☞ ☞

SWAD-Europe Deliverable 6.4: Using Modelling Methods to Build Semantic Web Applications - Modelling Experiments

Project name:
    Semantic Web Advanced Development for Europe (SWAD-Europe)
Project Number:
    IST-2001-34732
Workpackage name:
    WP 6: XML and RDF Integration
Workpackage description:
    ☞http://www.w3.org/2001/sw/Europe/plan/workpackages/live/esw-wp-6.html
Deliverable title:
    D6.4 Using Modelling Methods to Build Semantic Web Applications
URI:
Authors:
    ☞Alvaro Arenas, CCLRC.
    ☞Brian Matthews, CCLRC.
Abstract:
    This paper contributes to the area of software engineering for Semantic Web development. We describe how to apply MAS-CommonKADS, an agent-oriented extension of CommonKADS, to the development of the ITtalks Web Portal. Domain-specific knowledge is modelled by reusing well-known ontologies such as FOAF and RDFiCal. We also describe how to specify CommonKADS problem-solving methods as web services, expressed using the OWL-S language.
Status:
    Deliverable release: 2004-09-30
    Comments on this document should be sent to the public SWAD-Europe mailing list, ☞public-esw@w3.org,

1. Introduction

Realising the Semantic Web vision calls for the development and support of intelligent agents as well as the ontologies for sharing knowledge. Several methodologies have been proposed for developing agent-based system [16] and ontologies [5] as well as knowledge-based systems [1, 15]. How to combine these different modelling techniques and adapt them for the semantic web have been identified as an open research question that requires urgent attention [11, pp. 24].

This paper contributes to solve this research question by studying the application of MAS-CommonKADS [13], an agent-oriented extension of the knowledge-based methodology CommonKADS [15], to the design of the ITtalks Web Portal [6]. ITtalks is a case study developed as part of the DAML project at the University of Maryland, USA. It offers access to information about talks, seminars, colloquia and other information technology (IT) related events, making use of DAML+OIL ontologies for knowledge base representation, reasoning, and agent communication.

Section 2 summarises the MAS-CommonKADS methodology. Section 3 describes how to model agents and tasks for ITtalks. Section 4 presents the knowledge modelling. We show how domain-specific knowledge can be defined by making reuse of widely-used ontologies such as FOAF and RDFiCal. Further, CommonKADS problem-solving methods are modelled as web services using the OWL-S language. Section 5 relates our work with that of others. Finally, section 6 presents our conclusions and highlights future work.

2. The MAS-CommonKADS Methodology

Agent technology can be seen as an extension to object technology, supporting a much richer and sophisticated range of capabilities such as adaptability, cooperation, autonomy, negotiation and delegation. Several methodologies have been proposed for supporting this paradigm. In this paper we apply MAS-CommonKADS [13], an agent-oriented extension of the knowledge-based methodology CommonKADS [15], to developing semantic web applications.

MAS-CommonKADS is based on a set of models covering specific aspects of the development process. The *agent model* specifies agents' characteristics such as reasoning capabilities, sensor/effectors, and services. The *task model* describes the tasks that the agents can carry out. The *knowledge model* defines the knowledge needed by the agents to achieve their goals. The *organisation model* describes the social organisation of the agent society. The *coordination model* illustrates the conversation between agents. The *communication model* details the human-software agent interactions. Finally, the *design model* includes, in addition to the typical action of the design phase, the design of relevant aspects of the agent network, selecting the most suitable agent architecture and the agent development platform.

The application of the methodology consists in developing the different models. Due to space limitation, we concentrate here on the agent, tasks and knowledge models.

3. Modelling Agents and Tasks

ITtalks is organised around domains, which correspond to event-hosting organisations such as universities, research labs or professional groups. Each domain is represented by a separate web domain. Users can access the system through the web or through agents acting on her/his behalf. The web portal includes features such as registration, search, and domain administration. ITtalks users can view information such as location, speaker, hosting organisation and talk topic.

3.1 The Model Agent

For ITtalks, we have identified four agents:

- **User Agent**: Interface agent representing a user. It interacts with the system by servicing particular requests from users and by managing user profiles.
- **Talk Agent**: software agent that discovers the talks in a particular domain according to requests received from the User agent. There is one agent per domain. For instance, three possible domains in the Oxford area are CCLRC, University of Oxford and Oxford Brookes University. There will be an agent for each domain, managing the information about talks in these places.
- **Calendar Agent**: software agent that manages the calendar of the users of a particular domain. There is one agent per domain.
- **Location Agent**: software agent that provides the service of locating places near to a particular location.

