UNIVERSIDADE FEDERAL DE SANTA CATARINA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

O SISTEMA UNL NOS PROCESSOS DE INTERNACIONALIZAÇÃO E LOCALIZAÇÃO DE SOFTWARE

Diego Duarte de Aragão

Florianópolis, novembro de 2004.

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

O SISTEMA UNL NOS PROCESSOS DE INTERNACIONALIZAÇÃO E LOCALIZAÇÃO DE SOFTWARE

Autor:
Diego Duarte de Aragão

Orientador:
Prof. José Eduardo de Lucca, Dr.

Banca Examinadora:
Prof. Jovelino Falqueto, Dr.
Prof. Renato Cislaghi, MSc.

Palavras-chave:
Internacionalização de Software, Localização de Software, Tradução por Máquinas, Sistema UNL, Linguagem UNL.

Florianópolis, 18 de novembro de 2004

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

O SISTEMA UNL NOS PROCESSOS DE INTERNACIONALIZAÇÃO E LOCALIZAÇÃO DE SOFTWARE

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do grau de Bacharelado em Ciências da Computação e aprovado em sua forma final pelo Curso de Ciências da Computação da Universidade Federal de Santa Catarina.

Florianópolis – SC, 18 de novembro de 2004.

_____

Prof. José Eduardo de Lucca, Dr.
Universidade Federal de Santa Catarina

_____

Prof. Jovelino Falqueto, Dr.
Universidade Federal de Santa Catarina

_____

Prof. Renato Cislaghi, MSc.
Universidade Federal de Santa Catarina

"*A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original.*"

Albert Einstein

AGRADECIMENTOS

Agradeço especialmente a meus pais e minha irmã, sem os quais este trabalho não seria possível. Foram vocês que me propiciaram a oportunidade única de concluir essa graduação, dando o seu suporte inquestionável e incondicional durante toda essa jornada e sempre acreditando no meu potencial. É a vocês que eu devo minha eterna gratidão.

Agradeço também ao meu orientador, professor José Eduardo de Lucca, por ter aceitado o desafio da orientação deste trabalho. Agradeço também aos membros da banca examinadora pelas críticas e sugestões ao relatório e principalmente ao professor Renato Cislaghi, que sempre se mostrou solícito e disposto a sanar muitos dos percalços aos quais nos deparamos durante a elaboração deste trabalho.

Também registro minha gratidão para com a equipe do Projeto UNDL Brasil, que sempre se mostrou interessada no trabalho e indicou parte da bibliografia sobre o Sistema UNL. Agradeço também aos membros da UNDL Foundation, que me ofereceram acesso na condição de pesquisador ao seu conteúdo técnico restrito.

Não poderia deixar de agradecer aos meus amigos, que sempre estiveram presentes em todos os momentos da elaboração desse trabalho, me dando forças e incentivando nos momentos de desânimo. Agradeço especialmente a um deles, Ronald Gomes, que contribuiu com sugestões valiosas, tanto acerca desse trabalho quanto nos mais diversos assuntos profissionais e pessoais.

Por fim, agradeço àqueles que mesmo não estando presentes na minha vida hoje, foram parte fundamental nessa jornada. Espero que saibam a dimensão de suas importâncias, e que nossas vidas voltem a se cruzar no futuro.

SUMÁRIO

RESUMO **12**
ABSTRACT **13**

1. INTRODUÇÃO **14**
1.1. Apresentação **14**
1.2. Justificativa **14**
1.3. Objetivos **15**
1.3.1. Objetivos Gerais 15
1.3.2. Objetivos Específicos 15
1.4. Estrutura do Trabalho **16**

2. A INTERNACIONALIZAÇÃO E A LOCALIZAÇÃO DE SOFTWARE **17**
2.1. Internacionalização de Software **19**
2.1.1. Código Fonte Separado da Interface do Usuário 19
2.1.1.1. Header Files 20
2.1.1.2. Resource Files 20
2.1.2. Expansão de Texto 21
2.1.2.1. Restrições das Linguagens de Programação e dos Sistemas Operacionais 21
2.1.2.2. Controles de Tamanho das Caixas de Diálogo 22
2.1.3. Números, Moedas e Formatos de Data 22
2.1.3.1.Unidades de Pesos e Medidas 24
2.1.4. Hotkeys 24
2.1.4.1. O Teclado do Usuário 25
2.1.4.2. Strings Compostos 25
2.1.4.2.1. Construções em Plural nas Mensagens Compostas 26
2.1.5. Character Sets 27
2.1.5.1. Ordenamento de Caracteres 28
2.1.5.2. Maiúsculas e Minúsculas 28
2.1.5.3. Habilitação de Caracteres de Dois Bytes 29

2.1.6. Idiomas da Direita para a Esquerda 30
2.1.7. Comentários 30
2.2. Localização de Software 31
2.2.1. Como Começar um Projeto de Localização 31
2.2.2. Quanto Localizar 31
2.2.3. O que Localizar? 32
2.2.4. Juntando as Peças 32
2.2.5. Timing da Localização 33
2.2.6. Diretrizes de Estilo, Glossários e Listas de Terminologia 34
2.2.6.1. Diretrizes de Estilo 35
2.2.6.2. Glossário 35
2.2.6.3. Lista de Terminologias 36
2.2.7. Passos do Processo de Localização 37
2.2.7.1. Interface do usuário (UI) 37
2.2.7.2. Documentação Eletrônica 39
2.2.7.2.1. Arquivos de Help 39
2.2.7.2.2. Arquivos PDF para Manuais Eletrônicos 40
2.2.7.3. Websites de Suporte 40
2.2.7.3.1. O Processo de Localização de *Websites* 41
2.2.7.3.2. Character Sets Extendidos e de 16 Bits na Web 41
2.2.7.3.3. Considerações sobre a Programação de Interfaces Web 42

3. A TRADUÇÃO POR MÁQUINAS 43
3.1 Entendendo a Tradução por Máquinas 43
3.2. Tipos de Sistemas de Tradução por Máquinas 45
3.3. Usos da Tradução por Máquinas 46
3.4. Benefícios e Custos da Tradução por Máquinas 49
3.5. Avaliando, Escolhendo e Customizando um Sistema de Tradução por Máquinas 49
3.6. Aplicações e Tendências para os Sistemas de Tradução por Máquinas 52
3.7. A Tradução por Máquinas e o Sistema UNL 53

4. O PROJETO UNL 56
4.1. Objetivos do Projeto UNL 56
4.1.1. Metodologia de Desenvolvimento do Projeto UNL 57
4.1.2. O Papel do Brasil no Projeto UNL 58
4.2. A Linguagem UNL 59
4.2.1. A Representação do Significado na Linguagem UNL 61
4.2.2. A Estruturação do Significado 62
4.2.3. O Léxico Segundo a Perspectiva da UNL 64
4.3. O Vocabulário UNL 68
4.3.1. As Palavras Universais (UWs) 68
4.3.2. Os Rótulos de Relação (RLs) 69
4.3.3. Os Rótulos de Atributos (ALs) 74
4.4. Ilustração do Processo Completo de Conversão e Deconversão 81

5. O FUNCIONAMENTO DO ENCO E DO DECO 84
5.1. O Módulo Léxico 85
5.2. O Módulo Sintático 88
5.3. O Módulo Semântico 90
5.4. Visão Geral do Funcionamento do EnCo e do DeCo 91
5.4.1. Funcionamento do EnCo 92
5.4.2. Funcionamento do DeCo 93

6. A INTERNACIONALIZAÇÃO E A UNL NO CICLO DE VIDA DE DESENVOLVIMENTO DE SOFTWARE 94
6.1. O *Framework* 94
6.2. O Ciclo de Vida de Desenvolvimento de Software – O Modelo em Cascata 95
6.2.1. Fase de Análise de Requisitos 95
6.2.1.1. Impacto da Internacionalização e do Uso da UNL 96
6.2.2. Fase de Especificação 98
6.2.2.1. Impacto da Internacionalização e do Uso da UNL 98

6.2.3. Fase de Design 100
6.2.3.1. Impacto da Internacionalização e do Uso da UNL 101
6.2.4. Fase de Implementação e Integração 102
6.2.4.1. Impacto da Internacionalização e do Uso da UNL 102
6.2.5. Fase de Manutenção 104
6.2.6. Resumo do *Framework* 104

7. CONSIDERAÇÕES FINAIS 106
7.1. Resultados Obtidos 106
7.2. Dificuldades Encontradas 106
7.3. Sugestões para Trabalhos Futuros 107

REFERÊNCIAS 108

ANEXO 1 – Especificação Oficial da Universal Networking Language (UNL) 110
ANEXO 2 – Artigo 110

LISTA DE FIGURAS

Figura 1 – Sobreposição dos conceitos de globalização, internacionalização, localização e tradução 18
Figura 2 – A codificação e decodificação de documentos no sistema UNL.....60
Figura 3 – Exemplo de sentença UNL..... 82
Figura 4 – Grafo da sentença UNL..... 82
Figura 5 – Arquitetura geral do sistema de conversão e deconversão..... 84
Figura 6 – Os cinco submódulos do analisador léxico 85
Figura 7 – Exemplo da operação do analisador léxico..... 86
Figura 8 – Exemplo da operação do gerador léxico 88
Figura 9 – O módulo sintático 89
Figura 10 – O analisador sintático..... 90
Figura 12 – O módulo semântico 91
Figura 13 – Ilustração da UNL nos processos de internacionalização e localização de software 95

LISTA DE TABELAS

Tabela 1 – Definição do array gszFileExtractError[] para múltiplos strings 21
Tabela 2 – Exemplos de representações distintas para o mesmo número 22
Tabela 3 – Exemplos de representações para moedas distintas..... 23
Tabela 4 – Exemplos de representações para formatos de tempo distintos..... 23
Tabela 5 – Exemplos de representações para formatos de data distintos 24
Tabela 6 – Exemplo do uso de uma função de reordenamento de parâmetros 26
Tabela 7 – Exemplos de traduções anômalas 27
Tabela 8 – Exemplo do uso de #defines para diferenciar a codificação dos caracteres..... 29
Tabela 9 – Pontos fortes e fracos das diferentes modalidades de traduções automáticas de texto 44
Tabela 10 – Similaridades e diferenças entre os sistemas de MT e a tradução humana 48
Tabela 11 – Entradas do dicionário de UWs derivadas da palavra “threaten”..... 66
Tabela 12 – Exemplos de entradas lexicais analisadas e não analisadas..... 67
Tabela 13 – Exemplo de sentença UNL 79

LISTA DE ACRÔNIMOS

API	Application Program Interface
ASCII	American Standard Code for Information Exchange
AL	Attribute Labels
DeCo	DeConverter
DLL	Dynamic Link Library
EnCo	EnCoverter
HTML	Hypertext Markup Language
I18N	Internationatilization
IM	Instant Messaging
ISO	International Standards Organization
L10N	Localization
MT	Machine Translation
PALN	Processamento Automático de Linguagem Natural
PDF	Portable Document Format
RL	Relation Label
RTF	Rich Text Format
SMS	Small Message Service
TTS	Text-to-Speech
UI	User Interface
UNL	Universal Networking Language
UW	Universal Word
WWW	World Wide Web
WYSIWYG	What You See Is What You Get

RESUMO

O presente Trabalho de Conclusão de Curso de Ciências da Computação tem como objetivo apresentar os fundamentos teóricos e práticos acerca da Internacionalização e Localização de Software, integrando ferramentas do sistema UNL (Universal Networking Language) a estes processos, essenciais à globalização do software desenvolvido em nosso país, visando um aumento de produtividade nos processos de desenvolvimento dos mesmos.

O trabalho consiste de uma sólida revisão bibliográfica acerca da Internacionalização e Localização de Software, a Tradução por Máquinas e sobre o sistema UNL, dissecando o funcionamento das ferramentas de conversão e deconversão desse sistema. Por fim, desenvolveu-se um *framework* baseado no Ciclo de Vida de Desenvolvimento em Cascata, integrando aspectos cruciais da Internacionalização e Localização de Software a um processo formal de desenvolvimento, utilizando as ferramentas do Sistema UNL para este fim.

Palavras-chave: Internacionalização de Software, Localização de Software, Tradução por Máquinas, Sistema UNL, Linguagem UNL.

ABSTRACT

The present work of conclusion of the Computer Sciences course has the target to present the theorical and practical fundaments about the Software Internacionalization and Localization, integrating the tools of the UNL (Universal Networking Language) system to these processes, key points to the globalized software to be developed in our country, aiming a productivity increase in their software development processes.

The work consists in a solid bibliographic review about the Software Internacionalization and Localization, the Machine Translation and the UNL system, deploying the conversion and deconversion tools of this system. Thus, a *framework* based in the Waterfall Development Lifecycle was developed, integrating the key aspects of the Software Internacionalization and Localization to a formal development lifecycle, using the UNL system tools to accomplish this target.

Keywords: Software Internacionalization, Software Localization, Machine Translation, UNL System, UNL Language.

1. INTRODUÇÃO

1.1. Apresentação

As indústrias de base tecnológica brasileiras em geral, encontram-se, atualmente, participando de um processo de globalização da economia. Esta situação induz mudanças profundas nos mercados de atuação dessas empresas, pois indústrias e empresas, que normalmente atuavam sobre mercados regionalmente restritos, vêem-se, agora, confrontados com concorrentes do mundo inteiro.

Além disso, com a evolução cada vez mais rápida de uma economia globalizada e sem fronteiras, muitas empresas não possuem a consciência de que softwares que fazem sucesso no mercado nacional podem não ser aceitos ou não obterem o mesmo êxito em mercados internacionais. Isso acontece devido ao fato de que em cada novo país, existem diversos fatores que poderão afetar a receptividade do software pelos clientes locais e, conseqüentemente, a sua decisão em comprar o produto. O que é necessário ser feito, para aumentar a competitividade e aceitabilidade do software em um determinado país é submetê-lo aos processos de internacionalização e localização, para que este software tenha maiores chances de ser aceito no mercado internacional. Uma conseqüência evidente é que a partir do momento em que o software está totalmente adaptado ao país que se deseja atingir é maior a sua competitividade através do aumento de suas vendas.

No entanto, para continuarem competitivas nesse mercado global, empresas desenvolvedoras de software que pretendem atuar globalmente, precisam adaptar seus produtos com o objetivo de atender às expectativas, requisitos e necessidades dos clientes potenciais. Todos esses fatores, que estão cada vez mais diversificados, se forem tratados por um processo adequado de adaptação, permitem que seus softwares possam ser facilmente compreendidos por todos os clientes, seja qual for seu país, idioma ou cultura e como conseqüência imediata a maior aceitação e venda de seus produtos.

1.2. Justificativa

Com a globalização, a disseminação da internet e da informática em geral, é essencial para as empresas desenvolvedoras de software, expandir suas fronteiras e a

distribuição de seus produtos para mercados diferentes do mercado local. Considerando que cada mercado possui diferentes peculiaridades locais e aspectos culturais distintos, essas diferenças devem ser tratadas no projeto e desenvolvimento do software tanto a nível funcional quanto ao nível de interface.

Um fato é que o Brasil é o quinto maior mercado de software do mundo, tendo movimentado aproximadamente US$ 8,7 bilhões em 2003 [1], e a previsão é que as importações de software cheguem a 10 bilhões de dólares em 3 anos. Um valor muito alto de importações, uma vez que o Brasil poderia exportar mais softwares e importar menos.

Em encontro a esses fatos, vem a necessidade do uso, aprimoramento e disseminação das técnicas de internacionalização e localização de software, a fim de suprir lacunas tanto no mercado nacional, bem como viabilizar a exportação do software desenvolvido em nosso país.

1.3. Objetivos

1.3.1. Objetivos Gerais

O projeto visa utilizar ferramentas do Sistema UNL no processo de internacionalização e localização de software, através do levantamento de potenciais benefícios advindos do uso dessa tecnologia e criação de de um *framework* para o desenvolvimento de software utilizando essas ferramentas.

1.3.2. Objetivos Específicos

- Fazer uma revisão bibliográfica acerca da internacionalização e localização de software, coletando informações e técnicas acerca do tema;
- Apresentar o Sistema UNL, introduzindo seus conceitos, paradigmas e peculiariedades, através de estudo e revisão bibliográfica;
- Ilustrar o uso do Sistema UNL em metodologias de desenvolvimento para internacionalização e localização de software;

- Criar um modelo ilustrativo utilizando a linguagem UNL para o processo de internacionalização e localização de software.
- Desenvolver um *framework* utilizando o Sistema UNL através da linguagem UNL e decodificadores na língua portuguesa para estudar a viabilidade do uso do Sistema no processo de internacionalização e localização de software.

1.4. Estrutura do Trabalho

Este Trabalho de Conclusão de Curso está disposto da seguinte maneira:

O segundo capítulo lida com a internacionalização e localização de software, dando uma ênfase mais prática na aplicação dessas ferramentas.

A Tradução por Máquinas, elemento fundamental e base do Sistema UNL é tratado no terceiro capítulo, onde elucidam-se os fundamentos da MT.

No quarto capítulo é introduzido o conceito do Sistema UNL, mostrando suas aplicações e seu funcionamento.

O quinto capítulo está focado no funcionamento das ferramentas de conversão e deconversão utilizadas pelo Sistema UNL.

O sexto capítulo, por sua vez, apresenta a construção de um *framework* baseado no Ciclo de Desenvolvimento de Software em Cascata que introduz os aspectos fundamentais da internacionalização e localização de software ao processo de desenvolvimento, fazendo uso de ferramentas do Sistema UNL.

Por fim, no sétimo capítulo, são feitas as conclusões e considerações finais acerca do trabalho apresentado.

2. A INTERNACIONALIZAÇÃO E A LOCALIZAÇÃO DE SOFTWARE

Há muita confusão a respeito de como as expressões **globalização**, **internacionalização**, **localização** e **tradução** devem ser utilizadas. Estes termos são usados indiscriminadamente pelos desenvolvedores de software, autores de documentação, usuários, mídia, departamentos de marketing e pelos vendedores dos softwares de modos distintos. O correto entendimento destes termos é um passo crucial no processo de desenvolvimento de software. Esses termos são definidos da seguinte maneira [2]:

Globalização: O processo de conceituação da linha de produtos para mercados globais de modo que possam ser vendidos em qualquer lugar do mundo com pequenas revisões.Está relacionado com a estratégia global de marketing da empresa e associado a conceitos de marketing (marcas, estabelecimento de *market share*, etc). A globalização é particularmente importante em mercados como os das indústrias de roupas e comidas. Qualquer um sabe o que é e pode beber Coca-Cola ou usar calças jeans da Levis, por exemplo.

Internacionalização: O processo de criação de um produto que possa ser facilmente e eficazmente localizado. Essa internacionalização pode ser algo tão básico quanto a formatação do layout de um documento, por exemplo, até a habilitação do software para lidar com *character sets* de dois bits, mudança muito mais complexa.

Localização: O processo de personalizar um produto para consumidores em um mercado alvo de forma que quando eles o usem tenham a impressão da que o produto foi projetado por um nativo daquele próprio país.

Tradução: O processo de conversão das escritas de uma linguagem de origem para as escritas de uma linguagem de destino. A tradução é um componente crucial da localização.

Estes quatro conceitos ajustam-se tal como um diagrama em forma de alvo, como mostra a Figura 1. A **globalização** envolve o conceito inteiro de tornar uma linha de produto global. A **internacionalização** é executada de forma que o produto possa ser então localizado. Finalmente, a **tradução** é o componente-base para o processo completo, pois representa a transformação da linguagem.

Figura 1 – Sobreposição dos conceitos de globalização, internacionalização, localização e tradução [2]

Portar um software para um mercado internacional é um processo de duas partes: primeiro o software deve ser internacionalizado e só então localizado. A internacionalização de software envolve a preparação do código fonte original para o processo de localização. A localização de software transforma o idioma de origem do software em um ou mais idiomas de destino, dando para o produto o aspecto de ter sido criado no país designado. A internacionalização é usualmente referida pela abreviação **I18N** – I seguido de 18 letras, então a letra N. A localização é abreviada freqüentemente como **L10N**).

2.1. Internacionalização de Software

A internacionalização de software é o processo de desenvolvimento de produtos (ou a recriação de softwares já desenvolvidos) com o mercado global em mente. Há vários aspectos a se considerar antes da localização de software. Aqui é onde a internacionalização aparece. Dentre os principais pontos a serem considerados, os seguintes aspectos podem ser ressaltados [3]:

• Esquemas de cores e escolhas de gráficos que evitem ofensas a clientes potenciais,

• Caixas de diálogo largas o bastante para acomodar a expansão de texto,

• Funcionalidades que suportem vários formatos de data, hora e moeda,

• Funcionalidades de input e output que suportem vários *character sets* (inclusive *character sets* de dois bytes para o mercado asiático),

• Campos de texto justificados para prevenir que o texto expandido em outras linguagens se sobreponha aos gráficos,

• Uma interface do usuário facilmente adaptável para permitir aos clientes ocidentais lerem textos da esquerda para direita e que possibilite aos clientes árabes lerem seus textos da direita para esquerda.

Cada subseção abaixo elenca os pontos básicos acerca dos princípios básicos citados acima do processo de internacionalização de software.

2.1.1. Código Fonte Separado da Interface do Usuário

Na fase de implementação de software, certos princípios ajudam a determinar quão facilmente o software pode ser localizado e traduzido. O conselho mais básico é separar o texto que vai ser exibido do cerne do código de programa. Embora seja certamente mais fácil escrever código com mensagens de erro e elementos da interface colocados perto donde eles serão usados no código do programa, isto significa a necessidades de tradutor ter acesso ao código fonte para pode traduzir os *strings* que serão traduzidos. Os tradutores, enquanto peritos em falar e escrever um idioma estrangeiro, não são necessariamente

engenheiros de software. Separando o texto do código, aumenta-se a segurança para o correto funcionamento das engrenagens vitais do produto. Existem duas técnicas tradicionais para separar o texto do código: os *header files* e os *resource files*.

2.1.1.1. Header Files

Os *header files* são utilizados para definir as partes do programa que podem ser utilizadas em múltiplos pedaços do código e precisam ser modificados freqüentemente. Por exemplo, *strings* de texto devem ser colocados nos *header files*. Nesse caso, o tradutor pode editar os *strings* nos *header files* sem precisar ter acesso ao cerne do código. Além disso, para ajudar o tradutor a identificar o texto traduzível, os *header files* podem ser "marcados" de modo que esse texto seja facilmente identificável.

O principal porém no uso dos *header files* é o fato do tradutor não ser capaz de ver e entender o contexto no qual os *strings* estão sendo traduzidos. A falta de informação contextual pode acarretar em traduções dúbias e errôneas. Para evitar esse problema, o *header file* traduzido deve ser re-inserido na aplicação e compilado de modo que o tradutor possa revisar a UI completada com a informação contextual traduzida.

2.1.1.2. Resource Files

Outra abordagem de separação do texto do código que contorna o problema dos *header files* é o uso dos *resource files* (arquivos .rc em C++) para isolar o texto localizável. Os *resource files* contêm a definição das caixas de diálogo, *strings*, menus, botões e ícones utilizados na UI. Basicamente tudo que o usuário visualiza na UI pode ser configurado no *resource file*, fazendo com que os tradutores saibam o contexto de utilização dos textos.

Para levar esse processo um passo adiante, pode-se isolar o texto localizável em *resource files* que podem ser compilados em bibliotecas de ligação dinâmica (arquivos DLL do Windows), de modo que o gerenciamento das múltiplas linguagens é facilitado. Assim, o arquivo executável da aplicação acessa os recursos localizados das bibliotecas localizadas. Para modificar a linguagem, simplesmente se muda a biblioteca referenciada no programa.

2.1.2. Expansão de Texto

A maioria dos idiomas requer mais caracteres (e/ou palavras) para representar o mesmo pensamento expresso na língua de origem, levando a uma expansão dos *strings* de texto. O comprimento dos *strings* de texto é importante sob as perspectivas da programação e do sistema operacional.

2.1.2.1. Restrições das Linguagens de Programação e dos Sistemas Operacionais

Agora que a maioria do desenvolvimento de software é para sistemas operacionais de 32 bits ou acima, o comprimento das *string* não é tão restritivo, mas para sistemas de 16 bits tais recursos são limitados a 255 bytes. Se o *string* na língua de origem está perto deste limite, o string traduzido provável vai estar acima do limite. Existem modos para criar *strings* com mais de 255 bytes, mas estes requerem a recodificação dos recursos. Por exemplo, um *string* que exceda 255 bytes pode ser movido do *resource file* para o código fonte do executável e declarado como um *array*, com cada elemento do *array* tendo menos de 255 bytes, apesar de ir de encontro aos princípios de isolamento do texto do código. Os elementos são combinados para criar mensagens que possam exceder essa limitação. Por exemplo, o código abaixo define o *array* `gszFileExtractError[]` que contém um *string* de texto em cada elemento do *array*:

```
char gszFileExtractError[] =
"Um erro aconteceu na extração ou decriptação"
"dos arquivos do módulo decriptado!"
... {uma linha de texto para cada elemento do array} ...
"Por favor entre em contato com o seu distribuidor para"
"adquirir um módulo válido ou reinstale o programa";
printf ("ERROR:%s", gszFileExtractError);
```

Tabela 1 – Definição do array gszFileExtractError[] para múltiplos strings

2.1.2.2. Controles de Tamanho das Caixas de Diálogo

A expansão de texto afeta o design da interface do software. Um botão projetado para a palavra “Fechar” pode não ser grande o suficiente para a palavra “Schließen”, tradução para o alemão desse termo. Como veremos adiante, durante o processo de localização, esses botões e menus podem ser redimensionados conforme a nova terminologia, mas isso pode ser evitado com um simples planejamento da expansão dos textos durante o processo de internacionalização. Isso significa que os campos de input e output de textos devem ser em média 30% maiores do que a especificação normal na língua de origem.

2.1.3. Números, Moedas e Formatos de Data

A maneira como são mostrados os formatos numéricos e as unidades que acompanham tais formatos variam conforme o país. Felizmente, APIs padrão como as disponibilizadas no Windows fazem boa parte do trabalho de conversão quando ocorre a mudança de *locales*. Entretanto, quando localizando um software, erros de adaptação podem ocorrer.

Muitas linguagens usam virgulas e espaços para separar os milhares das centenas, por exemplo. O mesmo número (5134) pode ser representado de três maneiras diferentes:

Brasil	**5.143**
EUA	**5,143**
Suécia	**5 143**

Tabela 2 – Exemplos de representações distintas para o mesmo número

O software projetado deve suportar tais variações, a fim de evitar problemas de conversão, quando 5.000 US$ transformam-se em 5,00 US$, por exemplo.

Além das convenções de numeração, quase todos os países do mundo têm um símbolo diferente para a sua moeda corrente. Se este símbolo aparece antes ou depois da quantia monetária também varia. Alguns exemplos incluem:

Dólar dos EUA	**$ ou US$**
Libra inglesa	**£**
Iene japonês	**¥**
Euro europeu	**€**

Tabela 3 – Exemplos de representações para moedas distintas

As datas e formatos do tempo também variam de país para país. Alguns países como os EUA usam o relógio de doze horas enquanto o Brasil e muitos países europeus usam o de vinte e quatro horas:

Brasil	**21:35**
EUA	**9:35 PM**
Alemanha	**21.35**
Canadá francês	**21 h 35**

Tabela 4 – Exemplos de representações para formatos de tempo distintos

Semelhantemente, o formato de datas varia de acordo com o país. O padrão dos EUA para a representação de datas é como mês, dia, ano (MM/DD/AA) com vários tipos de separadores (como "/" e "-"). O padrão europeu e brasileiro é dia, mês, ano (DD/MM/AA) com algumas exceções. O padrão chinês é ano, mês, dia. Eis alguns exemplos:

Brasil	**14 de agosto de 2004, 14/08/2004**
Estados Unidos	**August 14, 2004 8/14/2004**
França	**14 agust 2004 14.8.2004**
Hong Kong	**2004 2004/8/14**

Tabela 5 – Exemplos de representações para formatos de data distintos

Até mesmo a determinação de qual é o primeiro dia da semana pode variar. Por exemplo, na América o primeiro dia da semana é o domingo, já no calendário francês cada semana começa na segunda-feira. Além disso, a abreviação dos dias da semana devem ser evitada desde que alguns idiomas têm as mesmas primeiras letras para alguns dias da semana.

2.1.3.1.Unidades de Pesos e Medidas

A maioria do mundo fora o EUA usa o sistema métrico internacional, de modo que o software internacional deve poder lidar com essas métricas e fazer as conversões necessárias corretamente. Problemas com unidades de medida são particularmente graves em softwares científicos e de engenharia, onde inexatidões e erros de arredondamento durante o processo de conversão podem ter conseqüências catastróficas. Um grande cuidado deve ser considerado para assegurar conversões corretas entre sistemas de medidas.

