# SEMANTIC E-LEARNING AGENTS
## *Supporting E-Learning By Semantic Web and Agents Technologies*

Jürgen Dunkel, Ralf Bruns
*Department of Computer Science, University of Applied Sciences and Arts Hannover,*
*Ricklinger Stadtweg 120, D-30459 Hannover, Germany*

Sascha Ossowski
*AI Group, E.S.C.E.T.,Universidad Rey Juan Carlos Madrid,*
*Campus de Mostoles, Calle Tulipan s/n, E-28933 Madrid, Spain*

Keywords: Semantic Web, Ontology, Software Agents, E-learning

Abstract: E-learning is starting to play a major role in the learning and teaching activities at institutions of higher education worldwide. The students perform significant parts of their study activities decentralized and access the necessary information sources via the Internet. Several tools have been developed providing basic infrastructures that enable individual and collaborative work in a location-independent and time-independent fashion. Still, systems that adequately provide personalized and permanent support for using these tools are still to come.

This paper reports on the advances of the Semantic E-learning Agent (SEA) project, whose objective is to develop virtual student advisors, that render support to university students in order to successfully organize und perform their studies. The E-learning agents are developed with novel concepts of the Semantic Web and agents technology. The key concept is the semantic modeling of the E-learning domain by means of XML-based applied ontology languages such as DAML+OIL and OWL. Software agents apply ontological and domain knowledge in order to assist human users in their decision making processes. For this task, the inference engine JESS is applied in conjunction with the agent framework JADE.

## 1 INTRODUCTION

E-learning has established itself as a significant part of learning and teaching at institutions of higher education worldwide. The students perform significant parts of their study activities decentralized and access the necessary information sources via the Internet. The emerged individual means of work are location-independent and time-independent, consequently requiring a permanent available and direct support that can only be provided by a software system.

The main focus of current E-learning systems is to provide an appropriate technical infrastructure for the information exchange between all user groups involved in the E-learning process. A recent comparison of modern E-learning environments (CCTT, 2002) revealed, that intelligent advisory agents are not applied so far in E-learning systems. However, the necessity of an intelligent support is unquestioned due to the individual and decentralized means of study (Cuena et al., 1999, Ossowski et al. 2002).

The objective of the semantic E-learning agent project is to develop virtual student advisors, that render support to university students, assisting them to successfully organize und perform their studies. These advisors are to behave both reactive and proactive: setting out from a knowledge base consisting of E-learning and user ontologies, their recommendations must be tailored to the personal needs of a particular student. For example, they should be able to answer questions regarding the regulations of study (e.g.: does a student possess all requirements to participate in an examination or a course?, is a student allowed to register for his/her



Dunkel J., Bruns R. and Ossowski S. (2004).
SEMANTIC E-LEARNING AGENTS - Supporting E-Learning By Semantic Web and Agents Technologies.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 271-278
DOI: 10.5220/0002613602710278

thesis?, etc). In addition, advisors should be capable of announcing new opportunities for students that are looking for suitable practical training jobs or thesis subjects.

To achieve these goals, we propose a software architecture (Dunkel et al., 2003) where virtual student advisors are developed with novel concepts from Semantic Web (Berners-Lee et al., 2001) and Intelligent Agent (Wooldbridge et al. 1995) technology. The basic idea is to model the structure of our E-learning domain by means of ontologies, and to represent it by means of XML-based applied ontology languages such as DAML+OIL and OWL. Due to the standardization of these technologies, knowledge models can easily be shared und reused via the Internet. Software agents apply the knowledge represented in the ontologies during their intelligent decision making process. Again, the use of widespread inference engines, such as JESS (Friedman-Hill, 2000a), and of agent frameworks that comply with the FIPA standard (FIPA, 2003) as JADE (Bellifemine et al., 2002), which facilitates maintenance and fosters interoperability with our system. We claim that this is quite a promising approach because – although first concrete practical application scenarios with Semantic Web technologies have been published, e.g. (Horrocks et al. 2002) – E-learning systems that successfully combine these techniques in order to render support to users are still to come.