The outcome of the agent model is a set of textual templates, one for each agent, which shows information such as agent description, role, etc. For instance, Table 1 presents the template for *Talk Agent*.

Talk Agent	
Type:	Software Agent
Role:	Information provider
Location:	Within each participating domain
Description:	This agent determines the talks that satisfy the criteria provided by the user. It contacts the *Location Agent* to determine the places within a predetermined number of miles of the location provided by the user. Then, it contacts the *Talk Agents* in other places to get the talks on the requested topic within the defined date range. It also searches its local talk database to get information on local talks. Once it has collected all talks in relevant places, it consults the local *Calendar Agent* to check the user availability on the talk dates. Finally, it notifies the user with the list of talks, including information whether the user is available for such talks.
Coordination:	Talk Agent, Calendar Agent, Location Agent
Resources:	Talk ontology

Table 1. Textual Template for *Talk Agent*

Similar templates are defined for the other three agents.

3.2 The Task Model

This model describes the tasks each agent can carry out. Use cases are useful for this activity. We have defined the following tasks to be performed for each agent:

- **User Agent**: introduce user profile; update user profile; request talk; confirm talk in user calendar; delete talk from user calendar.
- **Talk Agent**: search talk; contrast talk timing with user availability; notify users about new talk.
- **Calendar Agent**: include talk in user calendar; delete talk from user calendar; check availability for user.
- **Location Agent**: provide distance between two locations; provide nearby locations.

Tasks can be split into subtasks. UML activity diagrams can be used for representing the subtasks involved in realising a task. For instance, the *Search Talk* task realises the request "*give me talks in area X on topic Y during period Z*". This task involves subtasks such as selecting domains in area X; requesting talks on topic Y to all domains in the area; or qualifying a talk according to user's interest and speaker reputation. Figure 1 shows the subtask flow for *Search Talk* task.

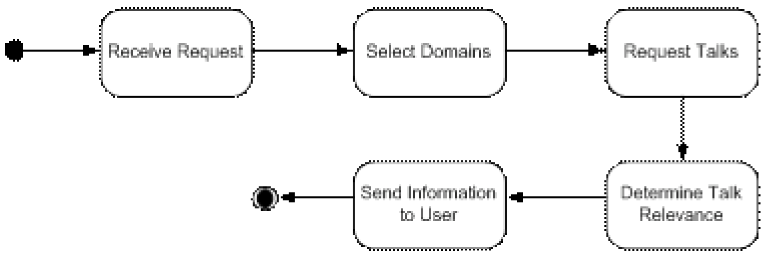

Figure 1. Subtasks involved in *Search Talk* Task

The methodology also offers templates for describing tasks. Table 2 describes subtask *Determine Talk Relevance*.

Task Determine Talk Relevance	
Objective:	Determine the relevance of a talk
Description:	This task assesses the relevance of the talk according to user's profiles (for instance, following user's topic of interests). It may include contacting a service such as CiteSeer to determine speaker reputation. It combines these two values to determine the talk relevance.
Dependency/Flow:	See Figure 1
Input:	Talk information; User Profile
Output:	An indicator of the relevance of the talk

Table 2 . Template for task *Determine Talk Relevance*

4 Knowledge Modelling

The more relevant model for this paper is the knowledge model. It describes the reasoning capabilities of the agents needed to carry our specific tasks and achieve their goals. This model includes the specification of domain knowledge, modelled as concepts, properties, and relationships. The methodology provides a semi-formal language for such modelling, called the Conceptual Modelling Language (CML) [15]. Instead of using CML, we have opted for UML-based graphical language to represent domain knowledge, then expressing this model in Semantic Web languages such as RDF and OWL.

4.1 Modelling Domain-Specific Knowledge

In this part we model specific aspects of the ITtalks domain. One of the important characteristics of the semantic web is the use of common ontologies. There are several simple ontologies available for modelling domains such as people and their interests, e.g. FOAF [2], and calendars, e.g. RDFiCal [14]. We have reused these ontologies for defining our main elements. Figure 2 illustrates the domain knowledge.