2.1.4. Hotkeys

A maioria dos programas tem *hotkeys* (atalhos de teclado) para várias tarefas. Por exemplo, no MS Word, apertando-se "Ctrl" e “N” simultaneamente abre-se a caixa de diálogo "Novo Arquivo" de modo que o usuário pode elaborar um novo documento. Freqüentemente, quando localizados, esses *hotkeys* não são traduzidos e devem ser modificados de acordo com o novo texto. Por exemplo, quando “Localizar" é traduzido para o português o *hotkey* ALT-F, referente ao "Find" do inglês, deveria mudar para ALT-L.

2.1.4.1. O Teclado do Usuário

Para introduzir caracteres especiais como **é**, **á** e **õ** cada idioma tem uma função no teclado correspondente. Estes teclados tornam mais fácil digitar a maioria dos caracteres especiais, mas alguns caracteres que são freqüentemente usados no Brasil podem ser mais difíceis de se digitar em outros teclados. Por exemplo o caracter "\" em muitos países do Leste Europeu é inserido digitando-se Alt Direito+Q (há uma diferença entre as teclas Alt da direita e da esquerda) e vários idiomas europeus como o alemão trocam a posição das teclas "z" e "y". Os teclados japoneses, chineses e coreanos exigem apertar teclas múltiplas para criar um caracter. É importante se lembrar isto de ao escolher *hotkeys* ou controles de teclado.

2.1.4.2. Strings Compostos

Um dos desafios maiores da internacionalização de software está na combinação com mensagens compostas. Mensagens compostas acontecem onde dois ou mais *strings* são combinados para criar uma mensagem. Mensagens compostas tipicamente têm o seguinte formato:

```
"O %s tem um erro: %e"
```

Nesta combinação, o `%s` e `%e` são substituídos com o texto apropriado, "disco rígido" e "falta de espaço":

```
"O disco rígido tem um erro: falta de espaço"
```

Trabalhando com *strings* compostos, não há nenhum modo para o tradutor saber o gênero dos substantivos. A ordem das variáveis não muda a menos que um programador altere o código, mas as exigências gramaticais de alguns idiomas muitas vezes necessitam uma mudança na ordem das palavras. Os conteúdos ao redor da variável podem ter que mudar a ordem na estrutura de oração por causa da gramática que rege o idioma de destino. Se essa mudança acontece e o código correspondente não é atualizado, podem ocorrer erros como o da oração abaixo:

```
"O erro é: disco rígido falta de espaço"
```

Um modo de focalizar o problema de ordem das palavras durante a localização é usar uma função para ordenar parâmetros. Isto permite que o reordenamento aconteça em um *resource file* em vez de no código. Quando se usa o Windows, por exemplo, a ordem das palavras pode ser fixada usando a função `FormatMessage()` com o `%1` e `%2` como parâmetro de texto, por exemplo:

```
#define STR_ERROR "o %1 tem um erro %2
...
...
sprintf(OutBuf, FormatMessage(STR_ERROR, dispositivo, msgErro))
```

Tabela 6 – Exemplo do uso de uma função de reordenamento de parâmetros

2.1.4.2.1. Construções em Plural nas Mensagens Compostas

Regras por criar construções plurais diferem de idioma para idioma. Até mesmo em português as regra por construção de substantivos plurais não é universal: o plural para "cama" é "camas", mas o plural para "coração" não é "corações". O exemplo seguinte ilustra o problema com plurais:

```
"%d programa%s procurado%s"
```

e o string:

```
"%d arquivo%s procurado%s"
```

Se `%d` é maior que um, o `%s` é usado para inserir um "s" e formar o plural então, de modo que a mensagem poderia ser lida:

```
"1 programa procurado" e "1 arquivo procurado"
```

ou

```
"3 programas procurados" e "3 arquivos procurados"
```

Isto pode funcionar para o português, mas não funcionará para o alemão e para outras traduções de idiomas de europeu, como ilustra a Tabela 7:

programa = programma
programas = programma's
arquivo = bestand
arquivos = bestanden

Tabela 7 – Exemplos de traduções anômalas

2.1.5. Character Sets

Sistemas operacionais usam diferentes metodologias para o *input* e *output* de caracteres. Estas metodologias unificam os caracteres que representam idioma, assim é fácil ver quais caracteres são suportados ou não para determinada codificação. Por exemplo, o alfabeto regular do português é suportado em quase todas codificações como um subconjunto, mas caracteres "especiais" como **é**, **è** e **å** podem não ser. Os *character sets* são mapas de caracteres usados no sistema operacional. Alguns sistemas UNIX usam um *character set* ASCII de 7 bits que só contém 128 caracteres (incluindo aspas, espaços, pontuação, símbolos, caracteres alfabéticos maiúsculos e minúsculos, números e quebras de linha). O *character set* ASCII de 7 bits é muito limitado para a maioria dos idiomas estrangeiros já que não contém caracteres especiais (como **é**, **å**, ou **â**). Um padrão mais novo é o *character set* ASCII extendido de 8 bits permitindo 256 caracteres. O Microsoft Windows usa este *character set* ASCII de 8 bits e para computadores UNIX há um *standard* da ISO (ISO 8859-X). Apesar de diferentes, os *standards* da Microsoft e padrões de ISO são bem parecidos e tendem a funcionar corretamente em seus ambientes.

Entretanto, até mesmo com os 256 caracteres do ASCII de 8 bits ainda não é possível representar todos os caracteres utilizados pelos diversos idiomas ao redor do mundo. Para resolver este problema existem vários diferentes *character sets* ASCII de 8 bits que contêm todos o caracteres para um grupo de idiomas semelhantes. Já o japonês, chinês e coreano têm muitos caracteres em seus idiomas para caber em um *character set*

ASCII de 8 bits. Para estes idiomas, um *character set* ASCII de 16 bits deve ser utilizado. Em um esforço para unificar tais codificações, foi desenvolvido o *character set* chamado de Unicode, que contém os caracteres para quase todos os idiomas em 16 bits. O Unicode é suportado pelos mais novos sistemas operacionais como Windows NT, Windows 2000, Windows XP e pelos sistemas UNIX. Além disso, a linguagem Java tem suporte nativo ao Unicode, configurando-se como uma opção interessante de desenvolvimento de software internacional.

2.1.5.1. Ordenamento de Caracteres

Ordenar alfabeticamente é fácil para os mercados brasileiro e dos Estados Unidos, mas como ordenar palavras acentuadas em outros idiomas, como **é**, **õ**, e **à**? Para o Brasil e a maioria dos idiomas europeus, os caracteres acentuados são considerados iguais aos caracteres não acentuados e são ordenados normalmente (ou seguem o caracter não acentuado). Já a maioria dos idiomas escandinavos põe os caracteres acentuados depois do z (y, z,æ, å). No norueguês e dinamarquês, as vocais duplas como "aa" aparecem no fim do alfabeto. Em países de língua espanhola latino-americanos, o "ch" é considerado um único caracter que aparece entre o "c" e o "d". Consideração a respeito do ordenamento dos caracteres devem ser feitos na fase de design para assegurar que software funcione da maneira correta.

2.1.5.2. Maiúsculas e Minúsculas

Deve ser tomado cuidado ao se determinar se um caracter é maiúsculo ou minúsculo. Há vários artifícios que podem ser usados para se determinar se um caracter é maiúsculo ou minúsculo. O seguinte código é um exemplo de um modo rápido para conferir se um caráter é maiúsculo determinando se é maior que ou igual a "A" e menor ou igual a "Z":

```
If ((Unknown >= 'A') && (Unknown <= 'Z'))
  Maiusculo = TRUE;
```

Apesar desta técnica funcionar bem para o inglês, os caracteres especiais encontrados em outros alfabetos estão além da gama especificada. Todo os caracteres

especiais seriam considerados minúsculos. Os mesmos problemas podem ser encontrados em testes visando determinar se um caracter é alfanumérico ou algum outro símbolo.

2.1.5.3. Habilitação de Caracteres de Dois Bytes

É freqüentemente necessário num programa determinar o comprimento de um string. A técnica mais comum (a função "`strlength()`" e equivalentes) é contar o número de bytes desses string. Um byte é um grupo de bits (binários 1s ou 0s) usados para criar um caracter. A maioria dos idiomas pode ser representada com um único byte de informação. Entretanto, como mencionado acima, os caracteres do japonês, coreano do e chinês são representados através de um ou dois bytes, dependendo do caracter que é representado.

Idiomas que usam mais que um byte para representar um caracter criam problemas ao contar bytes para determinar o número de caracteres em um string. Ao escrever programas para uso nestes países, há funções diferentes que devem ser utilizadas. Para determinar o comprimento (número de caracteres) de um string nesses casos deve-se utilizar as funções do tipo "`mbsXXX()`". Já para o tamanho (número de bytes) o melhor é usar a família de funções "`strXXX()`". Além disso, como discutido anteriormente, os *strings* e *arrays* devem ser de tamanho suficiente para lidar com dois bytes por caracter. Por exemplo:

```
#define MAXSTRLEN 10
#ifdef _DBCS
#define MAXCHARSIZE 2 /* dois bytes por caracter */
#else
#define MAXCHARSIZE 1 /* um byte por caracter    */
#endif
int numofchar; int numofbytes;
char outBuf[MAXSTRLEN * MAXCHARSIZE];
...
...
numofchar = mbslen(outBuf);
numofbytes = strlen(outBuf);
```

Tabela 8 – Exemplo do uso de #defines para diferenciar a codificação dos caracteres

Devem ser examinadas todas as funções usadas para ver se elas podem controlar caracteres de byte duplo.

O desenvolvimento de novas tecnologias fez com que a exibição e o *input* de caracteres de 2 bytes fique muito mais fácil. Com os sistemas operacionais Windows mais recentes (Windows NT, Windows ME, Windows 2000, e Windows XP), o suporte multi-linguagem permite visualizar esses caracteres em um sistema padrão. Claro que a aplicação que roda sob o Windows também deve suportar os caracteres de byte duplo, e muitos deles o fazem. Além disso, também existem pacotes de software de que terceiros que facilitam a entrada e o manuseio de texto em dois bytes, como o Richwin, Twinbridge, e NJWin.

2.1.6. Idiomas da Direita para a Esquerda

Alguns idiomas, como o árabe e o hebreu, escrevem da direita para a esquerda em vez da esquerda para a direita. Estes idiomas contêm também freqüentemente *strings* de número e caracteres normais que devem ser escritos da esquerda para a direita. Isto significa há uma combinação de textos da esquerda para a direita e textos de direitas para esquerda na mesma página ou exibição. Esta combinação é chamada de texto bidirecional. As interfaces para estes idiomas põem menus e botões no lado oposto da tela. Os menus começam à direita, as barras de rolagem estão na esquerda e os botões normalmente aparecem no lado esquerdo da janela. A maior parte desses aspectos é controlada rearranjando os controles nos *resource files*, mas os recursos de direita-para-esquerda e de esquerda-para-direita não podem ser compartilhados. Qualquer função especial do software deve ser considerada muito cedo dentro do processo de desenvolvimento para assegurar que eles possam suportar idiomas da direita-para-esquerda e texto bidirecional.

2.1.7. Comentários

Os comentários devem ser usados para elucidar *strings* ambíguas instruindo o tradutor acerca de informações como quantos caracteres podem ser usados num *string* ou como esses *strings* devem ser utilizados. Os comentários são particularmente interessantes

para indicar partes do código que são voltadas para *character sets* de 16 bits, textos da direita para a esquerda, etc.

2.2. Localização de Software

Depois de finalizado o processo de internacionalização de software corretamente o produto está pronto para a localização. O processo de localização envolve vários passos básicos que são aplicáveis a todos os tipos de projetos de software. Os detalhes podem variar, dependendo do tipo de software que é localizado. A equipe de localização deve trabalhar junto à equipe de desenvolvimento, determinando os processos exatos que são necessários para o projeto.

2.2.1. Como Começar um Projeto de Localização

Pela simplicidade, dividimos o processo de localização em pedaços mais manejáveis. Uma vez estabelecida a extensão do projeto, deve-se selecionar o método de localização que melhor atenda as necessidades do projeto.

2.2.2. Quanto Localizar

Quantos componentes do projeto realmente precisam ser localizados? A resposta pode ser qualquer coisa entre “não muito” e “todo o conteúdo em cima de todos os componentes de produto”.Em muitos casos, o orçamento ou o prazo para a localização podem ditar a quantia do conteúdo ser localizado. Entretanto, deve-se mensurar o impacto do conteúdo não ser localizado. Ao escolher não localizar certos aspectos do produto corre-se o risco de frustrar potenciais clientes em um mercado estrangeiro não provendo informação no idioma deles ou, talvez até pior, a empresa exportadora em potencial pode ser impedida por legislações e agências reguladoras de vender produtos que não são localizados para o mercado alvo. Na realidade, devido à tendência atual para a globalização

da economia mundial, é prudente consultar as autoridades apropriadas sobre as implicações legais de não localizar parte ou todo o conteúdo.

A decisão de quantas línguas a serem traduzidas é freqüentemente derivada da demanda do mercado e/ou das exigências legais. Felizmente, quanto mais idiomas traduzidos de uma vez, mais eficiente o processo é em termos de prazos e custos. Uma vez que a infra-estrutura de localização é estabelecida, o esforço para a tradução de novas linguagens decresce à medida que o número de línguas a serem localizadas aumenta.

2.2.3. O que Localizar?

Os principais aspectos do software a serem localizados são a interface do usuário (UI), os arquivos de ajuda eletrônicos, as garantias legais do produto, os arquivos *readme*, as capas dos CDs, o produto impresso, a documentação e a documentação on-line. Num segundo estágio, a informação de suporte no *website* e o material de treinamento utilizado pelas equipes de vendas internacionais também devem ser localizados.

Felizmente, a localização de virtualmente todos estes componentes pode se beneficiar de algumas das mais novas técnicas e tecnologias aplicadas à administração de conteúdo. Recentemente, foram desenvolvidas ferramentas e metodologias que permitem ao desenvolvedor reciclar o conteúdo já traduzido em diferentes tipos de contextos. Nestes casos há a redução dos custos de tradução (pelo fato do conteúdo ser reutilizado) e dos custos de localização (pelo fato o qual a editoração do conteúdo é feita apenas uma vez para os múltiplos formatos internacionais).

2.2.4. Juntando as Peças

Dado todos os vários pedaços envolvidos no *release* de um software, é importante pensar na ordem a qual submetê-los à empresa ou membros da equipe responsáveis pela localização. Se a documentação inclui 20 capturas de tela da UI do software, por exemplo, a UI precisa ser localizada antes da documentação A dificuldade que os responsáveis pela localização freqüentemente enfrentam é o fato de que com prazos muito curtos, a

localização da UI e da documentação acontecem simultaneamente, por exemplo. Pelo fato destes projetos freqüentemente envolverem a tradução de milhares de palavras, os responsáveis pela localização devem formar times de tradutores para trabalhar em ambos os componentes ao mesmo tempo. Em certo ponto do cronograma, depois que a UI estiver localizada, o tempo deve ser alocado para permitir que a documentação capture de forma correta a terminologia usada no UI. Felizmente, componentes como materiais de treinamento e o conteúdo de *web* podem esperar até que o cerne da localização de produto esteja completo.

2.2.5. Timing da Localização

Há duas alternativas típicas para o lançamento de softwares no mercado global: focar no mercado principal e nos mercados marginais simultaneamente ou lançar a versão do produto internacionalizada na versão para o mercado principal primeiro e ir desenvolvendo as versões localizadas do mesmo para os outros mercados à medida que as demandas vão surgindo. Ambas as alternativas funcionam bem, dados determinados contextos.

Uma vez que os mercado principais e marginais já estão identificados e existe uma demanda, normalmente há uma pressão dos departamentos de marketing e vendas para que haja a liberação simultânea dos produtos para o mercados principais e marginais.

A liberação simultânea é o desafio principal da localização. Para liberar a versão na linguagem de origem e a localizada do produto ao mesmo tempo, a localização geralmente começa enquanto a versão principal do programa ainda estiver sendo internacionalizada. Isso significa que à medida que os desenvolvedores fazem mudanças na UI (ou os autores de documentação ajustam o Help, por exemplo) essas mudanças têm de ser incorporadas pela equipe de localização. Obviamente, isto faz com que o gerenciamento de configuração e conteúdo no processo de localização torne-se mais desafiador e impacta diretamente no custo do projeto com as repetidas “paradas e recomeços” para acomodar as múltiplas e constante mudanças.

Dado estes desafios, parece natural esperar que localização seja mais fácil depois que a versão internacionalizada é liberada e a demanda dos mercados marginais for

surgindo. Entretanto, se por um lado é mais fácil (a localização do conteúdo na língua nativa dos desenvolvedores é muito mais rápida e direta), o mercado globalizado dos dias atuais demanda que todos os potenciais mercados consumidores do produto sejam atendidos imediatamente.

A equipe de localização deve ser experimentada com ambos cenários de localização do software e devem ter metodologias e procedimentos que suportem ambos os modelos. Se a equipe decide pela liberação simultânea, um ciclo de vida de desenvolvimento incremental em cascata pode ajudar no alcance das metas estabelecidas, como veremos mais adiante. A equipe de localização é abastecida com versões *alfa* e *beta* do software em internacionalização e então, quando o mesmo está funcionalmente completo, a equipe de localização pode finalizar as traduções. Esta alternativa gera um pouco mais de trabalho, mas o benefício de ter tudo localizado ao mesmo tempo tende a minimizar esse problema.

Uma vez determinado o ciclo e o *timing* da localização de software, deve-se iniciar o real processo de localização. O primeiro passo é entender a importância da administração da terminologia e do conteúdo.

2.2.6. Diretrizes de Estilo, Glossários e Listas de Terminologia

Para alcançar a mais alta qualidade lingüística possível, antes do processo de localização deve-se incluir o desenvolvimento de três itens:

- Uma diretriz de estilo,
- Um glossário no idioma de origem,
- Uma lista de terminologia no idioma de destino.

Enquanto o desenvolvimento destes três componentes acarreta num maior tempo ao iniciar um projeto, na fase de internacionalização, quando feitos corretamente eles economizam uma grande quantia de tempo durante a fase de localização (garantindo um nível muito maior de qualidade no produto final). Diretrizes de estilo, glossários e listas de terminologia asseguram a consistência na tradução de todos os componentes do produto localizado.

2.2.6.1. Diretrizes de Estilo

Diretrizes de estilo, ou folhas de estilo, são listas de “regras” específicas que a equipe de localização deve seguir durante o processo de tradução. Estas diretrizes geralmente são fornecidas pelos clientes. As diretrizes tipicamente regem os seguintes aspectos do conteúdo:

- Tom da documentação localizada,
- Termos que são traduzidos e aqueles que não o são,
- Regras para capitalização e acentuação,
- Tradução de títulos e subtítulos,
- Conversão de medidas,
- Regras para a tradução de números por extenso,
- Uso de abreviações,
- Regras de pontuação, etc.

A qualidade de uma documentação localizada é amplamente dependente da qualidade do texto fonte. Para que o texto fonte seja de boa qualidade para a localização, o autor ou desenvolvedor do mesmo deve ser informado previamente que o mesmo vai ser localizado. Nesse sentido, a interação entre os criadores do texto e o responsável pela localização é importante para que os aspectos culturais específicos de cada país sejam contemplados previamente. As diretrizes de estilo ajudam a criar documentos apropriados para o usuário final, para o cumprimento dos requisitos legais de cada país bem como para a manutenção das particularidades culturais e geográficas.

2.2.6.2. Glossário

Um glossário é uma lista de palavras na língua de origem que pela suas dificuldades ou características técnicas são detalhadas especificamente. Geralmente o glossário é desenvolvido por autores técnicos e desenvolvedores trabalhando num domínio específico,

de modo que o mesmo seja utilizado para garantir que a tradução correta seja dada para estes termos especializados.

2.2.6.3. Lista de Terminologias

A lista de terminologias é uma lista de termos definidos previamente, geralmente restritos ao domínio do problema que o software se propõe a resolver, na linguagem destino, a serem utilizadas no processo de localização. Ela garante:

• Que os desenvolvedores, autores de documentação e responsáveis pela localização usem a mesma terminologia específica daquele domínio em todos os componentes do projeto,

• Consistência de abreviações, nomes de produtos, termos não traduzíveis e medidas,

• Particularidades locais,

• Consenso entre as partes envolvidas no processo.

A lista de terminologias é baseada em:

• O glossário específico do domínio do produto desenvolvido pelos desenvolvedores e autores dos documentos fonte,

• A terminologia utilizada nas UIs já localizadas por grandes empresas de software (por exemplo, as caixas de diálogo padrão do Windows, localizadas pela Microsoft),

• Softwares e documentações que a empresa já tenha localizado,

• Documento ou especificação que representa os padrões adotados no país a ser localizado. Esse documento, chamado de *locale*, inclui padrões de formatação de números, de ordenação de listas, abreviações, tempo, datas, feriados, moeda corrente, medidas, etc.

• Todos os outros recursos localizados como planos de marketing e catálogos de produtos.

Para assegurar a consistência e precisão entre o software e a documentação, a lista de terminologia deve ser desenvolvida antes do início do processo de tradução.

2.2.7. Passos do Processo de Localização

Os passos básicos de um projeto de localização de software incluem:

• Estabecimento um sistema de gerenciamento de configuração que mantenha os registros de cada uma das versões localizadas do código de fonte,

• Elaboração da diretriz de estilo, glossário e lista de terminologias,

• Identificação do *strings* de texto no idioma de origem a serem localizados,

• Tradução dos *strings* para as linguagens de destino,

• Reinserção dos *strings* traduzidos no código fonte corretamente para cada versão dos idiomas de destino,

• Teste do software traduzido, editando a interface conforme o necessário para acomodar o texto traduzido,

• Elaboração uma revisão lingüística da versão localizada do software para assegurar que a forma e o conteúdo estejam corretos,

• Incorporação de comentários finais para a entrega do produto.

Cada subseção abaixo examina o processo de localização para os três principais aspectos a serem considerados neste processo: a Interface do Usuário (UI), a documentação eletrônica em anexo (como arquivos de Help e arquivos PDF), e os *websites* de suporte (arquivos HTML/SGML/ASP/JSP/etc associados, scripts e *applets*).

2.2.7.1. Interface do usuário (UI)

Não importa a linguagem de programação ou plataforma utilizada, sempre há um modo de isolar os *strings* de texto das caixas de diálogo que aparecem na tela. Usando o Microsoft C/C++, por exemplo, os *resource files* (. rc) são usados para isolar estes *strings*, como discutido acima sobre a internacionalização. Estes *resource files* podem ser compilados a fim de criar bibliotecas de ligação dinâmicas (.DLL) para cada um dos idiomas de destino. O processo de localização da UI requer que os *resource files* sejam tratados e "marcados" de modo que os tradutores possam identificar facilmente os *strings* que devem ser traduzidos. Existem ferramentas disponíveis, soluções proprietárias

desenvolvidas por empresas de localização, que ajudam nesse processo de extração de *strings*. Dependendo de que ferramenta está sendo utilizada, os *resource files* normalmente são controlados por um dos três seguintes modos:

• Os *strings* são extraídos dos *resource files* e colocados em um documento de texto com apontadores para a localização original no *resource file*. As ferramentas proprietárias funcionam deste modo. Uma vez que os *strings* localizados são reinseridos no *resource file*, as caixas de diálogo, menus e gráficos são modificados para ajustar-se aos novos *strings*. A principal vantagem desta abordagem é o fato da extração e reinserção do texto poder funcionar de maneira completamente automatizada. O aspecto negativo é o fato do tradutor que só está trabalhando com os *strings* não pode ver o contexto no qual esses *strings* estão sendo utilizados.

• Os *resource files* são tratados de modo que os *strings* de texto sejam realçados ajudando o tradutor a identificá-los. O tradutor ignora todas as partes de código no *resource file*, podendo ver o arquivo inteiro (entretanto com os *strings* de texto realçados para a correta edição). Como resultado, o contexto do *string* pode ser determinado, caso o tradutor tenha um pouco de familiaridade com a codificação. Então, os *resource files* localizados são modificados pelos desenvolvedores de software para editar as caixas de diálogo, menus e gráficos para o texto traduzido. Esta técnica normalmente é a mais fácil de se implementar, pelo fato do único requisito ser tradutor ter o mínimo de familiarização com os *resource files*.

• Algumas soluções de localização proprietárias fornecem uma interface que exibe os *strings* de texto que necessitam ser localizados. O tradutor trabalha utilizando esses softwares para acessar a UI dos componentes, traduzindo os *strings*. O software controla a reinserção no *resource file*, DLL ou no executável automaticamente. Estas ferramentas geralmente trazem editores gráficos para edição das caixas de diálogo, botões e menus à medida que a tradução é executada. Estes sistemas oferecem a melhor abordagem para o processo de localização, em que os *strings* e a UI são editados para a localização paralelamente. Problemas podem surgir, entretanto, pela necessidade do tradutor ter uma

certa familiaridade com o processo de desenvolvimento de software e pelo alto custo dessas soluções proprietárias.

2.2.7.2. Documentação Eletrônica

Os pacotes de software utilizam várias formas de documentação eletrônica para prover suporte aos usuários. Na pior das hipóteses, existe um "*readme*" ou um arquivo de *release notes*, elencando as funcionalidades implementadas em cada versão. Ultimamente, está crescendo o número de fabricantes de software que provém uma maior documentação aos seus softwares. Alguns fabricantes vêm eliminando materiais impressos e calcando a documentação de seus produtos em documentos eletrônicos em anexo.

Os dois maiores exemplos de documentação eletrônica são os arquivos de Help (tipicamente acessados pelo próprio programa) e manuais eletrônicos incluídos no pacote de distribuição (manuais de usuário, guias de instalação, etc).

2.2.7.2.1. Arquivos de Help

Os arquivos de Help são a forma mais comum de documentação eletrônica. A Microsoft tem dois padrões de arquivos de ajuda para o Windows: O WinHelp baseado em arquivos RTF, e o HTMLHelp baseado em HTML. A maioria dos arquivos de ajuda é escrita utilizando ferramentas como o RoboHelp.

Ao se traduzir arquivos de ajuda, a equipe responsável pela localização precisa de acesso aos arquivos usados na construção da Ajuda:

• arquivos de conteúdo (RTF, DOC, ou HTML), contendo o cerne do texto do Help,

• arquivos de projeto (utilizados para compilar o conteúdo do Help),

• gráficos e figuras que serão utilizados nos arquivos de Ajuda (freqüentemente contendo texto que requer tradução),

• outros arquivos de conteúdo (como os. arquivos CNT usados para criar o sumário dos arquivos de Ajuda).

2.2.7.2.2. Arquivos PDF para Manuais Eletrônicos

Outra forma muito popular de documentação on-line usa o Adobe Acrobat para criar arquivos PDF. O Acrobat é um formato de distribuição eletrônica de documentação multiplataforma, provendo 100% de fidelidade gráfica e de layout de página em uma grande variedade de sistemas operacionais. Com o Acrobat, os documentos são vistos da maneira que os autores originalmente os projetaram.

Para um tradutor poder trabalhar com documentação eletrônica usando o Acrobat ele necessitados arquivos de conteúdo, obviamente. Os arquivos de conteúdos são então traduzidos e depois convertidos em PDF. O arquivo de PDF é enviado então para o editor de cópia e revisor. Usando o Acrobat Reader, os editores de cópia e revisores podem editar a tradução até mesmo não possuindo o código fonte da aplicação.

2.2.7.3. Websites de Suporte

Apesar do estouro da "bolha" da Internet em 2001, o uso da WWW continua crescendo vertiginosamente. A sua importância no estabelecimento de mercados, incremento de vendas e hospedagem de serviços e aplicações está bem estabelecido. Até mesmo dentro de uma organização, o uso de tecnologias baseadas na WWW para a administração da informação é padrão.

A WWW gerou tremendas oportunidades tecnológicas para o fornecimento de informação para parceiros e clientes ao redor do mundo. Este é o cerne da questão: pessoas em qualquer lugar no mundo podem pesquisar uma companhia ou informações acerca de produtos em um *website*, de modo que essa informação deva estar mais próxima e acessível a estes clientes em potencial.

É fácil dizer que a informação de um *website* deve ser fornecida na linguagem nativa do cliente, mas há pontos importantes a serem considerados antes de decisão de localizar o conteúdo de um *website*. Os *websites*, por natureza, encorajam o administrador do site a atualizar e/ou modificar o seu conteúdo freqüentemente, à medida que os visitantes esperam encontrar informações atualizadas. É esta expectativa que faz a localização de *websites* uma atividade um pouco mais desafiadora. A mudança de uma página no *website* requer mudanças na mesma página dentre todos os idiomas suportados.

Evidentemente, a manutenção de conteúdo em *websites* torna-se mais complicada com o incremento do número de idiomas suportados.

2.2.7.3.1. O Processo de Localização de *Websites*

A complexidade de processo de localização em um *website* sobe sensivelmente quando comparado ao esforço de se localizar um software. Antes de se proceder a localização, o *website* deve ser avaliado em sua complexidade. São incluídas páginas *web* de conteúdo estático (texto), gráfico e imagens, *hyperlinks* e componentes técnicos avançados. Cada um destes componentes deve ser considerado no processo de localização. O conteúdo deve fazer parte da construção de página, carregado dinamicamente através de scripts ou por uma interface de banco de dados.

