Uma Linguagem de comunicação para agentes na Internet baseada em Ontologias

Carlos A. Estombelo Montesco, Dilvan de Abreu Moreira

Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação,
Av. do Trabalhador São-Carlense, 400 - Centro - Cx. Postal 668 São Carlos - São Paulo - Brasil CEP 13560-970
{cestombe, dilvan}@icmc.sc.usp.br

Abstract. To attain a level of behavior comparable to humans, the Behavior Based Model has to be augmented with the ability of handling language. Language is very important to human intelligence. It shapes the human brain (many parts of the brain are specialized in processing different aspects of language). To deal with any human language would be very difficult. To solve this problem we propose a simplified form of a human language, based upon the UNL (Universal Network Language). Unfortunately UNL is not an open standard. For this reason we have been forced to develop a new language based on the concepts of UNL, but implemented in XML (Extensible Markup Language), called Universal Communication Language (UCL). This language unites the simplicity of XML to the power of UNL and it is also Web ready.

1. Introdução.

A tecnologia de agentes de software desperta o interesse como ferramenta para criação de um novo modelo para sistemas de software complexos. No projeto de agentes, misturam-se muitas das propriedades tradicionais de inteligência artificial (como "raciocínio" no nível de conhecimento, flexibilidade, etc) com as experiências obtidas nas áreas de sistemas distribuídos, teorias sobre negociação e teorias de equipe de trabalho, como também das ciências sociais (Dignum, 2000).

Para que a cooperação entre agentes de software tenha sucesso, é requerida comunicação entre eles. Uma coleção de agentes trabalhando juntos em cooperação pode ser vista como uma pequena sociedade, e o funcionamento de qualquer sociedade coerente precisa de uma linguagem comum e um meio de comunicação. Agentes que trabalham de forma isolada provavelmente serão menos eficazes do que aqueles que são capazes de interagir. (David, 1999).

É tão vasto o campo da comunicação que se torna difícil encaixá-lo numa definição exata. Em geral, a comunicação é o intercâmbio intencional de informação que se dá mediante a emissão e percepção de sinais que pertencem a um sistema convencionado.

Para que estes atos de comunicação e a cooperação entre agentes sejam possíveis, precisa-se de uma Linguagem de Comunicação entre Agentes (em inglês: *Agent Communication Language*, ou ACL). Dentro de uma ACL, torna-se importante a forma como as mensagens são comunicadas, isto é, se as mensagens expressam adequadamente seu propósito sob um ponto de vista semântico. Na continuação, será mostrado um breve resumo de trabalhos relacionados a este assunto.

Embora existam algumas iniciativas e alguns trabalhos que foram feitos em relação às ACLs, de forma geral, percebe-se a falta de consenso nos fundamentos para a comunicação entre agentes de software. Não há um entendimento semântico claro das mensagens a serem transmitidas ou até mesmo os conceitos básicos que deveriam ser usados para definir uma semântica (Dignum, 2000).

2. Objetivos

O objetivo deste trabalho concentra-se na especificação de uma nova ACL, chamada UCL – *Universal Communication Language*, que se preocupa com a descrição da estrutura das mensagens, com o modelo semântico e com suporte a protocolos para interação entre agentes (de software ou humanos) usando a Internet.

Na comunicação entre agentes, é imprescindível um entendimento adequado do que vai ser comunicado

através da troca de mensagens. Uma boa representação do domínio de conhecimento pode colaborar para um melhor entendimento do contexto em que a troca de mensagens acontece. Como conseqüência, é importante explorar as tentativas de classificar e estruturar conceitos e suas relações dentro dum domínio, focalizando-se no compartilhamento e reuso destes conceitos. Além disso, é importante explorar, no contexto deste trabalho, a utilização do padrão XML (*Extensible Markup Language*), para atribuir à linguagem UCL meios para uma fácil integração à Internet. Por isso a linguagem UCL foi implementada no padrão XML.

Neste artigo serão mostrados os conceitos de agentes, destacando-se algumas das definições encontradas na literatura e das propriedades dos agentes; um meio para representar um domínio de conhecimento; a descrição de várias tecnologias úteis que auxiliam na especificação e implementação da linguagem UCL; e a especificação da linguagem UCL. Incluem-se ainda considerações sobre a linguagem e suas características e sobre a metodologia que orientou o seu desenvolvimento.

Está presente também uma breve descrição da implementação da linguagem UCL, tomando como base a especificação mostrada, e finalmente as conclusões deste trabalho, considerando as decisões de projeto, contribuições do trabalho e sugestões para pesquisas futuras.

2. COMUNICAÇÃO ENTRE AGENTES DE SOFTWARE

2.1. Agentes de Software

O termo "agente" existe na linguagem dos computadores desde a década de 80, porém a definição do termo é tão complexa para a comunidade científica quanto a definição do termo "inteligência artificial". Dessa forma, o que tem ocorrido é que cada autor adota uma definição do termo "agente" que melhor se adapte ao seu contexto de trabalho.

2.1.1. Definição de um Agente

Iniciando pela definição encontrada nos dicionários, agente é aquele que opera; é aquele que é encarregado dos negócios de terceiros. Tais definições consideram duas perspectivas diferentes: a primeira que associa um agente a uma entidade que é capaz de agir; e a segunda em que um agente é considerado como um ajudante, ou alguém que atua por intermédio de outra pessoa.

No contexto da ciência da computação, um agente de software pode ser utilizado para facilitar a criação de software capaz de interoperar, ou seja, trocar informações e serviços com outros programas e, dessa forma resolver problemas complexos (Moreira & Walczowski, 1997).

O termo "agente" não tem uma definição consensual, talvez por cobrir diversas áreas de investigação e desenvolvimento. As várias definições de agentes fornecem uma lista de atributos para eles, o que não significa que todos os agentes tenham que, necessariamente, ter todos os atributos, os quais dependem do tipo de aplicação que está sendo desenvolvida (Gouveia et al., 1998). Segundo (Franklin & Graesser, 1996). As características encontradas na maioria dos agentes são:

- Autonomia: um agente será tão mais autônomo, quanto mais controle tiver sobre as suas ações. Um agente pode ser considerado autônomo em relação ao ambiente ou em relação a outros agentes.

