UCL—Universal Communication Language

Carlos A. Estombelo-Montesco and Dilvan A. Moreira

Universidade de São Paulo,
Instituto de Ciências Matemáticas e de Computação
Av. do Trabalhador São-Carlense, 400 - Centro - Cx. Postal 668
São Carlos - SP - Brazil CEP 13560-970
`c_estombelo@yahoo.com`, `dilvan@computer.org`

Abstract. For successful cooperation to occur between agents they have to be able to communicate among themselves. To enable this communication an Agent Communication Language (ACL) is required. Messages coded in an ACL should adequately express their meaning from a semantic point of view. The Universal Communication Language (UCL) can fulfill the role of an ACL and, at the same time, be convertible to and from a natural language. UCL design is concerned with the description of message structures, their underlining semantic context and the support for protocols for agent interaction. The key point about UCL is that the language can be used not only for communication among software agents but among humans too. This is possible because UCL is derived from the Universal Network Language (UNL), a language created to allow communication among people using different languages. UCL was defined using the Extended Markup Language (XML) to make it easier to integrate into the Internet. In addition, an enconverter-deconverter software prototype was written to serve as a tool for testing and experimenting with the language specifications.

1 Introduction

The technology of software agents can be an interesting tool for the creation of new models for complex software systems. In the project of software agents, many of the traditional techniques of artificial intelligence can be mixed with techniques from the field of distributed computer systems, theories about negotiation and theories about working teams [2]. Software agents are basically designed to cooperate (either with others or with humans) in a seemingly intelligent way. But for cooperation to occur a communication language is necessary.

What does it mean to be able to communicate with someone? Simplifying it, useful communication requires shared knowledge. While this includes knowledge of language, words and syntactic structures, meaningful communication is even more focused on knowledge about a problem to be solved. To interact with a florist you need some knowledge about flowers.

The widespread use of the Word Wide Web (WWW) and growing Internet facilities have sparked enormous interest in improving the way people communicate using computers. To date, communication among software agents and humans has been done under limited conditions: communication is reduced to basic information exchange, ignoring the richness and flexibility implied by human language.

However to deal with any human language would be very difficult. To solve this problem, communication systems can use an Agent Communication Language (ACL) based in a simplified form of human language, which could be converted from and to natural language.

2 Objectives

The main objective of this work is the specification of a new ACL, called UCL—Universal Communication Language, that focus on the specification of the semantic model and structure of the messages it represents. It also adds support for message transmission over the Internet and can be translated into or generated from natural language (English or other language).

UCL is derived from the Universal Network Language (UNL) [6] and implemented using the language XML (Extensible Markup Language) [1]. XML is a W3C (World Wide Web Consortium) standard language, like HTML, this means an easy integration with the Internet.

Another goal of this paper is to show a working UCL enconverter-deconverter prototype using the tool *Thought Treasure* and its associated ontology.

3 Communication Among Agents

In the communication process among agents, it is indispensable an appropriate understanding of what will be communicated through the exchange of messages. A good representation of the knowledge domain, shared by the agents, can collaborate for a better understanding of the context where a message exchange takes place. As a consequence, it is important to explore concept classifications and their hierarchical structures for knowledge domain representation. The concepts in the knowledge domain have to be shared by the agents exchanging messages and be reusable in more than one context.

The specification of an ACL has to deal with the description of the message structure, his semantic model and the interaction protocols [4]:

- The message format defines the communicative acts primitives and the parameters of the message (as sender, receiver, etc.). The message content describes facts, actions, or objects in a content language (KIF, Prolog, etc).
- The semantic model of an ACL should allow for messages with a concise meaning and no ambiguity.
- The interaction protocols are projected to facilitate the communication among agents. Protocols are optional, but, in case they are used, the communication among agents should be consistent with the chosen protocols.

3.1 Ontologies for Communication

’Ontology’ is a term used to refer to the common sense of some domain of interest. The ontology can be used as a uniform framework to solve communication problems.

An ontology necessarily links or includes some type of "general vision" regarding a certain domain. This "general vision" is frequently conceived as a group of concepts (for example: entities, attributes, processes), their definitions and their interrelations. That is called a conceptualization.

A conceptualization can be concretely implemented, for example, in a software component, or it can remain abstract, being the implied concepts of a person. The use adopted in this work for ontology is that it is an explicit idea, or a representation (of some part) of a conceptualization.

