EXECUTION OF IMPERATIVE NATURAL LANGUAGE REQUISITIONS BASED ON UNL INTERLINGUA AND SOFTWARE COMPONENTS

Flávia Linhalis
Institute of Mathematics and Science Computing. São Paulo University, Av. Trabalhador São-Carlense, São Carlos, Brazil
Araraquara University Center (Uniara), Rua Voluntários da Pátria, Araraquara, Brazil

Dilvan de Abreu Moreira
Institute of Mathematics and Science Computing. São Paulo University, Av. Trabalhador São-Carlense, São Carlos, Brazil

Keywords: Natural language, Universal Networking Language (UNL), Ontology, Software Component.

Abstract: This paper describes the use of an Interlingua as a new approach to the execution of imperative natural language (NL) requisitions. Our goal is to embed a natural language interface into applications to allow the execution of users requisitions, described in natural language, through the activation of specific software components. The advantage of our approach is that natural language requisitions are first converted to an interlingua, UNL (Universal Networking Language), before the suitable components, methods and arguments are retrieved to execute each requisition. The interlingua allows the use of different human languages in the requisition (other systems are restricted to English). The NL-UNL conversion is preformed by the HERMETO system. In this paper, we also describe SeMaComp (Semantic Mapping between UNL relations and Components), a module that extracts semantic relevant information from UNL sentences and uses this information to retrieve the appropriated software components.

1 INTRODUCTION

The idea of a restricted natural language interface is very appealing because natural language is the way which humans communicate with each other. That is why several systems, developed throughout the last twenty-five years, have pursued the goal of describing user intentions in restricted natural language and have them executed by computers (Ballard & Bierman, 1979; Price et al., 2000; Cheyer & Martin, 2001; Tsai et al., 2003).

Despite the intuitive appeal of a natural language interface, it has been argued that a language like English has too many ambiguities to be useful for communicating with computers. The UNL (Universal Networking Language) project aims to represent, in the cyber world, the functions of natural languages used in human communication. But, different from natural languages, UNL expressions are unambiguous. UNL is an interlingua that enables computers to process information and knowledge across language barriers. UNL enables people to express knowledge conveyed in natural languages (English, French, Spanish, Portuguese, and so on). It also enables computers to intercommunicate, thus providing people with a linguistic infrastructure for distributing, receiving and understanding multilingual information (UNL Center, 2003).

Our goal is to be able to execute user requisitions described in several restricted natural languages, such as English, Portuguese, French, and so on. In order to do this, user requisitions are first converted into UNL. The UNL representation is used to extract relevant semantic information from the input sentences that will be necessary to retrieve and execute software components. The user requisitions will be executed through the activation of specific component methods.

In this paper, requisitions refer to user intentions described in a high level semantic abstraction and related to a specific domain. For example, considering the domain of web course management, valid requisitions could be:

Linhalis F. and de Abreu Moreira D. (2005).
EXECUTION OF IMPERATIVE NATURAL LANGUAGE REQUISITIONS BASED ON UNL INTERLINGUA AND SOFTWARE COMPONENTS.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 254-259
DOI: 10.5220/0002528902540259

(a) "Add student John Smith to the Hypermedia course."
(b) "Send an e-mail to the students of the Operating Systems course saying that the test will be on December 14th."

The main advantage and innovation of our approach is the use of UNL as an interlingua. In this way, natural language requisitions, expressed in different human languages, can be translated into the same UNL representation before being executed. To convert natural language into UNL, the HERMETO (Martins et al., 2004) system was used.

To achieve our goal, a new system, the SeMaComp (Semantic Mapping between UNL relations and Components) system is being developed. It uses ontologies to identify what components, methods and arguments will be necessary to execute requisitions expressed in UNL.