O aspecto mais complexo da localização de *websites* é a interface do banco de dados. É o banco de dados que faz os sistemas de administração de conteúdo funcionarem efetivamente. Muito do conteúdo é armazenado em um banco de dados e exibido na página *web* de acordo com a demanda. Um nível extra de complexidade deve ser somado ao gerenciamento do conteúdo apresentado na página: o banco de dados além de recuperar o conteúdo, tem que fazer a recuperação no idioma correto e ajustar esse conteúdo para que seja exibido corretamente. Isto é feito projetando o banco de dados para controlar o nível adicional de dificuldade com uma estrutura de tabelas “por idiomas”. Semelhantemente, a folha de estilo de *web* é modificada para controlar as exigências dos textos nos idiomas distintos de modo que o conteúdo seja exibido corretamente.

2.2.7.3.2. Character Sets Extendidos e de 16 Bits na Web

As páginas *web* devem ser capazes de exibir os caracteres de todas as partes do mundo. Os caracteres acentuados dos idiomas da Europa Ocidental (francês, espanhol, italiano, alemão, e português) são relativamente simples de exibir. A maioria dos computadores suporta o *character set* ASCII extendido para a representação destes caracteres. Para HTML, estes caracteres especiais podem ser exibidos através de códigos HTML específicos ou fixando-se o idioma que codifica a página. Por exemplo, um "**é**" é representado em HTML como **`&eacute`** (um acento agudo em cima da letra e). Estes

códigos especiais são gerados automaticamente ao se utilizar um editor HTML do tipo WYSIWYG (*What You See Is What You Get*).

Entretanto, a localização de *websites* para os idiomas de alguns países do Leste Europeu e da Ásia é ligeiramente mais complicada pelo fato dos *character sets* utilizados nessas regiões serem diferentes dos padrões mundiais. Felizmente, os browsers Internet Explorer e Netscape, acima das versões 4.0, suportam *metatags* de codificação de idiomas. Se o PC do usuário tiver suporte ao idioma apropriado (disponível como suporte multi-linguagem nas plataformas Windows e Mac), os *character sets* aparecem corretamente. Tanto a página como o *browser* devem estar configurados para suportar o *character set* desejado. Também devem ser instaladas fontes no computador para visualizar estes *character sets* de 2 bytes. Usando o tag **<META>** da linguagem HTML, a codificação necessária para visualizar uma página pode ser fixada automaticamente. Por exemplo, o tag

```
<META http-equiv="content-type" content="text/html; charset=big5">
```

indica que a página é codificada para o Chinês tradicional.

2.2.7.3.3. Considerações sobre a Programação de Interfaces Web

A programação de interface para *web* também se beneficia de técnicas que separam o texto a ser exibido do código. Infelizmente, os scripts mais comumente utilizados no desenvolvimento de interfaces *web* não são tão estruturados quanto aqueles utilizados no desenvolvimento de aplicações, fazendo com que seja mais difícil separar o código do texto. Em ver de isolar o texto em módulos especiais como *header files* e *resource files*, é necessário embutir o texto diretamente no script. Entretanto, para fazer com que o processo de localização torne-se mais fácil, comentários devem ser utilizados para formatar os arquivos de script de um modo que prontamente seja identificado o texto localizado. Desse modo, a equipe de tradução pode criar filtros que identifiquem e "marquem" o texto para tradução.

3. A TRADUÇÃO POR MÁQUINAS

A Tradução por Máquinas (MT) é uma ferramenta poderosa, largamente utilizada por governos, indústrias e pelos consumidores individualmente. A tecnologia de ponta evolui constantemente e usos inovadores da MT aparecem com uma freqüência cada vez maior. Nos últimos anos, o uso da MT vem se diversificando: por ano, já são traduzidas mais palavras utilizando MT em relação ao número de palavras traduzidas por tradutores humanos, e a demanda continua crescente [4].

Os sistemas de MT modernos começaram surgiram nos anos 50 a partir de uma pesquisa em parceria entre a Georgetown University e a IBM. Mesmo com a frustração inicial gerada pelo estudo do governo americano de 1966 que indicava a MT como sendo muito cara, ineficiente e lenta para a tradução automática, as pesquisas e o desenvolvimento da tecnologia continuaram em outros países como Japão, Rússia e na Europa. Com a globalização e os aumentos dos projetos internacionais nos anos 80, o uso da MT voltou a surgir como uma excelente alternativa no compartilhamento de conhecimento em línguas distintas. Nos dias de hoje, a MT é um componente crítico para as demandas lingüísticas do século XXI, permitindo aplicações e usos onde os tradutores humanos não poderiam atuar e maximizando sua performance em outros aspectos.

3.1 Entendendo a Tradução por Máquinas

A MT é um método de tradução onde um texto é traduzido de uma língua para outra automaticamente, sem a intervenção humana. Apesar de outras tecnologias aparentarem o mesmo uso, na realidade as mesmas diferem substancialmente. Os Dicionários Eletrônicos Bilíngües, por exemplo, apesar de serem capazes de traduzir palavras e expressões automaticamente, são incapazes de traduzir sentenças completas, além de não serem capazes de escolher a melhor alternativa de tradução entre as possíveis. Por outro lado, existem os produtos que utilizam a Tradução por Memória, que utilizam um número fixo de sentenças e parágrafos armazenados em memória e fazem a tradução baseada na similaridade entre os mesmos e o texto inserido. Tal paradigma exige que um tradutor humano preencha a memória com tais sentenças e parágrafos, ficando sujeito à tendência

das pessoas em dizer a mesma coisa de maneiras diferentes, minimizando a possibilidade do casamento correto entre o texto inserido e a tradução desejada.

Os sistemas de MT constroem automaticamente uma tradução para qualquer sentença, sendo independente de um número fixo de prévias traduções armazenadas em memória. A MT não provê uma tradução palavra-a-palavra: essa técnica processa o contexto da sentença a fim de determinar os significados das palavras e das sentenças. Logo, é muito mais flexível que os produtos que utilizam a Tradução por Memória e muito mais profundo que os que utilizam Dicionários Eletrônicos. Entretanto, pelo fato do conhecimento da gramática e das palavras das línguas envolvidas ser muito menor em um sistema de MT do que o conhecimento do tradutor humano, o texto traduzido não atinge total correção semântica.

A tabela abaixo resume os pontos fortes e fracos dos diferentes paradigmas de tradução automática de textos.

	Pontos Fortes	**Pontos Fracos**
Dicionários Eletrônicos Bilíngües	- Fácil desenvolvimento	- Trabalha apenas com palavras - Não analisa o contexto da sentença
Tradução por Memória	- Utiliza traduções já existentes - Geralmente a tradução não necessita de ajustes	- Depende da similaridade do texto a ser traduzido com o texto em memória
MT	- Aceita novas sentenças - Extremamente rápido	- Depende mais da qualidade do texto fonte do que as outras técnicas

Tabela 9 – Pontos fortes e fracos das diferentes modalidades de traduções automáticas de texto

Um ponto corriqueiro levantado acerca da MT é a substituição dos tradutores humanos pela técnica em questão. Os sistemas de MT foram desenvolvidos como uma ferramenta para ajudar os tradutores humanos a trabalhar mais eficientemente ou como uma alternativa “boa o suficiente” em situações onde os tradutores humanos não são ou não podem ser utilizados. Fica evidente que a MT deve ser utilizada para diminuir os custos da

tradução e aumentar a eficiência da tradução humana. Além disso, a demanda por conteúdo em várias línguas expande o mercado de traduções tanto para os tradutores humanos quanto para as máquinas.

3.2. Tipos de Sistemas de Tradução por Máquinas

Diferente das técnicas de tradução utilizando Dicionários Eletrônicos Bilíngües e Tradução por Memória, os sistemas de MT usam inteligência artificial a fim de fazer uma análise sofisticada do texto fonte, maximizando a correção semântica e ortográfica do texto gerado. Atualmente existem cinco tipos de sistemas de MT em uso [5]: os Sistemas de MT Baseados em Dicionários Simples (tradução para Simple Dctionary-based MT), a MT Baseada em Transferência (tradução de Transfer-Based MT), os Sistemas de MT Interlíngua (tradução de Interlingual MT Systems), a MT Direcionada aos Dados (tradução de Data-Driven MT) e os Sistemas Híbridos (tradução de Hybrid Systems)..

Os Sistemas de MT Baseados em Dicionários Simples são a forma mais simples de MT, sendo baseados na transferência léxica, palavra a palavra. Apesar de muito simples, esses sistemas podem ser úteis em situações onde é necessária apenas uma idéia básica do texto a ser traduzido.

A MT Baseada em Transferência constrói regras gramaticais que o sistema utiliza para analisar as sentenças da língua fonte, mapear gramaticalmente as estruturas da gramática da linguagem destino e gerar as sentenças do texto final. Essas regras gramaticais são de desenvolvimento caro e demorado. Como resultado, os padrões das sentenças geralmente não são analisados corretamente quando as regras não foram completamente desenvolvidas. Essa alternativa de MT é muito dependente do conhecimento do projetista e do lingüista do software, de modo que um sistema de MT Baseado em Transferência pode levar mais de dois anos para ser desenvolvido. A grande maioria dos sistemas comerciais de MT é desse tipo.

Além desses, existem os Sistemas de MT Interlíngua. Nesse contexto, o significado da sentença original é traduzido para uma interlíngua (que pode ser uma língua natural, como o inglês, ou uma linguagem computacional, como a UNL) usada para gerar as sentenças em qualquer outra linguagem. Nessa técnica, os Sistemas de MT Baseados em Transferência são utilizados para fazer a conversão entre a língua de origem e a interlíngua

e entre a interlíngua e a língua destino. Esse paradigma é muito interessante à medida que o número de línguas cresce, como por exemplo no caso de um sistema para a tradução automática de sete línguas, onde seria necessário o desenvolvimento de 42 algoritmos de tradução em pares entre as línguas envolvidas. Pela complexidade dessa abordagem, ainda não existem Sistemas de MT Interlíngua disponíveis comercialmente, estando os mesmo restritos à pesquisas acadêmicas, como o caso da UNL, detalhada mais adiante neste capítulo.

Por sua vez, a MT Direcionada aos Dados utiliza a estratégia de compilar um grande número de traduções como exemplo, empregando métodos estatísticos a fim de computar quais peças gramaticais de cada texto de origem encaixam-se melhor com as peças dos texto de destino. Como as peças podem variar de simples palavras até sentenças completas, o sistema gera um dicionário e a tradução é feita automaticamente. Essa alternativa é muito dependente da quantidade de dados inseridas no sistema, de modo que o seu desenvolvimento tende a ser rápido, desde que os dados sejam abundantes. Atualmente são poucos os sistemas comerciais desse tipo.

Por fim, os Sistemas Híbridos visam combinar os pontos fortes dos diferentes tipos de sistemas de MT, minimizando as fraquezas. Exemplos incluem sistemas que utilizam a análise e geração de sentenças baseadas em regras conectados com regras de transferência direcionadas aos dados ou sistemas estatísticos que tentem diminuir a diferença entre traduções diferentes geradas para os mesmos textos.

3.3. Usos da Tradução por Máquinas

Entender para que a MT é utilizada e como esta técnica se relaciona com os outros processos de tradução é vital para entender os problemas que a MT pode solucionar. Como qualquer ferramenta, a MT funciona melhor sob um determinado conjunto de condições para as quais foi desenvolvida, de modo que o uso e conhecimento de tais condições é essencial para o correto uso e aproveitamento pleno de tais benefícios.

As aplicações sobre MT podem ser divididas em duas grandes categorias: as Aplicações Otimizadas pela MT (tradução de MT-Enhanced Applications) e as Aplicações Possibilitadas pela MT (tradução de MT-Enabled Applications). Além disso, o

entendimento das diferenças básicas entre a tradução humana e a MT é fundamental para a escolha da técnica correta nas diversas situações aplicáveis das mesmas.

As Aplicações Otimizadas pela MT são utilizadas para diminuir o custo e aumentar a eficiência da tradução humana. Essa técnica também pode ser chamada de Tradução Assistida por Computadores [6] (tradução de Computer Aided Translation, ou CAT). Nessa categoria, um documento na linguagem fonte é traduzido para uma ou mais linguagens destino de modo que a informação seja disponibilizada para leitores de outras culturas. Esse modelo remete ao processo tradicional de localização, que utiliza tradutores humanos. Nesse caso, a MT acelera o processo de localização fornecendo uma tradução padrão para a edição dos tradutores humanos, em vez de exigir a necessidade da criação de tal tradução do zero.

Além disso, a MT possibilita novos tipos de tradução onde seria impossível o uso da tradução humana, ou onde o custo e/ou tempo de tradução tornariam impossíveis a aplicação de tal técnica. Nesse contexto, o das Aplicações Possibilitadas por MT [7], podemos dividir o uso da MT em três grupos: MT para Informação, MT para Disseminação e MT para Comunicação.

A MT para Informação é utilizada para suprir a necessidade de informação onde o fornecedor da mesma na linguagem de origem não oferece uma alternativa para os leitores da linguagem destino. Podemos citar como exemplo o *gisting*, utilizado amplamente para situações onde o valor da informação não é proporcional ao custo da tradução da mesma. Nesse contexto, o uso da MT é utilizado apenas como referência das idéias básicas do texto. Em termos de volume essa é a forma mais comum de tradução utilizada atualmente. Como exemplo podemos citar os tradutores integrados a mecanismos de busca, tais como os tradutores do Google, Yahoo, etc.

No uso da MT para Disseminação, a técnica de tradução é utilizada para traduções de altos volumes de informação, onde as mesmas não variam muito em contexto, forma e semântica. Além disso, também é interessante em situações onde o custo da informação decresce rapidamente, como no caso de previsões meteorológicas, informações financeiras, etc. Nestas, os sistemas de MT podem ser configurados e adaptados especificamente de modo que a tradução instantânea mantenha o valor e a precisão da informação a ser traduzida.

Já a MT para Comunicação é utilizada em aplicações como *chat*, IM (Instant Messaging) e SMS (Short Message Service), pois as mesmas requerem agilidade de tradução, capacidade para altos volumes, tradução bidirecional, disponibilidade constante e, em alguns casos, confidencialidade das informações. Tais necessidades evidenciam as vantagens da MT em relação à tradução gerada por tradutores humanos.

A tabela abaixo resume as similaridades e diferenças entre os sistemas de MT e os de tradução humana.

	Sistemas de MT	**Tradução Humana**
Velocidade	Média de 4.800 páginas por dia. Tradução em tempo-real é possível.	Média de 10 páginas por dia. Tradução em tempo-real só é possível com intérpretes.
Exatidão	Sistemas configurados para um domínio restrito funcionam bem. Domínios, vocabulários e estilos de escrita não familiares diminuirão a exatidão.	Depende da técnica e do domínio do tradutor, sendo em geral alta.
Preço	Custo incremental por palavra abaixo de US$0,01 por palavra.	Custos diretos entre US$0,10-US$0,30, dependendo da linguagem
Consistência	Muito boa. Sistemas de MT são muito consistentes, mesmo nos seus erros.	Pode ser muito boa com memória de traduções e ferramentas de gerenciamento de terminologia.
Textos Repetitivos	Altamente recomendados para a MT.	Não recomendados para tradução humana, pela possível fadiga e tédio decorrente da atividade.
Escalabilidade	Pode ser rapidamente escalável para acomodar picos de demanda.	Necessidade de mais tradutores, treinamento e gerenciamento.
Disponibilidade	Sempre disponível.	Nem sempre disponível.
Qualidade	Variável, mas geralmente não tão boa quanto a tradução humana.	Variável, mas geralmente muito boa.
Tipos de Texto	Melhor para textos técnicos e assuntos pontuais em determinados domínios.	Requer treinamento para campos de conhecimento mais específicos.

Tabela 10 – Similaridades e diferenças entre os sistemas de MT e a tradução humana.

3.4. Benefícios e Custos da Tradução por Máquinas

A experiência mostra que o menor custo da MT é um fator crucial que leva ao incremento dos investimentos em localização e em outras formas de tradução em geral. Em muitos casos, a implementação de sistemas de MT torna a localização uma estratégia muito mais efetiva em termos de custos e tempo. Quando comparados com a tradução humana, muitas qualidades da MT destacam-se, de modo que uma análise de tais qualidades e potenciais custos se torna necessária.

Como benefícios diretos da MT podemos citar a redução dos custos de tradução para grandes volumes de informação, os tempos de entrega reduzidos (limitado apenas ao tempo necessário à revisão dos textos), a disponibilidade (os texto são processados à medida que são enviados) e o *throughput* (devido ao alto grau de escalabilidade da técnica).

Entre os benefícios indiretos podemos citar a redução dos custos de suporte (pela maior disponibilidade de informação e documentação em outras línguas), menor tempo no desenvolvimento de produtos (graças à diminuição do tempo e do custo da localização) e o aumento da lealdade à marca e/ou produto (advinda da aproximação com clientes de outras culturas).

Como custos, o mais óbvio deles é o custo de compra, instalação, manutenção e treinamento dos usuários de um sistema de MT. Além desses, existem os custos para desenvolvimento de dicionários (necessários para domínios técnicos específicos) e o custo de implantação de ferramentas especiais (*plug-ins* para e-mails, interfaces com outras ferramentas linguísticas, etc).

3.5. Avaliando, Escolhendo e Customizando um Sistema de Tradução por Máquinas

A avaliação de um sistema de MT está diretamente vinculada ao uso que será destinado ao mesmo. No caso de simples usuários, a simples escolha de textos aleatórios seguidos da tradução do mesmo já dá uma idéia básica da qualidade e dos resultados alcançados por estes sistemas, de modo que o usuário apenas necessita definir se a qualidade da tradução é boa o suficiente para o uso destinado ao texto. Entretanto, para os usuários corporativos, os sistemas de MT são componentes de tecnologia que devem ser

integrados a processos específicos da empresa a fim de serem corretamente auditados, não podendo ser avaliados separadamente por serem ferramentas para a construção de soluções e não as soluções por si próprias.

Infelizmente ainda não existem métricas padronizadas para a avaliação da qualidade de uma tradução de um texto, seja esta tradução feita por tradutores humanos ou sistemas de MT. Além disso, a avaliação é completamente subjetiva: usuários finais, tradutores e desenvolvedores de MT tendem a ter diferentes visões da tradução de um mesmo documento. Outro ponto é o fato de que para os leitores de cada linguagem a tolerância aos erros apresenta diferentes níveis, de modo que quanto maior o conhecimento da linguagem dos mesmos, maior o nível de exigência em relação à tradução do documento.

A melhor estratégia para avaliarmos um sistema de MT é através da identificação de qual dos sistemas de MT disponíveis no mercado atendem às necessidades do cliente em termos de linguagem, formato de documentos, etc. A partir destas alternativas, avaliar o quanto de trabalho será necessário para a adaptação aos seus documentos e aos seus processos, bem como o treinamento necessário para a adoção da ferramenta. Dessa maneira, fica evidente que um sistema de MT não pode ser avaliado pontualmente, sendo necessário um estudo e um período de testes a fim de quantificar o retorno do mesmo nos processos produtivos em que o sistema estará inserido.

Entretanto, existem maneiras práticas para a maximização da qualidade dos sistemas de MT. Uma delas é o estudo das fontes de erros e alternativas para evitá-los. Como apontado anteriormente, geralmente o autor do texto tem muito mais conhecimento acerca do tema dissertado do que o tradutor (seja ela humano ou o autor do dicionário do sistema de MT), de modo que, muitas vezes, ajustes no texto de origem são necessários.

Dessa maneira, fica evidente que a qualidade tradução é uma função da simplicidade e clareza do texto de origem. Quanto mais usual e restrita for a sintaxe desse texto, maior a chance do alcance de uma tradução confiável. Logo, a simples reformulação de frases no texto de origem pode gerar grandes mudanças no texto traduzido. Além disso, técnicas como a divisão de sentenças em outras menores através de artifícios de pontuação, mudança nas ordens do sujeitos, complementos, advérbios, etc, tendem a aumentar a qualidade do texto gerado.

Porém, na maioria dos casos não é possível ou viável fazer alterações no texto fonte antes do processo de tradução. Nesses casos, o aumento da qualidade da tradução depende de implementos no sistema de MT. A customização dos dicionários, a modificação nos algoritmos de análise semântica e o aumento do volume de textos pré-traduzidos são alternativas para o contorno dessas questões.

Os sistemas de MT podem ser customizados de duas maneiras: editando-os e/ou adicionando regras gramaticais (feitas pelo desenvolvedor) e customizando o dicionário (feitas pelo desenvolvedor e pelo cliente).

No caso das regras gramaticais do sistema de MT não conter uma estrutura que apareça comumente nos textos a serem traduzidos ou havendo uma má interpretação dessa estrutura, o sistema gerará uma tradução incorreta. Essas regras podem ser adicionadas e/ou alteradas pela equipe de desenvolvimento, num trabalho feito por lingüistas e programadores, ficando evidente a necessidade do *feedback* dos usuários para o constante aprimoramento dos sistemas de MT.

Em relação aos dicionários, assim como os tradutores humanos, os sistemas de MT são extremamente dependentes da correta terminologia para a realização de uma tradução confiável. Porém, diferente dos tradutores humanos, os sistemas de MT não podem consultar referências externas para a identificação de termos estranhos, de modo que é vital que a base de dados do dicionário seja a mais completa possível. Logo, além do dicionário básico original do sistema de MT, é necessário o desenvolvimento de dicionários específicos para o domínio do negócio da empresa em questão, maximizando o alcance das traduções geradas em cima de textos submersos nesse domínio.

Além da customização dos sistemas de MT outra abordagem interessante para a otimização dos resultados gerados é a customização dos processos de gerenciamento do conteúdo a fim de tornar a tradução destes mais efetiva. Um exemplo disto é a já abordada manipulação e otimização do texto de origem. Essa otimização pode ser auferida através do gerenciamento de terminologias, por exemplo, evitando o uso de sinônimos ou através da padronização da estrutura das sentenças, fazendo com que as mesmas tendam a seguir um padrão pré-estabelecido. Esse paradigma de desenvolvimento de conteúdo é conhecido como Autoria Controlada de Linguagem [8], de tal maneira que existem editores e padronizações a fim de auxiliar nesse processo.

3.6. Aplicações e Tendências para os Sistemas de Tradução por Máquinas

Os sistemas de MT vem sendo utilizados numa ampla gama de contextos, seja em textos técnicos ou em páginas *web*. Outrora restritos aos computadores de grande porte, a evolução dos processadores permitiu que esses sistemas estivessem mais próximos do usuário final, massificando o seu uso.

Como discutido acima, as Aplicações Possibilitadas pela MT são o grande nicho a ser explorado pelos profissionais gerenciadores de conteúdo, sendo a área onde as aplicações estão surgindo com maior força. Neste paradigma, o volume de informação é tão grande que seria necessário um exército de tradutores trabalhando diuturnamente a fim de realizar tais tarefas.

Como exemplo de uso da MT podemos citar a crescente integração com sistemas de mensagens eletrônicas, tais como os e-mails, *chats*, SMS ou qualquer sistema de comunicação de servidor a servidor. Nesses casos, o atraso de algum segundos causado pelo processamento do sistema de MT é mais do que válido graças a facilidade de um leitor em ler o texto na sua língua nativa.

Outro exemplo é a integração com sistemas de reconhecimento automático de voz. Esses sistemas evoluíram a ponto de ser possível obter excelentes resultados falando-se um pouco mais lentamente e claramente que o usual. O texto obtido pode servir como *input* para os sistemas de MT. O caminho contrário também é viável: já existem sistemas de MT que contam com ferramentas de *Text-to-Speech* (TTS) [9], que transformam o texto gerado em sons. Apesar da fala gerada não ser tão natural e clara como a fala humana, tais sistemas já se mostram viáveis para atuar em quiosques de informações, onde a repetição das mesmas perguntas e informações torna desnecessária a atuação humana em tal contexto.

Além destas abordagens, também podemos citar o uso dos sistemas de MT em Bancos de Dados e *feeds* de Dados. Um sistema de MT pode facilmente utilizar essas sentenças geradas automaticamente e com uma baixa amplitude de variação terminológica e traduzir as mesmas automaticamente para diferentes culturas, maximizando o valor e a utilidade destas informações. Sistemas financeiros e meteorológicos são os principais candidatos a este tipo de abordagem.

A gama de aplicações onde é possível utilizar sistemas de MT é muito ampla e a tendência é que continue crescendo. Com o amadurecimento da tecnologia, o aumento do volume de informação a ser compartilhada em diferentes línguas e o irreversível processo de globalização que testemunhamos hoje fica evidente a valia dessa ferramenta no aumento da produtividade global.

3.7. A Tradução por Máquinas e o Sistema UNL

O Projeto UNL tem sido comparado a projetos de MT, devido a sua característica de transposição interlingual de significados. No entanto, embora ele explore o uso de computadores para a comunicação em massa, que envolve um alto grau de complexidade no que diz respeito ao tratamento da língua como instrumento de comunicação, o Sistema UNL é bastante diferente dos sistemas de MT, em função de seus objetivos, técnicas e organização. Algumas das principais razões para essa diferenciação seguem abaixo:

- O objetivo do Sistema UNL é agrupar tecnologias existentes para desenvolver um sistema robusto em grande escala, para uso cotidiano. Os sistemas de MT, por sua vez, são geralmente baseados em domínios limitados ou em tecnologias já conhecidas.

- A metodologia adotada no Sistema UNL é a metodologia de interlíngua, enquanto a maioria dos sistemas de MT usa técnicas diretas ou técnicas de transferência. Nas técnicas diretas ou de transferência, procura-se, em geral, uma correspondência real entre palavras ou padrões frasais e estruturas da língua fonte com palavras ou estruturas correspondentes da língua destino. Enquanto na tradução direta, por exemplo, rotinas de substituição e reconhecimento/casamento de padrões altamente específicos são utilizadas para permitir a troca de palavras ou estruturas da língua fonte por palavras ou estruturas correspondentes da língua destino, nos sistemas de transferência obtém-se uma estrutura representativa da sentença na língua original (por meio de um processo de análise) e é esta estrutura que é transferida aos padrões da língua destino, para então ser sintetizada na forma textual.

No caso do uso de interlíngua, não há a necessidade de módulos particulares para cada par de línguas envolvido na tradução. Somente dois estágios são necessários: o de

mapeamento de sentenças da língua fonte em uma representação neutra em relação às línguas envolvidas no processo e o de linearização dessa representação neutra em sentenças da língua destino. A Linguagem UNL pretende ser o componente neutro em relação às características particulares de cada língua, pois ela sugere a preservação dos conceitos semânticos independentemente das línguas envolvidas.

Dessa forma, a UNL visa solucionar o problema principal da tradução direta, que está, em geral, na incapacidade de se especificar uma correspondência um-para-um entre as construções lingüísticas das línguas envolvidas (este é o problema intrínseco da tradução literal, para o qual o esforço computacional é da ordem de n^2 mecanismos de tradução em pares, para n línguas envolvidas). Na tradução por transferência, o esforço computacional também é da ordem de n^2, pois são necessários n módulos de análise, n de síntese e n^2 de transferência. No Sistema UNL, ao contrário, o uso da UNL implica a complexidade de ordem n: para n línguas, são necessários $2n$ sistemas, n para a codificação para a interlíngua e n para a decodificação da interlíngua para a língua destino.

• O Sistema UNL está organizado de tal forma que vários grupos situados em posições geográficas distintas trabalhem independentemente, utilizando técnicas variadas para desenvolver módulos de processamento de línguas diferentes. Nos projetos de MT, é comum que um grupo de pessoas, dentre as quais projetistas do sistema, programadores e especialistas em cada língua sob enfoque, formem um único grupo, trabalhando na mesma localidade. O grupo é formado para que todos os seus membros trabalhem com base em um mesmo modelo teórico e sigam uma mesma metodologia. O maior problema deste tipo de organização está na dificuldade de interação entre profissionais para realizar a tarefa de MT: dificilmente indivíduos de origens, línguas e culturas distintas irão compartilhar um único modelo teórico que seja compatível com os objetivos do projeto. No Sistema UNL, ao contrário, os especialistas, distantes entre si, devem conhecer somente as questões relativas à sua própria língua, tendo em comum somente o Sistema UNL.

Neste projeto, os processos de codificação/decodificação são entendidos, assim, como processos de representação de sentenças em línguas distintas, sendo que a UNL

constitui uma representação intermediária que dispensa o trabalho de tradução, i.e., ela pretende ser uma linguagem suficientemente global e transparente a todas as línguas envolvidas. Sua vantagem principal, portanto, está na uniformidade alcançada com uma representação do conhecimento comum a várias línguas naturais. No capítulo seguinte abordaremos o Projeto UNL, elucidando seus aspectos técnicos e por fim, focalizando nos processos de conversão e deconversão de conteúdo.

