Ontology Based Application Server to Execute Semantic Rich Requests

Flávia Linhalis and Dilvan de Abreu Moreira

University of São Paulo - Brazil
Introduction

- **Application Servers**
  - Provide functionalities of conventional middleware + technologies for Web access
  - Facilitate the development of component based applications by providing transparency to programmers

- **Ontology Based Application Servers (OBAS)**
  - Ontologies to integrate the services of an application server, like J2EE (Oberle et al., 2004)
  - OBAS technology has the potential to be the future of today’s application servers for the Semantic Web
Introduction

- Web to help breaking the language barrier
  - The Universal Networking Language (UNL) project has this goal
  - Started in 1996, it embraces research institutions from several countries
  - A DeConverter and EnConverter for each language form a Language Server residing in the Internet to allow users to communicate in their native language
Introduction

- Our work joins that two ideas:
  1) to use UNL to make natural language computer readable
  2) to use an OBAS to semantically enrich the execution of services.

- An Ontology Based Application Server function is to execute Natural Language requests (NL-OBAS).
Introduction

- NL-OBAS allows users to write requests for servers, using their native language.

- To achieve this goal, the NL-OBAS performs a semantic mapping between UNL relations and software components using ontologies. It:
  - transparently hides the dynamic composition of a service to execute the request
  - searches for suitable software components to compose that service
The UNL Project

UNL represents sentences using three elements:

- Universal Words (UWs): Each UW relates to a concept represented as an English word.
- UWs can be optionally supplied with semantic information to restrict its meaning.
- Examples:
  - book
  - book (icl>publication)
  - book (icl>reserve)
Relation Labels (RLs): express semantic relations between UWs.
- RLs are represented as a pair relation_label(UW1, UW2).
- There are today 45 RLs defined.
- Example:
  - obj (move, table): defines a thing that is affected by an event. The example means the “table moved”.

Atribute Labels (ALs): express additional information about UWs (verb tense, intention, emphasis, etc).
- Example: obj(eat.@past, apple.@pl).
NL-OBAS General View

- **UNL-Enconverter (Hermeto)**: converts NL into UNL.
- **Semantic Mapping**: uses the Component Ontology and the UNL relations to extract semantic information to search the suitable components.

- **Component Loader**: loads components and executes specific methods to fulfill the natural language request.

- **The application domain is described in the Domain Ontology.**
The UNL-Enconverter Service (Hermeto)

- Hermeto system is the core of UNL-Enconverter Service.
- It can be used to convert any natural language into UNL.
- It receives as input a dictionary and a grammar for each language.

- English grammar and dictionary were developed to the course management domain (only to imperative sentences).
Component Ontology

- It describes the domain software components

**OntoDomainConcept** → concepts related to the domain. Each instance is a class of the Domain Ontology.

**Component** → represents the components. Each component representation is related to one concept in the Domain Ontology.

**Method** → methods of components.

**Parameter** → methods arguments.

**Action** → imperative verbs. Each method is related to one action.
NL-OBAS Description

- Component Ontology (UNLRelations class)
  - Its instances are UNL relations
  - The UWs of each relation are related with classes Component, Parameter or Action of the Component Ontology
NL-OBAS Description

- Domain Ontology
  - It was created and instantiated to the course management domain
NL-OBAS Description

- **Component Loader**
  - It uses Java Reflection to load the components and execute its methods

- **Domain Components Layer**
  - The components can perform queries and modify the instances of the Domain Ontology
  - They can perform external actions, such as send e-mails
  - Each component is related to a concept of the Domain Ontology
Semantic Mapping Service (SMS)

- It performs a semantic mapping between UNL relations and software components (using the Component Ontology).
- It identifies the component, methods and arguments to execute the request.
- Example: “Delete administrator Mary from course Java”.
  - The UNL-Enconverter Service generates the following UNL representation:

```
obj(delete,administrator)
gol(delete,course)
nam(administrator,Mary)
nam(course,Java)
```
Example:

The SMS uses the Component Ontology and the semantics of the UNL representation to identify the following information:

- **Action**: delete
- **Main Concept**: administrator
- **Other Concept**: course
- **Argument**: Mary
  - **Argument type**: administrator
- **Argument**: Java
  - **Argument type**: course
- **Number of arguments**: 2
- **Return type**: none

This information is used to search for the suitable component and, finally, the Component Loader is called.
Related Works

- Sugumaram and Storey (2003) use ontologies to search for components
  - Natural language to specify the components description
  - Their work do not execute requests

- OAA (Open Agent Architecture) and SOTA
  - They also use components or agents to execute natural language requests.
  - Our differential and advantage:
    - Requests are converted to an interlingua → communication in several languages.
Conclusion and Future Work

NL-OBAS can be used in different application domains. It is necessary to:
- build the appropriate software component set,
- define the dictionary and the grammar rules,
- create instances of the Component Ontology,
- define the Domain Ontology.

Future Works:
- Improve the dynamic service composition: it is limited by the number of imperative verbs.
- Perform the semantic mapping to other kind of sentences (not only imperatives)
- Extend NL-OBAS to support enterprise applications (using, for example, Jboss)
Acknowledgments

Thanks to Interinstitutional Center for Computational Linguistics (NILC - São Carlos/Brazil) for making the UNL-HERMETO system available to our research.

And the Protégé Team.
Contacts

- Flávia Linhalis: flavia@icmc.usp.br
- Dilvan de Abreu Moreira: dilvan@stanford.edu dilvan@gmail.com