An explicit ontology can take a variety of forms, but necessarily they will include a vocabulary of terms and some specification of their meanings (for example: definitions).

The level of formality for a vocabulary varies considerably. This variation can be shown in the following four points of view:

- Highly informal: expressed freely in natural language.
- Semi-informal: expressed in a restricted form and structure in natural language. Better clarity for ambiguity reduction.
- Semi-formal: expressed in an artificial language defined formally.
- Strictly formal: defined meticulously with formal semantics, theorems and proofs.

A shared ontology is necessary for communication between two agents. Unfortunately UNL does not have a public available ontology. For this reason, the ontology embedded in the tool *Thought Treasure* was used to implement the enconverter-deconverter prototype.

3.2 The Tool *Thought Treasure* (TT)

Thought Treasure (TT) is is a powerful tool for processing natural language, developed by Erik T. Mueller [5]. It is capable of interpreting natural language, as well as extending its ontology-based knowledge base. TT has a compiler for natural language that allows it to extract information of sentences.

TT has a database with 25,000 concepts organized in a hierarchical way. For example, Evian is a flat-water type, which is a drinking-water type, which is a food type and so on.

Each concept has one or more word translations what forms a total of 55,000 words and sentences of the English and French language. For instance, as it is observed in the Fig. 1, the association with the concept food in the English language are the words *food* and *foodstuffs* and in French *aliment* and *nourriture* (among others).

In addition, TT has approximately 50,000 assertions related to concepts such as: a green-pea is a seed-vegetable, a green-pea is green, the green-pea is part of pod-of-peas, and pod-of-peas is found usually at a store of foodstuffs.

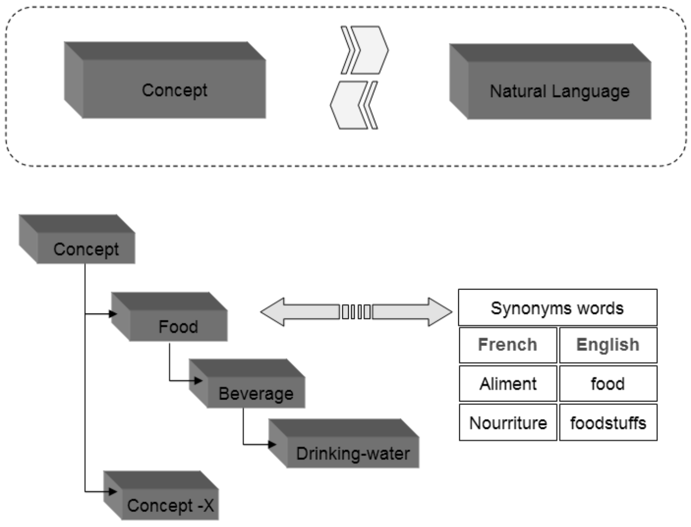

Fig. 1. Association of the ontology with a natural language.

4 UCL—Universal Communication Language

The language UCL represents information in the same way UNL does, but using syntax based in XML. XML is a meta-language, a simplified form of SGML, which developers can use to create new languages based in tag elements. The new tags, created to represent the new language elements, can be described in a special file called DTD (Document Type Definition). UNL is a formal language for representing the meaning of natural language sentences and exchange information over a network. Information that is written in a native natural language is "enconverted" into UNL and stored in a server. This information can be "deconverted" into other languages to be read by each native reader. Thus, UNL can play the role of an interface between different human languages to exchange information.

UNL represents information expressed in sentences as a set of relations between meanings, expressed by words, and a syntactic structure that makes up the sentence. The vocabulary of UNL consists of:

– Universal Words (UWs), to represent word meaning.
– Relation Labels, to represent relationships between UWs
– Attribute Labels, to express further definitions or additional information for the UWs that appear in a sentence.

In UNL, the information about a sentence includes its meaning, tense and aspect information (how the speaker grasp the event), intention of utterance, speaker's feeling or judgment upon contents, and sentence structure. In the language, the meaning of a sentence is represented by the description of the relationships between UWs and its structure is described by attaching attribute labels to these UWs.

4.1 UCL Goals

The language UCL is to be used for high-level communication among agents through the exchange of messages. Some characteristics that guided the definition of the language were:

– To aid the communication involving agents giving importance to the semantics of the message;
– To be easy to use;
– To facilitate its integration into the Internet environment writing it in XML.