- Pró-atividade: um agente pró-ativo toma a iniciativa para atingir os seus objetivos, não se limitando a responder a estímulos do ambiente.

- Reatividade: um agente tem capacidade de reagir às mudanças que sente no ambiente (estímulos).

- Continuidade Temporal: um agente está continuamente ativo. Nota-se que grande parte do software existente não tem essa

característica, já que esses programas executam uma ou mais tarefas e terminam.

- Capacidade Social: se um agente tem capacidade social, então ele pode se comunicar com outros agentes, o que poderá incluir humanos. Dessa comunicação poderá resultar uma cooperação. Para o caso específico da comunicação entre o agente de software e o usuário humano deverá ocorrer uma cooperação na construção do "contrato" sobre o que o agente deverá fazer e não uma simples ordem.

- Capacidade de Adaptação: um agente com capacidade de adaptação é capaz de alterar seu comportamento com base na experiência. Esse tipo de agente é o chamado "agente inteligente", pois possui a capacidade de aprendizagem. A adaptação pode ser relativa ao ambiente ou no sentido de melhorar a sua interação com outros agentes.

- Mobilidade: corresponde à capacidade do agente de se mover dentro do ambiente. Quando se trata de agentes de software, um agente móvel é aquele capaz de se transportar de uma máquina para outra durante a sua execução.

- Flexibilidade: um agente com flexibilidade é aquele que não executa ações pré-definidas em roteiros. Ou seja, possui a capacidade de escolher dinamicamente seqüências de ações em resposta a um estado do ambiente.

Para (Franklin & Graesser, 1996), autonomia, pró-atividade, reatividade e continuidade temporal são características essenciais em um agente.

2.2. Especificações de uma linguagem de comunicação para agentes

A especificação de uma linguagem preocupa-se com a descrição da estrutura da mensagem, seu modelo semântico e os protocolos de interação em que se apóia (Mamadou, 2000):

- O formato da mensagem define os atos de comunicação (communicative acts) primitivos e os parâmetros da mensagem (como sender, receiver, etc.). O conteúdo da mensagem descreve fatos, ações, ou objetos em uma linguagem de conteúdo (KIF, Prolog, etc). Outros parâmetros podem cuidar do significado da mensagem (ontologia) e sua entrega.

- O modelo semântico de uma ACL estabelece os fundamentos para obter um significado conciso e não ambíguo das mensagens do agente e dos protocolos de interação.

- Os protocolos de interação são conjuntos de padrões bem definidos projetados para facilitar a comunicação entre agentes. Protocolos são opcionais, mas, caso sejam usados, a comunicação entre agentes deve ser consistente com o protocolo escolhido.

3. ONTOLOGIAS PARA AGENTES

3.1. Motivação para o surgimento das Ontologias

Pessoas, organizações e sistemas de software devem ser capazes de se comunicar. Porém, devido às necessidades e aos contextos diferentes. Podem-se notar vários pontos de vista e suposições relacionados ao mesmo assunto. A existência de conjuntos de palavras específicas para cada contexto ou assunto de uma área permite observar discrepâncias, justaposições e/ou conceitos mal compreendidos, em estruturas e métodos. Como conseqüência, surge a falta de um entendimento comum. Isto conduz a:

- Uma comunicação pobre, dentro e entre estas pessoas e/ou organizações.

- Dificuldades na identificação de requisitos e deste modo na definição da especificação do sistema.

- Restringem a interoperabilidade;

- Restringem o potencial para reutilizar e compartilhar material.

O segundo ponto se refere ao contexto de construção de sistemas de tecnologia de informação (TI) e sua falta de entendimento comum. O terceiro e quarto ponto se referem as discrepâncias no modelamento dos métodos, paradigmas, linguagens e ferramentas de software.

Um caminho para resolver estes problemas é reduzir ou eliminar as confusões conceituais e terminológicas e conseguir uma compreensão comum. Esta compreensão pode funcionar como um *framework* unificador para os diferentes pontos de vista e servir de base para usos de ontologias (ver **Figura 3.1**):

- **Comunicação** entre pessoas com diferentes necessidades e pontos de vista que surgem em seus diferentes contextos.

- **Inter-Operabilidade** entre sistemas que interagem com diferentes métodos de modelamento, paradigmas, linguagens e ferramentas de software.

- **Benefícios na Engenharia de Sistemas:** em particular, a) Esta representação formal pode ordenar uma re-utilização de componentes compartilhados em um sistema de software, b) uma representação formal também torna possível uma automatização na verificação de resultados em um software confiável, c) a compreensão compartilhada pode ajudar no processo de identificação de requisitos e definição de uma especificação para um sistema de tecnologia da informação.

Alguns exemplos práticos, na unificação de áreas de pesquisa, na fabricação de semicondutores, e na missão de operações de astronaves, são descritos em (Ushold & Gruninger, 1996) como uma amostra das utilidades das ontologias.

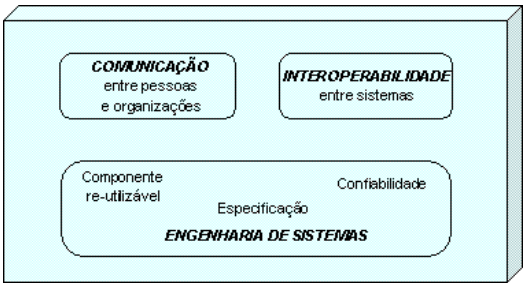

Figura 3.1 Usos de ontologias.

3.2. Definição de Ontologia

'Ontologia' é um termo usado para se referir ao senso comum de algum domínio de interesse. A ontologia pode ser usada como um *framework* unificador para resolver os problemas apresentados anteriormente.

Uma ontologia necessariamente vincula ou inclui algum tipo de "visão geral" referente a um domínio determinado. Esta "visão geral" é freqüentemente concebida como um conjunto de conceitos (i.e. entidades, atributos, processos), suas definições e suas inter-relações. Isto é chamada uma conceitualização (*conceptualisation*).

