Use of Semantic Web Technologies for Meeting Management Software Development

Aleksandar Nikov[1], Milos Jovanovik[2], Riste Stojanov[2], Milan Petkovski[2], Dimitar Trajanov[2]

[1] Netcetera, Partizanski Odredi 72a, 1000 Skopje
aleksandar.nikov@netcetera.com.mk
[2] Faculty of Electrical Engineering and Information Technologies – Skopje, Rugjer Boskovic bb, 1000 Skopje
milos@feit.ukim.edu.mk, ristes@feit.ukim.edu.mk, milan.petkovski@yahoo.com, dimitar.trajanov@feit.ukim.edu.mk

Abstract. One of the major concerns about the process of developing and maintaining a meeting software application is the complexity and robustness of the architecture and the underlying data model. That sometimes creates a problem with the need of restructuring the interface or the business logic, due to changes in requirements. Therefore, we need a flexible architecture framework, one that will provide easy and fast model modification. In this paper we present our ongoing work for developing an online meeting application, which is based on a flexible architecture framework. The application development is based on the following concepts: a data model based on ontologies and semi-structured data stored in RDF format, a service oriented architecture with the capability of automatic composition of semantic web services, and a user interface based on a guided natural language input, dynamic forms, and a graphic data representation tool, both based on ontologies.

Keywords: semantic web, meeting software, architecture, ontologies, OWL, RDF.

1 Introduction

One of the major concerns about the process of developing and maintaining a meeting software application, is the complexity and robustness of the architecture and the underlying data model: the developers need to know exactly which types of data will the client work with. The developers need this in order to construct the database, i.e. the data model, to develop the business logic, and to construct the forms for the user interface, so that the client can manipulate the data. Also, the developers sometimes have the problem with the lack of documentation and specification, which in the later development phases will create problems, like the need of restructuring the user interface or even major changes in the business logic. In real life, the clients are not concerned with the data model of their application. Their interest is the work

procedure that the application needs to provide, and the way they will communicate with the application.

Therefore, we need a flexible architecture framework, one that will provide easy and fast model modification. It would also be very convenient if the client can *describe* his requirements directly to the application, by using a natural language, i.e. by using semantics which both humans and the application can understand. Then, if we capture this semantic description provided by the client and represent it in a computer understandable format, we have the definition of the system and the data model required by the client.

With the current technological achievements, we already have the tools for developing such applications. The Semantic Web technologies have introduced a new way of information and knowledge management. The extensibility of ontologies, which are at the core of the Semantic Web, provides the application developers with the convenience of not having the need of a strict solution architecture. The ontology representation actually maps the semantic meaning of the words and phrases into an ontology graph, which can be understood by the computer. Therefore, if we choose to use ontologies to define the architecture of a system, we can have a very flexible architecture design.

In this paper we present our ongoing work for developing an online meeting application. The application development is based on the following concepts: a data model based on ontologies and semi-structured data stored in RDF format, a service oriented architecture with the capability of automatic composition of semantic web services, and a user interface based on a guided natural language input, dynamic forms, and a graphic data representation tool, both based on ontologies. We have already developed some of these features, as a proof of concept.

2 Related work

During the requirements definition phase, many of the existing web software and services for organizing and management of online meetings were reviewed. The process included trial and analysis of both meeting and collaboration software so that we could differentiate and locate the most essential features the common user likes to see in his web conferencing software. Currently there are several available commercial solutions on the market that are worth mentioning: GoToMeeting [1], Microsoft Live Meeting [2], Cisco WebEx [3], Adobe Connect Pro [4], Elluminate Live [5], etc. Furthermore, a number of free and open source solutions were reviewed, including DimDim [6], WebHuddle [7] and OpenMeetings [8]. All of the above mentioned introduce various feature and service type differences, most notably in means of implementing a fat or thin client architecture. However, none of them is based on semantic web technologies, and thus our main goal is to create a new type of meeting management software that will use the benefits of the semantic web technologies.

One of the main factors for success of the new application will be the creation of the innovative interface that will allow to end-users simple and a fast way to enter and use the heterogeneous information related to meetings. The end-user tools for creating

SW content can be classified into two groups [9]. The first one is GUI-rich environments for the construction of SW schemata (ontologies) and instances such as the Protégé ontology editor [10][11], and the ACE View editor [12]. The generation of forms in the user interface from ontologies has been suggested and implemented in the OIL to XML schema transformator [13]. The idea that we want to implement in our framework, where the clients themselves can define the applications they use, was proposed and implemented as a part of MIT's Haystack Platform [14][15][16].