The language UCL represents the information in sentences (that can form messages) that involves a syntactic structure with a group of concepts, relationships and attributes similar to UNL:

– Universal Words (UW),
– Relationship labels,
– Attribute labels.

To define a language based in XML a specific DTD file is used. This DTD is essentially a free context grammar, like the extended BNF form (*Backus Naur Form*) used to describe computer languages [3].

As in UNL, a Universal Word (UW) is the minimum unit that represents a concept, which denotes a specific meaning in a message. When a concept needs to be defined in more detail Relationship Labels and Attribute Labels are used. In addition, UCL uses a shared ontology, from the TT tool, to add meaning to the UWs. All agents participating in a communication process should share this ontology.

In a UCL sentence, each defined UW has an identifier label (id) that is used to identify a particular concept inside a sentence. A sequence of alphanumeric characters forms this labels. The label head corresponds to the place where the name of the concept will be defined. The concepts used are always related to the ontology being used (TT ontology). It is at this point that a sentence in UCL is connected to the ontology for a specify knowledge domain.

In UCL, messages possess a certain meaning involving concepts. This composition of concepts is represented by groups of binary relationships, which allow different relationships involving the concepts. The relationship labels used come from UNL. Figure 2 shows an English sentence and its translation to UCL.

5 Enconverter-Deconverter Implementation

UCL is defined in the meta-language XML, to work with it a XML parser should be used. As the enconverter-deconverter is written in the language Java, the Java