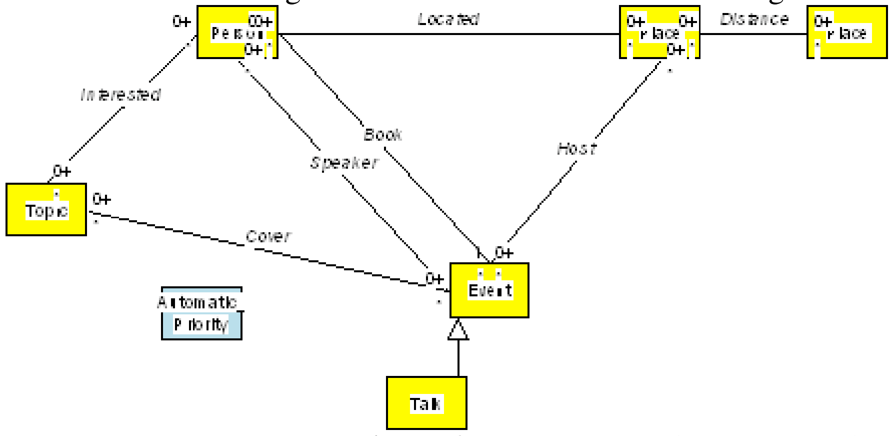

Figure 2. Domain Knowledge for ITtalks

A domain-knowledge description typically consists of two types of ingredients: one or more domain schemas, and one or more knowledge bases. In our case, domain schemas are given by general ontologies such as FOAF and RDFiCal. The knowledge bases correspond to instances of the ontologies. For example, the concept Person can be defined as follows.

```
<foaf:Person rdf:ID=Arenas>

   <foaf:name>Alvaro Arenas</foaf:name>

   <foaf:mbox rdf:resource=mailto:A.E.Arenas@rl.ac.uk/>

   <foaf:homeAddress>Oxford, OX4</foaf:homeAddress>
   <foaf:interest rdf:resource=http://www.w3.org/2001/sw/>

</foaf:Person>
```

4.2 Modelling Problem-Solving Methods

Problem-Solving Methods (PSMs) are software components that can be assembled with domain knowledge bases to create application systems. The knowledge-engineering community has identified and developed PSMs of general usefulness or for specific high-level tasks such as diagnosis, assessment, planning, etc. Knowledge-based methodologies such as CommonKADS [15] or MIKE [1] consider PSMs as essential structures for controlling the methodological activities that must be carried out to build expertise models. Here, we define PSMs as services described using semantic web technologies.

Let us first consider the *Assessment* PSM as described in CommonKADS. This method aims at finding a *decision category* for a *case* based on a set of domain-specific *norms*. For instance, determining the relevance of a talk can be seen as applying an assessment method, where the case corresponds to information about the talk to be qualified; the norms are the set of rules for qualifying a talk according to the user's profile and the importance of the speaker; and the decision category corresponds to the qualification of the talk. The usual solution involves steps such as abstracting information from the input case data; finding the norms that can be used for the case, this information is normally domain-specific; and evaluating the norm with respect to the case data.

We can represent PSMs as services using the mark-up language OWL-S. This has advantages associated to web services such as facilitating automatic composition and interoperability through their semantic description. Our goal here is interoperability and reuse though semantic description, aiming at providing a library of PSMs that can be used in the implementation of knowledge-intensive tasks for the web. We use OWL-S Processes to describe the functionalities of the method.

We give the process description for the assessment method as a subclass of OWL-S Process. It receives two parameters: a *Case*, which corresponds to a list of things - domain-specific objects that constitute the case to be assessed; and *Norms*, which corresponds to a list of things denoting the norms (evaluation guide) to assess the case. The output is *Decision*, a thing indicating a decision category. We define classes for each parameter data type.

```
<owl:Class rdf:ID="Case" />

<owl:Class rdf:ID="Norms"/>

<owl:Class rdf:ID="Decision"/>
```

To make it easier to express, we also give classes to define the input and output parameters (i.e. the parameters as opposed to the parameter values). We give only *CaseInput* for brevity

```
<owl:Class rdf:ID="CaseInput>

<rdfs:subClassOf rdf:resource="&process;Input">

<rdfs:subClassOf>

 <owl:Restriction>

 <owl:onProperty rdf:resource="&process;parameterType"/>

<owl:hasClass rdf:resource="#Case" />

 </owl:Restriction>
```

```
     </rdfs:subClassOf>

  </owl:Class>
```