This paper reports on the lessons learnt from the construction of a real-world application in the E-learning domain that draws upon an effective integration of both, Semantic Web and Intelligent Agent technology. It is organized as follows: In the next section the employed knowledge representation techniques and the developed knowledge models are presented in detail. The third section shows how automated inference can be carried out on base of the knowledge models, and how agents can provide reasoning capabilities using this ontology. In the following section the software architecture of the agent system is outlined. Finally, the last section summarizes the most significant features of the project and provides a brief outlook to the direction of future research.

# 2 ONTOLOGIES

The key concept of a semantic advisory system for university students is the semantic modeling of the E-learning domain knowledge (e.g. university regulations, course descriptions, admission regulations) as well as an individual user model, which reflects the current situation of study (e.g. passed exams, current courses). In these models the fundamental structures of the available domain knowledge as well as the basic facts (e.g. offered courses) are defined.

In our system, the structural part of this E-learning knowledge is modeled by means of ontologies which formally define domain entities and the relations among them. For this purpose, we use Semantic Web technology based on XML and RDF/ RDF Schema (WWW-RDF, 2003), respectively. Software agents use this information as the basis for their reasoning and, due to the standardization of these technologies, they are able to access distributed information sources from different universities. Thus the developed ontologies can serve as standardized and open interfaces for the interoperability of E-learning systems.

The ontology language DAML+OIL is an attempt to address the shortcomings of the RDF/ RDF Schema specification by incorporating additional features (DAML, 2003). DAML+OIL includes support for classification, property restriction and facilities for type definitions. In the last years, ontology languages have converged to the new W3C standard OWL (Web Ontology Language) (WWW-OWL, 2003), which is currently under development. In a first step, we have chosen the DAML+OIL language to model the E-learning knowledge. The main reason was the availability of different tools for the development of the knowledge base. As soon as possible, the knowledge base will be migrated to the new W3C standard language OWL.

Two different ontologies have been developed for our E-learning agents: on the one hand, an ontology describing the organization structure of a university department, on the other hand, an ontology holding the knowledge about a specific user of the system.

## 2.1 Department Ontology

The department ontology models the essential parts of the organizational structure of a university. The emphasis lies on the individual departments, the different roles of persons in a department and the courses.