Sentence: UNL is a common language that would be used for network communications.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sentence SYSTEM "Sentence.dtd">
    <sentence>
        <uw id="uw00" head="language">
            <icl direction="to"> <uw head="abstract thing"/> </icl>
            <tense attribute="present"/>
            <focus attribute="entry"/>
        </uw>
        <uw id="uw01" head="UNL">
            <icl direction="to"> <uw head="language"/> </icl>
            <focus attribute="topic"/>
        </uw>
        <uw id="uw02" head="common">
            <aoj direction="to"> <uw head="thing"/> </aoj>
        </uw>
        <uw id="uw03" head="use">
            <icl direction="to"> <uw head="do"/> </icl>
            <tense attribute="present"/>
        </uw>
        <uw id="uw04" head="language">
            <icl direction="to"> <uw head="abstract thing"/> </icl>
            <tense attribute="present"/>
            <focus attribute="entry"/>
        </uw>
        <uw id="uw05" head="communication">
            <icl direction="to"> <uw head="action"/> </icl>
            <convention attribute="pl"/>
        </uw>
        <uw id="uw06" head="network">
            <icl direction="to"> <uw head="thing"/> </icl>
        </uw>
        <relation label="aoj" uw-id1="uw00" uw-id2="uw01"/>
        <relation label="mod" uw-id1="uw00" uw-id2="uw02"/>
        <relation label="obj" uw-id1="uw03" uw-id2="uw04"/>
        <relation label="pur" uw-id1="uw03" uw-id2="uw05"/>
        <relation label="mod" uw-id1="uw05" uw-id2="uw06"/>
    </sentence>
```

Fig. 2. Definition of a UNL sentence.

API for XML Processing (JAXP) Version 1.1 from Sun, was used (other Java XML parsers could have been used).

As said before, UCL uses the ontology available on the TT tool (written in C). This tool includes program libraries to manipulate concepts of the ontology, to do consultations on the concepts network, and to analyze their hierarchy. An instance of TT can run as a server in a network and communicate with a Java program running in another process. A Java communication API is supplied with TT to handle the low level details of this communication.

The enconverter-deconverter prototype uses the Java communication API to contact a running instance of TT and use its functionality. Those include natural language treatment, ontology queries, etc. A high level Java interface was written to communicate with the TT server (through the API) and implement the high level functions needed by the prototype. This interface is called *UclLanguage*.

Figure 3 presents a diagram with the sequence of events that happens when the prototype makes use of the interface *UclLanguage* to generate UCL messages.

The process begins when a user enters a natural language sentence into the prototype. The prototype calls the method *understood* of the interface *UclLanguage*. The natural language sentence is interpreted (using TT) and some possible semantic interpretations are returned. The user chooses the most appropriate interpretation. The chosen interpretation is converted to TT format (method *takeAttofConcept*) and then to UCL format (method *convertTTtoUCLwrite*). The UCL format can be shown on the screen or saved in a file.

The reverse process, to transform a UCL message in natural language is easier. The prototype uses the method *deconvertUCLtoTT* to convert the UCL message in a list of TT concepts. Then it uses the method *deconverterTTtoLN* to transform this list of concepts in a natural language sentence, which represents the original UCL message. Figure 4 shows the prototype converting a sentence to UCL and then back to English (and French).

Figure 5 illustrates the use of UCL (using one TT server) in the communication process between two software agents.

6 Conclusions

The definition of the Universal Communication Language (UCL) includes all theoretical concepts of the Universal Networking Language (UNL). This was done to preserve the representative power of this language. The Web community currently regards XML as an important step toward semantic integration. Developing the language UCL using XML yielded some important benefits. The most important is the reuse of existing tools for creating, transforming, and parsing UCL documents.

The UCL enconverter-deconverter prototype shows the need for a shared ontology for the implementation of a successful enconverter-deconverter. UCL was developed to be used as a rich Agent Communication Language (ACL), which would make it easier for humans to communicate with and program software

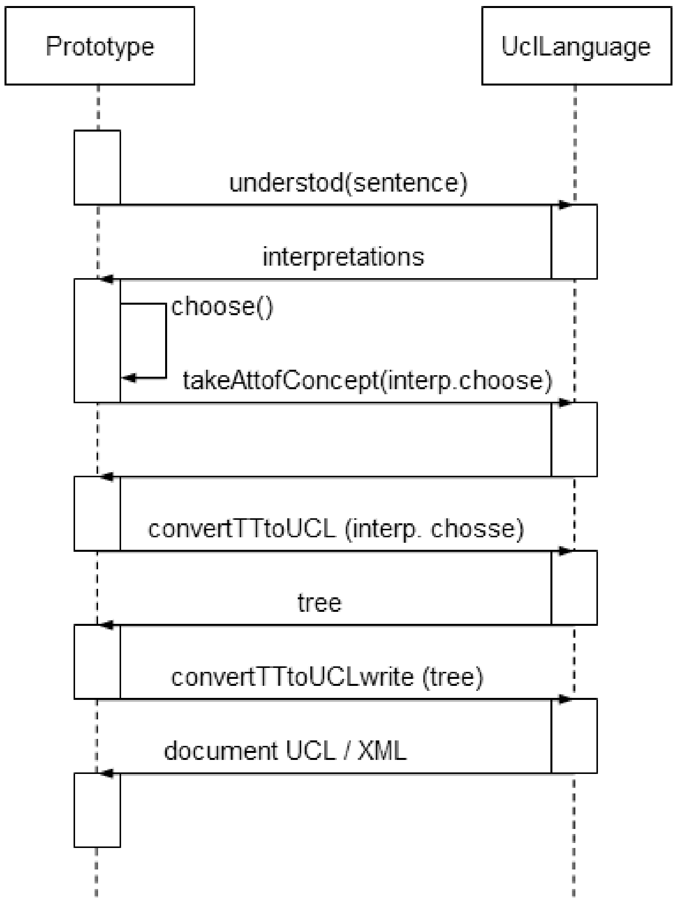
Prototype
UclLanguage
understod(sentence)
interpretations
choose()
takeAttofConcept(interp.choose)
convertTTtoUCL (interp. chosse)
tree
convertTTtoUCLwrite (tree)
document UCL / XML

Fig. 3. Diagram with the sequence of events during enconvertion.

Example: Monkey eats bananas

```
======= Input Natural Language ==========
Example: Monkey eats bananas.

============ Choose Option ==============
<0>An ape eats a banana.