Finally we define the new PSM Assessment. It is a subclass of the generic atomic process class.

```
<owl:Class rdf:ID="Assessment" >

   <rdfs:subClassOf rdf:resource="&process;AtomicProcess">



 <!-- at least one input is a Case -->

  <rdfs:subClassOf>

   <owl:Restriction>

   <owl:onProperty rdf:resource="&process;:hasInput" />

   <owl:someValuesFrom rdf:resource="#CaseInput" />

   </owl:Restriction>

   </rdfs:subClassOf>



  <!-- at least one input is Norms -->

  <rdfs:subClassOf>

   <owl:Restriction>

   <owl:onProperty rdf:resource="&process;hasInput" />

   <owl:someValuesFrom rdf:resource="#NormsInput" />

   </owl:Restriction>

   </rdfs:subClassOf>



   <!-- Only one output available -->

   <rdfs:subClassOf>

   <owl:Restriction>

   <owl:onProperty rdf:resource="&process;hasOutput" />

   <owl:cardinality rdf:datatype="&xsd;:nonNegativeInteger">1</owl:cardinality>

   </owl:Restriction>

   </rdfs:subClassOf>

   <!-- that output is a Decision -->

   <rdfs:subClassOf>

   <owl:Restriction>

   <owl:onProperty rdf:resource="&process;hasOutput" />

   <owl:hasClass rdf:resource="#DecisionOutput"/>
   </owl:Restriction>

   </rdfs:subClassOf>

  </owl:Class>
```

4.3 Linking Tasks and Problem Solving Methods

This part describes the invocation of the service specified in the previous subsection for the case of task *Determine Talk Relevance*. The input *Case* corresponds to a list including two elements: the grade that the user has giving to the topic of the talk (domain-specific information that is part of the user profile, corresponding a value between 0 –less relevant- and 1 –very relevant-) and the ranking of the speaker according to services such as CiteSeer (we supposed here that in a previous subtask, the number of times that the speaker is cited was requested, corresponding to an integer in the range of 0 to 1000, we assume relevance 0 in case is not in the first 1000 authors). The input *Norms* corresponds to the rule for determining the relevance of the talk: it is a method indicating that the relevance is obtained by giving 70% of value to the user interest and 30% value to the speaker reputation. The output category decision in this case is a real between 0 and 1 indicating the relevance of the talk.

To define the specific instance of the process for the task Determine Talk Relevance, we provide an instance of the Assessment class, with appropriate new datatypes. This is defined as follows. First we provide new classes for the specific types of the input and outputs.

```
<owl:Class rdf:ID="ITTalksCase" >

   <rdfs:subClassOf rdf:resource="Case" />

</owl:Class>
```

Then add properties for the two values given in the *Case*.

```
<owl:DatatypeProperty rdf:ID="TopicGrade" >

   <rdfs:domain rdf:resource="#ITTalksCase" />

   <rdfs:range rdf:resource="&xsd;#nonNegativeInteger" />

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="SpeakerRanking" >

   <rdfs:domain rdf:resource="#ITTalksCaseData" />

   <rdfs:range rdf:resource="&xsd;#nonNegativeInteger" />

</owl:DatatypeProperty>
```

Similar definitions are provided for *Norms* and *Decision*. Now for the specific service we give the following instance of the *Assessment* class which we have defined. We bind the input and output parameters to the specific ITtalks types.

```
<Assessment rdf:ID="ITTalksAssessment" >

  <process:hasInput>

<process:Input ref:ID="UserCase">

   <process:parameterType rdf:resource="#ITTalksCaseData"/>

</process:Input>
```

```
  </process:hasInput>

  <process:hasInput>

<process:Input ref:ID="ITTalksNorms">

   <process:parameterType rdf:resource="#ITTalksNormsDefn"/>

</process:Input>

  </process:hasInput>

  <process:hasOutput>

<process:Output ref:ID="DecisionValue">

   <process:parameterType rdf:resource="#ITTalksDecisionType"/>

</process:Output>

  </process:hasOutput>

</Assessment>
```

5 Related Work

Our work has been inspired by works in the agent and ontology communities. There has been a fresh interest in studying the application of agent-oriented methodologies for Semantic Web applications. In [3], it is presented an agent-oriented approach to build Semantic Web applications. A web service is seen as an agent itself or as part of an agent – an extension that the agent can incorporate into itself once it finds and chooses to adopt a service. Behaviour-Oriented Design (BOD) methodology is employed, extended with DAML-S to mark up services. The Nuin agent platform [10] is designed for Semantic Web applications around the belief-desire-intension (BDI) principles. Nuin agents are deliberative reasoners in the tradition of first-order logic-based inference, implemented on top of the Jena toolkit [12]. The Agentcities initiative has been also studying the interaction between agents and Semantic Web applications through web services. In [9], it is explored the access of services by agents, when services are represented using the DAML-S model. They are interested in the automatic discovery of services and their composition rather than methodological aspects related to the development of agent-oriented applications.

