Ontology Based Application Server to Execute Semantic Rich Requests

Dilvan de Abreu Moreira and Flávia Linhalis
Institute of Mathematics and Science Computing (ICMC)
University of São Paulo, São Carlos, Brazil
{dilvan, flavia}@icmc.usp.br

1. Introduction

Since computing earlier days, people have wanted computers that could interface and be programmed using easy to use tools like speech or gesture. However, until now, such a computer system remains in the realm of scientific fiction. Now, with recent advances in human computer interfaces (such as speech understanding, ubiquitous computing, tablet computers), and in the processing of semantic information by computers (through initiatives such as the Semantic Web), new technologies and tools (such as Protégé and Jena) are maturing to a level where such a system becomes a possibility.

Our work is about the application of semantic information to make computer systems smarter (using ontologies to add knowledge about application domains and common sense) and, with the help of new kinds of human-computer interfaces, to produce systems that a computer layperson can understand and program in useful ways.

In particular, this work concentrates on the processing of semantic rich requests through the activation of software components stored in an application server. Semantic information from each request is combined with semantic information from ontologies, describing the application domain and the available components, to generate arguments and activate the most appropriate component (or components) to attend the request. Ontology Based Application Servers (OBAS, section 2) can be used to improve component activation. We are testing this idea with the development of a prototype of an OBAS to execute imperative natural language requests expressed in several natural languages (NL-OBAS). In section 3 we focus on NL-OBAS ontologies.

2. Ontology Based Application Servers

The idea of improving component activation using ontologies is outlined in [1], where the authors propose an Ontology Based Application Server (OBAS): an application server, like the ones described in the Sun's J2EE architecture, but with the capability of using semantic information about the software components it holds. In an OBAS, an ontology captures properties of, relationships between, and behaviors of components. These component descriptions may be queried, may foresight required actions, e.g. preloading of indirectly required components, may be checked to avoid inconsistent system configurations, or may improve the dynamic composition of services [1, 2].

Expanding this idea for an OBAS, a request does not need to be addressed to a particular component implementation. For instance, if one needs to send an email, an email request can be made to a generic software component described in the OBAS as an email sender component. This request will contain only the information needed to

send an email (recipient, sender, subject, body, etc) the actual software component and its parameters are going to be determined by the OBAS using domain and component ontologies. Once a system capable of attending such requests is available, the next step is to connect it to human-computer interfaces that could generate the requests, from user interaction, and show the results of these requests in the context of the interaction.

To test these ideas, a prototype of an Ontology Based Application Server for the execution of imperative natural language requests (NL-OBAS) is being developed [3], as described in the next section.

3. The NL-OBAS

The NL-OBAS architecture (Figure 1) provides the UNL-Enconverter service to convert natural language requests into an interlingua named UNL (Universal Networking Language) [4]. The interlingua allows the use of different human languages to express the requests (other systems are restricted to English). Currently, the input requests must be imperative sentences; however the system can be extended to other sentence types.

The architecture also provides a Semantic Mapping Service, that uses an ontology to extract relevant information about the domain components and semantic information from the UNL representation of a request. The Component Loader service uses this information to dynamically load specific software components and execute methods to fulfill a request.

The NL-OBAS software components are related to a specific domain. These components are described in the Component Ontology and the application domain is described in the Domain Ontology. The components can query and modify the Domain Ontology.

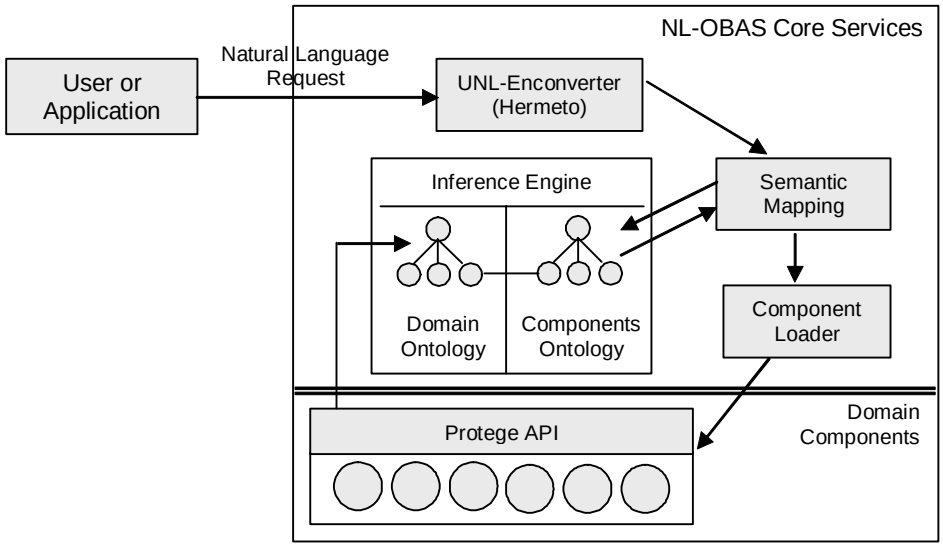

Figure 1 – NL-OBAS Architecture.

The Domain and Component ontologies were both developed using the Protégé tool [5] and are represented in OWL (Ontology Web Language) [6]. The remaining of this section is about these ontologies. Details about the services of the NL-OBAS (UNL-Enconverter, Semantic Mapping, Component Loader) are described at Linhalis & Moreira [3].