Option: 0
============ Message UCL  ===============
<?xml version="1.0" encoding="UTF-8"?>
<sentence>
    <uw id="uw2" head="present-indicative">
        <icl direction="to">
            <uw head="present-tense" />
        </icl>
        <focus attribute="entry" />
    </uw>
    <uw id="uw4" head="eat">
        <icl direction="to">
            <uw head="ingest" />
        </icl>
    </uw>
    <uw id="uw5" head="ape">
        <icl direction="to">
            <uw head="mammal" />
        </icl>
    </uw>
    <uw id="uw7" head="banana">
        <icl direction="to">
            <uw head="fruit-tropical" />
        </icl>
    </uw>
    <relation id="uw1" label="icl" id1="uw2" id2="uw6" />
    <relation id="uw6" label="icl" id1="uw3" id2="uw7" />
    <relation id="uw3" label="agt" id1="uw4" id2="uw5" />
</sentence>

======== Deconverter Message UCL  ===========
=>Debug : [present-indicative [eat ape banana ]]

English: An ape eats a banana.
French : Un singe croque la banane.
```

Fig. 4. Prototype converting a sentence to UCL and back to English.

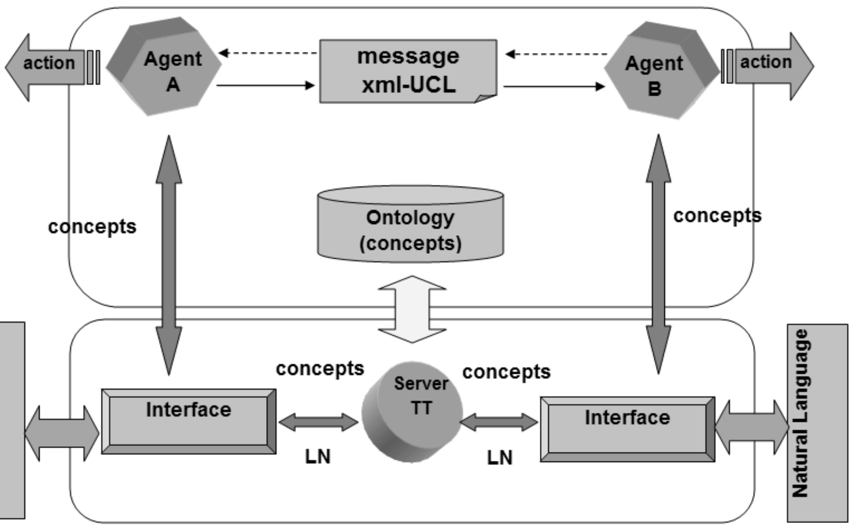

Fig. 5. Architecture of a system that uses the UCL language.

agents (using multiple natural languages). But UCL can be used in the same role as UNL.

The prototype also points out the need for an open shared ontology for UNL. UNL relation and attribute labels have some ontological knowledge already embedded in them. This makes impossible to map all possible UNL (and consequently UCL) constructs into *Thought Treasure* (TT) ontology based representation. The prototype cannot be expanded into a full featured UCL enconverter-deconverter. For the time being this prototype is good enough to help the development of a prototype UCL interpreter for software agents.

The full power, of the approaching of using UCL as an ACL and programming tool for software agents, will only be realized when an open shared ontology for UNL and enconverters-deconverters, for many natural languages (using this shared ontology), are available. One will be able to program a software agent using his own native language and share this program with many other people, which will see and interact with the program in their own native languages.

Finally, UCL is still a proposal, but we hope that others in the Web community will help to shape its final format.

7 Acknowledgements

The authors would like to thank the CNPq—Brazilian National Council for Research for the financial support for this work.

References

1. Connolly, D.: *Extensible Markup Language (XML).* February (2000). Available on-line: http://www.w3.org/XML/
2. Dignum, F., Greaves, M. (ed.): *Issues in Agent Communication.* (Lecture notes in computer science; Vol 1916: Lecture notes in artificial intelligence) Berlin: Springer, (2000).
3. Grosof, B. N.; Labrou, Y.: *An Approach to using XML and a Rule-based Content Language with an Agent Communication Language.* IBM Research Report. RC 21491 (96965), 28 May (1999). Available on-line: http://www.research.ibm.com
4. Mamadou, T. K., Shimazu, A., Tatsuo, N.: *The State of the Art in Agent Communication Languages.* Japan Advanced Institute of Science and Technology, Japan, (2000).
5. Mueller, E.T.: *Natural Language processing with ThoughtTreasure.* New York: Signiform (1998). Also available on-line: http://www.signiform.com/tt/book/
6. Ushida, H., Zhu, M., Senta, T.D.: *The UNL a Gift for a Millennium.* UNU/IAS, ISBN:4-906686-06-0, November (1999). Also available on-line: http://www.unl.ias.unu.edu/publications/index.htm