Uma conceitualização pode estar concretamente implementada, como por exemplo, em um componente de software, ou pode ser abstrata, estando os conceitos subentendidos por uma pessoa. A palavra 'Ontologia' é algumas vezes usada para chamar isto de uma conceitualização implícita. Porém, o uso mais padronizado e que será adotado neste documento é que ontologia é uma idéia explícita, ou uma representação (de alguma parte) de uma conceitualização.

3.2.1. Características de uma ontologia

Uma ontologia explícita pode tomar uma variedade de formas, mas necessariamente esta incluirá um vocabulário de termos e alguma especificação dos seus significados (i.e. definições).

Os graus de formalidade pelos quais um vocabulário é criado e especificado varia consideravelmente. Esta variação pode-se mostrar nos seguintes quatro pontos de vista:

- Altamente informal: expresso livremente em linguagem natural.

- Semi-informal: expresso em uma forma restrita e estruturada na linguagem natural. Maior clareza pela redução de ambigüidade.

- Semiformal: expressa em uma linguagem artificial definida formalmente.

- Rigorosamente formal: termos meticulosamente definidos com uma semântica formal, teoremas e provas de tais propriedades como completitude e boa qualidade.

4. RECURSOS PARA A COMUNICAÇÃO ENVOLVENDO AGENTES DE SOFTWARE NA INTERNET

4.1 A linguagem UNL – Universal Networking Language

A Universidade das Nações Unidas (UNU), sediada em Tóquio, resolveu patrocinar um projeto de longa duração – 10 anos – para o desenvolvimento de ferramentas de software, como representação intermediaria de mensagens, para vencer a barreira da língua para a comunicação entre os povos. Nesse projeto, ao invés da tradução de uma língua para outra se faz a codificação do conteúdo de um texto em uma dada língua natural para uma artificial, chamada *Universal Networking Language* (UNL), criada especificamente para textos escritos para o ambiente da Internet. O texto já codificado em UNL pode então ser decodificado para a língua destino. O Projeto UNL busca uma integração comunicativa em um ambiente de processamento automático de linguagem natural (PALN) integrado em rede, que permitirá que usuários de qualquer parte do mundo possam se comunicar sem que, para isso, tenham que aprender uma linguagem especial de comunicação.

4.1.1 O léxico segundo a perspectiva da UNL

Os componentes do léxico incluem conceitos padrões ou *Universal Words* (UWs) e um conjunto de relações conceituais e atributos que pode ser expresso estruturalmente, em termos de relações sentenciais. Essas relações são rotuladas adequadamente, por meio de "rótulos de relações" (*Relation Labels* - RLs) e "rótulos de atributos" (*Attribute Labels* - ALs). As UWs são baseadas em palavras da língua inglesa e são relacionadas segundo as relações hierárquicas da taxonomia conceitual ou segundo os rótulos de relacionamento sentencial fornecidos pelo usuário especialista. Dessa forma, o léxico forma uma hiper-rede de relações entre UWs que abrange conceitos genéricos e específicos, indicando parte de seu inter-relacionamento semântico.

Assim como se faz uso de RLs para se chegar ao significado pertinente e, logo, a um UW particular, faz-se uso dos ALs para limitar o significado das UWs. Desse modo, o uso dos componentes sentenciais indica a semântica lexical incorporada ao léxico.

4.2. A meta-linguagem XML – Extensible MarkUp Language

A Linguagem XML (Connolly, 2000) surge como uma forma simplificada da SGML, sendo uma alternativa para a solução dos problemas encontrados em HTML e tendo maior flexibilidade para manipular conteúdos propriamente ditos. O objetivo da XML é fornecer muito dos benefícios encontrados em SGML e não disponíveis em HTML. Além disso, facilitar o seu aprendizado e a sua utilização se comparada com a complexa linguagem SGML (Mace et. al., 1998; MCGrath, 1999).

A XML é uma metalinguagem definida como um subconjunto da SGML e oferece uma abordagem padrão para descrição, captura, processamento e publicação de informações.

Em XML, os desenvolvedores podem criar seus próprios elementos de acordo com a aplicação que está sendo modelada, dando importância ao conteúdo e à estrutura da informação, sem se preocupar com a apresentação.

4.3. O Documento que define as regras para XML

Um DTD (*Document Type Definition*) define regras para a especificação de uma classe de documentos, tais como (Trindade, 1997):

- que tipos de elementos podem existir em um documento (por exemplo, Aluno, Professor, Curso);
- que atributos esses elementos podem ter (por exemplo, um Curso é formado pelos elementos Nome_Curso, Professor, Período, Alunos);
- como as instâncias desses elementos estão hierarquicamente relacionadas (como por exemplo, um Curso possui Alunos e cada Aluno possui Identificação_Curso, Nome, Email, Homepage e Notas, que por sua vez podem conter várias Notas e uma Nota final).

A estrutura especificada em um DTD, segundo sua definição no padrão SGML, possui uma propriedade importante: apenas a estrutura lógica de um documento é descrita, não sendo fornecida informação sobre a apresentação dos elementos definidos (Brown, 1989).

Se o documento XML estiver associado a um DTD, o *parser* (Interpretador) deve verificar se o documento XML está correto (*parser* de validação). Para isto, o *parser* processa o DTD que corresponde ao documento XML e obtém sua estrutura. Posteriormente, o *parser* obtém as informações do documento XML realizando a validação com a estrutura definida no DTD. Por outro lado, se o documento XML não está associado a um DTD, então apenas a estrutura sintática do documento XML pode ser verificada (*parser* sem validação).

4.4. A ferramenta *Thought Treasure* (TT)

Esta é uma poderosa ferramenta de processamento de linguagem natural (ferramenta de código aberto), desenvolvida por Erik T. Mueller.(1998). Ela é capaz de interpretar linguagem natural, como também estender sua base de conhecimento que está baseada em ontologias. É um compilador para linguagem natural que permite extrair informação de sentenças ou frases.

TT é conformada por uma base de dados de 25,000 conceitos organizados hierarquicamente. Por exemplo, *Evian* é um tipo de *flat-water*, o qual é um tipo de *drinking-water*, o qual é um tipo de *food* e assim por diante.