This paper is organized in the following way: Section 2 discusses related works about interfaces for the execution of natural language requisitions. Section 3 describes the UNL project and the HERMETO system. Section 4 presents an imperative natural language requisition system using SeMaComp. Section 5 describes an application in the web course management domain. Section 6 concludes the paper with some remarks on future work.

2 RELATED WORKS

The first efforts to execute user requisitions expressed in natural language began in the later 70s. The NLC (Natural Language Processing) system (Ballard & Bierman, 1979) was designed to process data stored in matrices or tables. It enables a computer user to type English commands into a display terminal and watch them executed on the screen. A more recent example of the same idea is NaturalJava system (Price et al., 2000). Its interface accepts English sentences as input and generates the Java source code to execute the sentences. Both systems are very limited because input must be in a restricted algorithmic fashion. Higher semantic level sentences are not allowed.

Some approaches, such as OAA (Open Agent Architecture) (Cheyer & Martin, 2001) and SOTA (Tsai et al., 2003), have worked with software components and agents to get a higher level of abstraction. OAA is a framework for constructing agent-based systems that makes it possible for software services to be provided through the cooperative efforts of distributed collections of agents. OAA provides an interface that accepts English sentences as input that are converted to ICL (Interagent Communication Language), a Prolog-based language. ICL is used, by the agents, to communicate with each other and to register their capabilities with a facilitator agent. The facilitator is responsible for matching ICL requests to choose the most suitable agents to execute these requests.

SOTA is an office task automation framework that uses web services, ontology, and software agents to create an integrated service platform that provides user-centric support for automating intranet office tasks. SOTA can take plain English text sentences as input and serve users with a single and integrate user-interface form to access web services, thus avoiding the need to access each distributed service manually. SOTA performs its tasks in three phases: first it parses user input sentences to identify possible web services using an ontology, next it prepares most of the input data fields the services requires, and, finally, it combines related services to define a single task flow.

Such as OAA and SOTA, our work aims to use a restricted natural language interface to describe user requisitions and software components to execute these requisitions. One of the major differentials of our approach is that the natural language requisitions are first converted to an interlingua (UNL (Ushida & Zhu, 2001)), and then the requisitions are analyzed and the appropriated component methods are called (to process the requisitions). References to systems that convert user requisitions into an interlingua and use that interlingua semantic information to choose the appropriated software components have not been found in the literature.

3 THE UNL PROJECT

The UNL project started in 1996 and currently embraces several universities and research institutions in the world. The project proposed an interlingua, entitled Universal Networking Language (UNL), which has sufficient expressive power to represent relevant information conveyed by natural languages. For each natural language, two systems should be developed: a "Deconverter" capable of translating texts from UNL to this natural language, and an "Enconverter" which has to convert natural language texts into UNL.

UNL represents sentences using three elements (UNL Center, 2003):

- Universal Words (UWs): Each UW relates to a concept and is represented as an English word that can be optionally supplied with semantic specifications to restrict its meaning. The following are examples of UWs: book, book(icl>publication), book(icl>reserve). In the

two last examples, the meaning of book is restricted by other UWs ("publication" and "reserve"). The restrictions allow representing UWs as disambiguated English words.

- Relation Labels (RLs): RLs express semantic relations between UWs. There are today 44 RLs defined. The RLs are represented as a pair relation_label(UW1, UW2). For example:
  - obj (move, table): This relation defines a thing in focus that is directly affected by an event or state. In our example, it means the "table moved".
- *Atribute Labels* (ALs): ALs express additional information about UWs, such as verb tense, intention, emphasis, etc. ALs are represented as UW.@atrib$_1$.@atrib$_2$...@atrib$_n$. For example: obj(eat.@past, apple.@pl). The AL "@past" indicates past and "@pl" indicates plural.

We do not intend to describe the UNL language here in details. A full specification of UNL can be found at http://www.undl.org.