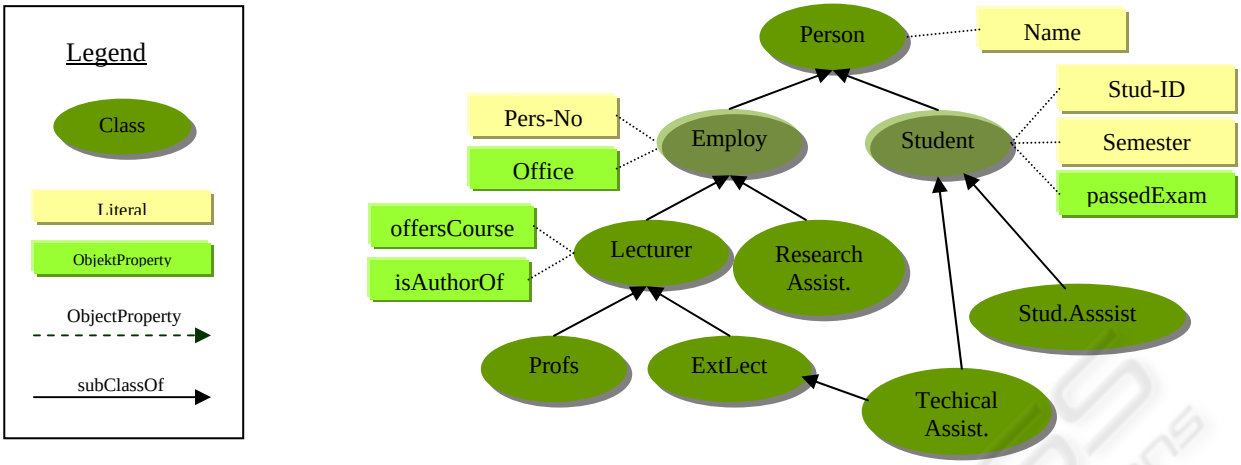


Figure 1: Department Ontology – person branch

It is modeled as follows. Every organizational unit is defined as a subclass of organization. For this super class a transitive property is defined, thus a hierarchy of instances can easily be modeled. In DAML this transitivity is modeled as follows:

```
<daml:TransitiveProperty
                    rdf:ID="subOrg">
   <rdfs:label>subOrg of</rdfs:label>
   <rdfs:domain
        rdf:resource="#Organization"/>
   <rdfs:range
        rdf:resource="#Organization"/>
</daml:TransitiveProperty>
```

The transitivity is used in the instance files to model a concrete hierarchy. For example, a student project is a sub-organization of a department and the computer science department is a sub-organization of the university FH Hannover.

```
<fbi:department rdf:ID="CS">
   <fbi:subOrg>
      <fbi:FH rdf:about="#FHHannover"/>
   </fbi:subOrg>
</fbi:department>
<fbi:project rdf:ID="Project1">
   <fbi:subOrg>
     <fbi:department
              rdf:about="#CS"/>
  </fbi:subOrg>
</fbi:project>
```

All further parts of the ontology belong to an organization. This is modeled by the property `<daml:ObjectProperty rdf:ID=”isPartOf”/>`, which is restricted to a concrete subclass of organization.

The part of the ontology that models a person is shown in figure 1. The semantic of inheritance in this taxonomy is slightly different compared to object-oriented programming. In object-oriented programming it expresses a specialization of an “is-a”-relation, while in the context of ontologies, it serves mainly as a categorization of knowledge.

For the sake of clarity, the graphical representation does not show all information of the relations. In particular, is not shown which class or property of another branch of the ontology is referred to. One example is the property `offersCourse` of the class `Lecturer`. In the XML notation it is defined as follows:

```
<daml:ObjectProperty
                rdf:ID="offersCourse">
  <rdfs:label> offers course
  </rdfs:label>
  <rdfs:domain
          rdf:resource="#Lecturer"/>
  <rdfs:range
    rdf:resource="#Course"/>
  <daml:minCardinality>1
  </daml:minCardinality>
</daml:ObjectProperty>
```

A lecturer teaches one or more courses and it is possible to navigate from a lecturer to a specific

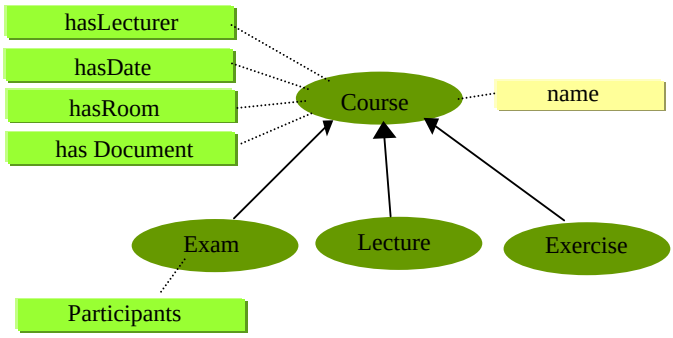


Figure 2: Department Ontology – course branch

course. In the course branch of the ontology one can find a property `hasLecturer` with a similar semantics with inverse direction of navigation. This can be defined as an inverse property in DAML.

```
<daml:ObjectProperty
                  rdf:ID="hasLecturer">
  <daml:label>is offered by
  </daml:label>
  <daml:inverseOf
rdf:resource="#offersCourse"/>
</daml:ObjectProperty>
```

Figure 2 displays the course branch of the E-learning ontology. Not visualized by the graphical notation are further characteristics of subclasses. For example a course is a disjunctive union of its subclasses. In DAML this is modeled as follows.

```
<daml:Class
        rdf:about="#Course">
 <daml:disjointUnionOf rdf:parseType=
  "http://www.daml.org/2001/
             03/daml+oil#collection">
   <daml:Class rdf:about="#Lecture"/>
   <daml:Class rdf:about="#Exercise"/>
   <daml:Class rdf:about="#Exam"/>
 </daml:disjointUnionOf>
</daml:Class>
```

This construct ensures that a course is either a lecture, an exercise or an examination.