4. O PROJETO UNL

O Projeto *Universal Networking Language* (Projeto UNL) é um projeto financiado pela Universidade das Nações Unidas (*The United Nations University - UNU)*, mais particularmente pelo Instituto de Estudos Avançados (*Institute of Advanced Studies - IAS/UNU)*, com sede em Tóquio. Seu principal objetivo é promover e facilitar a comunicação internacional por meio do uso de sistemas computacionais de Tradução por Máquinas (MT).

A idéia central do Projeto UNL está no desenvolvimento da Linguagem UNL, que deverá ser usada em um ambiente Internet a fim de facilitar o fluxo de informação entre indivíduos de qualquer parte do mundo. Desse modo, a UNL deve servir de meio de troca de informação para que as pessoas se comuniquem em suas próprias línguas nativas [9].

O problema central deste projeto se coloca a partir da necessidade de superar a barreira lingüística, com a utilização de uma representação intermediária das mensagens que cada usuário de computador, uma vez conectado a uma rede de computadores, planeja enviar a destinatários cujas línguas nativas não necessariamente sejam comuns.

O Projeto UNL também procura estimular e promover o desenvolvimento de tecnologias de engenharia lingüística e da MT, além de criar um fórum para discussões internacionais e colaborações em grande escala.

4.1. Objetivos do Projeto UNL

O Projeto UNL tem por objetivo disponibilizar a UNL em larga escala, por meio de pacotes de software de conversão (de cada língua nativa para a UNL) e deconversão (da UNL para uma determinada língua nativa) armazenados, por exemplo, em servidores WWW, a fim de serem manipulados por *browsers*. Informações disponíveis nesse ambiente poderão ser convertidas/deconvertidas, dependendo da necessidade do usuário. O Projeto UNL busca, assim, uma integração comunicativa em um ambiente de 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. Torna-se possível, portanto, que:

• os cidadãos obtenham e enviem informações para usuários de qualquer parte do mundo em suas próprias línguas nativas,

• pesquisadores, educadores, empresários e industriais tenham a mesma oportunidade de acesso à informação disponível em línguas que não a sua, a fim de garantir um desenvolvimento industrial e econômico igualitário entre as nações,

• estados-membros das Nações Unidas eliminem as barreiras da linguagem, visando um melhor entendimento global,

• estados-membros das Nações Unidas compartilhem a tecnologia resultante do Projeto, a fim de produzir novos conhecimentos.

Para viabilizar o Projeto UNL, servidores padrões de rede devem ser compatíveis mundialmente. O software UNL produzido também deve ser compatível com os servidores de rede e com as plataformas computacionais em uso atualmente.

4.1.1. Metodologia de Desenvolvimento do Projeto UNL

O Projeto UNL é desenvolvido fundamentalmente em regime de parceria, sendo que o IAS/UNU lidera os trabalhos das instituições signatárias, detendo os direitos sobre o mesmo. Em particular, o IAS/UNU é responsável pela criação e especificação da Linguagem UNL e pelo fornecimento de pacotes de desenvolvimento dos codificadores e decodificadores para qualquer língua nativa. Esses pacotes consistem em protótipos de codificadores e decodificadores fornecidos a institutos de pesquisa internacionais e parceiros industriais, que serão responsáveis pelo desenvolvimento dos módulos individuais de conversão e deconversão específicos para cada língua nativa.

Os módulos específicos podem ser desenvolvidos independentemente do sistema central, permitindo que novos parceiros se integrem ao projeto em diferentes fases do mesmo.

Um sistema experimental começou a ser implementado pelo IAS/UNU, em colaboração com institutos de pesquisa e companhias do Brasil, China, Egito, França, Alemanha, Índia, Indonésia, Itália, Japão, Jordânia, Mongólia, Rússia e Espanha. Os seguintes resultados são almejados:

• a Linguagem UNL em si mesma, disponibilizada a todas as nações;

• os softwares de conversão e deconversão para qualquer língua nativa, compatíveis com *browsers* da WWW e com servidores padrões de rede;

• especificações e diretrizes técnicas para o desenvolvimento de cada módulo específico.

4.1.2. O Papel do Brasil no Projeto UNL

Segundo a perspectiva dos pesquisadores brasileiros envolvidos no Projeto UNL, atualmente há três áreas de aplicação sob enfoque, sendo elas: a de MT, a de Redes de Computadores e a de Ferramentas de Suporte. Essas áreas determinam, respectivamente, o projeto e desenvolvimento dos módulos relativos à língua portuguesa e à comunicação e interface dos pacotes de conversão/deconversão para a língua portuguesa com a Internet.

O módulo de processamento do português está a cargo do Instituto UNDL Brasil, sediado em Florianópolis-SC. É sobre esse módulo que se fundamenta esse trabalho. A este subprojeto específico damos o nome de *Projeto UNDL Brasil* [10].

Especialistas de diversas instituições estão envolvidos no Projeto UNDL Brasil. Pode se destacar, particularmente, a USP-São Carlos, UFScar e UNESP-Araraquara. O Instituto UNDL Brasil tem a função de projetar e desenvolver os módulos de conversão e deconversão para o português, de acordo com as diretrizes do Projeto, que envolvem: um processo semi-automático de conversão de qualquer língua nativa para a Linguagem UNL, e um processo automático de deconversão de sentenças UNL para qualquer sentença em língua natural. Tais aplicações foram mais informalmente batizadas de EnCo e DeCo.

O primeiro processo permite que o próprio projetista efetue alterações no sistema, uma vez que tenha avaliado os resultados automáticos obtidos da análise de sentenças em uma língua nativa. Assim, ele pode modificá-la em tempo real para obter uma representação mais consistente com a sentença original.

4.2. A Linguagem UNL

A UNL, no contexto do projeto coordenado pela UNU, é vista como uma interlíngua capaz de representar de forma única o conteúdo semântico de uma sentença escrita em qualquer língua natural. Juntamente com sistemas de conversão e deconversão, ela deverá permitir a comunicação entre indivíduos de diversas línguas nativas, devendo residir como um *plug-in* para *browsers* WWW populares na Internet. Para tanto, a UNL deverá ser compatível com servidores padrões de rede. Qualquer pessoa com acesso à Internet poderá codificar um texto escrito em sua língua nativa usando a UNL, para que o mesmo seja decodificado por qualquer leitor de uma língua distinta.

Mais especificamente, a UNL é uma metalinguagem que serve para descrever aspectos especiais do significado de sentenças, tais como as relações semânticas que podem ser representadas por relações formais (morfológicas ou sintáticas) entre palavras de uma sentença [10]. A Figura 2 representa a codificação e decodificação de documentos no sistema UNL.

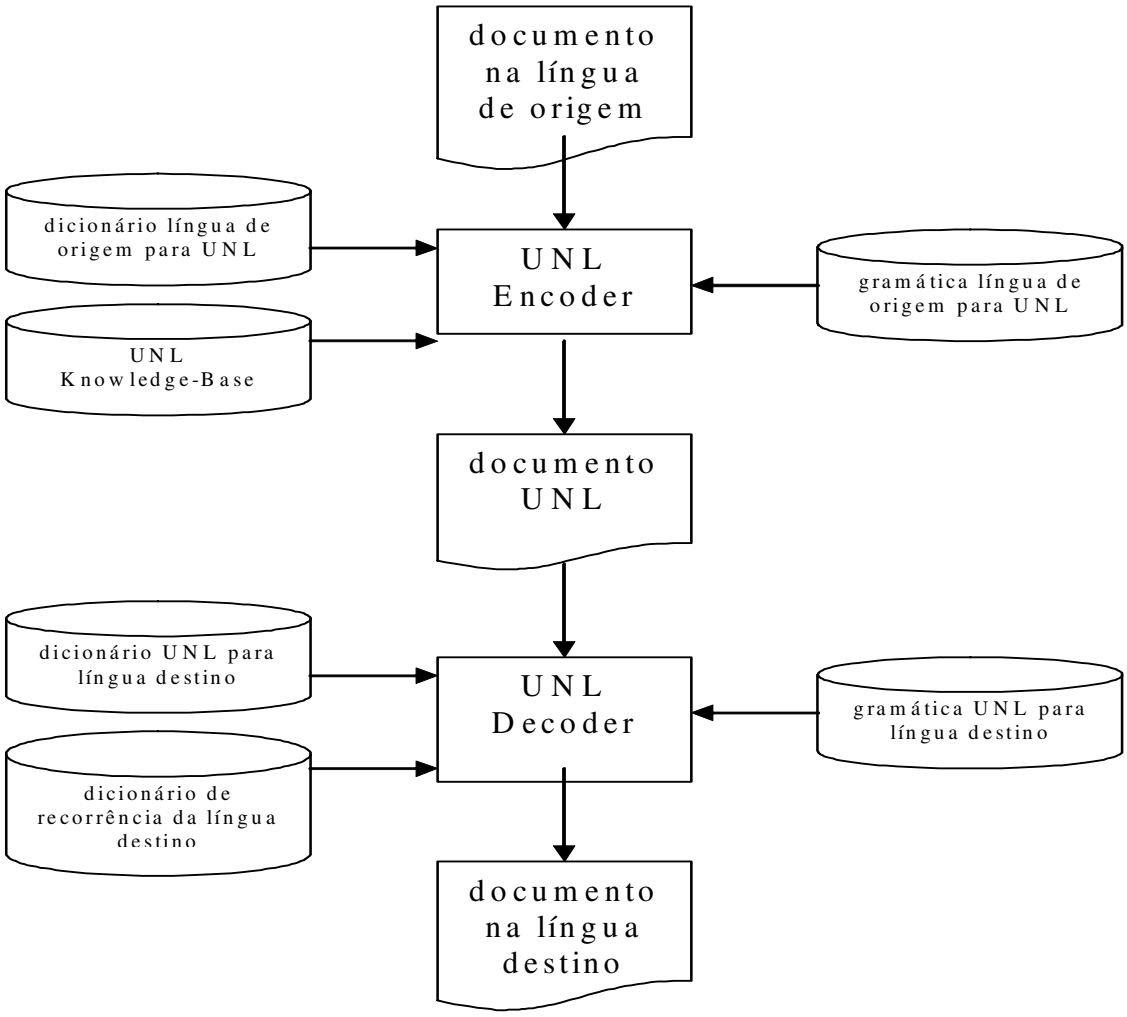

Figura 2 – A codificação e decodificação de documentos no sistema UNL

Buscando atingir objetivos similares, a UNL é capaz de representar de maneira uniforme o significado de orações que podem ser descritas e explicadas a partir de diferentes modelos gramaticais, pois se baseia na análise gramatical das mesmas e, logo, na representação superficial de suas estruturas textuais. Embora ainda limitada, ela apresenta uma proposta viável de automação de processos lingüístico-computacionais, devendo permitir que expressões lingüísticas de um mesmo fenômeno, ainda que variadas, possam ser representadas em UNL de forma similar e consistente. Torna-se possível, então, que as estruturas semânticas de descrições de fenômenos sejam compartilhadas por indivíduos do mundo todo.

4.2.1. A Representação do Significado na Linguagem UNL

Apesar de ainda rudimentar, a UNL fornece uma visão muito prática de um sistema de MT: a de permitir o tratamento de aspectos semânticos de forma sistemática e independente do conhecimento profundo sobre questões não textuais. Ela se restringe, por exemplo, ao significado literal (e, logo, denotativo) de sentenças e trata de descrições de significados frasais seguindo a abordagem tradicional de relacionamento de papéis semânticos a entidades ou objetos componentes da frase. Ela permite representar somente uma parte da frase, constituída nas relações entre elementos e relações gramaticais, tais como sufixação ou subordinação. Ela não abrange alguns pressupostos teóricos importantes para a comunicação, tais como as questões comunicativas, estilísticas, intencionais ou retóricas, que levam ao significado conotativo, de natureza pragmática. Apesar dessas limitações, a UNL permite reconhecer algumas expressões idiomáticas, que são indexadas no léxico com base em interpretações literais. Este é o elo mais explícito, na UNL, da tentativa de se processar informações de caráter conotativo. Desse modo, suas limitações indicam também a exclusão do tratamento da recepção da linguagem, tida no Projeto UNL como um problema exclusivo do leitor. Ela promove, portanto, somente a comunicação básica entre indivíduos de línguas nativas distintas e não a comunicação natural e abrangente que existiria entre indivíduos que partilhassem a mesma língua. Apesar de básica, a UNL apresenta, no momento, trinta e cinco relações semânticas sentenciais, que a situam na média das linguagens de PALN (Processamento Automático de Linguagem Natural) existentes [11].

A característica principal da UNL é a de privilegiar a predicação entre elementos lingüísticos, por sua caracterização individual ou relacional (com outros elementos lingüísticos) durante a ocorrência de eventos ou outras relações complexas entre eventos. Neste caso, os próprios argumentos verbais ou adjuntos frasais se manifestam como predicados, sendo responsáveis por preservar a relevância da informação textual. As predicações podem, portanto, ser relativamente simples, como um simples adjetivo, ou complexas, como em uma subordinação. Privilegiando as predicações, a UNL não fornece mecanismos para tratar questões referenciais, tais como construções elípticas ou referências introduzidas por pronomes demonstrativos (este, esta, aquele, etc.), pois ela não consegue

recuperar o sentido e a referência aos componentes individuais em questão. No entanto, ela sugere uma técnica para se recuperar itens co-referentes no contexto imediato, pela co-indexação dos componentes das frases. Porém, antes que testes extensivos sejam realizados não é possível antever como (e se) esta técnica permitirá garantir o correto relacionamento semântico em uma estrutura abrangente de significado.

4.2.2. A Estruturação do Significado

Em seu vocabulário, a UNL privilegia os aspectos estruturais do significado, ao invés dos aspectos da semântica lexical (ou composicional). Ela se baseia em conceitos relacionados a sistemas de *relações primitivas semanticamente universais* [10]. O componente composicional, entretanto, não é ignorado no processo de representação interlingual, pois a semântica lexical é incorporada ao léxico, como um ponto de partida para o processamento subseqüente. Desse modo, parte da representação semântica já é recuperada pelas próprias entradas lexicais. Em especial, incorpora-se ao léxico as interpretações particulares da língua em cada um de seus verbetes e são esses que passam a corresponder ao vocabulário UNL.

Sintaticamente, uma gramática de estrutura de frase representa as formas lingüísticas, definidas por elementos que pertencem a classes de palavras e por estruturas mais complexas (sintagmas ou frases) decorrentes destas. Logicamente, a UNL sugere uma representação proposicional cujos conceitos são classificados hierarquicamente, de tal forma que sua taxonomia permite criar estruturas derivadas. A articulação do significado pode dar-se, p.ex., por relações agente-recipiente, causa-efeito, ação-instrumento, entidade-atributo.

No momento, o conjunto de relações primitivas semânticas ainda não se encontra suficientemente "estabilizado" na UNL, pois ela passa por um processo de análise de sua representatividade e abrangência. Uma das dificuldades no tratamento morfossintático necessário para a associação léxico-semântica nesse projeto diz respeito ao próprio objetivo da UNL: embora ela sugira a utilização de conceitos semânticos na tentativa de se partilhar o significado entre línguas distintas e embora a própria semântica tenha, tradicionalmente, como escopo "o estudo do significado", duas concepções se contrapõem em termos de

teorias sobre o significado, que são importantes do ponto de vista de fundamentação teórica no Projeto UNL. São elas [12]:

- a *externalista*, segundo a qual o significado é identificado com as coisas e características do meio em que uma expressão é usada e, dessa forma, o conjunto de significados passa a ser igual ao conjunto de objetos do mundo (i.e., a semântica passa a ser o estudo das coisas do mundo e suas características) ou passa a ser identificado com o comportamento do locutor ou do ouvinte (i.e., a semântica passa a ser o estudo do comportamento humano);

- a *internalista*, segundo a qual o significado é identificado com entidades ou características mentais, tais como idéias ou estruturas conceituais (i.e., a semântica passa a ser o estudo das entidades mentais).

A UNL toma por base ambas as concepções, uma vez que lança mão dos padrões comuns a cada uma delas: o significado de expressões do locutor (no caso, o locutor de qualquer língua a ser processada no âmbito do Projeto UNL), o significado de expressões do ouvinte e o significado das próprias entidades conceituais, construídas em função dos conceitos semânticos primitivos e expressas estruturalmente na forma de predicações entre os objetos do mundo.

Entretanto, por priorizar a *semântica relacional*, i.e., a abordagem da descrição do significado pela associação relativa entre palavras, o Projeto UNL também promove a semântica como o "estudo das relações de significação", no que diz respeito à referência, denotação e designação. Nesta concepção a semântica compreende "a construção de teorias-ponte entre a gramática e o significado" [12]. Esta é a visão que deve ser adotada: a da UNL como uma ferramenta para elaborar a relação entre gramática e significado e, portanto, como uma linguagem que permite o estudo das relações de significação pela associação relativa das entidades e características lingüísticas a entidades do significado. Nessa visão, a UNL torna possível o estudo do inter-relacionamento entre os componentes lingüísticos a fim de superar as dificuldades de representação semântica para a descrição do significado. Dessa forma, o potencial de conhecimento de falantes nativos de cada língua envolvida no projeto, independentemente do ponto de vista do ouvinte, é explorado. No estágio atual, esse potencial é explorado pelo próprio lingüista, no papel que ele tem de

engenheiro do conhecimento na construção do sistema de deconversão. A elaboração da relação gramática/significado, propriamente dita, se dá pelo conjunto de regras gramaticais especificadas para o funcionamento do DeCo.

Procurando conjugar os dois sistemas de representação – o de estrutura gramatical e o de interpretações semânticas – a UNL serve, portanto, para projetar entidades de sistemas de representação da estrutura gramatical em sistemas de representação de interpretações semânticas e vice-versa. Por exemplo, na conversão, ela é usada para mapear uma sentença escrita em língua natural (sentença de origem) em um conjunto de relações entre significados (sentença UNL), sendo que conjuntos de relações são relativos a proposições, no sistema de representação. Na deconversão, regras gramaticais de uma língua destino qualquer são aplicadas à sentença UNL, para gerar a estrutura gramatical correspondente, a partir da qual se obtém a forma superficial na língua destino. Na conversão, faz-se a projeção entre gramática e significado delimitando-se os componentes frasais da sentença de origem e extraindo-se sua estrutura sintática, para então se obter a sentença UNL. Na deconversão, faz-se a projeção do significado em características gramaticais da língua destino, resolvendo-se as questões léxico-gramaticais particulares daquela língua. Da forma como foi proposto, o Projeto UNL se concentra, prioritariamente. em informações textuais e, logo, em estruturas textuais, opostamente a estruturas discursivas, sendo que a associação entre língua de origem e língua destino durante a comunicação se dá por meio da UNL, que serve de interface entre línguas diferentes. Os processos de conversão e deconversão serão abordados em mais detalhes no capítulo a seguir.

4.2.3. O Léxico Segundo a Perspectiva da UNL

A especificação dos conceitos no léxico abrange informações sobre a semântica lexical (visão externalista), assim como informações sobre as relações semânticas entre os próprios conceitos (visão internalista). Ou seja, o significado das palavras é expresso por algumas relações usadas para descrever o significado de uma sentença, fazendo com que a própria representação lexical contribua para a ênfase aos aspectos estruturais do significado.

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*). Há ainda uma ontologia que estrutura o léxico. Suas 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 (no caso, um lingüista). 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.

As UWs são, portanto, entradas do léxico que especificam conceitos individuais em termos de sua relação com outros conceitos, assim como a correspondência de cada conceito com a forma que ele deve tomar em determinada língua. A relação interconceitual diz respeito à semântica composicional das palavras: uma UW principal e todas as suas ramificações são definidas em termos da composição de significados que ela possa apresentar. Devido a essa característica, o léxico é, na verdade, um dicionário que faz a correspondência entre UWs e verbetes no português (dicionário UWs-português).

Assim como se faz uso de RLs para se chegar ao significado pertinente e, logo, a uma UW particular, faz-se uso dos ALs (rótulos de atributos) para limitar o significado das UWs. Desse modo, o uso dos componentes sentenciais indica a semântica lexical incorporada ao léxico: a cada UW é associado um significado particular, representativo das acepções mais freqüentes da língua portuguesa (essas acepções recebem o nome de *headwords*).

A sintaxe do dicionário UWs-português obedece o seguinte formato [13]:

[headword] canônica "UW" (grammatical features) <P,f,p>;

onde: *headword* é a palavra do português, correspondente ao significado expresso pela UW;

"canônica" é a forma canônica da palavra em português;

(*grammatical features)* é o conjunto de traços gramaticais e semânticos da *headword*;

P: denota português;

f e **p** são valores naturais que exprimem a freqüência e a prioridade de uso da *headword.*

As entradas da Tabela 11 ilustram o conteúdo de tal dicionário antes do processo de associação com o português. Todas as entradas derivadas de uma única palavra em inglês correspondem às suas diferentes acepções. Vale notar as restrições semânticas impostas à UW mais genérica "*threaten*" em cada caso, originando diferentes acepções. Estas restrições envolvem RLs (p.ex., *agt*, *obj*) e outras UWs (p.e., *danger*, *human*, *trouble*). Vale notar, ainda, que cada restrição distinta associada à UW genérica implica uma nova UW.

```
[]{} "threaten" ();
      []{} "threaten(agt>human,obj>danger)" ();
      []{} "threaten(agt>human,obj>entity)" ();
      []{} "threaten(agt>human,obj>human)" ();
      []{} "threaten(agt>human,obj>trouble)" ();
      []{} "threaten(icl>event)" ();
      []{} "threaten(icl>event,obj>human)" ();
```

Tabela 11 – Entradas do dicionário de UWs derivadas da palavra "threaten"

Em vista das características do DeCo, adota-se como forma de representação das *headwords* em português, as formas analisadas, i.e., as representações de radicais. O principal impacto dessa decisão está na representação da classe de verbos, cujo processo flexional passa a ocorrer via regras. Entretanto, na classe de nomes (substantivos, adjetivos), prevê-se tanto formas analisadas quanto formas flexionadas, a fim de se contemplar a geração das flexões de gênero e número. A Tabela 12 mostra um exemplo de entrada analisada ("perfeit") e um exemplo de entrada flexionada ("transmissão"). Esta figura também ilustra a forma final das entradas do dicionário, após sua associação com o português.

smooth(aoj>movement)

[perfeit] {} perfeito

"smooth(aoj>movement)"(stem,plural,1arg,rege(de)(em)) <P,0,0>;

communication(icl>connection)

[transmissão] {} transmissão

”communication(icl>connection"(stem,^alomorfe,fem,2arg,rege(a)(para)(por)(de),deverbais,comum) <P,0,0>;

"communication(icl>connection)"

[transmissõ] {} transmissão

(stem,alomorfe,plural(es),2arg,rege(a)(para)(por)(de),deverbais) <P,0,0>;

Tabela 12 – Exemplos de entradas lexicais analisadas e não analisadas

O processo de associação de *headwords* é feito de duas formas: a) tratando-se as classes fechadas, ou seja, artigos, pronomes, conjunções, etc.; b) tratando-se as classes abertas, de verbos e nomes. Neste último caso, por se tratarem de classes grandes, duas etapas são realizadas: a associação da palavra em português correspondente àquela acepção e a atribuição dos traços gramaticais e semânticos a cada entrada do dicionário. As classes fechadas não oferecem maiores desafios e, atualmente, é considerada completa. Para as classes abertas, no entanto, as etapas são complexas e demoradas, embora possamos lançar mão de algum processamento automático. As restrições semânticas que compõem a maioria das UWs, no entanto, impedem que este processo seja completamente automático. O preenchimento dos traços semânticos e gramaticais também pode envolver um pré-processamento automático, uma vez que a base do léxico é um dicionário do português categorizado gramaticalmente, com cerca de 1,5 milhão de entradas [14]. No entanto, como não existem informações semânticas neste dicionário, o preenchimento é exaustivo e requer supervisão humana. As classes de itens gramaticais no dicionário UWs-português compreendem os substantivos, adjetivos, verbos, advérbios, pronomes, conjunções, artigos, numerais, preposições, assim como locuções proposicionais, pronominais e adverbiais. Os traços gramaticais são particulares e pertinentes a cada uma das classes.

De acordo com o exposto, o significado é identificado por restrições paradigmáticas ou sintagmáticas. No primeiro caso, distingue-se sinonímia e hiperonímia; no segundo caso, distingue-se os tipos de argumentos das palavras na frase. Cada significado distinto corresponde a uma entrada lexical no dicionário, sendo que as diversas acepções do seu verbete são definidas em função dos RLs. Os ALs, por sua vez, são usados para expressar variações ou restrições de significado, sem que haja qualquer interferência na construção ou acesso ao dicionário.

As trinta e cinco relações semânticas sentenciais atualmente definidas na UNL representam as possíveis formas de relacionamento intra-sentencial das UWs. O objetivo central de cada software conversor e deconversor de UNL é automatizar o mapeamento entre essas relações semânticas e as características gramaticais correspondentes em determinada língua nativa.

4.3. O Vocabulário UNL

O vocabulário UNL apresentado a seguir se baseia na especificação atualda Linguagem UNL [15], incluída como anexo ao fim desse trabalho. É sobre esta versão que foi desenvolvida a primeira etapa do mapeamento UNL-português utilizada no EnCo e no DeCo. Atualmente, essa correspondência está sendo avaliada e criticada em função de alterações do próprio vocabulário, que serão apresentadas futuramente.

4.3.1. As Palavras Universais (UWs)

A função de uma UW é denotar um significado específico. Sua representação genérica é ou um rótulo simples (que indica o significado genérico de uma palavra em inglês), ou um rótulo limitado por um intervalo específico, que denota significados distintos, quando há ambigüidade em relação à palavra original. Por exemplo, o significado "book" permite a representação das seguintes UWS: **book** = (*livro*), **book(icl>publication)** = (*publicações*), **book(inc>account)** = (l*ivro comercial)* e book(obj>room) = (*reserva de um quarto*). A primeira UW, **book**, é a representação mais genérica do significado. As

demais limitam este significado a conceitos particulares, desfazendo, desse modo, a ambigüidade natural dessa palavra [15].

4.3.2. Os Rótulos de Relação (RLs)

RLs servem para expressar relações binárias entre significados, i.e., entre duas UWs distintas. Sua representação geral é dada por um par ordenado do tipo *relation_label(UW1, UW2)*, onde UW_1 e UW_2 são duas UWs diferentes relacionadas pela relação semântica indicada por *relation_label*. Há diversas classes de RLs. Por exemplo:

RLs entre Componentes Intra-sentenciais (em número de trinta e três)

***agt*: Agente.** Um agente que causa uma ação autônoma, i.e., um objeto animado com intenções. Exemplo:

Sentença original: *A lebre corre*.

Representação UNL: `agt(run.@entry.@pred.@present, rabbit.@def)`

***and*. Conjunção.** Conjunção de objetos ou eventos. Exemplo:

Sentença original: *Gatos e ratos são animais.*

Representação UNL: `aoj(and(cat.@generic,rat.@generic),animal)`

***aoj*: Objeto atributivo.** Um objeto de um atributo. Exemplo:

Sentença original: *Folhas são verdes*.

Representação UNL: `aoj(green,leaf.@generic.@pl)`

***bas*: Base.** Critério para comparação. Exemplo:

Sentença original: *Um gato é maior que um rato.*

Representação UNL: `aoj(cat.@generic,big), bas(big,mouse.@generic)`

***ben*: Beneficiário.** Beneficiário de eventos. Exemplo:

Frase original: *Trabalhar para uma família.*

Representação UNL: `ben(work,family.@indef)`

***cag*. Concomitância/co-agência.** Concomitante/co-agente. Exemplo:

Frase original: *Andar com um amigo.*

Representação UNL: `cag(walk,friend.@indef)`

***cau*: Causa.** Causa de um evento. Exemplo:

Sentença original: *Ele morreu de câncer*.

Representação UNL: `obj(die.@entry.@pred.@past, He),`
`cau(die.@entry.@pred.@past,cancer.@generic)`

***con*: Condição.** Condição que causa (voluntária ou involuntariamente) a ocorrência de um evento. Exemplo:

Sentença original: *Eles voltaram para casa devido à chuva*.

Representação UNL: `agt(come_back.@entry.@pred.@past, They),`
`gol(come_back.@entry.@pred.@past, house.@generic),`
`con(come_back.@entry.@pred.@past, rain.@generic)`

***coo*: Co-ocorrência.** Progressão simultânea de eventos. Exemplo:

Sentença original: *Eles voltaram para casa chorando*.

Representação UNL: `agt(come_back.@entry.@pred.@past, They),`
`gol(come_back.@entry.@pred.@past, house.@generic),`
`coo(come_back.@entry.@pred.@past, cry.@progressive)`

***fmt*: Origem-destino.** Abrangência de objetos/eventos. Exemplo:

Frase original: *Uma passagem de Florianópolis a Curitiba.*

Representação UNL: `mod(ticket.@indef,fmt(Florianópolis,Curitiba))`

***gol*: Objetivo.** Local (físico ou lógico) de um agente/objeto relativo a um evento. Exemplo:

Sentença original: *As crianças foram para Londres*.