Cada conceito tem uma ou mais aproximações sinônimas e que chega a um total 55,000 palavras e frases do idioma inglês e francês. Por exemplo, como se vê na **Figura 4.2**, a associação com o conceito *food* no idioma inglês são as palavras *food* e *foodstuffs* e em francês *aliment* e *nourriture* (entre outras).

ThougTreasure tem aproximadamente 50,000 asserções relacionados a conceitos, assim como: a *green-pea* is a *seed-vegetable*, a *green-pea* is *green* , a *grean-pea* é parte de *pod-of-pea*, *pod-of-peas* é encontrado geralmente em uma loja de comestíveis.

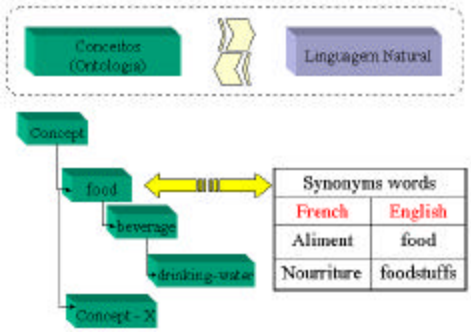

Figura 4.2 Associação da ontologia com a linguagem natural

ThougTreasure tem ao redor de 100 *scripts*, descrevendo atividades típicas, que permitem entender o mundo real.

5. A LINGUAGEM UCL PARA A COMUNICAÇÃO ENVOLVENDO AGENTES

5.1 Abordagem da linguagem UCL

A linguagem UCL visa estabelecer uma comunicação de alto nível envolvendo agentes (de software), tendo em vista que as mensagens a

serem transmitidas representam de forma apropriada o domínio de conhecimento. Algumas características que guiaram a definição da linguagem foram:

- Auxiliar a comunicação envolvendo agentes dando importância à semântica da mensagem;
- Ser de fácil utilização;
- Permitir sua integração à rede Internet seguindo o padrão XML (*Extensible Markup Language*)

Para definir uma linguagem seguindo as características acima, foi considerada a abordagem de uma linguagem baseada em mensagens (comandos) e argumentos. O usuário expressa cada mensagem através de uma estrutura formada pelos conceitos relacionados ao domínio de conhecimento, seguida das relações existentes envolvendo estes conceitos. Além disso, cada conceito contém uma estrutura de atributos que descreve o conceito de forma que se evite a ambigüidade com outros conceitos. A linguagem UCL apresenta a vantagem de permitir uma descrição formal e semântica das mensagens.

Para minimizar os inconvenientes desta abordagem, bem como atender as características listadas anteriormente como sendo as desejadas para a linguagem de comunicação, foram estabelecidos alguns critérios que devem ser seguidos na definição da linguagem:

- As mensagens que são compostas por conceitos, atributos e suas relações devem ser representadas de forma organizada e estruturadas para que sejam de fácil leitura para o usuário.
- Permitir a escalabilidade e extensibilidade da linguagem, de tal forma que se possa agregar novos conceitos relacionados ao domínio de conhecimento.
- Codificar a linguagem em uma meta-linguagem padrão que permita a comunicação entre sistemas heterogêneos e ao mesmo tempo que seja de fácil integração na Internet.
- Indiferença entre letras maiúsculas e minúsculas e minimização de pontuações especiais.

Incorporar estes aspectos à linguagem proposta visa oferecer uma representação e formato mais legíveis. No entanto, a UCL ainda tem o inconveniente de exigir por parte da linguagem uma validação das mensagens a serem transmitidas (análise sintática); e por parte do usuário, uma memorização das regras da linguagem.

5.2 Metodologia utilizada

O desenvolvimento da linguagem pode ser dividido em duas etapas: Definição da Linguagem e Implementação da Linguagem.

a) **Definição da linguagem** Nesta primeira fase procurou-se especificar a linguagem, tendo como ênfase abranger os aspectos conceituais de como representar o significado das mensagens. Para esta representação, utilizou-se a base teórica da linguagem *Universal Networking Language* (UNL). Como foi dito anteriormente, a UNL é uma linguagem de representação semântica baseada em relações envolvendo predicados (conceitos), desta forma, a linguagem UCL proporciona os meios para descrever as mensagens.

b) **Implementação da Linguagem** Utilizando a especificação inicial da linguagem que se baseia nas características e critérios mostradas anteriormente, passou-se a preencher os aspectos gramaticais necessários para a implementação da linguagem. Para isso foi necessário:

- Estabelecer com rigor os critérios sintáticos da linguagem inicialmente definida, o que se

traduziu em: definir palavras reservadas, separadores, estabelecer a gramática da linguagem;

- A utilização da meta-linguagem XML para definir a linguagem proposta. Esta meta-linguagem uniria a simplicidade da XML ao poder representativo da UNL e nos aliviaria de muitas dificuldades de escrever um *parser* (interpretador) para ela.

- Junto com cada documento XML a associação a definição da gramática (regras) em um arquivo DTD (*Document Type Definition*) que representa a estrutura lógica da mensagem envolvendo elementos, atributos e instâncias.

Como foi dito na introdução deste trabalho, a linguagem UNL não foi adotada por não ser um padrão aberto e por não seguir o padrão SGML (ou um subconjunto dele), o que dificulta a sua integração com a rede Internet. Mas os conceitos teóricos de UNL foram usados por seu poder de representar mensagens.

Como resultado desta fase, obteve-se uma definição da linguagem para representar mensagens relacionadas a um determinado domínio de conhecimento.

5.3 Características da linguagem UCL

A linguagem UCL representa a informação em sentenças (também chamadas de mensagens para o caso deste projeto) que envolve uma estrutura sintática que forma a mensagem, e um conjunto de conceitos, relações e de atributos que são definidos a seguir:

- UW (*Univeral Word*), que representa o significado de uma palavra ou conceito.

- Rótulo de relação, que representa uma relação entre *Universal Words*.

- Rótulo de atributo, que contém alguma informação adicional ou definição que se acrescenta à *Universal Word* e que está presente na mensagem.