## 2.2 User Ontology

The user ontology serves as the knowledge model of a specific user, e.g. a student or a faculty member. The core class of the ontology is `User`. A user is a person with respect to the department ontology. This is modeled by the object property `sameClassAs`, which is the DAML element to model inter-ontological equivalence.

```
<daml:Class rdf:about="#User">
   <daml:sameClassAs rdf:resource=
      "http://localhost:8080/Agents/
                FB_Onto.daml#Person"/>
</daml:Class>
```

The additional properties model all relevant data of a person, e.g. login name, student ID, current semester, passed/failed courses, last login date, skills etc.

# 3 AGENTS AND INFERENCE

The semantic E-learning agents should act like a human advisor according to the knowledge modeled in the ontology. This is achieved by using a rule-based inference engine to carry out the automated inferences entailed by the semantics of DAML.

## 3.1 Inference

To provide the semantic E-learning agents with reasoning capabilities, the rule-based Expert System Shell JESS (Java Expert System Shell) (Friedmann-Hill, 2000] is employed. JESS was initially developed as a Java version of CLIPS (C Language Integrated Productions System) and provides a convenient way to integrate reasoning capabilities into Java programs. With the JESS language complex rules, facts and queries can be specified.

## 3.2 Ontology Reasoning

To make use of the knowledge modeled in the ontology, the DAML semantics must be mapped into facts and rules of a production system, like JESS.

Because JESS does not provide any interface to import a DAML ontology in its knowledge base, we choose DAMLJessKB (Kopena et al., 2003), a reasoning tool for DAML that uses JESS as inference engine. In some more detail DAMLJessKB processes the following steps.

First DAMLJessKB loads and parses RDF documents using the RDF-Parser ARP of the Jena toolkit (Hewlett Packard Labs, 2003). ARP generates RDF triples. DAMLJessKB reorders the triples from subject-predicate-object form into predicate-subject-object form. Each RDF triple represents an unordered fact in JESS.

To assert triples in JESS minor transformations are necessary. DAMLJessKB translates URIs (Uniform Resource Identifiers) into JESS symbols by removing invalid characters (e.g. ~), and inserts the dummy predicate `PropertyValue` in front of each triple. The following example shows a generated fact for JESS, which means, that `Professor` is a subclass of `Lecturer`.

```
(PropertyValue
  http://www.w3.org/2000/01/rdf-
                    schema#subClassOf
     file:///C:/FB_Onto.daml#Professor
     file:///C:/FB_Onto.daml#Lecturer )
```

Because in our DAML ontology `Lecturer` is defined as a subclass of `Person`, it follows that `Professor` is also a subclass of `Person`.

To support reasoning, DAMLJessKB includes some built-in rules of the DAML semantics, which are asserted into JESS, e.g. that an instance of a subclass is also an instance of the super class:

```
(defrule subclassInstances
  (PropertyValue daml:subClassOf
                 ?child ?parent)
  (PropertyValue rdf:type
                 ?instance ?child)
   =>
  (assert
  (PropertyValue rdf:type
             ?instance ?parent)
  )
)
```

The semantics of a JESS rule is similar to an if-then-statement in a programming language. Whenever the `if` part (the left-hand-side) which consists of several patterns is satisfied, the rule is executed, i.e. in our example a new fact is asserted into JESS. Details about the JESS language can be found in (Friedman-Hill, 2000).

Beside the DAML rules, which are directly supplied by DAMLJessKB, it is necessary to develop own domain-specific rules to model the complete expert knowledge. These rules make it possible to cope with complex queries related to a domain.

First, all facts are produced; then, the DAML rules are added; and finally the domain-specific rules are asserted into JESS. The reasoning process is performed by JESS applying all rules to deduce new facts which are successively added to the knowledge base.