Representação UNL: `agt(go.@entry.@pred.@past, child.@pl.@def),`
`gol(go.@entry.@pred.@past,Londres)`

***ins*: Instrumento.** Um instrumento utilizado em uma ação volitiva. Exemplo:

Sentença original: *A criança se cortou com a faca*.

Representação UNL: `agt(cut.@entry.@pred.@past,child.@def),`
`ins(cut.@entry.@pred,knife.@def)`

***lpl*: Lugar lógico.** Cenário de uma ação. Exemplo:

Sentença original: *Martina se machucou na competição*.

Representação UNL: `obj(hurt.@entry.@pred.@past, Martina),`
`lpl(hurt.@entry.@pred.@past, competition.@def)`

***man*: Maneira.** O modo de uma ação ou mudança introduzida pela ação. Exemplo:

Sentença original: *A criança corre rápido*.

Representação UNL: `agt(run.@entry.@pred.@present,child.@def), man(run.@entry.@pred,fast)`

***mat*: Material.** Material usado em uma ação. Exemplo:

Sentença original: *Ela fez manteiga com a nata do leite*.

Representação UNL: `agt(make.@entry.@pred.@past, She), obj(make.@entry.@pred.@past,butter.@generic), mat(make.@entry.@pred.@past,milkskim.@generic)`

***met*: Método.** Relação de meios ou métodos segundo os quais uma ação volitiva é realizada. Exemplo:

Sentença original: *Ele sarou por meio de uma cirurgia*.

Representação UNL: `agt(cure.@entry.@pred.@past,He), met(cure.@entry.@pred.@past,surgery.@indef)`

***mod*: Modificador.** Modificação de um objeto. Exemplo:

Frase original: *O terceiro homem.*

Representação UNL: `mod(man.@def,third)`

***num*: Número.** Número de unidades de um objeto. Exemplo:

Frase original: *3 kg*.

Representação UNL: `num(kg,3)`

***obj*. Objeto.** Um objeto de uma ação ou uma mudança que afeta o objeto. Exemplo:

Sentença original 1: *Pedro come maçãs.*

Representação UNL: `agt(eat.@entry.@pred.@present,Pedro), obj(eat.@entry.@pred,apple.@generic.@pl)`

***opl*: Lugar objetivo.** O local relacionado a uma ação. Exemplo:

Sentença original: *Ela anda na lama.*

Representação UNL: `agt(walk.@entry.@pred.@present, She), opl(walk.@entry.@pred.@present, mud.@generic)`

***or*. Disjunção.** Disjunção de objetos ou eventos. Exemplo:

Sentença original: *Manter um gato ou um rato.*

Representação UNL: `obj(keep.@entry,or(cat.@indef,rat.@indef)`

***per*. Unidade de medida.** Unidade de medida de um objeto. Exemplo:

Frase original: *Dois dias por semana.*

Representação UNL: `per(num(day,2),week)`

***pos*: Possuidor.** Dono de um objeto. Exemplo:

Frase original: *O cachorro de João.*

Representação UNL: `pos(dog.@def,João)`

***ppl*: Lugar físico.** Local onde ocorre uma ação. Exemplo:

Sentença original: *As crianças brincavam no jardim*.

Representação UNL: `agt(play.@entry.@pred.@past, children.@def.@pl), ppl(play.@entry.@pred.@past, garden.@def)`

***ptn*: Companheiro.** Companheiro de uma ação que requer cooperação simétrica. Exemplo:

Frase original: *Competir com um amigo*

Representação UNL: `ptn(compete,friend.@indef)`

***pur*: Propósito.** O propósito de uma ação volitiva. Exemplo:

Sentença original: *Venha para me ver.*

Representação UNL: `pur(come.@entry.@pred.@imperative,see), obj(see,I)`

***qua*: Quantidade.** Unidade associada à quantidade de um objeto/quantia ou grau de mudança. Exemplo:

Sentença original 1: *Ana comprou 3 kg de maçãs.*

Representação UNL: `agt(buy.@entry.@pred.@past,Ana), obj(buy.@entry.@pred.@past,apple.@generic), qua(apple,kg),num(kg,3)`

***seq*: Seqüencia.** Um evento de seu sucessor. Exemplo:

Sentença original: *Ana foi à biblioteca e retirou um livro*.

Representação UNL: `agt(go.@entry.@pred.@past,Ana), gol(go.@entry.@pred.@past,library.@generic), agt(draw.@entry.@past,Ana), obj(draw.@entry.@past,book.@indef), seq(go.@entry.@pred.@past,draw.@entry.@past)`

***soj*: Objeto estativo.** Um objeto de um estado. Exemplo:

Sentença original: *O mar está calmo*.

Representação UNL: `soj(calm,sea.@generic)`

***src*: Origem.** Local (físico ou lógico) ou estado de um agente/objeto relativo a um evento. Exemplo:

Sentença original: *As crianças vieram de Londres*.

Representação UNL: `agt(come.@entry.@pred.@past, child.@pl.@def), src(come.@entry.@pred.@past,Londres)`

***tim*: Tempo.** Horário de um evento. Exemplo:

Sentença original: *Comemos ao meio-dia*.

Representação UNL: `agt(eat.@entry.@pred, We), tim(eat.@entry.@pred,noon)`

***tmf*: Tempo inicial.** Horário inicial de um evento. Exemplo:

Sentença original: *Trabalhamos a partir do meio-dia*.

Representação UNL: `agt(work.@entry.@pred, We), tmf(work.@entry.@pred, noon)`

***tmt*: Tempo final.** Horário final de um evento. Exemplo:

Sentença original: *Trabalhamos até o meio-dia*.

Representação UNL: `agt(work.@entry.@pred, We), tmt(work.@entry.@pred, noon)`

RLs Entre Palavras Universais (UWs)

***icl*: Inclusão.** Representação de hiperonínima (super e subclasses) ou meronímia (relação parte-todo). Exemplo:

`icl(cachorro, animal)`

`icl(braço, corpo_humano).`

***equ*: Sinonímia.** representação de significados equivalentes entre UWs, como em `equ(book, accounts)`

Esses são os únicos rótulos de relação entre UWs definidos para estabelecer um paralelo entre UWs e são especificados diretamente no dicionário UWs-português. Na especificação de UNL anexada ao fim deste trabalho estão disponíveis a sintaxe, funcionamento e exemplos das RLs elencadas acima.

4.3.3. Os Rótulos de Atributos (ALs)

ALs servem para limitar o significado de uma UW genérica, i.e., para particularizar seu significado. Informações adicionais tais como tempo verbal, aspecto, intenção ou estrutura sentencial são exemplos de atributos específicos de uma UW. A representação genérica de um AL é dada pela UW, seguida por tantos atributos quantos forem necessários para restringir o significado do conceito genérico. Cada um dos atributos é identificado na sentença UNL pelo símbolo inicial "@". Por exemplo, a representação genérica de uma UW com *n* atributos tem a forma:

$$\mathbf{UW.@atrib_1.@atrib_2.....@atrib_n}$$

Se não houver ALs vinculados a uma UW, esta representa o significado mais genérico de sua classe. Da mesma forma que para RLs, pode haver diferentes classes de ALs. Apresentamos, a seguir, algumas delas.

a) ALs que limitam o poder de expressão de uma UW**,** tais como:

@*generic*: Representação genérica de uma UW. Exemplo:

Sentença original: *Pedro come maçãs*.

Representação UNL: `agt(eat.@present.@entry.@pred,Pedro),`
`obj(eat.@present.@entry.@pred,apple.@generic.@pl)`

@*def*: indica uma UW que já foi referenciada anteriormente. Exemplo:

Sentença original: *Pedro fechou a porta*.

Representação UNL:`agt(shut.@entry.@pred.@past,Pedro),`
`obj(shut.@entry.@pred.@past,door.@def)`

@*indef*. UW não especificada, mas particular ou não genérica. Exemplo:

Sentença original: *Um macaco caiu de um galho.*

Representação UNL: `obj(fall.@entry.@pred.@past,monkey.@indef),`
`src(fall.@entry.@pred.@past,treebranch.@indef).`

@*pl*. Pluralidade de UWs que não são genéricas. Exemplo:

Sentença original: *Comprei facas.*

Representação UNL: `agt(buy.@past.@entry.@pred,I),`
`obj(buy.@past.@entry.@pred,knife.@pl).`

@*not*. Negação. Exemplo:

Sentença original: *Eu não vou ao cinema.*

Representação UNL: `agt(go.@present.@not.@entry.@pred,I), gol(go.@present.@not.@entry.@pred,cinema.@def).`

b) ALs que expressam tempo verbal**.** Estes seguem a gramática da língua inglesa e incluem:

@*past*. Evento que aconteceu no passado. Exemplo:

Sentença original: *Eu não fui ao cinema.*

Representação UNL: `agt(go.@past.@not.@entry.@pred,I), gol(go.@past.@not.@entry.@pred,cinema.@def).`

@*present*. Evento que está acontecendo. Exemplo:

Sentença original: *Pedro fecha a porta*.

Representação UNL:`agt(shut.@present.@entry.@pred,Pedro), obj(shut.@present.@entry.@pred,door.@def)`

@*future*. Evento que irá acontecer. Exemplo:

Sentença original: *Ela virá.*

Representação UNL: `agt(come.@future.@entry.@pred,She)`

c) ALs que expressam aspecto**:** Esses ALs se referem à noção de aspecto da língua inglesa. Vale observar que a maioria deles se aplica também ao português, embora essa noção possa variar de uma língua para outra.

@*begin-soon*: "evento que vai começar". Exemplo:

Sentença original: *O avião está para aterrissar*.

Representação UNL:

`agt(land_in.@beginsoon.@entry.@pred.@present,plane.@def)`

@*begin-just*: "evento que recém-começou". Exemplo:

Sentença original: *O jogo acabou de começar*.

Representação UNL: `obj(start.@beginjust.@entry.@pred.@past,game.@def)`

@*end-soon*. Evento que está quase terminando. Exemplo:

Sentença original: *Estamos chegando ao final*.

Representação UNL:

```
agt(arrive.@end-soon.@entry.@pred.@present,We),gol(arrive.@end-
soon.@entry.@pred.@present,end.@def).
```

@*end-just*. Evento que recém-terminou. Exemplo:

Sentença original: *Acabamos de chegar em casa*.

Representação UNL:

```
agt(arrive.@end-just.@entry.@pred.@past,We),
gol(arrive.@end-just.@entry.@pred.@past,home.@def).
```

@*progress*. Evento em progressão. Exemplo:

Sentença original: *A menina está chorando*.

Representação UNL:

```
agt(cry.@progress.@entry.@pred.@present,girl.@def).
```

@*repeat*. Repetição de um mesmo evento, envolvendo o mesmo agente/objeto. Exemplo:

Sentença original: *A menina está pulando*.

Representação UNL:

```
agt(jump.@repeat.@entry.@pred.@present,girl.@def).
```

@*state*. Estado final de um objeto/evento, após a ocorrência de um outro evento. Exemplo:

Sentença original: *O vaso quebrou*.

Representação UNL:

```
obj(break.@state.@entry.@pred.@past,vase.@def).
```

@*complete*. Evento já concluído. Exemplo:

Sentença original: *Ela tocou piano*.

Representação UNL: `agt(play.@complete.@entry.@pred,She),`

```
obj(play.@complete.@entry.@pred.@past,piano.@generic).
```

d) ALs que expressam intenções

Intenções do locutor, tais como emoções, interpretações subjetivas e ênfase, são expressas por ALs desta classe. Como intenções podem se manifestar léxica ou estruturalmente, os ALs são divididos em duas subclasses: os associados a intenções

expressas por elementos intra-sentenciais e os associados a intenções expressos pela sentença inteira.

@*focus*: Informação sobre o foco de uma sentença. Exemplo:

Sentença original: *Foi você quem saiu?*

Representação UNL: `agt(leave.@entry.@pred.@interrogation.@past, you.@focus)`

@*emphasis* : Entidade sentencial enfatizada. Exemplo:

Sentença original: *Do caráter dele, eu desconfio*.

Representação UNL: `agt(distrust.@entry.@pred.@present, I), obj(distrust.@entry.@pred.@present,character.@emphasis), pos(He,character.@emphasis)`

@*theme*. Informação temática.

Há outros rótulos de atributos de UWs que se aplicam a uma sentença como um todo, em vez de serem aplicados a elementos intra-sentenciais. Neste caso, eles representam intenções no contexto geral da sentença e, em geral, são expressos pelo modo (interrogativo, imperativo, subjuntivo, etc.) Os ALs são associados ao núcleo da predicação central da sentença. Como a UNL se baseia na língua inglesa, os verbos auxiliares desta são usados como ALs, para representar intenções correspondentes. Combinações de ALs e UWs são também usadas para complementar as informações necessárias de uma UW particular. A forma geral de ALs desse tipo é dada por

`<verbo-auxiliar-no-inglês>@AL`

Veremos, a seguir, alguns ALs desse tipo, juntamente com sua função e sua correspondência com o inglês.

e) ALs que expressam intenções em uma sentença inteira

e.1) *ALs aplicadas a uma UW central ou combinadas com UWs que expressam verbos auxiliares*

@*ability*. Habilidade, competência para executar algo (*can, be able to, be capable of*). Exemplo:

Sentença original: *Ele pode falar inglês*.

Representação UNL:

```
agt(speak.@entry.@pred.can@ability.@present,He),
obj(speak.@entry.@pred.can@ability.@present,English)
```

@*apodosis*. Apódose (*could, should, would).* Exemplo:

Sentença original: *Se tivéssemos mais dinheiro, poderíamos comprar um carro.*

Representação UNL:

```
agt(buy.@entry.@pred.could@apodosis.@past,We),
obj(buy.@entry.@pred.could@apodosis.@past,car.@indef),
con(buy.@entry.@pred.could@apodosis.@past,money),
mod(money,more).
```

@*grant.* Consentir (*can, could, might, may*). Exemplo:

Sentença original: *Posso fumar aqui?*

Representação UNL:

```
agt(smoke.@entry.@pred.@interrogative.@present.can@grant,I),
ppl(smoke.@entry.@pred.@interrogative.@present.can@grant,here)
```

@*grant-not.* Não consentir (*must not, be not allowed to, may not*). Exemplo:

Sentença original: *Você não pode emprestar meu carro.*

Representação UNL:

```
agt(borrow.@entry.@pred.may@grant-not.@present,You),
obj(borrow.@entry.@pred.may@grant-not.@present,car),
mod(car,mine)
```

@*intention.* Desejo ou intenção (*shall, will*). Exemplo:

Sentença original: *Pretendo pintar um retrato.*

Representação UNL:

```
agt(paint.@entry.@pred.may@intention.@present,I),
obj(paint.@entry.@pred.may@intention.@present,portrait.@indef)
```

Outras formas mais complexas de combinações de UWs podem ainda ser expressas na UNL. Por exemplo, para a sentença

Ele pode falar inglês mas não pode escrever muito bem.

a combinação de AL com o verbo auxiliar *can* resulta na forma `can@ability`, que será o atributo complexo da UW `speak`. A sentença UNL completa para tal sentença é expressa abaixo:

```
agt(speak.@entry.@pred.@present.can@ability,He),
obj(speak.@entry.@pred.@present.can@ability,English),
agt(write.@entry.@present.@not.can@ability,He),
mod(write.@entry.@present.@not.can@ability,very_well)
```

Tabela 13 – Exemplo de sentença UNL

Línguas naturais distintas podem, ainda, ter representações mais detalhadas para expressar informações restritivas sobre conceitos semânticos. Para ambas as situações, a UNL permite a definição de subcategorizações pela inclusão de novos atributos, tornando possível tratar as particularidades de cada língua. Outros ALs que se aplicam a uma UW ou combinação de UWs incluem, p.ex., aqueles cuja expressão natural em inglês se dá pelos verbos modais *could, may, might, ought to, shall, should, will, would*, tais como:

@*custom*. Ação habitual (*would*).
@*insistence*. Desejo contundente para fazer algo (*shall, will, would*).
@*inevitability*. Suposição sobre o que é inevitável (*must*).
@*may*. Suposição sobre o que é possível (*may, might*).
@*obligation*. Atribuir obrigações a alguém (*shall, must, have to*).
@*obligation-not*. Proibir (*must not, need not, don't have to*).
@*possibility*. Suposição sobre o que é possível (*can, could*).
@*probability*. Suposição sobre o que é provável (*would*).
@*should*. Sentir-se obrigado (*should, ought to*).
@*will*. Desejar (*shall, will*).

e.2) ALs que podem ser aplicados somente a uma UW central

@*confirmation*. Solicitação de concordância, p.ex., com ponto de vista (*tag questions*).

@*exclamation*. Sentença exclamatória.

@*if*. Suposição sobre um evento incerto (*if* + *modo subjuntivo*).

@*imperative*. Sentença imperativa.

@*interrogation*. Sentença interrogativa.

@*invitation*. Indução para fazer algo (*let us* + *verb*).

@*politeness*. Demonstração de educação (*would you ...*).

@*recommendation*. Recomendação para fazer algo (*had better, rather, would rather ...*).

@*respect*. Demonstração de respeito (*...sir*).

@*thought*. Demonstração de dúvida, contradições ou suposições (*modo subjuntivo*).

@*underestimate*. Demonstração de subestima (*nothing but*).

f) ALs que expressam estruturas sentenciais**:** uma das formas de se preservar o significado original de sentenças é procurar refletir sua estrutura sentencial por meio da ênfase dada a informações dominantes na mesma. Na UNL, os ALs têm também essa função, quando associados às UWs relevantes. Caso não haja uma UW dominante, várias formas de conversão são possíveis. Os ALs que expressam estruturas sentenciais na UNL são os seguintes:

@*pred*. Predica uma UW. Por exemplo,

Sentença original: *Ele falou que vai sair.*

Representação UNL: `agt(speak.@past.@pred.@entry,He),`
`obj(speak.@past.@pred.@entry,leave.@future.@pred),`
`agt(leave.@future.@pred,He)`

@*entry*. Aponta a UW dominante em uma sentença UNL. Por exemplo,

Sentença original: *Ele falou que vai sair.*

Representação UNL: `agt(speak.@past.@pred.@entry,He),`
`obj(speak.@past.@pred.@entry,leave.@future.@pred),`
`agt(leave.@future.@pred,He)`

@*sub*. Aponta a UW dominante em um nó do hipergrafo que representa a sentença UNL. Uma UW marcada por @sub é essencial em um hipergrafo. Por exemplo,

Frase original: *Uma passagem de Londres a Oxford.*

Representação UNL: `mod(ticket.@indef,fmt(Oxford.@sub,Londres).`

@title. Aponta uma UW correspondente ao título, como forma sentencial. Por exemplo,

Sentença original: *Manifestação sintática do português*

Representação UNL: *`mod(expression.@title,syntax),`*
`mod(syntax,Portuguese)`

Na Especificação da Linguagem UNL anexada ao fim desse trabalho pode-se ter acesso à lista completa dos ALs, com exemplos de usos e leituras e a sintaxe correta dos mesmos.

4.4. Ilustração do Processo Completo de Conversão e Deconversão

A Figura 3 mostra a representação em UNL da seguinte sentença, enquanto a Figura 4 mostra o grafo resultante desta sentença:

Long ago, in the city of Babylon, people begun to build a huge tower, which seemed to reach the heaven.

```
      [S]
       tim(begin(icl>event).@entry.@pred.@past, long_ago)
       nam(city(icl>place).@def, Babylon(icl>city))
       ppl(begin(icl>event).@entry.@pred.@past,
city(icl>place).@def)
       agt(begin(icl>event).@entry.@pred.@past,
people(icl>human).@def)
       obj(begin(icl>event).@entry.@pred.@past,
       build(equ>construct,agt>human,obj>structure).@pred)
       agt(build(equ>construct,agt>human,obj>structure).@pred,
       people(icl>human).@def)
       obj(build(equ>construct,agt>human,obj>structure).@pred,
       tower(icl>building).@indef)
       aoj(huge(aoj>entity), tower(icl>building).@indef)
```

```
      aoj(seem(icl>event).@pred.@past,
tower(icl>building).@indef)
      obj(seem(icl>event).@pred.@past,
      reach(gol>entity,obj>entity).@pred.@begin-soon)
      obj(reach(gol>entity,obj>entity).@pred.@begin-soon,
tower(icl>building).@indef)
      gol(reach(gol>entity,obj>entity).@pred.@begin-soon,
heaven(ant>hell).@def.@pl)
      [/S]
```

Figura 3 – Exemplo de sentença UNL

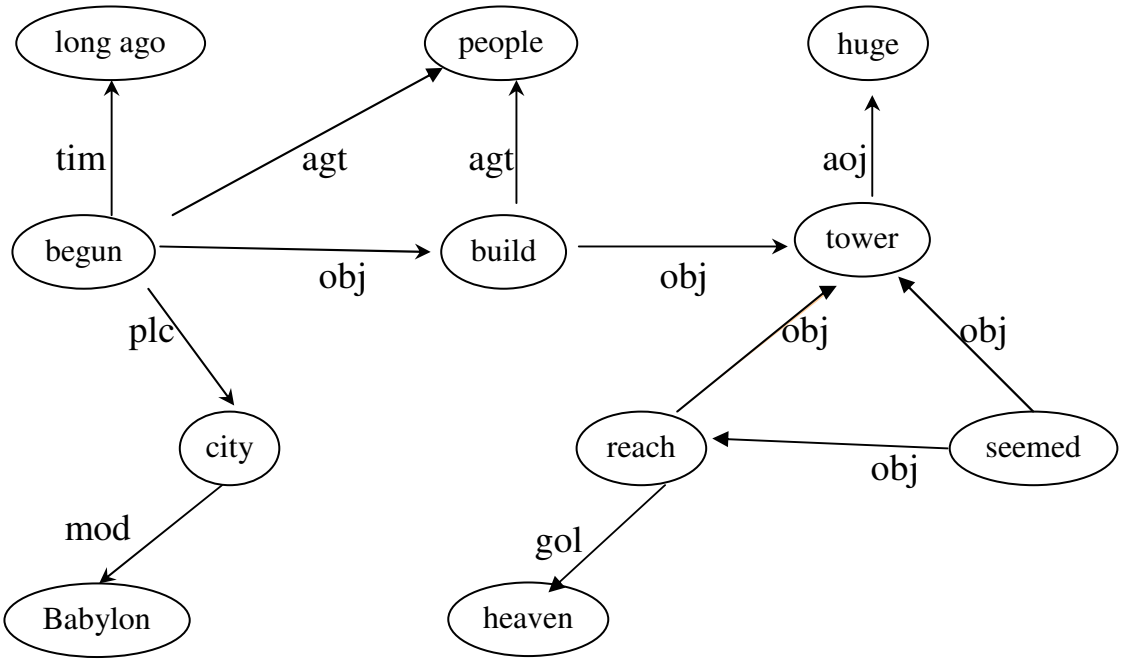

Figura 4 – Grafo da sentença UNL

O próximo exemplo mostra ambas as representações: a resultante do processo de conversão, i.e., a sentença UNL, assim como uma possível derivação de uma sentença UNL em português. Seja a seguinte sentença:

As meninas não mataram o menino.

com correspondente sentença UNL:

```
[S]
```

```
 agt(kill.@past.@not, girl.@def.@pl)
 obj(kill.@past.@not, boy.@def)
[/S]
```

Seja, agora, a seguinte sentença UNL, estendida da anterior:

```
[S]
 agt(kill.@past.@not, girl.@def.@pl)
 obj(kill.@past.@not, boy.@def)
 agt(buy.@past, boy.@def)
 obj(buy.@past, book.@pl)
[/S]
```

Em um processo de deconversão, esta sentença pode ser expressa em português, p.ex, de duas formas:

As meninas não mataram o menino. O menino comprou livros. (1)

As meninas não mataram o menino que comprou livros. (2)

Ambas as formas correspondem à mesma sentença UNL. No entanto, (2) apresenta uma resolução do elo referencial entre "o menino que comprou livros" e "o menino que não foi morto" estilisticamente mais elegante que (1). Este tipo de resolução é possível porque a UNL permite delimitar os constituintes de uma sentença pelos símbolos especiais [S] e [/S] (respectivamente, começo e fim da sentença). Com essa delimitação, consideramos que qualquer repetição de componentes de uma UW em uma sentença UNL corresponde à mesma entidade conceitual. Este é o caso da UW "menino.@def" no exemplo acima.

Vemos, nessas ilustrações, que tanto o processo de conversão quanto o processo de deconversão fazem uso da associação da UNL com o português: na conversão, para extrair a sentença UNL de uma sentença em português; na deconversão, para gerar a sentença em português correspondente a uma representação UNL. Para ilustrar esses processos, é necessária a análise do funcionamento do EnCo e do DeCo UNL-português, apresentada a seguir.

5. O FUNCIONAMENTO DO ENCO E DO DECO

Esta parte do trabalho contém uma descrição breve do funcionamento do EnCo e do DeCo, aplicações com a finalidade de converter o português do Brasil para a Universal Networking Language (UNL) e deconverter as representações de UNL para a nossa língua. Esta descrição será seguida por uma análise detalhada de suas limitações e benefícios, bem como a sua aplicação ao idioma português [16].

Experiências com a Tradução por Máquinas (MT) mostram que pelo menos três módulos independentes são necessários para o processamento de linguagens: os módulos léxicos, sintáticos e semânticos. Nas aplicações avaliadas, estes três módulos são usados por ambas as ferramentas, o "enconverter", também conhecido como EnCo (português-UNL) e o "deconverter", também chamado de DeCo (UNL-português), sem uma especificação à priori para a arquitetura interna dessas ferramentas. A arquitetura geral para o sistema é descrita na Figura 5, e cada um de seus módulos é apresentado nas seções seguintes.

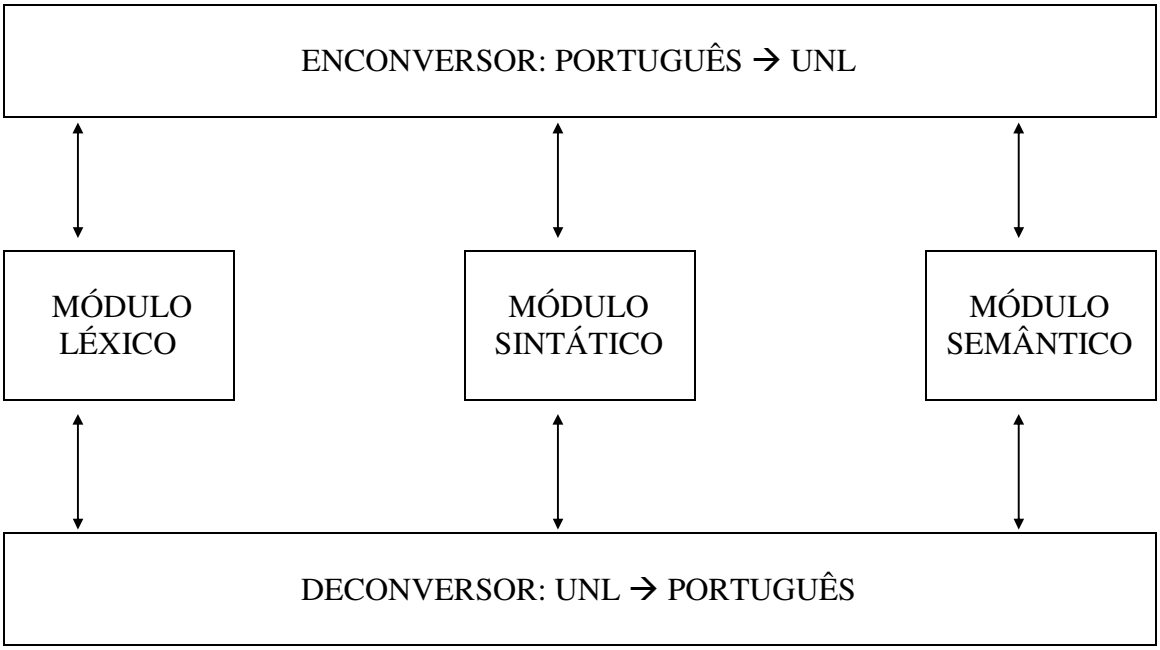

Figura 5 – Arquitetura geral do sistema de conversão/deconversão [16]

5.1. O Módulo Léxico

Este módulo engloba cinco submódulos, como ilustrados na Figura 6: i) o léxico, ou dicionário, onde ficam armazenadas as terminologias em português ii) uma tabela para converter o português em *attribute labels* (tabela de ALs em português) , iii) um analisador léxico, iv) um gerador léxico e v) um sistema de desambiguamento léxico. Os dois primeiro submódulos representam o banco de dados no qual o EnCo e o DeCo irão operar. O analisador léxico executa a análise morfológica e a conversão das palavras em português para palavras universais (UWs) e *attribute labels* (ALs), conforme a necessidade. O gerador léxico, por sua vez, leva a cabo a tarefa de conversão correspondente, convertendo UWs e ALs em palavras em português. Ambos, o analisador léxico e o gerador dependem do sistema de desambiguamento para escolher as classes gramaticais apropriadas, bem como os atributos semânticos e gramáticos das palavras em português. Eles também mapeiam UWs e ALs para as expressões portuguesas, conforme a necessidade.