5.3.1 Considerações gerais sobre a linguagem

a) Para que a linguagem proposta cumpra com a característica relacionada à representação semântica da mensagem, considerou-se a utilização de uma ontologia que define formalmente o domínio de conhecimento.

b) A representação conceitual da mensagem está representada pela ontologia *ThoughtTreasure.*

c) Considerando-se que as mensagens devem evitar ambigüidade no momento de serem representadas, cada *Universal Word,* que visa representar um conceito, é limitada na abrangência de seu significado por meio do uso de rótulos de relação. Desta forma, evita-se que dois conceitos se sobreponham.

d) Definidos os conceitos que pertencem à mensagem, precisa-se relacioná-los utilizando os rótulos de relação para construir a mensagem.

5.3.2 A notação utilizada

Como foi visto anteriormente, a XML é uma meta-linguagem usada para definir outras linguagens. Para definir uma linguagem baseada em XML usa-se um DTD específico. Este DTD é essencialmente uma gramática de livre contexto, tal como, a forma BNF estendida (*Backus Naur Form*) usada para descrever linguagens de computador (Grosof & Labrou, 1999).

Fisicamente a especificação sintática da linguagem será definida em um arquivo denominado *Document Type Definition* que estará associado a um documento XML. A utilização deste padrão especificado pelo *World Wide Web Consortium* (W3C) para a definição da gramática é uma vantagem considerando que a linguagem deve ser

independente de plataforma, como acontece na Internet.

5.4 Especificação da linguagem

5.4.1 UW (Universal Word)

A *Universal Word* é a unidade mínima que representa um conceito, que em conjunto denotam um significado específico em uma mensagem. Sua representação pode ser correspondente a palavras em inglês ou de qualquer outra linguagem ou mesmo por símbolos, sendo que o mais importante é o significado ou conceito que ela envolve.

Quando existe a necessidade de representar um conceito de forma mais exata, a linguagem possui métodos para que o conceito seja restrito no momento da definição de seu significado. Estes métodos referem-se a utilização de rótulos de relação e rótulos de atributo que serão mostradas nos tópicos seguintes. Na **Figura 5.1** é mostrada a estrutura (sintaxe) geral de uma *Universal Word*.

```
<!ELEMENT uw ((%label;)*, (tense | aspect |
                      reference | focus |
                      attitudes | viewpoint
                      | convention)*)>

<!ATTLIST uw

        id ID #IMPLIED

        head CDATA #REQUIRED>
```

Figura 5.1 Sintaxe geral de uma *Universal Word*

Na estrutura mostrada acima, pode se observar que cada `uw` definida deve ter um identificador `id`. Este identificador é formado por uma cadeia de caracteres alfanuméricos e servirá como único identificador do conceito em toda a mensagem. Este identificador pode ser referenciado em qualquer outra parte da mensagem, mas não poderá ser redefinido.

O rótulo `head` corresponde ao lugar onde o conceito será definido. Os conceitos utilizados sempre estão relacionados à ontologia que se esteja usando, é neste ponto que a linguagem UCL se conecta com a ontologia de um domínio especifico de conhecimento.

Na **Figura 5.2** é apresentado um exemplo de documento XML com a definição de conceitos, utilizando a sintaxe da **Figura 5.1**:

```
<uw id="uw02" head="area">
   <icl direction="to">
      <uw head="place"/>
   </icl>
   <reference attribute="indef"/>
</uw>

<uw id="uw03" head="strategic"/>

<uw id="uw04" head="designate">
   <icl direction="to">
      <uw head="do"/>
   </icl>
   <focus attribute="entry"/>
   <viewpoint attribute="may"/>
</uw>

<uw id="uw05" head="read">
   <icl direction="to">
      <uw head="do"/>
   </icl>
</uw>

<uw id="uw06" head="home"/>
```

Figura 5.2 Definição de conceitos em um documento XML

Neste exemplo, pode-se observar a definição de cinco conceitos, cada um rotulado como `uw`. Cada conceito tem um identificador único incluído como um atributo (`id`). Pode-se observar que cada conceito está incluído com o atributo `head`.

O rótulo `label` e seguintes atributos mostrados na estrutura são explicados no tópico seguinte.

5.4.2 Rótulos de relação

A linguagem UCL representa mensagens que possuem um determinado significado envolvendo conceitos. Esta composição de conceitos é representada por um conjunto de relações binárias que permitem distinguir as diferentes relações envolvendo os conceitos.

Em concordância com a teoria da *Universal Networking Language*, para a representação de composição de conceitos, adotaram-se os rótulos de relação definidos nessa linguagem. Estes rótulos estão formados com, no máximo, três caracteres de comprimento.

Um exemplo utilizando rótulos de relação é mostrado na **Figura 5.4** uma mensagem é representada na linguagem UCL, construída a partir da seguinte sentença.

- *UNL is a common language that would be used for network communications John breaks.*

```
<?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>
```

Figura 5.4 Definição de Rótulos de relação em um documento XML

5.4.3 Rótulos de atributo

Os rótulos de atributos foram introduzidos para limitar o significado dos conceitos. Com os rótulos de atributos pode-se particularizar o significado de uma uw.

As representações destes atributos estão agrupadas de forma tal que se possa identificar as informações de tempo verbal, o aspecto do conceito ou estrutura da mensagem, entre outros. Estas podem ser inseridas dentro da abrangência do rótulo uw até conseguir restringir seu significado como desejado.

6. IMPLEMENTAÇÃO DA LINGUAGEM UCL

6.1 Gramática

Para discutir a implementação da linguagem, será necessário situar alguns conceitos manipulados pela implementação.

A linguagem UCL é especificada através de regras que descrevem a estrutura da mensagem. O conjunto destas regras compõe a gramática da linguagem. O reconhecimento de uma mensagem UCL é feito pelo interpretador ou "*parser*" (usando XML) que faz o controle de fluxo dos ítens fornecidos pelo usuário. A seguir, na **Tabela 6.1** são mostradas as principais etiquetas que representam a estrutura de uma mensagem UCL que contem informações sobre a sentença, conceitos (UWs) e relações.