The second group is a text based ontology editors, where the user writes sentences in a guided natural language with the assistance of the user interface and the application, which allows him to manipulate the ontologies and the instances [17][18][19][20].

Both approaches have good and bad sides. Our idea is to create an interface that is somewhere in between. We want to use the flexibility of text based interfaces but to have speed and easy for use of rich GUI based interface. We currently have implemented the rich GUI based interface and also one form of text based interface to SW.

3 The software architecture

Capturing the semantic description provided from the client, and representing it in an ontology based language, such as OWL, can provide complete mapping of the requirements into the system. This information are stored in semi-structured OWL format, where they can be easily extended or even changed. The architecture of the proposed system is given on Figure 1. This architecture has four modules: GUI module, Grammar, Knowledge Base and Persistence module. All of these modules are developed separately, in different technologies, and communicate with each other through web services.

The GUI module has two ways of operation. The first one is the rich GUI interface, where the user is entering and retrieving data over forms and trees provided from the system. The improvement of the user interface is achieved with the development of graphical visualization for the model which enables direct manipulation of all classes, properties and instances in the currently active ontology. The goal of this approach is to simplify the way the user manipulates and organizes the ontology by amplifying many of the benefits that rich graphical interfaces introduce, generally in means of effectiveness, easy-use and easy-learning. For this purpose, the interface was developed as a RIA (Rich Internet Application) solution by using Silverlight 3.0 technologies [22]. The ontology model itself is visualized by a graph-like structure consisted of different types of user controls, each representing classes or properties from the ontology definition or their appropriate instances from the knowledge base.

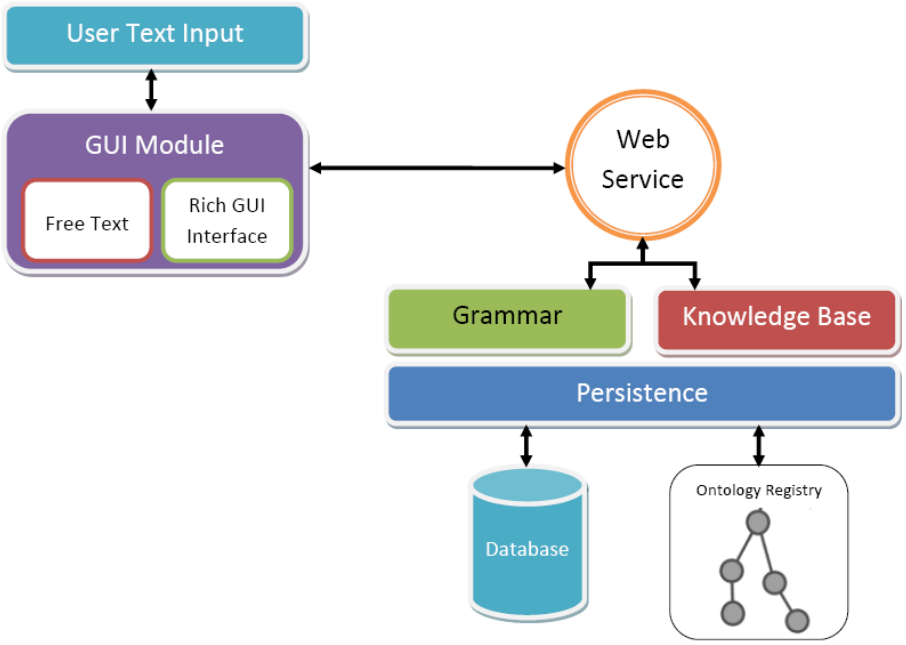

Fig. 1. Meeting software architecture.