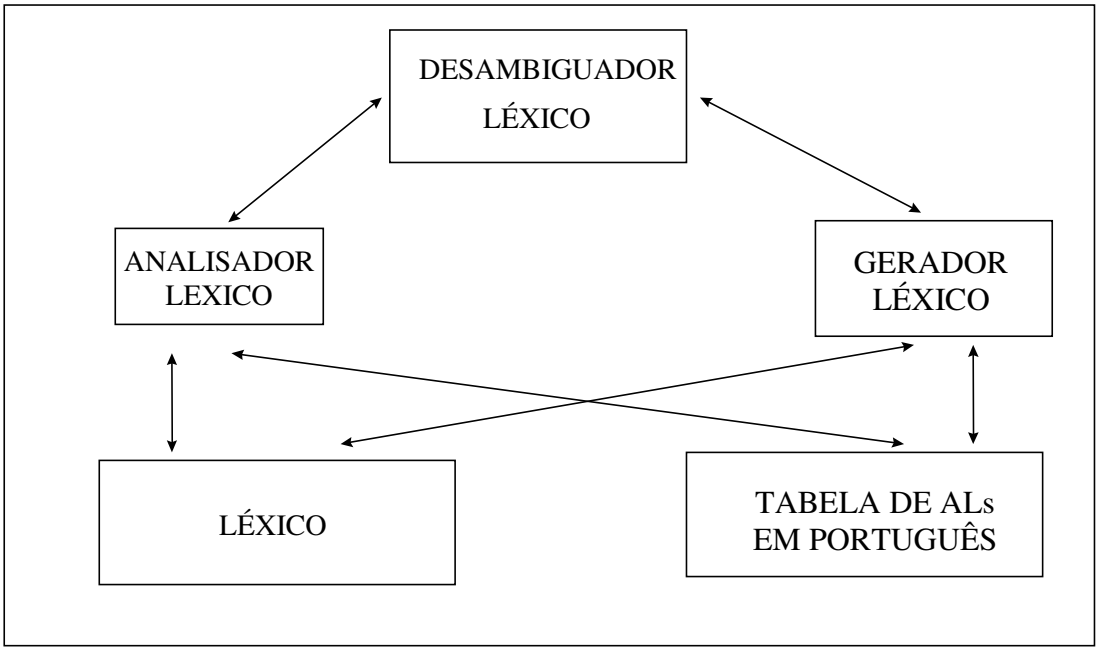

Figura 6 – Os cinco submódulos do analisador léxico [16]

A representação interna do léxico é praticamente igual a do dicionário de UWs. O acesso ao léxico é feito através de palavras padronizadas escolhidas, que são palavras em português não-flexionadas. Por exemplo, em sua forma padronizada, um substantivo, ou

um adjetivo, é representado de acordo com seu número e gênero, p.e., singular, masculino. Para um verbo, é adotada a representação de infinitivo. O léxico é indexado para permitir acesso de qualquer UW que corresponda a uma palavra em português (que não é flexionada neste caso) ou vice-versa. Além disso, colocações que são muito freqüentes em textos em português também são representadas como unidades únicas no léxico (i.e., por exemplo, a colocação 'por outro lado' é representada pela UW '`on_the_other_hand`', tradução para o inglês desta colocação). Os pedaços de informação relacionados a itens léxicos podem ser: as próprias palavras em português, suas UWs correspondentes, papéis gramaticais (substantivo, adjetivo, verbo, etc.), categorias morfológicas (número, gênero, etc), papéis sintáticos (sujeito, predicado, etc.) e atributos semânticos.

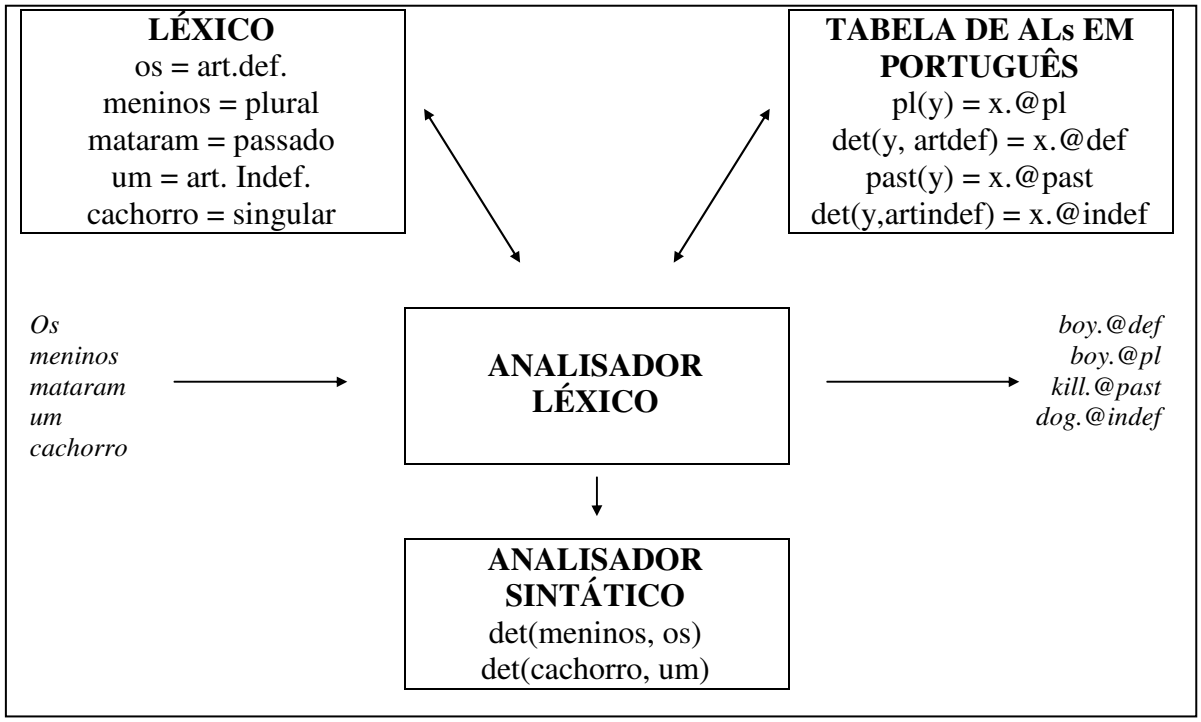

Figura 7 – Exemplo da operação do analisador léxico [16]

A tabela de ALs em português faz a correspondência entre as ALs e as representações em português (papéis sintáticos e gramaticais) que é projetada para permitir o processamento automático das categorias morfológicas tanto para o EnCo (por meio do analisador léxico) quanto para o DeCo (por meio do gerador léxico). O analisador léxico é necessário para codificar substantivos e verbos do português para UNL. Dois

procedimentos são necessários para isto: i) uma análise morfológica que opera diretamente no léxico e na tabela de ALs em português, por obter o *attribute label* (ALs) correspondente à palavra em português (como o número para um substantivo ou a flexão para um verbo); ii) uma análise sintática, por obter ALs que não são aparentes na representação morfológica das palavras, como o sujeito de uma frase. Esta tarefa é executada obtendo informação tanto do analisador sintático como da tabela de ALs em português. A Figura 7 exemplifica o comportamento do analisador léxico ao receber uma oração em português.

O gerador léxico opera precisamente na direção oposta ao analisador léxico, convertendo atributos léxicos representados em UNL em palavras em português. Também são seguidos dois passos: i) uma síntese morfológica pela qual palavras em português flexionadas são obtidas a partir das UWs; ii) o cumprimento de papéis sintáticos por meio de ALs que tragam informação sobre categorias adicionais representadas pelas UWs. São ilustrados os processos subjacentes do gerador léxico na Figura 8.

O sistema de desambiguamento léxico é necessário pelo fato das palavras em português, ou mesmo as UWs, poderem ter mais de uma classificação, como ilustrado abaixo:

a) Categoria gramatical:

Português: *canto* (substantivo), *canto* (verbo)

UNL: *cause* (substantivo), *cause* (verbo)

b) Morfologia:

Português: *lápis* (singular), *lápis* (plural)

Português: *dentista* (masc), *dentista* (fem)

UNL: presumivelmente nenhum.

c) Sintaxe:

Português: *ser* (auxiliar), *ser* (intransitivo)

UNL: presumivelmente nenhum.

d) Semântica:

Português: *manga* (fruta), *manga* (da camiseta)

UNL: *book* (publicação), *book* (de contabilidade)

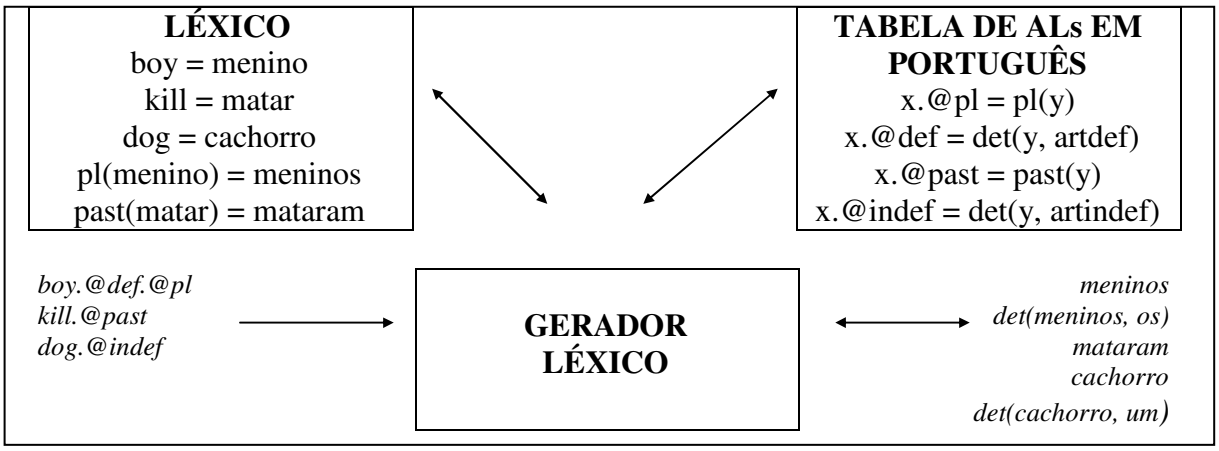

Figura 8 – Exemplo da operação do gerador léxico

O desambiguador será utilizado quando o analisador léxico, o gerador léxico ou o analisador sintático acessarem o léxico para procurar uma classificação de uma palavra e acharem mais de uma opção.

5.2. O Módulo Sintático

O módulo sintático, descrito na Figura 9, também inclui cinco submódulos: uma tabela de regras de produção sintática, um *parser* (para análise sintática automática), um gerador de estruturas sintáticas, um desambiguador de papéis sintáticos e um regularizador sintático. A tabela de regras de produção contém, em princípio, todas as regras sintáticas para gerar construções corretas em português. O ponto de partida aqui é uma tabela baseada em uma gramática livre de contexto. Atualmente, a tabela está sendo ampliada e modificada introduzindo-se tais características desse tipo de gramática.

.

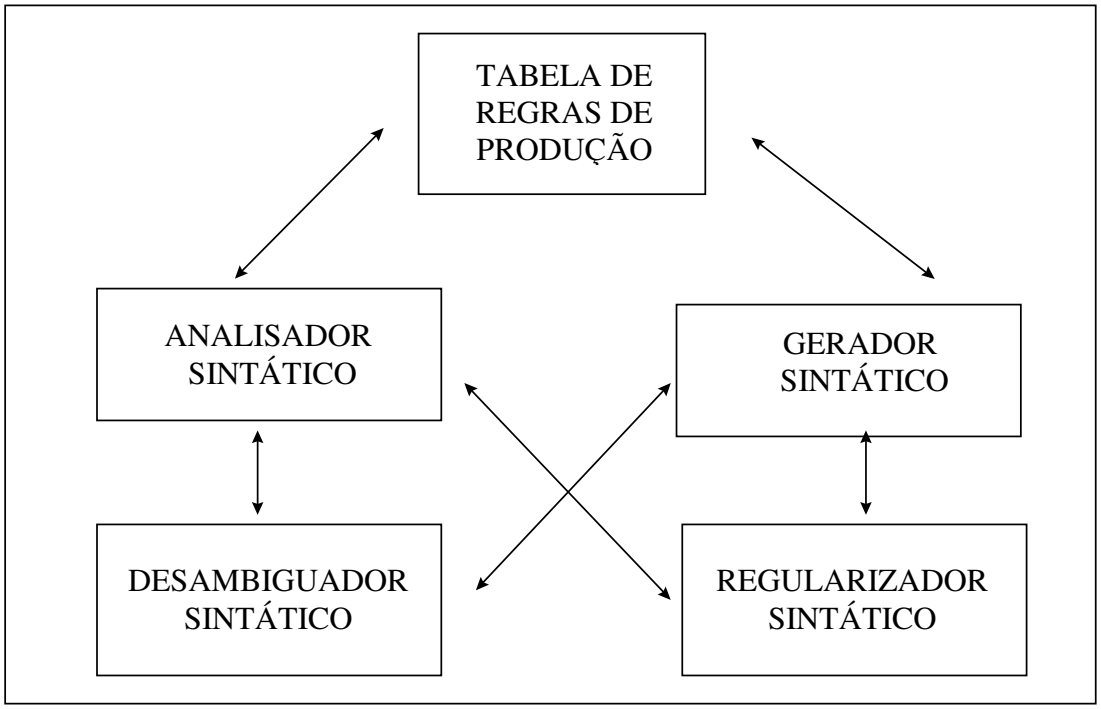

Figura 9 – O módulo sintático

O analisador sintático, por sua vez, trata-se de um *parser* de esquerda para a direita e de cima para baixo que inclui algumas estratégias de *lookahead* e *backtracking* até o último ponto de tomada de decisão. Estas estratégias são usadas quando um casamento para uma construção sintática não obtiver êxito. O analisador identifica o papel sintático de cada item léxico de uma oração de entrada, incluindo todas suas relações de dependência que são essenciais para o analisador semântico. A Figura 10 ilustra o funcionamento do analisador sintático. Pode ser interessante, em fases posteriores do projeto, introduzir estratégias de *parsing* do tipo *bottom-up* para casos de *mismatch* e ambigüidade sintática. Além disso, poderia ser feito o uso de um modelo de *parsing* que analise os núcleos das orações individualmente, o que é particularmente útil para geração de orações.

Obviamente, ambigüidades aparecem quando se estabelecem os papéis sintáticos para algumas palavras. Logo, um quarto submódulo, o desambiguador sintático, é necessário. As duas funções básicas deste módulo são: a) selecionar um entre os vários possíveis casamentos de uma determinada oração em português; b) selecionar o papel sintático mais adequado de cada atributo semântico na representação em UNL, quando

correspondências entre sintaxe e semântica ocorrem. O desambiguador é ativado através dos analisadores sintáticos e semânticos.

O gerador sintático foi projetado para converter relações semânticas UNL em papéis sintáticos do português. A correção das relações é conferida então pelo regularizador sintático, a fim de uma produção gramaticalmente correta. Este sistema também é responsável por preencher as faltas de informações quando a associação entre os termos é feita. Em outras palavras, o gerador sintático é o responsável por reconstruir a profunda estrutura da linguagem UNL a partir dos dados de entrada inseridos pelo usuário.

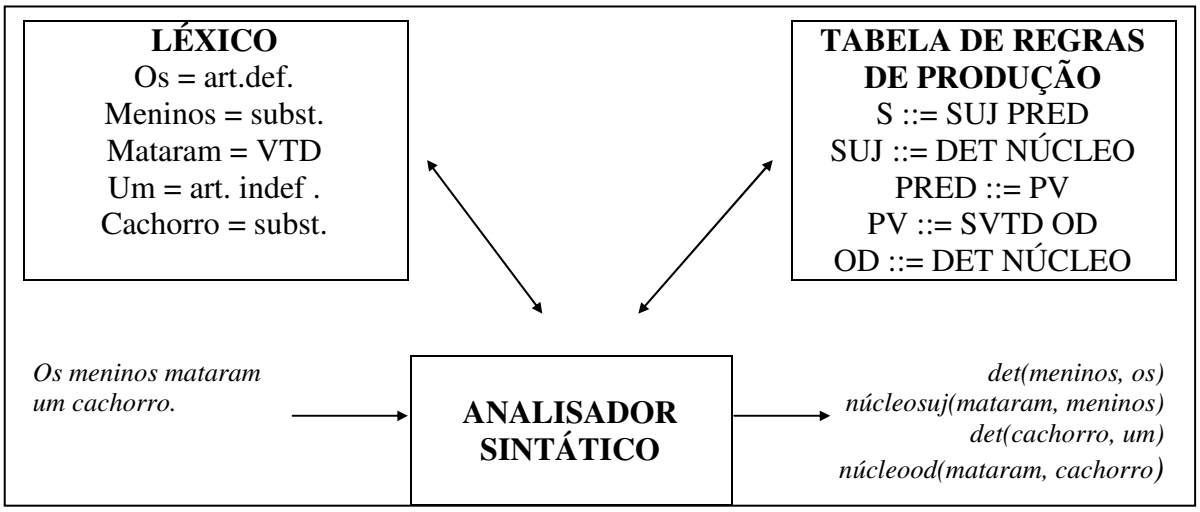

Figura 10 – O analisador sintático

5.3. O Módulo Semântico

O módulo semântico inclui três submódulos (Figura 11), como segue: uma tabela para a conversão do português em *relation labels* (RLs), chamada de tabela de RLs em português, um analisador semântico, e um desambiguador semântico. A tabela de RLs em português armazena as relações semânticas entre variações sintáticas correspondentes às entidades conceituais contidas em uma oração escrita em português. A mesma transforma a sintaxe em semântica, i.e., são recuperados os atributos semânticos baseados na descrição sintática de artigos léxicos do português. Além disso, tal mapeamento permite que atributos sintáticos sejam obtidos através de relações semânticas indicadas pela linguagem UNL. O

analisador semântico é responsável por transformação da informação sintática em atributos semânticos, operando no output do analisador sintático para prover as relações semânticas necessárias. Por exemplo, o analisador pode reconstruir relações a partir de relações de ligação, como agente (agt) e objeto (obj).

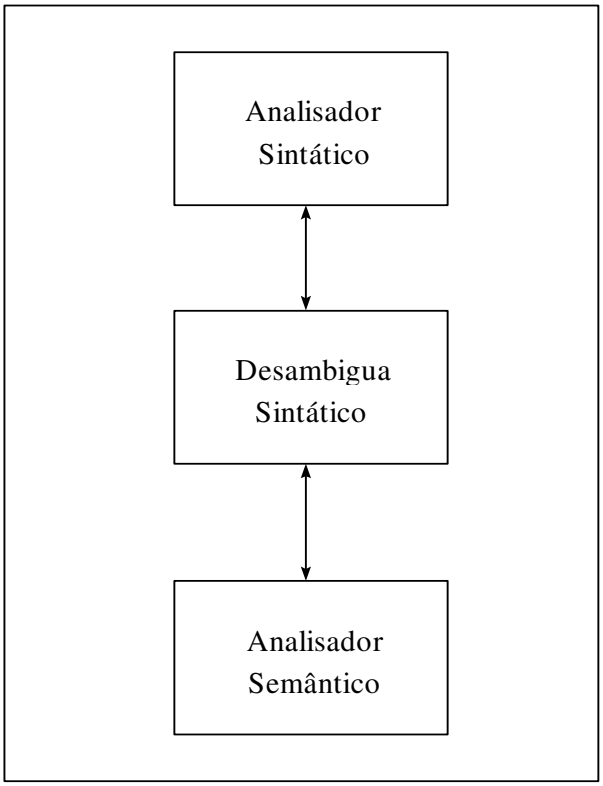

Figura 11 – O módulo semântico

O desambiguador semântico é responsável para solucionar ambigüidades estruturais que não tenham sido resolvidas pelos desambiguadores sintáticos e léxicos.

5.4. Visão Geral do Funcionamento do EnCo e do DeCo

Em prol da simplicidade, são mostrados apenas os passos principais do processos de conversão e deconversão.

5.4.1. Funcionamento do EnCo

O processo de conversão é constituído dos seguintes passos:

1. Leitura da oração em português. As sentenças aqui são um pedaço de texto entre dois marcadores de limite. Como tal, uma unidade a ser analisada pode ser uma cláusula, em vez de uma oração inteira. Marcadores de limite podem ser conjunções ou, mais usualmente, a pontuação da oração (ponto, vírgula, etc).

2. Ativação do analisador sintático para identificar o papel sintático de cada artigo léxico do texto de entrada. Uma mensagem de falha é devolvida se não houver nenhum casamento com uma regra de produção.

3. Ativação do analisador léxico para identificar os atributos das palavras.

4. Ativação do analisador semântico para identificar o papel semântico de cada artigo léxico. Os artigos léxicos que não são reconhecidos (atualmente, a única possibilidade é que os mesmos não estejam definidos no léxico) são devolvidos sem atributos semânticos.

5. Formatação da informação gerada pelos processos anteriores de acordo com a especificação de UNL.

Exemplo:

1. A menina comprou um livro.
2. `suj(comprou, menina)`
   `obj(comprou,livro)`
   `det(menina,a)`
   `det(livro,um)`
3. `menina.@def`
   `comprou.@past`
   `livro.@indef`
4. `agt(comprou, menina)`
   `obj(comprou, livro)`
5. **`agt(buy.@past,girl.@def)`**
   `obj(buy.@past,book.@indef)`

5.4.2. Funcionamento do DeCo

O processo de deconversão consiste nos passos seguintes:

1. Ativação do gerador léxico por identificação dos atributos léxicos. Os papéis sintáticos que correspondem às ALs são preenchidos. As palavras em português correspondentes às UWs da entrada são flexionadas de acordo com a especificação do AL.

2. Ativação do gerador semântico por identificar as relações entre as sentenças.

3. Ativação do gerador sintático por construção da sentença. As relações semânticas são recuperadas a partir da representação em UNL e os papéis sintáticos de cada item em UNL são preenchidos. A oração assim representada é traduzida para o português.

Exemplo:

1. `comprar(passado)`
2. `suj(comprou, menina)`
   `obj(comprou,livro)`
   `det(menina,a)`
   `det(livro,um)`
3. **A menina comprou um livro.**

6. A INTERNACIONALIZAÇÃO E A UNL NO CICLO DE VIDA DE DESENVOLVIMENTO DE SOFTWARE

Baseado nas informações coletadas durante a revisão bibliográfica sobre a internacionalização e localização de software, foi detectada uma falta de documentação visando integrar a internacionalização aos procedimentos padrão de desenvolvimento de software. Esta lacuna motivou o desenvolvimento de um *framework*, aonde são enumerados os pontos nos quais a internacionalização aliada ao uso do sistema UNL impactam no processo de desenvolvimento significativamente. A meta deste *framework* é fazer com que os aspectos inerentes à internacionalização de software integrem-se ao processo de desenvolvimento normal de um produto de software, utilizando as funcionalidades providas pela UNL para esse propósito. Esse *framework* é baseado no ciclo de desenvolvimento de software em cascata [17], pela grande disseminação e eficiência do mesmo.

Em cada fase do ciclo de desenvolvimento foram feitas notas sobre os problemas detectados, não só com o *framework*, mas também com a internacionalização do software e com o sistema UNL em geral. Uma vez que o desenvolvimento foi completado, todas essas notas foram agrupadas a fim de viabilizar uma análise do *framework* como um todo.

6.1. O *Framework*

O objetivo é que o *framework* de internacionalização aja como um guia pelo qual uma equipe de desenvolvimento de software possa começar a projetar software de modo completamente internacionalizado. O *framework* foi desenvolvido visando uma audiência sem uma experiência prévia do funcionamento e das técnicas de internacionalização.

O *framework* foi construído sobre o modelo em cascata do ciclo de vida de desenvolvimento de software. Há duas razões por trás desta escolha. Primeiro, este modelo é um dos mais difundidos atualmente, sendo um dos padrões da indústria de desenvolvimento de software nacional. Dessa maneira, fica mais fácil a integração da internacionalização com este ciclo de vida, bem como o foco nos aspectos fundamentais da internacionalização de software e do sistema UNL é mantido. A segunda razão para a escolha do modelo em cascata é que este modelo cobre sistematicamente todas as fases do processo de desenvolvimento de software [18], de tal sorte que desenvolvedores que

queiram utilizar outros modelos para o ciclo de vida de desenvolvimento ainda poderiam analisar do princípio ao fim este *framework* e adaptar o mesmo às necessidades de cada modelo em específico.

O Sistema UNL aparece no *framework* na medida em que os *resource files* e os documentos contendo os textos e as especificações do software mantêm-se desacoplados do mesmo, como ilustra a Figura 12. Utilizando ferramentas como o EnCo e o DeCo em fases específicas do processo de desenvolvimento, o desenvolvimento do software fica facilitado e desvincula-se o produto de qualquer linguagem específica, ficando o mesmo facilmente localizável.

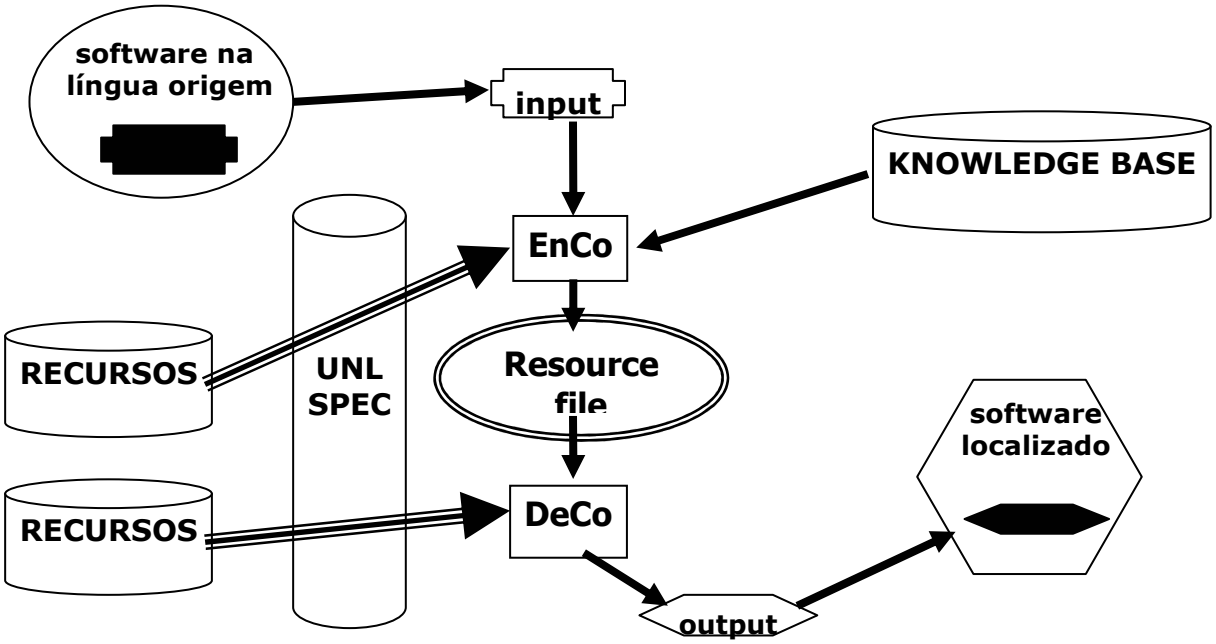

Figura 12 – Ilustração do Sistema UNL nos processos de internacionalização e localização de software

6.2. O Ciclo de Vida de Desenvolvimento de Software – O Modelo em Cascata

6.2.1. Fase de Análise de Requisitos

Durante a Fase de Análise de Requisitos, a equipe de desenvolvimento tem que determinar o que o novo software deve fazer. Os desenvolvedores baseiam-se em softwares

atuais ou em sistemas que executam as mesmas tarefas e analisam seus pontos fortes e fracos. Métodos que são freqüentemente usados para determinar os requisitos do novo produto incluem a entrevista com o cliente, a distribuição de questionários para os usuários potenciais e análise das ferramentas e métodos utilizados no sistema atual para a realização das tarefas almejadas. [19]

É importante saber diferenciar o cliente – pessoa ou companhia que encomenda o produto – do usuário – a pessoa que na verdade usará o produto. O desenvolvedor tem que receber a contribuição adequada e balanceada de ambos para projetar o sistema corretamente. É importante durante esta fase do ciclo de vida de desenvolvimento assegurar que todas as partes pertinentes tenham a sua margem de contribuição sobre os requisitos do novo sistema.

O resultado da Fase de Análise de Requisitos é um documento que detalha as necessidades do cliente para este pedaço em particular de software [20]. A viabilidade destas necessidades como também as considerações sobre a implementação não são analisadas neste momento. Na realidade, nesta fase nenhuma decisão ainda foi tomada sobre o software, nem mesmo se o mesmo será realmente desenvolvido.