Além das etiquetas mostradas acima, temos as denominadas: rótulo de atributo e rótulo de relação que foram especificados.

Tabela 6.1 Principais etiquetas para a construção de mensagens UCL

Etiqueta	Descrição
<sentence>	Inicio da mensagem
</sentence>	Fim da mensagem.

<uw>	Inicio de um conceito/objeto.
</uw>	Fim de um conceito/objeto.
<relation>	Definição de Inico de uma relação envolvendo dois conceitos previamente definidos.
</relation>	Fim de uma relação

6.2 Implementando a linguagem

A linguagem *Universal Communication Language* é implementada tendo-se em mente características mostradas no item 5.3, e a abordagem da linguagem, do item 5.1. Tais características se resumem em a) permitir que a linguagem seja de fácil integração à rede Internet; b) ser de fácil utilização, isto inclui, que a linguagem seja de fácil leitura para o usuário (legível); c) permitir sua escalabilidade e estensibilidade. Além disso, na comunicação envolvendo agentes tem de se dar importância à semântica da mensagem.

Em conseqüência, na implementação da linguagem, se utilizou a meta-linguagem XML para definir as mensagens. A estrutura destas mensagens está baseada no DTD definido e descrito no item 5. Para validar os documentos XML com o DTD, precisou-se de um interpretador de documentos XML, no caso o *Java API for XML Processing* (JAXP) Version 1.1 da Sun.

Para a representação semântica da linguagem, precisa-se de uma ontologia que represente os conceitos envolvidos na mensagem. Neste ponto, adotou-se a ontologia usada na ferramenta *ThoughtTreasure*. Ela inclui a utilização de bibliotecas que permitem manipular os conceitos da ontologia, realizar consultas na rede de conceitos, e analisar a hierarquia.

Além disso, implementa-se um protótipo que permitirá gerar e manipular as mensagens UCL considerando a utilização das outras ferramentas mostradas no tópico anterior. Na **Figura 6.1** se mostra as classes que foram utilizadas e a interface que foi implementada no protótipo.

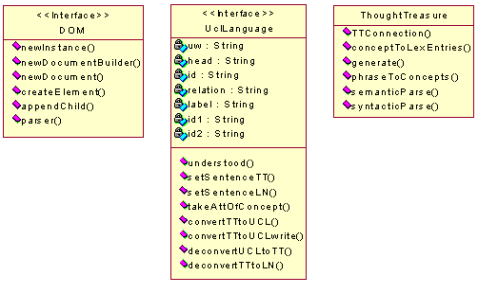

Figura 6.1 Classes e interface do protótipo.

Na **Figura 6.2** é mostrado o diagrama de classes da implementação do protótipo, este diagrama mostra quais são as relações que tem a Interface *UclLanguage* com as Classes e Interfaces.

A **Figura 6.3** apresenta o diagrama de seqüência de eventos que segue o protótipo quando o protótipo faz uso da interface *UclLanguage* para gerar mensagens na linguagem UCL.

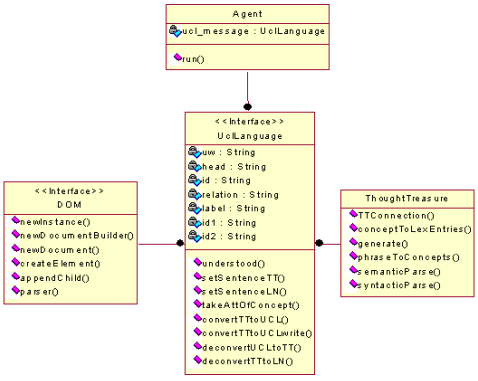

Figura 6.2 Diagrama de Classes.

O processo começa quando o protótipo chama o método *understood* da interface *UclLanguage,* desta forma uma sentença em linguagem natural é interpretada retornando-se várias interpretações desta sentença (de acordo com o modo que ThoughtTreasure interpreta a sentença em linguagen natural). Na continuação o usuário

escolhe a interpretação mais adequada. Desta interpretação escolhida se realiza um processo codificação na linguagem UCL, que é representado pelo documento UCL em formato XML.

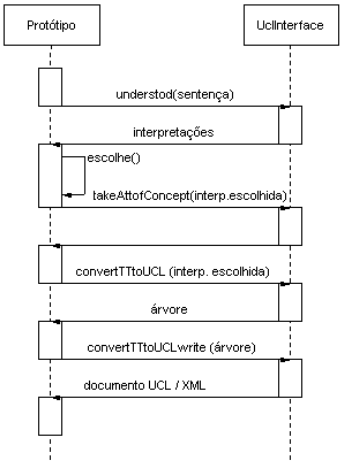

Figura 6.3 Diagrama de seqüência de eventos.

A seguir, são mostrados todos os métodos implementados (e que formam parte da API do protótipo) e como o processo envolvido para gerar as mensagens UCL se dá a partir de uma sentença em linguagem natural.

6.2.1 Processo de geração das mensagens UCL

Este tópico contem uma descrição do processo de geração das mensagens UCL, ao mesmo tempo, descreve-se o funcionamento de cada método no processo. O inicio se dá com a entrada de uma sentença em linguagem natural, posteriormente ela é codificada em uma mensagem UCL (utilizando a ontologia de conceitos) que expressa semanticamente a sentença inicial. Esta mensagem pode ser manipulada e/ou enviada a outro agente.

O processo de decodificação de uma mensagem UCL é iniciado com a recepção de uma mensagem codificada em UCL, para posteriormente ser transformada em uma sentença em linguagem natural equivalente à que inicialmente se deu.

Interpretando uma sentença em linguagem natural Este método (Understood)é chamado para realizar a interpretação (*parser*) de uma sentença em linguagem natural, e retornar uma lista de sentenças numa estrutura de dados de TT (ThoughTreasure) que expressam a mensagem inicial.

Transformar um elemento da lista de conceitos numa mensagem UCL Este método (*ConvertTTtoUCL*) cria a mensagem UCL baseado nos conceitos anteriormente cadastrados no agente emissor.