Our work has been also influenced by research on ontologies. In [7], Crubezy and Musen analyses the relation between problem-solving methods and ontologies. They propose a methodology in which domain knowledge bases and problem-solving methods are described as ontologies that can be reused in different applications. Their approach has been further applied to the Internet Reasoning Service [8], a web-based front-end which provides online problem-solving resources. They have influenced our work, although our interest has been in analysing the use of problem solving methods within the context of agent-oriented methodologies for the case of semantic web applications. Close-related is the topic of knowledge-based service composition [4], in which domain-specific knowledge is exploited to compose Web/Grid services into a workflow specification. They also exploit mark up languages as DAML-S for providing semantic characterisation of available services for discovery and appropriate utilisation.

6 Conclusions

This paper presents our experience in using MAS-CommonKADS for modelling semantic web applications. A central feature is the representation of problem-solving methods using the OWL-S language. By contrast to traditional web-service discovery, we show how tasks (services) can be determined when modelling a system based on the involved knowledge, so that the development of the service profile can be derived from the development process.

The work reported here is part of an ongoing project aiming at generating a library of problem-solving methods using semantic web technologies. There are many open issues that require further research to meet our goal. We have noticed that some methods require more expressive power than OWL-S, so we plan to incorporate languages such as RuleML to the description of the method components.

References

1. J. Angele, D. Fensel, D. Landes, and R. Studer. Developing Knowledge-Based Systems with MIKE. Journal of Automated Software Engineering, 5(4):389-419, 1998.

2. D. Brickley. Friend of a Friend RDF Vocabulary. ☞http://xmlns.com/foaf/0.1/, 2001.

3. J. J. Bryson, D. L. Martin, S. A. McIltraith, and L. A. Stein. Toward Behavioural Intelligence in the Semantic Web. IEEE Computer, pages 48-54, November 2002.

4. L. Chen, N. R. Shadbolt, C. Goble, F. Tao, S. J. Cox, C. Puleston, and P. R. Smart. Towards a Knowledge-Based Approach to Semantic Service Composition. 2 nd International Semantic Web Conference. Lecture Notes in Computer Science, vol. 2870, 2003.

5. O. Corcho, M. Fernández-López, and A. Gómez-Pérez. Methodologies, Tools and Languages for Building Ontologies. Where is Their Meeting Point? Data & Knowledge Engineering, 46(1):41-64, 2003.

6. R. Scott Cost, T. Finin, A. Joshi, Y. Peng, C. Nicholas, I. Soboroff, H. Chen, L. Kagal, F. Perich, Y. Zou, and S. Tolia. ITtalks: A Case Study in the Semantic Web and DAML+OIL. IEEE Intelligent Systems, pages 40-47, January/February 2002.

7. M. Crubezy and M. A.Musen. Ontologies in Support of Problem Solving. Handbook on Ontologies, pages 321-341, 2003.

8. M. Crubezy, E. Motta, W. Lu, and M. A. Musen. Configuring Online Problem-Solving Resources with the Internet Reasoning Service. IEEE Intelligent Systems, 18:34-42, 2003.

9. J. Dale, and L. Ceccaroni. Pizza and a Movie: A Case Study in Advanced Web Services. In: Agentcities: Challenges in Open Agent Environments Workshop, Autonomous Agents and Multi-Agent Systems Conference 2002, Bologna, Italy, 2002.

10.I. Dickinson and M. Wooldrige. Towards Practical Reasoning Agents for the Semantic Web. In Second International Conference on Autonomous Agents and Multiagent Systems, Lecture Notes in Artificial Intelligence, 2003.

11.J. Euzenat. Research Challenges and Perspectives of the Semantic Web. Report of the EU-NSF Strategic Workshop, Sophia-Antipolis, France. 2002.

12.HP Labs. The Jena Semantic Web Toolkit. ☞http://www.hpl.hp.com/semweb/jena-top.html. 2002.

13.C. A. Iglesias, M. Garijo, J. Centeno-Gonzalez, and J. R. Velasco. Analysis and Design of Multiagent Systems using MAS-CommonKADS. In Agent Theories, Architectures, and Languages, Lecture Notes in Artificial Intelligence, pages 313- 327, 1997.

14. L. Miller and D. Connolly. RDFiCal: iCalendar in RDF. ☞http://sw1.iltr.org/discovery/2003/11/rdfical/final.html, 2004.

15. G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Vand de Velde, and B. Wielinga. Knowledge Engineering and Management: The CommonKADS Methodology. The MIT Press, 2000.

16. A. Tveit. A Survey of Agent-Oriented Software Engineering. In NTNU Computer Science Graduate Student Conference, Norwegian University of Science and Technology, 2001.