6.2.1.1. Impacto da Internacionalização e do Uso da UNL

A internacionalização não muda o modo o qual os requisitos de um sistema são determinados, mas amplia as áreas nas quais tais requisitos devem ser buscados. A primeira tarefa extra que a internacionalização traz ao modelo de desenvolvimento é a determinação dos *locales* iniciais e futuros. Em outras palavras, em quais mercados o produto será utilizado ou vendido inicialmente e quais os possíveis mercados adicionais para quais o produto possa ser ampliado no futuro.

A segunda tarefa extra que a internacionalização traz ao processo de desenvolvimento é a determinação dos requisitos para cada um dos *locales* iniciais e futuros. Tal possibilidade deve ser contemplada pelas prováveis diferenças nos requisitos entre os países envolvidos. Por exemplo, um software bancário deve calcular taxa de juros em países ocidentais de maneira diferente dos países islâmicos, onde a cobrança de juros é ilegal.

Isto conduz ao segundo passo crucial que o processo de internacionalização traz à Fase de Análise de Requisitos: a determinação de se ou não é necessária a internacionalização do produto neste momento. Primeiro, o sistema precisa ser internacionalizado? Há qualquer tipo de interface onde os *inputs* e/ou *outputs* do sistema precisam diferir de local para local? Se o sistema vai ser embutido – i.e não é necessário nenhum *input* do usuário nem é gerado nenhum *output* para o mesmo, então o sistema não precisa ser internacionalizado. Um exemplo disto poderia ser um software embutido em um automóvel. O programa é projetado para um carro específico e monitora os vários sistemas para assegurar que eles estão trabalhando corretamente. Este programa funcionará da mesma maneira independentemente do país para o qual o carro é exportado, de modo que não requer internacionalização e os desenvolvedores podem proceder como de praxe.

A segunda pergunta que a equipe de desenvolvimento deve fazer é se realmente é lógico internacionalizar este produto? Durante esta fase eles determinaram os requisitos do software para os vários locais envolvidos. Se os requisitos para o sistema diferirem largamente entre os locais designados, então é mais viável desenvolver aplicações separadas para cada país e verifica-se que este projeto não é um bom candidato para a internacionalização. Um exemplo disto pode ser um software para preencher declarações de renda. A estrutura tributária, leis, formas e procedimentos variam tanto de um país para o outro de modo que os requisitos para um sistema seriam muito distintos entre os diversos *locales*, tornando a criação do software internacionalizado inviável.

Além disso, nessa fase são gerados documentos importantes para a correta localização do software a ser desenvolvido: as diretrizes de estilo, glossários e listas de terminologia, para garantir que a tradução dos termos empregados atinja um patamar elevado de correção. É importante que o levantamento dessa terminologia seja feita em conjunto com os usuários, pois os mesmo é que tem maior familiaridade com os termos do domínio específico em questão.

Nessa fase, o uso da UNL não é necessário. O software ainda limita-se apenas às especificações e requisitos, de modo que o foco deve ficar nas funcionalidades do software a ser desenvolvido.

6.2.2. Fase de Especificação

Durante a Fase de Especificação, o desenvolvedor deverá preparar um documento de especificação que represente o escopo do produto ser desenvolvido [21]. O documento deve ser técnico o suficiente para ser conclusivo e não-técnico o bastante para ser entendido pelo cliente. Este documento enumera todo o conhecimento que a equipe de especificação utilizará como parâmetros para o desenvolvimento do novo produto. Este documento não só faz o esboço do que o produto tem que fazer, mas também gera um cronograma para conclusão, bem como métricas confiáveis e exigências de tempo exeqüíveis para a conclusão do projeto, além de especificar qualquer outra condição que deva ser conhecida para que o produto seja aceitável ao cliente.

Também é nesta fase que as estratégias de solução são desenvolvidas para os problemas levantados através dos requisitos. O projeto é dividido em módulos e a estrutura inteira do programa é decomposta. Um método para documentar esta estrutura que de certo modo é compreensível tanto ao cliente como para a equipe de desenvolvimento é o uso de diagramas de fluxo de dados – uma série de caixas e linhas que representam o fluxo de dados através do sistema.

Finalmente, os desenvolvedores e o cliente passam por um processo de decisão em dois passos. No primeiro passo, o cliente e/ou equipe de desenvolvimento desenvolvem uma análise dos custos para verificar a viabilidade financeira do desenvolvimento do software. Se eles chegam à conclusão que é, um das estratégias de solução é o acordo em que o documento de especificação torna-se um contrato para a conclusão do produto de software [22].

6.2.2.1. Impacto da Internacionalização e do Uso da UNL

Muitas das decisões cruciais à internacionalização do software são feitas durante a Fase de Especificação. Neste momento a equipe envolvida na especificação tem que determinar os *locales do* produto que serão habilitados para a localização, como também quais *locales* serão desenvolvidos neste momento. Por exemplo, um cliente poderia escolher habilitar o software para vários idiomas bidirecionais, mas só desenvolver os

locales para os Estados Unidos e para o Brasil. A idéia seria então desenvolver outras versões como o chinês e japonês como parte de alguma versão futura do software.

A equipe de desenvolvimento também tem que decidir nesse ponto a respeito do *character set*. A decisão mais crítica será sobre a utilização de caracteres de 2 bytes (Unicode), haja vista que influencia em todo o processo de desenvolvimento do produto. A habilitação do software para Unicode provavelmente custará mais, mas economiza a dificuldade de redesenhar o software se o cliente quiser ampliar a cobertura para os mercados emergentes do Oriente.

Saber sob qual sistema operacional o programa será rodado é importante para o desenvolvimento de qualquer produto de software, mas é mais crítico ao se considerar a internacionalização. Um das melhores práticas de internacionalização é deixar o máximo das funções locale-específicas o possível a cargo do sistema operacional. Isto só pode acontecer se houver uma versão localizada do sistema operacional para o *locale* desejado pelo cliente. Se o cliente deseja projetar um programa para um *locale* no qual não há nenhum sistema operacional localizado, será necessário um trabalho extra a fim de se prover as funcionalidades localizadas fornecidas usualmente pelo sistema operacional. Neste momento a equipe deve considerar se este trabalho extra é viável.

O cliente também tem que determinar quais softwares de terceiros com que este produto precisa ser compatível. Todos os *locales* precisam ser analisados em separado devido ao fato que softwares e hardwares que são padrão ou líderes de mercado em um *locale* podem não usufruir desta posição em outros.

É nesse ponto que se atinge o ponto crítico do processo de desenvolvimento de software internacional: a definição dos textos a serem localizados e o isolamento dos mesmos nos *resource files*, como discutido nos capítulos anteriores. Com o auxílio da UNL, usa-se tais textos como *input* do EnCo, criando um *resource file* independente de qualquer linguagem e pronto para ser decodificado no processo de localização.

Também é na Fase de Especificação que os aspectos locais serão contemplados, como os formatos de data, moeda, hora, etc. Essas características estão definidas no arquivo de *locale* correspondente a cada região a ser internacionalizada, estando esses arquivos amplamente disponíveis na Internet.

Todas as decisões feitas pela equipe de especificação até este ponto determinarão até que nível o novo produto poderá ser internacionalizado. Para ser internacionalizado completamente, o produto deve ser compatível com o Unicode. Também deve ser igualmente possível construir uma versão localizada para qualquer *locale* definido na Fase de Análise de Requisitos. Entretanto, satisfazer os requisitos para todos os *locales* pode ser financeiramente inviável, assim como o uso dos caracteres Unicode e outros métodos de internacionalização podem fazer o código tornar-se inaceitavelmente grande. É neste momento que o cliente e o desenvolvedor têm que executar uma análise de custo/benefício, não só para a versão atual do produto, mas também para versões futuras do mesmo a fim de determinar o grau de viabilidade financeira dos futuros processos de localização do software.

À medida que os diagramas de fluxo de dados são desenhados, os desenvolvedores começam a determinar os tipos de dados e objetos que serão importantes para o sistema, bem como os objetos que serão afetados pelos aspectos de codificação para a internacionalização (caracteres de 16 bits, por exemplo). Essa análise tornar-se-á cada vez menos necessária ao passo que mais produtos internacionalizados são desenvolvidos, porém a mesma é muito importante quando se trata de equipes que trabalham em projetos com graus variáveis de internacionalização – como no caso da determinação se uma rotina de 8 bits será aceitável em um projeto ou não.

6.2.3. Fase de Design

Durante a Fase de Design do software, os desenvolvedores decompõem o software em módulos, de modo que o design desses produtos possa ser completamente efetivado. Os desenvolvedores partem do documento de especificação que detalha o que o software tem que fazer e devem transformar essas informações no documento de design, que descreve como o software executará estas tarefas.

Durante o design lógico, a especificação de documento é decomposta modularmente em unidades funcionais do programa desejado e a interação entre estas unidades é traçada e definida A segunda fase do processo de design envolve o design modular detalhado. São identificados e criados algoritmos e estruturas de dados para cada um dos módulos. Durante

todo o processo de design, são necessárias verificações e testes constantes para assegurar a consistência do produto ao longo do ciclo de desenvolvimento. O resultado da fase de design é um documento que detalha como codificar cada módulo do novo programa explicitamente.

É nessa fase que os aspectos técnicos da internacionalização devem ser abordados. Aspectos derivados dos *character sets*, tais como os *strings* compostos, os *hotkeys* e principalmente aspectos de codificação como o ordenamento de caracteres e a correta identificação de maiúsculas e minúsculas são projetados nesse momento.

Se o projeto não será internacionalizado completamente, o time deve ao menos documentar as partes do programa que seriam afetadas por futuros esforços de internacionalização – por exemplo, indicando módulos que são sensíveis á caracteres de 8 ou 16 bits.

6.2.3.1. Impacto da Internacionalização e do Uso da UNL

O impacto de internacionalização nesta fase do ciclo de desenvolvimento está basicamente derivado a omissão. Como são decompostos os módulos e a estrutura deles é desfeita, os designers têm que prestar atenção às partes do programa que podem ser sensíveis aos aspectos da internacionalização. Isto inclui os textos de input e output, elementos da interface e outras preocupações tais como as apontadas na revisão bibliográfica feita neste trabalho. Algumas empresas desenvolvedoras de software podem escolher ter um analista cuja função exclusiva é revisar os módulos de design para a devida internacionalização, mas uma alternativa melhor seria treinar todos os designers em todos os aspectos da internacionalização de forma que eles possam implementá-las da mesma forma que eles implementariam qualquer outro aspecto arquitetônico nos módulos a serem projetados.

Se o produto não vai ser internacionalizado completamente (só habilitado para caracteres de 8-bits, porém capaz de lidar com uma grande variedade de idiomas Ocidentais, por exemplo) a equipe de design deve documentar as áreas sensíveis à internacionalização do modo que sejam facilmente identificados no processo de design. Se este passo é dado da maneira correta, qualquer internacionalização completa que seja feita no futuro será

facilitada, pois a equipe de design já identificou as partes de código que precisam ser reescritas.

6.2.4. Fase de Implementação e Integração

Durante a Fase de Implementação e Integração, o documento de design é traduzido em código da linguagem de programação utilizada para o projeto. A primeira tarefa nesta fase, porém, é escolher quais as linguagens de programação apropriadas. Cada desenvolvedor escreve os módulos que serão testados separadamente, integrados um ao outro e retestados. O cerne dos testes acontece durante esta fase.

6.2.4.1. Impacto da Internacionalização e do Uso da UNL

Devido ao fato da Fase de Implementação e Integração ser onde as palavras se transformam em ação no ciclo de vida de desenvolvimento de software, é crucial que os desenvolvedores envolvidos nesta fase estejam intimamente instruídos sobre os aspectos do design do software e da codificação que são afetados pela internacionalização. A primeira tarefa da equipe envolvida nesta fase de desenvolvimento, como previamente foi mencionada, é selecionar uma linguagem de programação. Devem ser feitas considerações acerca do suporte adequado da linguagem à internacionalização. Isto significa o fornecimento de rotinas e ferramentas que simplifiquem a codificação dos módulos internacionalizados.

Os testes que usualmente ocorrem durante esta fase do ciclo de desenvolvimento são consideravelmente mais complexos quando o produto está sendo internacionalizado. O produto deve ser testado para todos os potenciais *locales* em que o mesmo será utilizado. Isso inclui a execução do programas em versões locais do sistema operacional a fim de assegurar a interação correta. Também significa testar sua compatibilidade com o software de terceiros de cada um desses *locales*. Finalmente, significa executar testes de usabilidade em cada uma das versões localizadas com assuntos que sejam pertinentes aos nativos desses locais.

Os encarregados pelos testes também devem testar as versões localizadas do software, o que não só inclui o código original internacionalizado, mas também os arquivos de localização, tais como os *resource files*. Isto significa que existem vários níveis de testes que devem ser efetuados imediatamente. O *resource files* e suas habilidades de interação com o núcleo internacionalizado do software devem ser testados simultaneamente. Será trabalho dos testadores determinar se qualquer erro resultante surge de erros usuais de programação, erros de internacionalização ou erros nos *resource files*.

6.2.5. Fase de Manutenção

A fase de manutenção somente é uma repetição das fases prévias num menor contexto. Seu propósito é incrementar o software com novas funcionalidades ou consertar os erros detectados durante o seu uso.

6.2.6. Resumo do *Framework*

Ciclo de Vida do Desenvolvimento de Software - Atividades de Internacionalização

1. Fase de Análise de Requisitos

1.1. determinação do *locales* iniciais e futuros.

1.2. execução da análise de requisitos para cada *locale* inicial e/ou futuro.

1.3. determinação se é necessário ou lógico internacionalizar este produto de software, baseado nos requisitos para cada *locale.*

1.4. criação das diretrizes de estilo, glossários e listas de terminologias.

2. Fase de Especificação

2.1. decisão quais *locales* serão desenvolvidos agora e quais *locales* somente serão habilitados para desenvolvimento posterior (se aplicável).

2.2. decisão sobre um *character set*.

2.3. determinação de quais sistemas operacionais o produto deve rodar.

2.4. determinação de quais softwares de terceiros o produto deve ser compatível, conforme o *locale*.

2.5. finalização da determinação do nível de internacionalização que será atingido no desenvolvimento deste produto de software.

2.6. separação do texto localizável do código do software, através dos *resource files*.

2.7. identificação de aspectos regionais a serem localizados, tais como formato de numeração, moedas e formatos de data e hora.

2.8. identificação de quais processos, módulos ou objetos serão afetados pela internacionalização.

3. Fase de Design

3.1. determinação de como os arquivos de *locale* serão dispostos e acessados.

3.2. determinação das partes do software que serão deixadas desinternacionalizadas, no caso do produto não ser completamente internacionalizado (para que a internacionalização futura ser feita).

3.3. definições acerca de aspectos técnicos da internacionalização relacionados à implementação do programa.

4. Fase de Integração e Implementação

4.1. escolha de uma linguagem que forneça o apoio adequado para o nível de internacionalização almejado para este projeto.

4.2. teste completo do produto para todos as versões localizadas, incluindo os scripts, *character sets* utilizados e as interações com sistemas operacionais locais e software de terceiros locais.

5. Fase de Manutenção

5.1. executar os passos 1 - 4 num menor contexto.

7. CONSIDERAÇÕES FINAIS

7.1. Resultados Obtidos

A Internacionalização e Localização de software são fatores estratégicos tanto para o governo brasileiro, que almeja o aumento das relações comerciais internacionais quanto para as empresas brasileiras que lutam para vencer as barreiras comerciais internacionais. Os mercados a serem explorados são imensuráveis e as oportunidades advindas desse cenário são infinitas.

De encontro a isso vem o crescimento do sistema UNL, criado a fim de diminuir os entraves causados pelas diferenças lingüísticas e culturais. A linguagem UNL é um bem da humanidade e mantida por uma fundação ligada às Nações Unidas. Esta foi homologada por uma rede de cientistas de renome internacional. O Sistema UNL conta com diversos utilitários que permitem à linguagem UNL ter "vida". Alguns destes sistemas já estão operacionais e outros ainda são objetos de pesquisa.

De tal sorte, tornam-se evidentes os benefícios advindos da integração desse promissor sistema com as metodologias de desenvolvimento para a Internacionalização e Localização de sofware.

7.2. Dificuldades Encontradas

Durante a elaboração do trabalho foram encontradas algumas dificuldades que serão relatadas nessa seção do trabalho. Alguns dos objetivos originais tiveram que ser alterados, bem como alguns aspectos da pesquisa tiveram que ser reestruturados.

O maior entrave encontrado no trabalho foi, sem sombra de dúvidas, o uso do Sistema UNL. Apesar do crescimento considerável nos últimos anos, o Sistema UNL ainda apresenta-se numa fase experimental, acarretando inúmeras dificuldades na integração a um aspecto teórico como o proposto por esse trabalho.

Os conversores e deconversores disponibilizados pela UNDL Foundation mostraram-se inviáveis para a utilização no tema proposto, em função de seu estado rudimentar de desenvolvimento.

Outra grande dificuldade diz respeito à ideologia adotada pela UNDL Foundation para com o Sistema UNL. Trata-se de um sistema fechado, onde as contribuições e incrementos nas ferramentas desse Sistema mostram-se impossíveis, o que impossibilita uma maior interação com o meio acadêmico. Apesar disso, o Sistema UNL mostra-se promissor. Os aspectos técnicos que fundamentam o funcionamento do Sistema são sólidos e inovadores, mesmo no estado experimental do desenvolvimento do mesmo.

Mesmo com essas dificuldades, o *framework* mostrou-se viável e aplicável. A integração dos aspectos da internacionalização e da localização de software a um processo de desenvolvimento de software mostra-se de grande valia, podendo servir de base às empresas que almejam desenvolver software global.

7.3. Sugestões para Trabalhos Futuros

As principais sugestões para trabalhos futuros dizem respeito à evolução do Sistema UNL. Á medida que esse Sistema se sedimentar e a base instalada de conversores e deconversores para as mais variadas línguas forem se estabelecendo no mercado, mais aspectos da internacionalização e localização de software poderão ser adicionadas ao trabalho.

Além disso, outras sugestões dizem respeito ao uso de outros modelos para Ciclos de Vida de Desenvolvimento de Software. O *framework* desenvolvido mostra-se facilmente adaptável ao Modelo em Espiral, ao Modelo Incremental, dentre outros.

REFERÊNCIAS

[1] SOFTEX. Disponível em: http://www.softex.br/ Acesso no dia: 14/10/2004

[2] WATKINS, J. **The Guide to Translation and Localization: Preparing Products for the Global Marketplace**. Portland, Oregon: Lingo Systems, 2001.

[3] ESSELINK, B. **A Practical Guide to Localization**. 2nd Edition. Amsterdam/Philadelphia: John Benjamins, 2000.

[4] LISA. **Best Practices Guide – Implementing Machine Translation**. Disponível em: http://www.lisa.org/products/bestPractice/. Acesso no dia 15/08/2004

[5] HUTCHINS, J.: **Towards a New Vision for MT**. Disponível em: http://www.mt-archive.info/MTS-2001-Hutchins.pdf . Acesso em 16/08/2004

[6] LANGLAIS, P., et al. **Computer-Aided Translation Typing System**.Disponível em: http://www.mt-archive.info/ACL-2000-Langlais.pdf. Acesso no dia: 16/08/2004

[7] TURIAN, J. P., **Evaluation of Machine Translation and its Evaluation** Disponível em**:** http://www.mt-archive.info/MTS-2003-Turian.pdf. Acesso no dia: 16/08/2004.

[8] GOUGH, N. **Example-based Controlled Translation**. Disponível em: http://www.mt-archive.info/EAMT-2004-Gough.pdf. Acesso no dia: 16/08/2004,

[9] ALLEN, J., et al. **Text-to-speech: the MITalk System**, Cambridge University Press, 1997.

[10] PROJETO UNDL BRASIL. Disponível em**: http://www.undl.org.br/**. Acesso no dia 22/09/2004.

[11] HUTCHINS, J., et al . **An Introduction to Machine Translation**. Academic Press, 1992.

[12] DILINGER, M. **Notas sobre Semântica**. Departamento de Lingüística, UFMG. Minas Gerais, 1993.

[13] UNL FOUNDATION. **UNL: Universal Networking Language - An Electronic Language for Communication, Understanding and Collaboration**. UNU/IAS/UNL Center.Tóquio, Japão. 1997.

[14] UCHIDA, H., et al. **The UNL, a Gift for a Millenium**. UNU/IAS/UNL Center. Tóquio, Japão. 1999.

[15] UNL FOUNDATION. **DeConverter Specification. Version 3.0** (Tech. Rep. UNL - TR1997 - 010). UNU/IAS/UNL Center. Tóquio, Japão. 2003.

[16] MARTINS, R., et al. **As Regras Gramaticais para a Decodificação UNL-Português no Projeto UNL**. Relatório Técnico 67. Instituto de Ciências Matemáticas e da Computação. Universidade de São Paulo, São Carlos.

[17] LARMAN, C. **Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process**. São Paulo: Prentice-Hall, 2002.

[18] JACOBSON, I., et al. **Object-Oriented Software Engineering – A Use Case Driven Approach**. Harlow: Addison-Wesley, 1992.

[19] PRESSMAN, R. **Engenharia de Software**. São Paulo: McGraw Hill, 2003.

[20] SOMMERVILLE, N. **Engenharia de Software**. São Paulo: Prentice-Hall, 2001.

The Universal Networking Language (UNL)
Especificações
Versão 3.0

UNL Center
UNDL Foundation
1 de Julho de 2004

Introdução

A Internet emergiu porque a infra-estrutura de informação global revolucionou não apenas o acesso à informação, como também a velocidade que é transmitida e recebida. Com a tecnologia do correio eletrônico, por exemplo, os povos podem comunicar-se rapidamente através de longas distâncias. Nem todos os usuários, entretanto, podem usar seu próprio idioma para se comunicar. A Universal Networking Language (UNL) é um idioma artificial na forma de uma rede semântica para que os computadores expressem e troquem todo tipo de informação. Desde o advento dos computadores, os pesquisadores em torno do mundo trabalham para desenvolver um sistema que supere as barreiras do idioma. Enquanto muitos diferentes sistemas foram desenvolvidos por diversas organizações, cada um tem sua representação especial de um determinado idioma. Isto resulta em incompatibilidades entre sistemas. É então impossível quebrar barreiras de idioma no mundo inteiro, até mesmo se todos os resultados forem combinados em um único sistema. Frente a este cenário, surgiu o conceito de UNL como um idioma comum para todos os computadores existentes. Com o surgimento da UNL, poderemos aplicar os resultados de pesquisa do passado aos desenvolvimentos atuais e construirmos uma infra-estrutura de pesquisa e desenvolvimento do futuro.

O que é a UNL?

A UNL consiste em palavras universais (UWs), relações, atributos e na base de conhecimento da UNL. As palavras universais constituem o vocabulário, as relações e os atributos constituem a sintaxe e a base de conhecimento constitui a semântica da UNL.

Por que é a UNL é necessária?

No futuro, um computador precisará ser capaz de processar o conhecimento. Conhecimento este que torna o computador capaz de assumir os pensamentos e julgamentos de humanos, usando o conhecimento de humanos. O processamento deve ser baseado em conteúdos. Para os computadores terem conhecimento eles precisam sabe como funciona o processo do conhecimento. É necessário que os computadores tenham um idioma ordenado em forma de conhecimento para processarem conteúdos como humanos. A UNL é um idioma para os computadores alcançarem isto. A UNL pode expressar conhecimento como um idioma natural. A UNL também pode expressar conteúdos como um idioma natural.

A vantagem de uma linguagem comum para computadores

A UNL reduz drasticamente os custos de conhecimento e em desenvolvimento ou conteúdos necessários para processar conhecimento compartilhando e conteúdo. Além disso, se o tipo de conhecimento requerido por fazer algo através de software e descrever numa linguagem para computadores, como a UNL, o software só precisará interpretar instruções escritas nesta linguagem para poder executar suas funções. E então outro software poderá compartilhar estas instruções. É então possível acumular tal conhecimento da mesma maneira por computadores como numa biblioteca para humanos.

Como a UNL expressa a informação?

A UNL representa a informação, buscando o sentido, sentença por sentença. A informação da sentença é representada como um hipergrafo que tem palavras universais (UWs) como nós e relações como arcos. Este hipergrafo é representado também por um conjunto de relações binárias dirigidas, uma entre cada duas das UWs atuais na sentença. A UNL expressa a informação que classifica a objetividade e a subjetividade. A objetividade é expressa usando UWs e relações. A subjetividade é expressa usando atributos ligados às UWs..

Capítulo 1: Expressões UNL

As relações binárias são os blocos de construção de expressões UNL. Elas são compostas de uma relação e de duas UWs. Esta seção trata da definição e da interpretação das relações binárias da expressão UNL.

Há duas formas para expressar as expressões UNL, uma é em forma de tabela e a outra é a forma de lista. A forma de tabela de uma expressão UNL é mais legível do que a forma de lista, mas a forma de lista de uma expressão UNL é mais compacta do que a forma de tabela. Aqui, somente a forma de tabela será explicada e a forma de lista será mostrada no Capítulo 5.

Qualquer componente, tal como uma palavra, frase ou título e, naturalmente, uma sentença de uma língua natural pode ser representada com as expressões UNL. Uma expressão UNL consiste então numa UW ou de uma (ajuste de) relação(s) binária. Num documento UNL, a expressão UNL para a sentença é incluída pelos Tags {unl} e {/unl} dentro dos tags [S] e [/S]. Se uma expressão UNL consistir em uma UW, esta UW deve ser incluída também pelo Tag [W] e pelo [/W].

Assim, a expressão UNL de uma sentença é a seguinte:

{unl}
<Binary Relation>
...
{/unl}

ou,

{unl}
[W]
<UW><Attribute List>
[/W]
{/unl}

Sintaxe de uma relação binária

Uma relação binária é composta como segue:

<Binary Relation> ::= <Relation Label> [“:”<Compound UW-ID>] “(“
{{ <UW 1> [":" <UW-ID 1>]} | { “:” <Compound UW-ID$_1$> }}[<Attribute List>] “,”
{{ <UW 2> [":" <UW-ID 2>]} | { “:” <Compound UW-ID$_2$> }}[<Attribute List>] “)”
<Relation Label> ::= a relação em si. Veja o "Capítulo 2: Relações"
<UW> ::= uma UW. Veja o "Capítulo 3: Palavras Universais"
<Attribute List> ::= { “.” <Attribute> } …
<Attribute> ::= um atributo. Veja o "Capítulo 4: Atributos"
<UW-ID> ::= dois caracteres de ‘0 - 9' e 'A - Z'
<Compound UW-ID> ::= número decimal de dois-dígito (00 - 99)
00 é usado para representar a sentença principal, que pode ser omitido.

As UW-IDs compostas são sequências de dois dígitos usadas para identificar cada instância especificada por UWs compostas. UWs compostas são grupos de relações binárias – (também conhecidas como "Espaço-Nós") que podem se referir como uma UW.

A seguir, um exemplo de uma expressão UNL da sentença: "eu posso ouvir um cão latindo lá fora".

```
{unl}
aoj(hear(icl>perceive(agt>person,obj>thing)).@entry.@ability, I)
obj(hear(icl>perceive(agt>thing,obj>thing)).@entry.@ability, :01)
agt:01(bark(agt>dog).@entry, dog(icl>mammal))
plc:01(bark(agt>dog).@entry, outside(icl>place))
{/unl}
```

Nas expressões UNL acima, o **"aoj"**, **"agt"** e **"obj"** são os **relations labels**, o "**hear(icl>perceive(agt>person, obj>thing))**" , o "**I**", o "**bark(agt>dog**)", “**dog(icl>mammal)**" e o "**outside(icl>place)**" são as UWs, e "**:01**", que aparece três vezes no exemplo, mostram a UW-ID composta. A UW-ID composta aparece na posição de uma UW, também conhecida por “espaço-nó”, é usada para citar ou se referir a uma UW composta definida previamente. As relações binárias indicadas pela UW-ID composta definem o conteúdo do escopo. Um scope-node começa sempre com ":" seguido pelos dois dígitos de uma UW-ID composta.

As UW-IDs são omitidas da expressão UNL acima. Quando uma UW é única em uma expressão UNL, a UW-ID pode ser omitida.

A UW-ID é usada para indicar alguma informação referencial, onde há duas ou mais ocorrências diferentes do mesm o conceito (não são co-referentes). Normalmente, se a mesma UW ocorrer ma is de uma vez, está entendido para se referenciar à mesma entidade ou oco rrência. Por exemplo, se um homem cumprimentasse um outro homem, a mesma UW seria usada duas vezes -- “man(icl>male person)” tornando possível distinguir um do outro, com as UW-IDs:

`man(icl>male person):01` para o primeiro e
`man(icl>male person):02` para o outro, pois o primeiro homem não cumprimentou a si mesmo.