Figure 2 presents the Component Ontology classes, attributes and relationships. This ontology has to be instantiated in accordance with the syntactic and semantic characteristics of the components in the Domain Components Layer of the NL-OBAS.

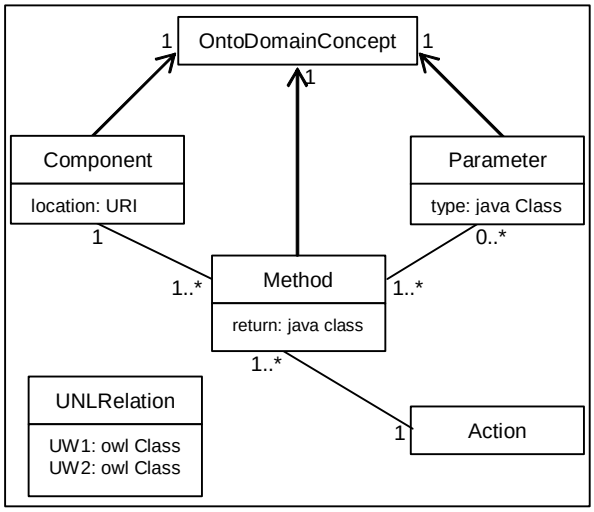

Figure 2 – Component Ontology.

The instances of the OntoDomainConcept class correspond to concepts of the application domain that are also concepts represented as classes in the Domain Ontology (Figure 3). Each Component class instance corresponds to the representation of an application domain software component that can be related to one or more concepts (represented as instances) of the OntoDomainConcept class. For example, considering the course management domain, "Student", "Teacher" and "Course" are concepts and belong to the OntoDomainConcept class. An instance of the Component class, like TeacherComponent, represents a component that is responsible for the execution of actions related to the "Teacher" concept.

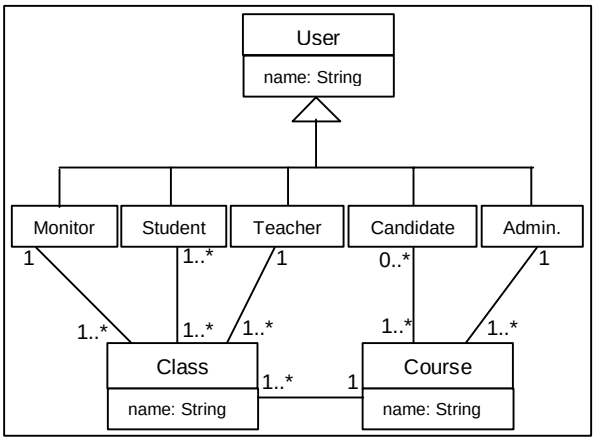

Figure 3 – Domain Ontology.

The Method class instances correspond to the methods of each component. The Parameter class instances correspond to the arguments of each method. And finally, the Action class instances correspond to imperative verbs. Each verb (action) is related to one or more methods, and each method is related to one verb.

The UNLRelation class indicates the mapping between the UNL interlingua and the software components. This class has instances representing all UNL relations

being used in the imperative sentences of the application domain. Each instance of this class is related with Component, Parameter or Action classes of the Component Ontology. More information about the UNLRelation class can be found at [3].

The natural language requests are related to a specific domain that is represented in the Domain Ontology. We defined and instantiated the Domain Ontology with relationships between the concepts of the course management domain.

4. Conclusion and Future Work

The NL-OBAS architecture can be used in different application domains; it is just necessary to write the appropriate software components, define the dictionary and grammar rules (that will be used by the UNL-Enconverter service), create instances of the Component Ontology and define the Domain Ontology.

Much work still needs to be done. We plan to improve and expand the NL-OBAS prototype and transform it in a system that can be used, in restricted domains, by experts in other fields, such as biologists, physicians, geneticists, stock market traders and others, to write useful programs, without the help of computer experts.

Using natural language or other kinds of human-computer interfaces, the OBAS technology has the potential to be a hot research topic in the coming years and to offer interesting new opportunities for research as this technology can be central to the development of the future Semantic Web applications.

References

[1] Oberle, D., Staab, S. and Eberhart, A. Towards Semantic Middleware for Web Application Development. *IEEE Distributed Systems Online*, February, 2005.

[2] Sugumaran, V. and Storey, V. C. A Semantic-Based Approach to Component Retrieval. *ACM SIGMIS Database*, v. 34, n. 3, 2003, pp. 8-24.

[3] Linhalis, F. and Moreira, D. A. "Ontology-Based Application Server to the Execution of Imperative Natural Language Requests". H.L. Larsen et al. (Eds.): 7th International Conference on Flexible Query Answering Systems (FQAS), June 2006, Milano, Italy. *Lecture Notes in Artificial Intelligence (LNAI) 4027*, pp. 589–600, 2006. Springer-Verlag, Berlin Heidelberg, 2006.

[4] Ushida, H. and Zhu, M. The Universal Networking Language beyond Machine Translation. International Symposium on Language and Cyberspace, Seoul (South Korea), 2001

[5] Noy, N. F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R. W. and Musen, M. A. Creating Semantic Web Contents with Protégé-2000. IEEE Intelligent Systems, v.16 n.2, 2001, pp. 60-71.

[6] McGuinness, D. L. and van Harmelen, F. Web Ontology Language Overview. W3C Recommendation, February 2004. http://www.w3.org/TR/owl-features/.