, S., & Omicini, A. (2002). Coordination knowledge engineering. *The Knowledge Engineering Review, 17*(4), 309-316.

OWL Inference Engine. (2004). Retrieved from http://mycampus.sadehlab.cs.cmu.edu/public_pages/OWLEngine.html

The Protégé Project. (2004). Retrieved August 25, 2004, from http://protege.stanford.edu/

Wong, H.C., & Sycara, K. (2000). A taxonomy of middle-agents for the Internet. In *Proceedings of 4th International Conference on Multi Agent Systems (ICMAS-2000)*, Boston, (pp. 465-466).

Wooldridge, M. (2002). *An introduction to multiagent systems*. Chichester, UK: John Wiley & Sons.

Wooldridge, M., & Jennings, N. (1995). Intelligent agents—Theory and practice. *Knowledge Engineering Review, 10*(2), 115–152.

W3C-OWL. (2004). *OWL Web ontology language reference—W3C recommendation 10 February 2004.* Retrieved August 25, 2004, from http://www.w3.org/TR/owl-ref/

W3C-RDF. (2004). *RDF primer—W3C recommendation 10 February 2004*. Retrieved August 25, 2004, from http://www.w3.org/TR/rdf-primer/

Ralf Bruns is a professor of computer science at the Fachhochschule Hannover (University of Applied Sciences and Arts). His research interests include software architecture, distributed systems, and internet technologies. He received a Diploma and a PhD in computer science from University of Oldenburg (Germany). He is a member of the German Computer Science Society.

Jürgen Dunkel is a professor of computer science at the Fachhochschule Hannover (University of Applied Sciences and Arts). He received a Diploma degree in computer science from University of Hagen and a PhD from University of Dortmund (Germany). His research interests include software architecture, distributed systems, and agent technologies. He is a member of the German Computer Science Society.

Sascha Ossowski is a professor in computer science at University Rey Juan Carlos in Madrid where he leads the Artificial Intelligence Laboratory. He obtained his MSc degree in Informatics from the University of Oldenburg (Germany) in 1993, and received a PhD in Artificial Intelligence from Technical University of Madrid in 1997. Dr. Ossowski holds several research grants focussing on the development of advanced software systems based on artificial intelligence and coordination technology. He is particularly interested in models and mechanisms for emergent coordination in open multiagent systems.