Transformar a mensagem UCL na lista de conceitos Este método (*deconvertUCLtoTT*), decodifica a mensagem UCL transformando-a em uma lista de conceitos.

Transformar a lista de conceitos em uma sentença em linguagem natural *Este processo (deconverterTTtoLN) transforma a lista de conceitos em uma sentença em linguagem natural que representa a mensagem UCL*

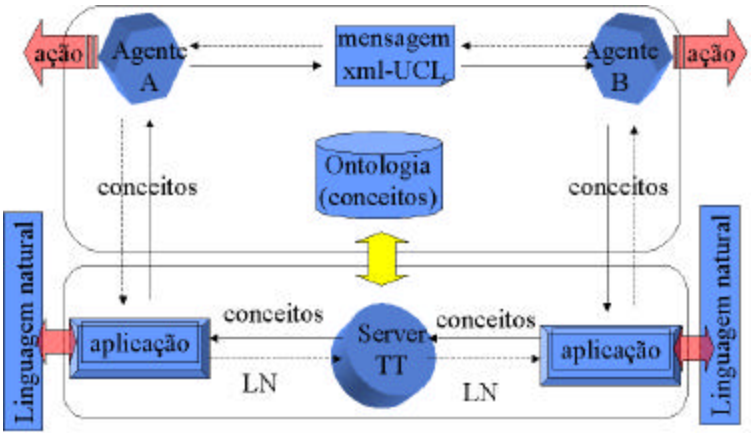

Figura 6.9 Arquitetura de um sistema que utiliza a linguagem UCL

6.3 Arquitetura de um sistema que utiliza a linguagem UCL

Considerando a utilização da linguagem UCL na comunicação entre dois agentes (de software), pode-se observar na **Figura 6.9** a seguinte arquitetura do sistema.

No suposto caso em que o agente A precisa se comunicar com o agente B, existem as seguintes possibilidades de comunicação:

- O usuário quer se comunicar com o Agente A
- O usuário quer se comunicar com o Agente B por meio do Agente A
- O usuário quer se comunicar com outro usuário

7. CONCLUSÕES

7.1. Decisões de Projeto

Neste item são feitas as descrições de algumas das decisões de projeto assumidas durante este trabalho, de forma a ressaltar aquelas já descritas no texto e complementar outras que não foram anteriormente citadas.

Durante a definição da linguagem *Universal Communication Language* (UCL) procurou-se abranger todos os conceitos teóricos da linguagem base, que neste caso é a *Universal Networking Language* (UNL). Desta forma se preservou o poder representativo de sentenças desta linguagem.

Os vários requisitos de uma linguagem para comunicação entre agentes, tais como, uma linguagem com ênfase na semântica da mensagem, facilidade de utilização, e integração com a rede Internet, dentre outros, levaram nos a optar por um conjunto de recursos específicos no nível de implementação do protótipo. Sendo esta linguagem definida para ser utilizada na comunicação entre agentes, foi importante ressaltar como a mensagem iria representar conceitos do mundo real, neste caso optou-se por utilizar ontologias como meio de representação. Neste nível se tinha algumas bases de dados ontológicas que podiam ser utilizadas, tais como a WordNet, Cyc, *ThoughtTreasure (TT),* dentre outras. A maioria destas bases ontológicas era proprietária o que dificultava a utilização das mesmas, em outras não existia uma forma clara de acessar os dados (por exemplo. Uma API disponibilizada). Neste sentido a única que cumpria os requisitos de disponibilidade (não proprietária) e oferecia uma interface de acesso à base de dados ontológica foi a *ThoughtTreasure.*

Uma outra justificativa para a utilização da base ontológica *ThoughtTreasure,* foi porque ela contem juntamente com os conceitos definidos dentro dela, várias características e atributos para representar linguagem natural que pode ser processada através da sua API TT. Desta forma pudemos integrar no protótipo que usa a linguagem proposta a geração de mensagens UCL a partir de sentenças em linguagem natural (em Inglês ou Francês).

Considerando a facilidade de utilização da linguagem na Internet, Optou-se pela utilização da meta-linguagem XML para definição da linguagem, através da definição de um documento DTD, sendo este um padrão utilizado para transferir dados na Internet. Desta forma a UCL não terá maiores problemas para se integrar a rede Internet. Além disso, a sintaxe desta linguagem é de fácil leitura e poupa trabalho no momento da implementação de um *parser*, já que, existem APIs ,padronizados pela W3C, que manipulam este tipo de documentos.

7.2. Contribuições

As pesquisas em linguagens de comunicação entre agentes vêm acompanhando a evolução dos sistemas multi-agentes, na medida em que permitam aos agentes se comunicar e assim cooperar entre si para resolver um problema.

Neste sentido, este trabalho contribui com a especificação da linguagem UCL, dando ênfase a importância da semântica da mensagem a ser representada com esta linguagem. A UCL facilita a representação de sentenças (mensagens) oriundas da linguagem natural, utilizando-se de ontologias, reduzindo desta forma confusões conceituais e terminológicas que podem existir na construção da sentença original em linguagem natural.

Além disso, a utilização de ontologias na linguagem auxilia na interoperabilidade entre diferentes agentes que trocam mensagens.

O uso do padrão XML, na implementação, contribuiu para que a linguagem seja facilmente integravel na rede Internet e em outras aplicações que usam este tipo de tecnologia.

A implementação do codificador da linguagem UCL serviu como subsídio para experimentação e também testam a viabilidade da proposta apresentada. Esta implementação foi desenvolvida como um interpretador de sentenças (mensagens), utilizando a linguagem Java e o programa TT.

7.3. Sugestões para Trabalhos Futuros

Uma extensão deste trabalho é implementar um interpretador de UCL para uso em agentes de software. Desta forma Auxiliando na implementação de agentes de software que podem se comunicar e assim cooperar para a realização de alguma tarefa complexa.

BIBLIOGRAFIA

(Bray et. al., 2000),Bray, T.; Paoli, J.; Sperberg-McQueen, C. M. *Extensible Markup Language (XML) 1.0*. (Second Edition) Outubro 2000. Disponível on-line: http://www.w3.org/TR/REC-xml

(Brown, 1989),Brown, H. *Standard for Structured Documents*. The Computer Journal, v.32, n.6, p.505-514, 1989.