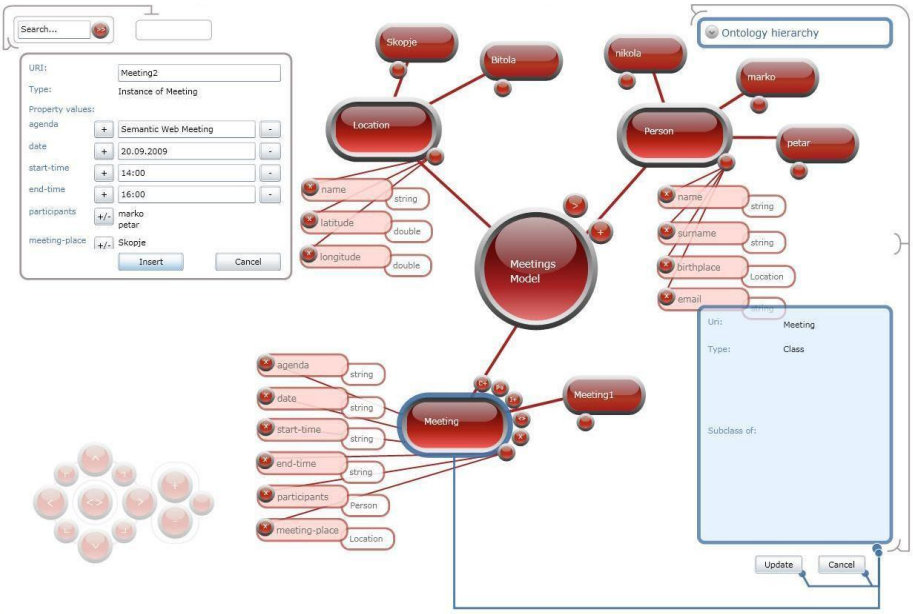

Fig. 2. Example model representation with the OntoBrowser Silverlight web application.

Figure 2 shows an example where different types of user controls, representing the class, property and instance objects of the ontology and knowledge base, are illustrated. In the given scenario, the model consists of three classes, Person, Location and Meeting, where the Person class has a property “birthplace” of type Location and the Meeting class has properties “participants” of type Person and “meeting-place” of type Location. All three classes have corresponding instances.

The second one is the "Free Text" mode, where the user is interacting with the application through a controlled natural language. On the Figure 1, the second mode is shown. As we can see, the GUI module is responsible for retrieving the user inputs and for performing the calls of the services that will manipulate this data. Also after it will retrieve the results from this services (modules), it displays the results.

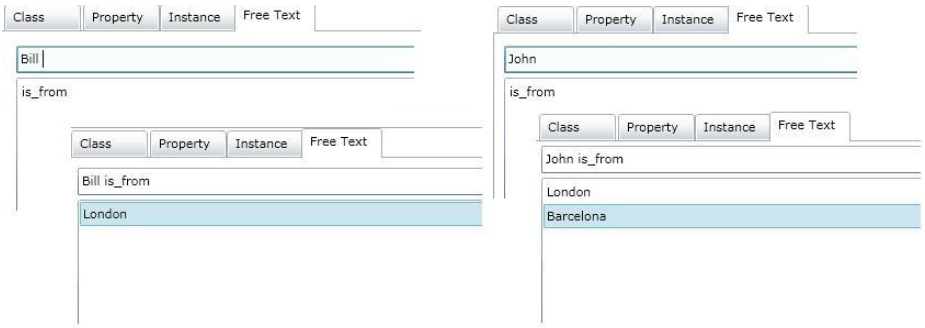

Fig. 3. Free text entry.

For the text based communication, it first parses the text, and forms so called *Common Objects***.** We use five common objects: CommonClass, CommonProperty, CommonInstance, CommonStatement and CommonEnvironment. The first are used to simply wrap the ontology Class, Property and Statement objects. We don't use the standard ontology objects for flexibility reasons i.e. because of the possible need of additional properties that will be used by our system. One possible example of such additional property may be the probability that the returned object represents the given literal object. The CommonEnvironment object is used for performance improvement reasons. It caches the data for a given user, prefetches data, describes additional visual properties and the commands that need to be executed.

The common objects formed by the GUI module contain only the literal without additional class or property values. This information is added to the common objects in the Grammar and Knowledge base modules. The grammar module contains rules for the classes and properties found in the knowledge base. It is more relevant than the results returned from the knowledge base module, and therefore it is first called to fill the properties that will fit its rules. The results from this module are sent back to the GUI module, which then sends them to the Knowledge Base module. Here, the common objects are used for querying the ontology. The results from this queries are again assembled in common object and sent back to the GUI module. It then shows the suggestions (results) to the user, or executes an action if the user specified so.

In the current implementation we are using Jena as a communication mediator with the ontology. Jena is a Java framework for developing Semantic Web applications [23]. It has been developed by HP Labs and it is an open source project. Basically, Jena provides Java environment for working with RDF, RDFS, OWL, SPARQL and reasoning engines. The Jena framework creates an additional layer of abstraction that translates the statements and constructs of the Semantic Web into Java artifacts, such as classes, objects, methods and attributes. These artifacts reduce the effort needed for programming Semantic Web applications. One of the strongest sides of Jena lies in its excellent documentation. The large number of resources, including descriptions and tutorials that can be found on the Web, encourage programmers to further develop their Semantic Web applications utilizing this framework.