3.1 HERMETO

HERMETO is a standalone environment for fully automatic syntactic and semantic natural language analysis (Martins et al., 2004). It can be used to convert any natural language into the Universal Networking Language (UNL). It receives as input a dictionary and a grammar that should be parameterized for each language, in a way very similar to the one required by the UNL Center Enconverter program (UNL Center, 2003). However, HERMETO brings together three special distinctive features: 1) it takes rather high-level syntactic and semantic grammars; 2) its dictionaries support attribute-value pair assignments; and 3) its user-friendly interface comprises debug, compiling and editing facilities. In this sense, it provides a better environment for the automatic production of UNL expressions.

Figure 1 shows examples of HERMETO dictionary entries and Figure 2 presents examples of its grammar rules. We do not intend to describe the dictionary syntax and the rules formalism in this paper. This information can be found in Martins et al (2004).

```
[a.m.] {} a.m. "a.m.(icl>ante meridiem)" (pos:abr)
<EN,1,1>;
[AM] {} a.m. "a.m.(icl>ante meridiem)" (pos:abr)
<EN,1,1>;
[a] {} a "a" (pos:art,typ:ndf) <EN,1,1>;
[access] {} access "access" (pos:ver) <EN,1,1>;
[add] {} add "add(icl>do)" (pos:ver) <EN,1,1>;
[admin] {} administrator "administrator" (pos:nou)
<EN,1,1>; [after] {} after "after(icl>how)"
(pos:adv,typ:tme) <EN,1,1>;
[after] {} after "after(icl>time)" (pos:pre) <EN,1,1>;
```

Figure 1: Examples of dictionary entries

```
; 2. PHRASE LEVEL
; 2.1. IMPERATIVE VERB PHRASE (IVP)

IVP[1] := VER.@entry + NOU + NOU -> nam(:02,:03),
obj(:01,:03)
IVP[2] := VER.@entry + NOU + NOU + PRE + ART +
NOU + NOU -> nam(:02,:03), nam(:06,:07),  bj(:01,:02),
gol(:01,:07)
IVP[2] := VER.@entry + NOU + NOU + PRE + NOU +
NOU -> nam(:02,:03), nam(:05,:06), obj(:01,:02),
gol(:01,:06)

; 3. WORD LEVEL
; 3.1. VERB
VER[1] := ver.@entry

; 3.2. NOUN
NOU[1] := nou.@entry
NOU[2] := ppn.@entry
```

Figure 2: Grammar rules examples

4 THE PROPOSED SYSTEM

As stated in section 1, our goal it to execute user requisitions described in restricted natural language. Figure 3 illustrates our approach.

Currently, the input requisitions must be imperative sentences. The input requisitions also have to obey grammar rules and words defined in a dictionary. Both, grammar and dictionary have to be defined according to a specific application domain. HERMETO will use them to convert natural language sentences into UNL.

The UNL sentence is the input for the SeMaComp (Semantic Mapping between UNL relations and Components) module. The SeMaComp goal is to identify what components, methods and arguments will be required to execute the UNL requisition. To achieve this goal, the SeMaComp module uses the Component Ontology (described in session 4.1). Some concepts of this ontology are shared with a Domain Ontology.