(Cohen & Levesque, 1995),Cohen, Philip R.; Levesque Hector. Communicative Actions for Artificial Agents, p.419-436. AAAI press/The MT press, Junho 1995. J. M.

(Connolly, 2000),Connolly, D. *Extensible Markup Language (XML)*. Fevereiro 2000. Disponível on-line: http://www.w3.org/XML/

(Chimezie, 2000),Chimezie T. *The future of natural-language processing*. Unix insider. Visitado em 17, Janeiro de 2001. Itworld.com. Disponível on-line: http://www.unixinsider.com/swol-12-2000/swol-1229-ontology.html

(David, 1999),David Reilly, *Agent Communication.*, Fevereiro 1999. Disponível on-line: http://www.webdevelopersjournal.com/articles/agent_communication.html

(Dignum, 2000),Dignum, Frank; Greaves, Mark. (ed.). *Issues in agent communication*. – (Lecture notes in computer science; Vol 1916: Lecture notes in artificial intelligence) Berlin; Heidelberg; New York; Barcelona; Hong Kong; London; Milan; Paris; Singapore; Tokyo: Springer, 2000.

(Finin et al., 1993),Finin, Tim; Yannis Labrou; Mayfield, James. *KQML as an Agent Communication Language*, p.291-315. AAAI press/The MIT press, 1993.

(Franklin & Graesser, 1996),Franklin, S.; Graesser, A. *Is it an agent or Just a Program?: A Taxonomy for Autonomous Agents*. In Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, 1996. Disponível on-line: http://www.msci.memphis.edu/~franklin/AgentProg.html.

(Gouveia et al., 1998),Gouveia, D; Neto, A. B.; Silva, M. J. *ACE:Um Agente de compras na Internet*. XII Simpósio Brasileiro de Engenharia de Software, Anais, p.281-296, 1998.

(Grosof & Labrou, 1999),Grosof, Benjamin N.; Labrou, Yannis. *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, Disponível on-line: http://www.research.ibm.com

(Gruninger & Fox, 1995),Gruninger M.; Fox, M. S. *Methodology for the design and evaluation of ontologies*. In Workshop on Basic Ontological Issues in Knowledge Sharing. International Joint Conference on Artificial Intelligence, 1995.

(Jennings & Wooldridge, 1995),Jennings, N. R.; Wooldrige, M. *Agent Theories, Architectures, and Languages: a Survey. Intelligent Agents*, p.55-67, 1995.

(Ketchpel & Genesereth, 1994),Ketchpel, S.; Genesereth, M. *Software Agents*. Communications of ACM, Vol37, Nro 7, p.48-53, julho de 1994.

(Labrou & Finin, 1997a),Labrou, Y.; Finin, T. *A proposal for a new kqml specification*. Technical Report TR-CS-97-03, University of Maryland, Baltimore Country (UMBC), Disponível on-line: http://www.cs.umbc.edu/kqml/papers, 1997.

(Labrou et. al., 1999),Labrou; Yannis; Finin, Tim; Yun Peng. *Agent Communication Languages: The Current Landscape*. IEEE Intelligents Systems, p45-52, March/April 1999.

(Lenat, 1995),Lenat, D. B. CYC: *A large-scale investment in knowledge infrastructure*.

Communications of the ACM, 38(11), 33-48. 1995

(Mace et al., 1998),Mace, S.; Flohr, V.; Dobson, R.; Graham, T. *Weaving a Better Web*.Byte,. p58-68, Março 23, 1998.

(Mamadou et al., 2000),Mamadou, T. K.; Shimazu, A.; Tatsuo, N. *The State of the Art in Agent Communication Languages*. Japan Advanced Institute of Science and Technology., Japan, 1999.

(McGrath, 1999),McGrath, S. XML Aplicações Práticas - Como Desenvdver Aplicações de Comércio Eletrônico. Editora Campus, 1999.

(Miller, 1995b),Miller, G. A. WordNet: A lexical database for English. Communications of the ACM, 38(11), 39-48. 1995.

(Moreira & Walczowski, 1997),Moreira, D. A.; Walczowski, L. T. *Using Software Agents to Generate VLSI Layouts*. IEEE Expert Intelligent Systems, p.26-32. Nov/Dez. de 1997.

(Mueller, 1998),Mueller, Erik T. *Natural Language processing with ThughtTreasure*. New York: Signiform. Disponível on-line:
http://www.signiform.com/tt/book/

(Pimentel et. al, 1999),Pimentel, M.G.C., Teixeira, C.A.C., Pinto, C. C. – *Hiperdocumentos Estruturados na WWW: Teoria e Prática*. IN: XVIII Jornada de Atualização em Informática. JAI99 –SBC. Julho de 1999. p. 367-424.

(Sierra et. al., 2000),Sierra, Carles; Wooldridge, Michael; Sadeh, Norman. *Agent Research and Development in Europe*. IEEE Internet computing, p81-83, September/October 2000.

(Singh, 1998),Singh, M. P. *Agent communication languages: Rethinking the principles*. IEEE Computer, 31(12), p.40-47, December 1998.

(Singh, 1999),Singh, M. P. *A Social Semantics for Agent Communication Languages*. ACM Press, August 1999. Stockholm, Sweden.

(Trindade, 1997),Trindade, C. C. *Minimal Hyperlink Documents: Especificação e Apresentação de Estruturas Clásicas de Hipertextos*. São Carlos, Dissertação (Mestrado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.

(Ushold & Gruninger, 1996),Uschold, M. Gruninger, M. *Ontologies: Principles, Methods and Applications*. To appear in Knowledge Engineering Review, v.11, Number 2, June 1996. University of Edinburgh, 1996.

(White, 1994),White, E. J. *Telescript technology: The foundation for the electronic marketplace*. Technical report, General Magic, Inc., Disponível on-line: http://www.generalmagic.com/technology, 1994.

(Yokoi, 1995),Yokoi, T. *The EDR electronic dictionary*. Communications of the ACM, *38*(11), 42-48. 1995.-