DAMLJessKB can be considered as an interface to JESS, which is capable of translating DAML documents in accordance with their formal semantics. We are aware of several other tools with similar functionality, for example DAML API (DAML API, 2003), or the SWI Prolog distribution (SWI-Prolog 2003), which includes a package to parse RDF and assert as Prolog facts, but none of them fully meet the integration requirements of our E-learning system

## 3.3 Agent access to the knowledge base

In order to cope with their specific tasks, semantic E-learning agents can pose queries to access the JESS knowledge base. These queries are special rules with no right-hand sides. The results of a query are those facts, which satisfy all patterns. For example, if a personal agent for a lecturer `tom` is interested in all courses he has to give, it can use the query:

```
(defquery getCourses
    "find IDs of all my courses"
  (declare (variables ?lecturerID)
    (PropertyValue lecture:givesCourse
        ?lectureID ?course)
   )
```

where `lecturerID` is the identifier of the lecturer, which serves as parameter of the query, and `?course` is an internal variable. All elements in a

query must be fully qualified with their namespace, as they are used in the knowledge base. Executing the query yields all facts that satisfy all patterns specified in the query. E.g. a fact that fits the query could be:

```
(PropertyValue
    file://C:/FB_User.daml#givesCourse
    file://C:/FB_User.daml#tom
    file://C:/FB_Onto.daml#Math1)
```

In this case the lecturer `tom` gives the `Math1` course. The following example shows a more complex query that yields all documents of a course that are more recent than a certain time. It has two parameters: the time `mydate` and the identifier of a course, e.g. `file://C:/FB_Onto.daml#Math1`.

```
(defquery getNewerDocs
  (declare (variables ?mydate ?course))
    (PropertyValue rdf:type
             ?course fb:course)
    (PropertyValue fb:hasDocument
             ?course ?doc)
    (PropertyValue fb:changeDate
             ?doc ?doc_modified)
    (PropertyValue date:longDate
             ?doc_modified? long_date)
    (PropertyValue rdf:value
             ?long_date
    ?doc_date&:(>= ?doc_date ?mydate))
    )
```

The last pattern contains the condition that the last time the document was modified is greater than `mydate`.

# 4 JADE-AGENTS

In the previous sections we have modeled the knowledge of the E-learning system in two different ontologies: the department and the user ontology. The two knowledge bases are related to different domain concepts: to a department advisor and to a specific user. A human advisor and a human user communicate and exchange information to find a solution for an individual problem.

To implement a software system reflecting this situation we chose agent technology. Software agents provide a direct way to implement conversations or negotiations. The FIPA (Foundation of Intelligent Physical Agents) organization (FIPA, 2003) has defined several standards for agent communication, e.g. ACL (Agent Communication Language). Agent technology provides natural means of communication and information exchange, which is on a high abstraction level and independent of certain technologies, e.g. protocols or inter-process communication mechanisms.

The semantic E-learning agents are developed with JADE (Java Agent Development Framework) (Bellifemine et al., 2002), which complies with the FIPA standards. JADE is completely written in Java and includes two main components: a FIPA-compliant agent platform and a framework to develop Java agents.

## 4.1 Agent structure

Figure 3 outlines the structure of the E-Learning system with two different types of agents: a user and a department agent.

**User Agent**
The user agent is implemented in a class `UserAgent` and contains the user ontology with all relevant information about personal data, courses, skills, etc. When the user agent is started, it reads in the user ontology with its personal data using DAMLJessKB. Then the user agent-specific rules and queries are loaded and asserted in JESS.

```
void setUP(){
  damljesskb = new DAMLJessKB();
  damljesskb.loadDAMLResource(
                    userOntology);
  ...
  loadRules(userRules);
  ...
```

Corresponding to each JESS query the agent includes a dedicated method like `getCourses()`, which execute the query via DAMLJessKB, and receives an iterator object containing the query result.

```
String[] getCourses(){
  ...
  Iterator e = damljesskb.query (
      "getCourses", new String[] {""});
```