Fig. 4. The meetings ontology shown in Protégé.

4 The meetings ontology

Considering the goal of creating a semantic web application with specialization for online meetings management, the application's initial development was focused on defining a proper ontology for the domain of interest. Having said that, the ontology model itself had to represent the common taxonomy met in meetings organization with addition to the relationships and properties of the classified concepts. The process in its basics included use of the Protégé ontology editor [10][11] paired with research on existing ontology libraries [21] and resulted in the creation of MeetingsOntology.owl as the fundament for further development.

Figure 4 depicts visualized representation of the classes defined with the ontology model. As presented, the current ontology version unifies various meeting domain concepts, including member roles (participant, initiator, moderator, viewer), thinking templates (Simplex, DoIt, Planning Cycle, Six Thinking Hats) and tools (brainstorming, SWOT, decision trees, etc.), collaboration types and appropriate resources, etc.

5 Conclusion and future work

Ontology-based knowledge management gives the opportunity to the users to store their data in semi-structured repositories, which can be easily managed with the use of the semantic web technologies and tools. What we want to achieve, is a better user environment through the use of a different user interface concept, so that both developers and end-users can manage the data, i.e. knowledge base with the use of guided free-text entry, which is close enough to the natural language, and with the use of a graphic ontology editor tool. The architecture of the application and the knowledge base are defined by ontologies, so by managing the ontologies from the same (or similar) user interfaces, both the user and the developer can make the necessary changes. In this way, the needed flexibility of the system architecture can be easily achieved.

The solution presented in this paper is the starting concept of the system we are building. We plan to extend the grammar, enrich the user interface, solve some of the issues of the Jena framework which prevent us from using large amounts of data, and introduce a complete service oriented architecture (SOA) solution. The SOA approach will allow us to publish the functionalities of every module as a web service, into a common web service repository. By annotating the web services with semantic meaning from the ontologies of the system, we plan to develop a semantic web service repository. Then, we will be able to introduce a service engine which will make automatic semantic web service compositions. This automatic semantic web service composition allows the modules to get results for more complex goals, goals for which there is no particular web service defined.

References

1. GoToMeeting, http://www.gotomeeting.com
2. Microsoft Live Meeting, http://office.microsoft.com/en-us/livemeeting/
3. Cisco WebEx, http://www.webex.com/
4. Adobe Connect Pro, http://www.adobe.com/products/acrobatconnectpro/
5. Elluminate Live, http://www.elluminate.com
6. DimDim, http://www.dimdim.com/
7. WebHuddle, http://www.webhuddle.com/
8. OpenMeetings, http://code.google.com/p/openmeetings/
9. M. V. Kleek, M. Bernstein, P. Andre, M. Perttunen, D. Karger, M. C. Schraefel: Simplifying Knowledge Creation and Access for End-Users on the Semantic Web. Workshop on Semantic Web User Interaction (2008)
10. J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubezy, H. Eriksson, N. F. Noy, and S. W. Tu.: The evolution of Protégé: An environment for knowledge-based systems development. Technical Report SMI-2002-0943, Stanford Medical Institute (2002)
11. The Protégé Ontology Editor, http://protege.stanford.edu/
12. K. Kaljurand: ACE View - an ontology and rule editor based on Attempto Controlled English (2008)
13. I. Gurevych, S. Merten, R. Porzel: Automatic Creation of Interface Specifications from Ontologies (2003)
14. D. R. Karger, K. Bakshi, D. Huynh, D. Quan, V. Sinha: Haystack: A Customizable General-Purpose Information Management Tool for End Users of Semistructured Data (2005)
15. K. Bakshi, D. R. Karger: End-User Application Development for the Semantic Web (2005)
16. MIT Haystack Group, http://groups.csail.mit.edu/haystack/
17. A. Bernstein, E. Kaufmann: GINO - A Guided Input Natural Language Ontology Editor (2006)
18. A. Funk, V. Tablan, K. Bontcheva, H. Cunningham, B. Davis, S. Handschuh: CLOnE: Controlled Language for Ontology Editing (2007)
19. J. Bao, P.R. Smart, N.R. Shadbolt, D. Braines, G. Jones: A Controlled Natural Language Interface for Semantic Media Wiki (2009)
20. V. Tablan, T. Polajnar, H. Cunningham, K. Bontcheva: User-friendly ontology authoring using a controlled language (2006)
21. DAML Ontology Library, http://www.daml.org/ontologies/
22. Microsoft Silverlight, http://silverlight.net/
23. Jena - A Semantic Web Framework for Java, http://jena.sourceforge.net/