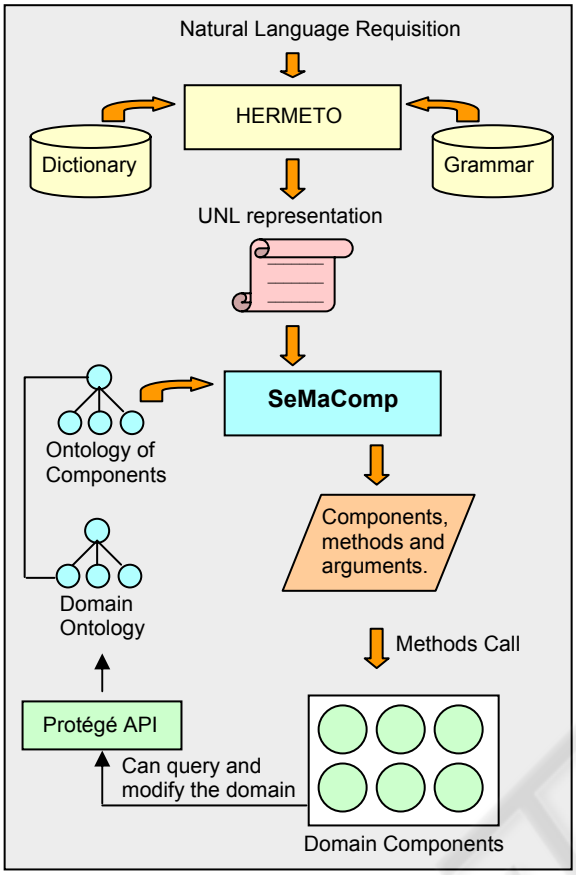

Figure 3: The Proposed System

The application domain software components have to be already installed in the system and ready to be used. These components can make simple queries and modify the Domain Ontology (and its instances) according to user requisitions. They also can perform external actions to the system, such as send e-mail. The Domain Ontology is currently developed using the Protégé tool (Noy et al., 2001), hence the components access it through the Protégé API. This API can be used directly by external applications to access Protégé knowledge bases without running the Protégé tool.

4.1 The Component Ontology

Figure 4 presents the Component Ontology classes and relationships. This ontology has to be instantiated in accordance with the characteristics of the application domain software components.

The instances of the class **OntoDomainConcepts** correspond to concepts of the application domain that are present in the Domain Ontology. The instances of the class **Components** correspond to the components of the application domain that can be related to one or more concepts of the **OntoDomainConcepts** class. The instances of class **Method** correspond to the methods of each software component of the application domain. The instances of class **Params** correspond to the arguments of each method. Finally, the instances of class **Actions** correspond to imperative verbs. Each verb (action) is