**Department Agent**
The department agent has all knowledge about the department, e.g. the curriculum and the examination regulations, which are modeled in its own DAML

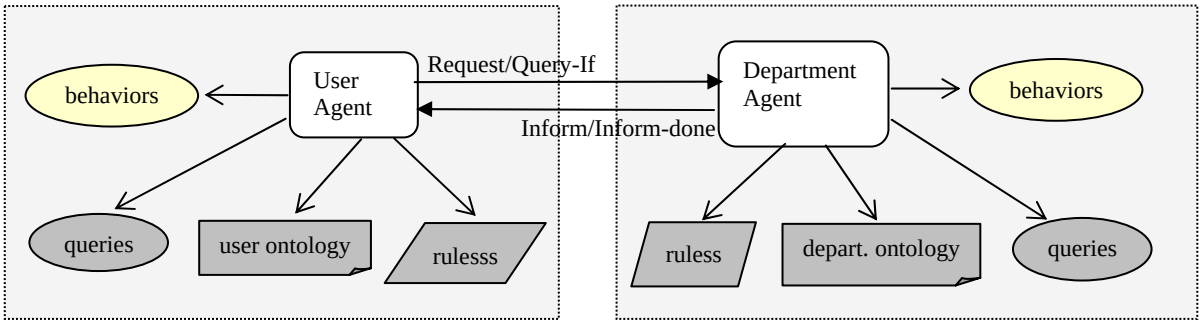


Figure 3: Agent structure

ontology. The corresponding class `Department-Agent` has a similar structure as `UserAgent`: In its `setup()`-Method DAMLJessKB is used to load the department ontology, specific rules and the necessary Jess queries. Each query can be executed in a corresponding agent method. One example is `getNewerDocs()`, which yields all documents related to a course which are newer than a specified date.

## 4.2 Agent Behavior and Communication

An agent must be able to execute several parallel tasks in response to different external events. In JADE all agent tasks are modeled as objects of the `Behavior` subclass, which determine the reactions of an agent: e.g. when it receives a message, and how it reacts on requests from another agent.

The JADE-method `addBehavior()` adds a behavior to the task queue of a specific agent. Behaviors a registered in an agent's `setup()`-method or on response to an user event.

In a round-robin policy a scheduler executes the `action()`-method of each behavior in the task queue. If the `action()`-method is finished, the `done()`-method is invoked. If it returns `true`, the task is removed from the event queue. To model cyclic tasks `done()` returns always `false`. Details about behaviors can be found in (Bellifemine et al., 2002).

For example, the following behaviors are defined in the E-learning system.

The user and the department agent use the `RegisterAtDF` behavior, which registers an agent with its name and type at the agent platform.

The user agent uses the `UA_SearchDepartment-Agent`-behavior to ask the name service of the platform, the Directory Facilitator (DF) for all department agents, and to establish a connection to them.

The `UA_SendRequest`-behavior requests information from the department agent. This behavior object is created by an event on the agent GUI. According to a parameter `content` (e.g. `SEND_DOCUMENTS`) the user agent collects the necessary request parameters, e.g. the courses of a user, and sends them via an ACL message to the department agent. Furthermore a command string (here: DOCUMENTS) is set to specify the request.

```
void action() {
  ...
  if(content== SEND_DOCUMENTS){
     ...
     parameters.setCourses(
              this.getCourses());
     ...

msg.setContentObject(parameters)
;
     msg.setLanguage(DOCUMENTS);
     ...
   }
   msg.addReceiver(agent);
   myAgent.send(msg);
```

The `UA_ReceiveRequests`-behavior waits in an infinite loop for messages from a department agent. If a message arrives it is analyzed and the results are sent to the agent's GUI.

The department agent uses the `DA_SearchUser-Agent`-behavior to get all user agents, and to establish a connection to them.

The `DA_ReceiveRequest`-behavior analyzes arriving messages from user agents. It extracts the command string and the parameters of the message, to execute the specified query. Then the query results are packed into a message and returned to the corresponding user agent.

## 5 CONCLUSION

In this paper, we have described the use of Semantic Web languages and agent technology for building an intelligent advisory system for E-learning environments. Our goal is to create and deploy semantic E-learning agents capable of supporting university students in successfully organizing and performing their studies. In the project we have developed a software architecture, which integrates Semantic Web and Intelligent Agent technologies.

Due to the use of Semantic Web languages the developed knowledge models can easily be used in distributed systems and shared among software agents via the Internet.

The major difficulty encountered was the integration of the different concepts – on the one hand the knowledge base written in RDF and DAML+OIL, on the other hand the inference engine JESS and the agent environment JADE. Further problems emerged from the unsatisfactory tool support for developing the ontology and the concrete instances of the ontology. However, after the mentioned problems were solved we could implement a prototype system, where the agents were able to reason upon the knowledge base in the desired manner. Actually the migration of our system to the upcoming W3C standard language OWL is under work.

## REFERENCES


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

Bellifemine, F, Giovanni, C., Trucco, T., Rimassa, G., 2002, JADE Programmers's Guide, http://sharon.cs-elt.it/projects/jade/, retrieved October, 2003.

Dunkel, J. Holitschke, A., Software Architecture (In German), 2003. Springer Verlag.

Bruns, R., Dunkel, J., von Helden, J., 2003. Secure Smart Card-Based Access To An eLearning Portal. In ICEIS'03, 5th International Conference on Enterprise Information Systems. ICEIS Press.

CCTT - Center for Curriculum, Transfer and Technology, 2002.http://www.edutools.info/course/compare/all.jsp, retrieved October, 2003.

Cuena J., Ossowski S., 1999. Distributed Models for Decision Support. In: Weiß (ed.): Multi-Agent Systems — A Modern Approach to DAI. MIT Press, 459–504.

DAML-The DARPA Agent Markup Language Homepage: http://www.daml.org, retrieved October 10, 2003.

DAML API, 2003. http://codip.grci.com/Tools/Components.html, retrieved October, 2003.

Friedman-Hill, E., 2000a. JESS, The rule engine for the Java platform,. http://herzberg.ca.sandia.gov/jess/ retrieved October, 2003.

Friedman-Hill, E., 2000b, Jess. The Rete Algorithm, Sandia National Laboratories, http://herzberg.ca.sandia.gov/jess/docs/52/rete.html, retrieved October, 2003.

FIPA - Foundation of Intelligent Physical Agents, 2003. www.fipa.org, retrieved October, 2003.

Horrocks, I., Hendler, J. (eds.), 2002. The Semantic Web, First International Semantic Web Conference, Sardinia, Italy, Springer LNCS 2342.

Hewlett Packard Labs: Jena Semantic Web Toolkit, 2003. http://www.hpl.hp.vom/semweb, retrieved October, 2003.

Kopena, J. Regli, W., 2003, DAMLJessKB: A Tool for reasoning with the Semantic Web. IEEE Intelligent Systems, 18(3).

Ossowski, S., Hernández, J., Iglesias, C.A.; Fernández, A., 2002. Engineering Agent Systems for Decision Support. In: Engineering Societies in an Agent World III (Petta, Tolksdorf & Zambonelli, eds.), Springer-Verlag.

Ossowski, S., Omicini, A., 2002. Coordination Knowledge Engineering. Knowledge Engineering Review 17(4), Cambridge University Press.

SWI-Prolog, 2003. http://www.swi-prolog.org, retrieved October, 2003.

WWW – The World Wide Web Consortium, 2003a. RDF Primer – W3C Working Draft 05 September 2003: http://www.w3.org/TR/2002/WD-rdf-primer-20020319/, retrieved October 10, 2003.

WWW – The World Wide Web Consortium, 2003b. OWL (Web Ontology Language): http://www.w3.org/TR/-owl-ref/ , retrieved October 10, 2003.

Wooldridge, M.; Jennings, N., 1995. Intelligent Agents - Theory and Practice. Knowledge Engineering Review 10 (2), pp. 115–152.