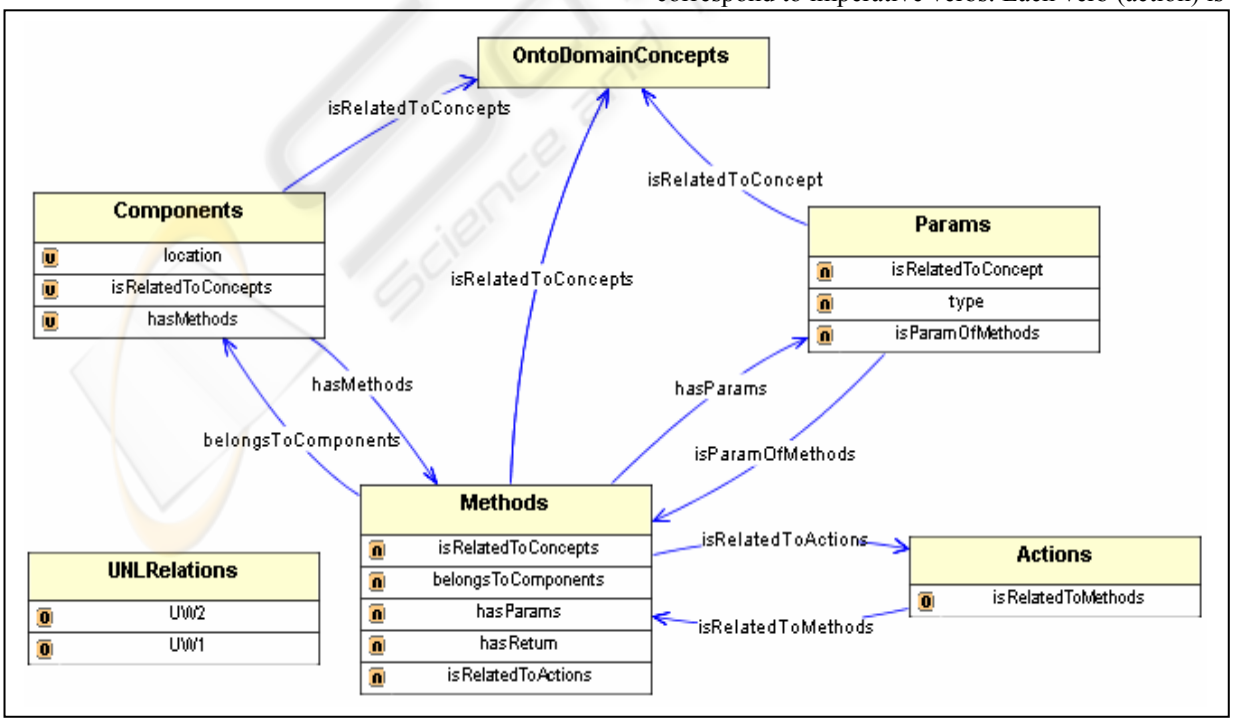

Figure 4: Component Ontology

related to one or more methods, and each method is related to one verb.

The class **UNLRelations** relates UNL relations to information about components. The aim of this class is to indicate the mapping between a particular relation_label(UW1, UW2) and the components, methods, arguments and actions in the Component Ontology. This class contains instances representing all the UNL relations currently being used in the imperative sentences related to the application domain.

Before defining the **UNLRelation** instances, it is necessary to observe what semantic information can be extracted from the UNL relations that are relevant to the UWs-Components mapping.

5 WEB COURSE EXAMPLE

We can demonstrate our approach with a scenario that involves a web course management domain. The software components must be set and ready to use. For this particular domain, we defined a set of components; each one related to a specific concept of the domain. For example, we have a "TeacherComponent" that is responsible for the execution of actions related to the "Teacher" concept. This component has methods to create, delete and list teachers, to assign a specific teacher to a specific course, to update information about a particular teacher, and so on. In a similar way, we have components related to the concepts "User", "Student", "Candidate", "Course", "Class", "Monitor" and "Administrator". After the development of these components, we defined a Domain Ontology, with relationships between the concepts of the underlined domain.

Figure 5 shows some examples of natural language imperative sentences (requisitions) that can serve as input to the system.

The requisitions must obey the grammar rules and dictionary entries created for the domain. Figures 1 and 2 showed a small part of the dictionary and rules created for this domain. The requisition will be converted into UNL (using HERMETO). For example, consider the following requisition:

"Delete administrator Mary from course Java." (c)

HERMETO will generate the UNL representation showed on Figure 6.

- Create course Operating System.
- Delete course Java.
- Add student John Smith to the class xxx.
- List classes of teacher Susan.
- Delete administrator Mary from the course Java.
- Update course Java candidate name from Mary Smith to Maria Smith.

Figure 5: Imperative Sentences Examples

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

Figure 6: UNL representation generated by HERMETO for the sentence on (c)

The UNL relations, shown in Figure 6, will serve as input to the SeMaComp module. It will use the Component Ontology to detect which components methods and arguments should be used to execute the requisition. The Component Ontology should have been previously instantiated according to the characteristics of the domain software components.

Special attention should be given to the class **UNLRelations**. Before defining this class, in the Component Ontology, it is necessary to observe what semantic information can be extracted from the UNL relations, present on the application domain, that is relevant to the UWs-Components mapping. Figure 7 illustrates the UNL relations present in the imperative sentences of the domain as instances of the class **UNLRelations**. This class instances and their relationships indicate which information can be extracted from the UNL representation of the sentence to help finding the most suitable components, methods and arguments to execute the requisition. This class should state if a UW, related to a particular relation label, corresponds to a component, an action, an argument type, an argument value or a return value.

As shown in Figure 7, UW1 of **obj** relation is

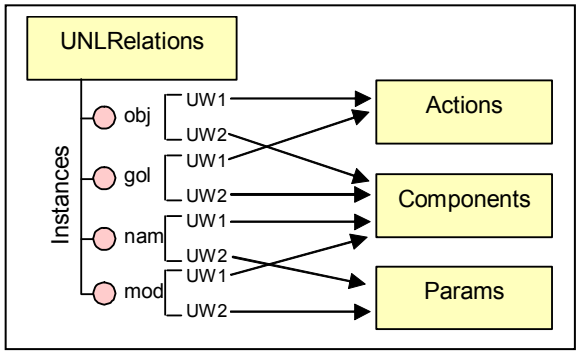

Figure 7: UNL-Components Mapping

always related to an Action (it means that the value of UW1 can be any instance of class Actions in the Component Ontology). Similarly, UW2 is related to a Concept – that should have one or more software components related to it and have to be present at the Domain Ontology.

SeMaComp separates the tokens of the UNL sentence and classify them using the Component Ontology. For the UNL representation of the requisition in Figure 6, SeMaComp will identify the relevant information shown in Figure 8.

```
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
```

Figure 8: Relevant information extracted from the UNL sentence of Figure 6

With this information, SeMaComp searches the Component Ontology to discover which methods are related to the action "delete" and belong to one of the components associated to the "administrator" concept. This search returns the methods `deleteAdmin` and `deleteAdminCourse`. Still in the Component Ontology, SeMaComp retrieves data about the number of arguments, argument types and return type of each identified method. These information are used to analyze the available methods and to conclude which one is the most suitable to execute the requisition.

6 CONCLUSIONS AND FUTURE WORK

This paper described a new approach to the execution of natural language requisitions. This approach proposes a semantic mapping between UNL relations and software components. UNL is an interlingua, the advantage of using an interlingua to describe user requisitions is that it can represent requests derived from different languages (English, Spanish, Portuguese, French, etc.).

In our system, natural language requests can be translated into UNL and the SeMaComp module can perform the semantic mapping between those UNL requests and software components and activate methods in these components to fulfill the requests.

The semantic mapping 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 HERMETO), create instances of the Component Ontology and define the Domain Ontology.

The semantic mapping between UNL relations and software components currently performed is limited to the information given by the user in the natural language requisition. As future work, we intend to extend the Component Ontology to support context information. Another future work is to perform the semantic mapping between UNL relations and software components using not only imperative sentence structures, but also interrogative and conditional sentence structures.

Our ultimate goal is to provide a restricted natural language interface that could be used by other systems to allow computer-based actions to be described in several natural languages.

REFERENCES

Ballard, B. A.; Bierman A. W., 1979. Programming in Natural Language: NLC as a Prototype. In: *ACM SCS'79, ACM Annual Computer Science Conference*. Proceedings. ACM Press. pp. 228-237.

Cheyer, A.; Martin, D., 2001. The Open Agent Architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, v.4, n.1, pp.143-148.

Martins, R. T.; Hasegawa, R.; Nunes, M. G. V., 2004. HERMETO: A NL Analysis Environment. In: *TIL'04, 2nd Workshop da Tecnologia da Informação e da Linguagem Humana*. Proceedings. Brazil, pp. 64-71.

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

Price, D.; Riloff, E.; Zachary, J.; Harvey, B., 2000. NaturalJava: A Natural Language Interface for Programming in Java. In: *5th ACM International Conference on Intelligent User Interfaces*. Proceedings. ACM Press. pp. 207-211.

Tsai, T et al., 2003. Ontology-Mediated Integration of Intranet Web Services. *IEEE Computer*. v. 36, n. 10, pp. 63-71.

UNL Center, 2003. The Universal Networking Language (UNL) Specifications. Version 3, edition 2.

Ushida H.; Zhu M., 2001. The Universal Networking Language beyond Machine Translation. In: *International Symposium on Language and Cyberspace*, Seoul (South Korea).