Capítulo 2: Relações

Esta seção trata da definição e da interpretação dos rótulos da relação da UNL. As relações entre UWs em relações binárias têm rótulos diferentes de acordo com os diferentes papéis que representam. Estas relações-Rótulo (Relation-Labels) são listadas e definidas abaixo.

Rótulos de Relação (Relation Labels)

Um rótulo da relação é representado por uma palavra de 3 caracteres ou menos. Há muitos fatores a serem considerados em escolher um inventário das relações. Os princípios para escolher relações são como segue:

Principio-1: A condição necessária

Quando uma UW tem relações entre mais de duas outras UWs, cada rótulo da relação deve ser ajustado para poder identificar cada relação na premissa que há bastante conhecimento sobre o conceito de cada UW expressada.

Principio-2: A condição suficiente

Quando há relações entre UWs, cada rótulo da relação deve ser ajustado para poder compreender o papel de cada UW somente se referindo ao rótulo da relação. São as seguintes as relações definidas de acordo com os princípios acima.

agt (agent) agente

Agt define uma coisa que inicia uma ação.

agt (ação, coisa)

Sintaxe

```
agt [":"<Compound UW-ID>] "(" {<UW1 >|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"
```

Definição Detalhada

Um agente é definido como a relação entre:
UW1 – uma ação (do), e
UW2 - uma coisa (thing)
onde:
UW2 inicializa UW1, ou
UW2 é tida como um papel direto em fazer que UW1 aconteça.

Exemplos e Leituras

`agt(break(agt>thing,obj>thing), John(icl>person))`	John breaks …
`agt(translate(agt>thing,gol>language,obj>information, src>language), computer(icl>machine))`	computer translates …
`agt(run(icl>act(agt>volitional thing)), car(icl>vehicle))`	car runs …
`agt(break(agt>thing,obj>thing), explosion(icl>event))`	explosion breaks …

Relações Relacionadas

Um agente é diferente do **cag** já que um agente inicia a ação, enquanto que um co-agente inicia uma ação diferente, acompanhada.

Um agente é diferente do **ptn** já que um agente é o iniciador focalizado da ação, enquanto um sócio é um iniciador não-focalizado.

Um agente é diferente do **con** já que um agente é o iniciador focalizado da ação, enquanto uma condição indireta, geralmente não focalizada, influencia na ação.

and (conjunction) conjunção

And define uma relação conjuntiva entre conceitos.

`and (UW,UW)`

Sintaxe

```
and [":"<Compound UW-ID>] "(" {<UW 1>|":"< Compound UW-ID>} ","
{<UW2>|":"< Compound UW-ID>} ")"
```

Definição Detalhada

Uma conjunção é definida como a relação entre:

UW1 - um conceito, e
UW2 - um outro conceito,
onde:

as UWs são diferentes, e
UW1 e UW2 é vista como agrupamento, e
o que é dito de UW1 é dito também de UW2.

Exemplos e Leituras

`and(quickly, easily)`	… easily and quickly
`and(sing(agt>person), dance(agt>person))`	… singing and dancing
`and(Mary(icl>person), John(icl>person))`	… John and Mary

Relações Relacionadas

Uma conjunção é diferente de **or** pelo fato das coisas serem agrupadas para dizer a mesma coisa sobre ambos, visto que com or nós os separamos para indicar que o que é ou não é verdadeiro sobre o outro.

Uma conjunção é diferente do **cag** pelo fato que os agentes são associados, ambos iniciam um evento explícito, visto que com cag, o co-agente inicia um evento implícito.

Uma conjunção é diferente do **ptn** pelo fato que os agentes e os sócios são associados, ambos estão no foco, visto que com ptn, o sócio não está no foco (em comparação ao agente).

Uma conjunção é diferente do **coo** e **seq** no significado, embora em muitos casos as mesmas expressões possam ser usadas para ambos. Uma conjunção significa somente que os termos estão agrupados juntos; nenhuma informação sobre o tempo é implicada. Coo, de outra forma, significa que os termos se realizam no mesmo tempo, se estão considerados agrupados ou não. Por sua vez, seq significa que os termos são requisitados a tempo, um após o outro.

aoj (thing with attribute) coisa com atributo

Aoj define uma coisa que esteja em um estado ou tenha um atributo.

```
aoj (*(aoj> coisa), coisa)
aoj (coisa, coisa)
aoj (ser, coisa)
```

Sintaxe

```
aoj [":"<Compound UW-ID>] "(" {<UW1 >|":"< Compound UW-ID>} ","
{<UW2>|":"< Compound UW-ID>} ")"
```

Definição Detalhada

Uma coisa com um atributo é definida como a relação entre:
UW1 – um estado ou uma coisa que represente um estado, e
UW2 – uma coisa,
onde:
UW1 é um atributo ou um estado de UW2, ou
UW1 é um estado associado com o UW2.

Exemplos e Leituras

`aoj(red(aoj>thing), leaf(pof>plant))`	... leaf is red.
`aoj(available, information)`	This information is available for …
`aoj(nice, ski(agt>person))`	Skiing is nice.
`aoj(teacher(icl>occupation), John(icl>person))`	John is a teacher.
`aoj(have(aoj>thing,obj>thing), I)` `obj(have(aoj>thing,obj>thing), pen(icl>writing instrument))`	I have a pen.
`aoj(know(aoj>thing,obj>thing), John(icl>person))`	John knows …
`aoj:01(difficult(aoj>thing,obj>thing), it)` `aoj(:01, John(icl>person))`	It is difficult for John.

Relações Relacionadas

Uma coisa com um atributo é diferente de **mod** pelo fato que mod dá alguma limitação, enquanto que aoj dá um estado ou uma característica.

Uma coisa com um atributo é diferente de **ben** pelo fato que um beneficiário é completamente independente de um evento ou de um estado focalizado. Este evento ou estado podem ser considerados como exercendo uma influência boa ou má, enquanto o aoj tem uma relação mais próxima e pode ser considerado como a descrição de um estado ou uma característica.

Uma coisa com um atributo é diferente do **obj** pelo fato que o obj define uma coisa que seja afetada diretamente por uma ação ou por um fenômeno, enquanto, o aoj define uma coisa em um estado.

bas (basis for expressing a degree) base para expressar um grau

Bas define uma coisa usada como a base (padrão) expressando um grau.

bas (grau, coisa)

Sintaxe

bas [":"<Compound UW-ID>] "(" {<UW1 >|":"< Compound UW-ID>} ","
{<UW2>|":"< Compound UW-ID>} ")"

Definição Detalhada

Uma base é definida como a relação entre:
UW1 – um grau, e
UW2 – uma coisa,
onde:
UW1 é um grau que expressa a semelhança ou a diferença, tal como "mais", "menos", "igual", "parecido", "tanto quanto", "ao menos" etc., e
UW2 é algo usado como a base para avaliar as características ou a quantidade de alguma outra coisa (enfocada).

Exemplos e Leituras

`bas(more(aoj>thing), 7)`	Ten is three more than seven.
`bas(more(icl>how), Jack(icl>person))`	Betty weighs more than Jack (does).
`bas(same(icl>how), girl(icl>female person).@pl)`	We treat boys exactly the same as girls.
`bas(at least, :01)` `qua:01(dollar(icl>money).@pl, 500)`	It'll cost at least 500 dollars.
`man(beautiful, more(icl>how))` `bas(more(icl>how), rose(icl>flower))`	A tulip is more beautiful than a rose.

`aoj(:01, John(icl>person))`
`man:01(quiet(aoj>thing), more(icl>how))`
`bas:01(more(icl>how), shy(aoj>thing,mod<thing))`
	John is more quiet than shy.

ben (beneficiary) beneficiário

Ben define um beneficiário indiretamente relacionado ou vítima de um evento ou estado.

ben (ocorrência, coisa)
ben (fazer, coisa)
ben ((aoj>coisa), coisa)

Sintaxe

ben [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Um beneficiário é definido como uma relação entre:
UW1 – um evento ou estado, e
UW2 – uma coisa,
onde:
UW2 é visto como sendo indiretamente afetado por UW1, como o beneficiário ou a vítima.

Exemplos e Leituras

`ben(give(agt>thing,gol>thing,obj>thing)), Mary(icl>person))`	To give … for Mary.
`ben(good(aoj>thing,mod<thing)), John(icl>person))`	It is good for John to …

Relações Relacionadas

Um beneficiário é diferente do aoj em que o aoj tem uma relação próxima e pode ser considerado como a descrição de uma característica do estado, visto que um beneficiário é bastante independente de um evento ou de um estado focalizado, mas este evento ou estado podem ser considerados como exercendo uma influência boa ou má.

cag (co-agent) co-agente

Cag define uma coisa secundária que inicia um evento implícito que seja feito em paralelo.

`cag (fazer, coisa)`

Sintaxe

cag [“:”<Compound UW-ID>] “(“ {<U W1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Um co-agente é definido como a relação entre:
UW1 – uma ação, e
UW2 – uma coisa
onde:
Há uma ação implícita de que seja independente, mas acompanha, UW1, e
UW2 é visto como iniciando a ação implícita, e
UW2 a ação implícita é vista como não estando no foco (em comparação à ação do agente).

Exemplos e Leituras

`cag(walk(icl>do), John(icl>person))`	To walk with John
`cag(live(icl>do), aunt(icl>person))`	To live with … aunt

Relações Relacionadas

Um co-agente é diferente do **agt** naquelas ações independentes diferindo daquelas que ocorrem para um agente e um co-agente. Além disso, um agente e sua ação estão no foco, emquanto um co-agente e sua ação não estiverem no foco.

Um co-agente é diferente do **ptn** quando o co-agente inicia uma ação que seja independente da ação de um agente, enquanto um sócio inicia a mesma ação junto com um agente. Um co-agente é diferente do con quando um co-agente inicia uma ação não-focalizada, considerando que uma condição é uma influência indireta na ação focalizada.

cao (co-thing with attribute) co-coisa com atributo

Cao define uma coisa que não está no foco em um estado paralelo.

```
cao ((aoj>coisa), coisa)
cao (coisa, coisa)
```

Sintaxe

cao [“:”<Compound UW-ID>] “(“ {<U W1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Uma co-coisa com um atributo é definida como a relação entre:
UW1 – um estado ou uma coisa que represente um estado, e
UW2 – uma coisa,
onde:
Há um estado implícito de que seja independente, mas acompanha, UW1, e
UW2 está em um estado implícito, ou
UW2 é associado com um estado implícito.

Exemplos e Leituras

`cao(exist(icl>be), you)`	… be with you

Relações Relacionadas

Uma co-coisa com um atributo é diferente do **aoj** naquele lá é um diferente, estado independente para uma coisa com um atributo e uma co-coisa com um atributo, respectivamente.

cnt (content) conteúdo

Cnt define um conceito equivalente.

```
cnt (coisa, coisa)
```

Sintaxe

cnt [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Um conteúdo é definido como a relação entre:
UW1 – uma coisa, e
UW2 – uma coisa,
onde:
UW2 é o conteúdo ou uma explanação de UW1.

Exemplos e Leituras

`cnt(UNL(icl>Universal Networking Language), Universal Networking Language)`	UNL, Universal Networking Language
`cnt(Internet(icl>communication network), amalgamation(icl>harmony))`	The Internet: an amalgamation
`cnt(language generator, deconverter.@double_quotation)`	a language generator "deconverter"…

cob (affected co-thing) co-coisa afetada

Cob define uma coisa que seja afetada diretamente por um evento implícito feito em paralelo ou em um estado implícito em paralelo.

```
cob (ocorrência, coisa)
cob (fazer, coisa)
cob ((aoj> coisa,obj> coisa), coisa)
```

Sintaxe

cob [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Um "co-objeto" é definido como a relação entre:
UW1 - um evento ou um estado, e
UW2 - uma coisa,
onde:
UW2 é vista como afetada diretament e por um evento implícito feito em paralela ou um estado implícito em paralela.

Exemplos e Leituras

cob(die(obj>living thing), Mary(icl>person))	… dead with Mary
obj(injure(icl>hurt(agt>thing,obj>living thing), John(icl>person)) cob(injure(icl>hurt(agt>thing,obj>living thing), friend(icl>comrade).@pl) pos(friend(icl>comrade).@pl, he)	John was injured in the accident with his friends

Relações Relacionadas

Um co-objeto é diferente do **obj** em que o obj está no foco, enquanto o cob é relacionado a um segundo, a um evento implícito ou a um estado não-focalizado.

con (condition) condição

Con define um evento não-focalizado ou indica as condições de um evento ou um estado focalizado.

```
con (ocorrência, ocorrência)
con (ocorrência, fazer)
con (ocorrência, (aoj> coisa))
con (fazer, ocorrência)
con (fazer, fazer)
con (fazer, (aoj> coisa))
con ((aoj> coisa), ocorrência)
con ((aoj> coisa), fazer)
con ((aoj> coisa), (aoj> coisa))
```

Sintaxe

con [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Uma condição é definida como a relação entre:
UW1 - um evento ou um estado focalizado, e
UW2 - um evento ou um estado condicionando,

onde:
UW1 e UW2 são diferentes e
UW2 é visto como tendo um papel indireto ou externo em fazer UW1 acontecer, isto é, como algum fator condicionando ou inibindo (real ou hipotético) essas influências se quando UW1 pode acontecer.

Exemplos e Leituras

```
aoj:01(tired(aoj>thing,mod<thing), you)
con(go(icl>move(agt>thing,gol>place,src>place)), :01)
```

If you are tired, we will go straight home.

coo (co-occurrence) co-ocorrência

Coo define um evento co-ocorrente ou um estado para um evento ou um estado focalizado.

```
coo (ocorrência, ocorrência)
coo (ocorrência, fazer)
coo (ocorrência, (aoj>coisa))
coo (fazer, ocorrência)
coo (fazer, fazer)
coo (fazer, (aoj> coisa))
coo ((aoj> coisa), ocorrência
coo ((aoj> coisa), fazer)
coo ((aoj> coisa), (aoj>coisa))
```

Sintaxe

coo [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Uma co-ocorrência é definida como a relação entre:
UW1 - um evento ou um estado focalizado, e
UW2 - um evento co-ocorrente ou um estado,
onde:
UW1 e UW2 são diferentes, e
UW1 ocorre ou é verdadeira ao mesmo tempo que UW2.

Exemplos e Leituras

coo(run(icl>act(agt>volitional thing)), … was crying while running
cry(icl>weep(agt>volitional thing))

coo(red(aoj>thing,mod<thing), ... is red while … is hot
hot(aoj>thing,mod<thing))

Relações Relacionadas

Uma co-ocorrência é diferente de **seq** em que seq descreve os eventos ou os estados que não ocorrem ao mesmo tempo, mas um após o outro, enquanto o coo descreve os eventos que ocorrem simultaneamente.

Uma co-ocorrência é diferente do **tim** em que o coo relaciona os tempos dos eventos ou dos estados com outro s eventos ou estados, visto que o tim relaciona eventos ou estados diretamente com pontos ou intervalos do tempo.

dur (duration) duração

Dur define um período de tempo durante o qual um evento ocorre ou um estado existe.

```
dur (ocorrência, período)
dur (ocorrência, evento)
dur (ocorrência, estado)
dur (ocorrência, ocorrência)
dur (ocorrência, fazer)
dur (ocorrência, *(aoj> coisa))
dur (do, período)
dur (do, evento)
dur (do, estado)
dur (do, ocorrência)
dur (do, fazer)
dur (do, *(aoj>coisa))
dur (*(aoj>coisa), período)
dur (*(aoj>coisa), evento)
dur (*(aoj>coisa), estado)
dur (*(aoj>coisa), ocorrência)
dur (*(aoj>coisa), fazer)
dur (*(aoj>coisa), *(aoj>coisa))
```

Sintaxe

dur [“:”<Compound UW-ID>] “(“ {< UW1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Uma duração é definida como a relação entre:
UW1 - um evento ou um estado, e
UW2 - um período durante que o evento ou o estado continuam.

Exemplos e Leituras

`dur(work(agt>person), hour(icl>period))` `qua(hour(icl>period), 9)`	… work nine hours (a day)
`dur(talk(icl>express(agt>thing,gol>person,` `obj>thing), meeting(icl>event)`	… talk … during meeting
`dur(come(icl>move(agt>thing,gol>place,` `src>place), absence(icl>state))`	… come during (my) absence

fmt (range: from-to) escala: de-para

Fmt define uma escala entre duas coisas.

`fmt (coisa, coisa)`

Sintaxe

fmt [":"<Compound UW-ID>] "(" {<UW1 >|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Uma escala (de-para) é definida como a relação entre:
UW1 - uma coisa escala-inicial, e
UW2 - uma coisa escala-final,
onde:
As UWs são diferentes, e
UW2 descreve o começo de uma escala e UW1 descreve o final.

Exemplos e Leituras

`fmt(a(icl>letter), z(icl>letter))`	The alphabets from A to Z
`fmt(Osaka(icl>city), NewYork(icl>city))`	The distance from Osaka to New York
`fmt(Monday(icl>day), Friday(icl>day))`	The weekdays from Monday to Friday

Relações Relacionadas

Uma escala é diferente do **src** e **gol** naquele para o src e o gol os estados iniciais e finais de determinado obj são caracterizados com respeito a algum evento, enquanto o fmt faz uma caracterização similar mas sem ligar o ponto final de uma escala a algum evento.

Uma escala é diferente do **plf** e do **plt** ou do **tmf** e do **tmt** que o fmt define ponto final de uma escala sem referência a toda a sorte de evento, enquanto o plf, plt, tmf e o tmt limitam eventos.

frm (origin) origem

Frm define a origem de uma coisa.

```
frm (coisa, coisa)
```

Sintaxe

frm [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Uma origem é definida como a relação entre:
UW1 - uma coisa, e
UW2 - a origem da coisa,
onde:
UW2 descreve a origem tal como a posição original de UW1.

Exemplos e Leituras

`frm(visitor(icl>person), Japan(icl>country))` a visitor from Japan

gol (goal: final state) objetivo: estado final

Gol define um estado final do objeto ou de uma coisa associada finalmente com o objeto de um evento.

```
gol (ocorrência (gol>coisa), coisa)
gol (fazer(gol> coisa), coisa)
```

Sintaxe

gol [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Um estado final é definido como a relação entre:
UW1 - um evento, e
UW2 - um estado ou uma coisa,
onde:
UW2 é o estado específico que descreve o obj (de UW1) no fim de UW1, ou
UW2 é uma coisa que seja associada com obj (de UW1) e o fim de UW1.

Exemplos e Leituras

`gol(change(gol>thing,obj>thing,src>thing), red(aoj>thing,mod<thing))`	the lights changed from green to red
`gol(deposit(icl>save(agt>thing,obj>thing)), account(icl>record))`	millions were deposited in a Swiss bank account.

Relações Relacionadas

Um estado final é diferente do **tmf** e **plf** que o gol descreve características qualitativas e não o tempo ou o lugar.

Um estado final é diferente do **src** que o gol descreve as características do obj no estado final do evento.

ins (instrument) instrumento

Ins define um instrumento para realizar um evento.

`ins (fazer, coisa concreta)`

Sintaxe

ins [“:”<Compound UW-ID>] “(“ {<U W1>|“:”<Compound UW-ID>} “,” {<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Um instrumento é definido como a relação entre:
UW1 - um evento, e
UW2 - uma coisa concreta,
onde:
UW2 especifica a coisa concreta que é usada a fim fazer UW1 acontecer.

Exemplos e Leituras

`ins(look(agt>thing,obj>thing), telescope(icl>optical instrument)`	look at stars through [with] a telescope
`ins(write(icl>express(agt>thing,obj>thing)), pencil(icl>stationery))`	write [draw] with a pencil
`ins(cut(agt>thing,obj>thing,opl>thing), scissors(icl>cutley))`	He cut the string with a pair of scissors

Relações Relacionadas

Um instrumento é diferente do **man** com isso man descreve um evento ao todo, enquanto o ins caracteriza um dos componentes do evento: o uso do instrumento.

Um instrumento é diferente do **met** com isso met é usado para coisas abstratas (meios ou métodos abstratos), enquanto o "ins" é usado para coisas concretas.

man (manner) maneira

Man define uma maneira realizar um evento ou as características de um estado.

```
man (occur, how)
man (do, how)
man ((aoj>thing), how)
```

Sintaxe

man [“:”<Compound UW-ID>] “(“ {<U W1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Uma "maneira" é definida como a relação entre:
UW1 - um evento ou um estado, e
UW2 - uma maneira,
onde:
As UWs são diferente, e
UW1 é feito ou existe em uma maneira caracterizada por UW2.

Exemplos e Leituras

`man(move(agt<thing,gol>place,src>place), quickly)`	move quickly
`man(visit(agt>thing,obj>thing)), often)`	I often visit him.
`man(beautiful, very(icl>how))`	it is very beautiful.

Relações Relacionadas

Uma maneira é diferente do **ins** ou **met** com isso met descreve como um evento é realizado fora dos termos das etapas dos instrumentos ou do componente do evento, enquanto man descreve outras características quantitativas ou qualitativas do evento como um todo.

met (method or means) o método ou meios

Met define meios para realizar um evento.

```
met (fazer, coisa abstrata)
met (fazer, fazer)
```

Sintaxe

met [“:”<Compound UW-ID>] “(“ {<U W1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Um "método ou os meios" são definidos como a relação entre:
UW1 - um evento, e
UW2 - uma coisa abstrata ou uma ação,
onde:
UW2 especifica a coisa abstrata usada ou as etapas realizadas a fim fazer
UW1 acontecer.

Exemplos e Leituras

`met(solve(icl>resolve(agt>thing,obj>thing)), dynamics(icl>science))`	… solve … with dynamics
`met(solve(icl> resolve(agt>thing,obj>thing)), algorithm(icl>method))`	… solve … using … algorithm
`met(separate(agt>thing,obj>thing,src>thing)), cut(agt>thing,obj>thing,opl>thing))`	… separate … by cutting …

Relações Relacionadas

Um método ou os meios são diferentes de **man** em que man descreve um evento ao todo, enquanto met caracteriza as etapas, os procedimentos ou os instrumentos do evento.

Um método ou os meios são diferentes do **ins** em que met é usado para coisas abstratas (meios ou métodos abstratos), enquanto o ins é usado para coisas concretas.

mod (modification) modificação

Mod define uma coisa que restrinja uma coisa focalizada.

```
mod (coisa, coisa)
mod (coisa, (mod>coisa))
```

Sintaxe

mod [“:”<Compound UW-ID>] “(“ {<U W1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Uma "modificação" é definida como a relação entre:
UW1 - uma coisa focalizada, e
UW2 - uma coisa que restrinja UW1 em alguma maneira.

Exemplos e Leituras

`mod(story(icl>tale), whole(mod<thing))`	the whole story
`mod(plan(icl>idea), master(mod<thing))`	a master plan
`mod(part(pof>thing), main(aoj>thing))`	the main part
`qua(block(icl>concrete thing), 3)`	… three blocks of ice …
`mod(ice(icl>solid), block(icl>concrete thing))`	

Relações Relacionadas

Uma modificação é diferente do **aoj** em que o aoj descreve um estado ou uma característica de uma coisa, visto que mod indica meramente uma limitação, que possa indiretamente sugerir algumas características da coisa descrita. A maioria das relações mod requer uma tradução (parafrasear) que introduza algum evento implícito para tornar-se mais desobstruídas, e nivelam então muitas possibilidades estão geralmente disponíveis.

Uma modificação é diferente de **man** em que man descreve uma maneira de realizar um evento ou as características de um estado.

nam (name) nome

Nam define um nome de uma coisa.

`nam (coisa, coisa)`

Sintaxe

nam [“:”<Compound UW-ID>] “(“ {<U W1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Um nome é definido como a relação entre:
UW1 - uma coisa, e
UW2 - uma coisa usada como um nome,
onde:
UW2 é um nome de UW1.

Exemplos e Leituras

```
nam(tower(icl>building), Tokyo(icl>city))
```
Tokyo tower

obj (affected thing) coisa afetada

Obj define uma coisa no foco que é afetado diretamente por um evento ou por um estado.

```
obj (ocorrência, coisa)
obj (fazer, coisa)
obj (ser, coisa)
obj ((aoj>coisa,obj>coisa), coisa)
```

Sintaxe

obj [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Uma coisa afetada é definida como a relação entre:
UW1 - um evento ou um estado, e
UW2 - uma coisa,
onde:
UW2 é pensado de como afetada diretamente por um evento ou por um estado.

Exemplos e Leituras

`obj(move(gol>place,obj>thing,src>place), table(icl>furniture))`	the table moved.
`obj(melt(gol>thing,obj>thing), sugar(icl>seasoning))`	the sugar melts into ...
`obj(cure(agt>thing,obj>thing), patient(icl>person))`	to cure the pacient.
`obj(have(aoj>thing,obj>thing), pen(icl>writing instrument))`	I have a pen.

Relações Relacionadas

Uma coisa afetada é diferente de **cob** em que obj está no foco, enquanto cob é relacionado a um segundo, a um evento ou a um estado implícito não-focalizado.

opl (affected place) lugar afetado

Opl define um lugar no foco afetado por um evento.

```
opl (fazer, lugar)
```

Sintaxe

opl [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Um lugar afetado é definido como a relação entre:
UW1 - um evento, e
UW2 - um lugar ou uma coisa que define um lugar,
onde:
UW2 é o lugar específico onde a mudança descrita por UW1 é dirigida, ou
UW2 é um lugar que seja visto como sendo afetado durante o evento.

Exemplos e Leituras

```
opl(pat(icl>touch(agt>thing,obj>thing,opl>thing)),
 shoulder(pof>trunk))
```
… pat … on shoulder

```
opl(cut(agt>thing,obj>thing,opl>thing),
 middle(icl>place))
```
… cut … in middle

Relações Relacionadas

Um lugar afetado é diferente de **obj** e **cob** em que quando afetado pelo evento é de um lugar melhor do que outros tipos das coisas.

Um lugar afetado é diferente do **plc** em que um lugar afetado está caracterizado pelo evento, quando o lugar físico e lógico definir o ambiente em que o evento acontece.

or (disjunction) disjunção

Or define uma relação disjuntiva entre dois conceitos.

```
or (coisa, coisa)
```

Sintaxe

or [":"<Compound UW-ID>] "(" {< UW1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Uma disjunção é definida como a relação entre:
UW1 - uma coisa, e
UW2 - um conceito,
onde:
As UWs são diferentes, e
Alguma descrição é verdadeira para UW1 ou UW2 (mas não ambas), ou

Alguma descrição é verdadeira para UW1 ou UW2 (e talvez ambas).

Exemplos e Leituras

`or(stay(icl>do), leave(icl>do))`	Will you stay or leave?
`or(red(icl>color), blue(icl>color))`	Is it red or blue?
`or(John(icl>person), Jack(icl>person))`	Who is going to do it, John or Jack?

Relações Relacionadas

Uma disjunção é diferente de uma conjunção em que os itens da disjunção estão agrupados a fim de dizer que algo é verdadeiro para um ou para o outro, visto que em uma conjunção ambos são agrupados para dizer que os dois são verdadeiros. Uma disjunção na lógica formal permite três situações para que seja verdadeira: 1) é verdadeiro para UW1, 2) ele é verdadeiro para UW2, 3) ele é verdadeiro para ambos. Por outro lado, uma conjunção permite somente a terceira situação.

per (proportion, rate or distribution) proporção, taxa ou distribuição

Per define uma base ou uma unidade da proporção, da taxa ou da distribuição.

`per (coisa, coisa)`

Sintaxe

per [“:”<Compound UW-ID>] “(“ {< UW1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Uma proporção, uma taxa ou uma distribuição são definidas como a relação entre:
UW1 - uma quantidade, e
UW2 - uma quantidade, ou uma coisa vista como uma quantidade,
onde:
UW1 e UW2 formam uma proporção, onde UW1 é o numerador e UW2 é o denominador, ou
UW2 é a base ou a unidade para compreender UW1, ou
Cada UW expressa uma dimensão diferente, do tamanho, por exemplo.

Exemplos e Leituras

`per(hour(icl>period), day(icl>period))`	eight hours a day
`qua(hour(icl>period), 8)`	
`per(time(icl>unit), week(icl>period))`	… twice a week

```
qua(time(icl>unit), 2)
```

plc (place) lugar

Plc define um lugar onde um evento ocorre, ou um estado que seja verdadeiro, ou uma coisa que existe.

```
plc (ocorrência, coisa)
plc (fazer, coisa)
plc ((aoj>coisa), coisa)
plc (coisa, coisa)
```

Sintaxe

plcl [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Um lugar é definido como a relação entre:
UW1 - um evento, um estado, ou uma coisa, e
UW2 - um lugar ou uma coisa compreendida como um lugar.

Exemplos e Leituras

`plc(cook(icl>do), kitchen(pof>building))`	… cook … in the kitchen
`plc(sit(icl>do), beside(icl>relative place))`	… sit beside me
`plc(cool(aoj>thing), here(icl>how))`	It's cool here.

Relações Relacionadas

Um lugar é diferente do **plf** e **plt** ou **src** e **gol** em que plc descreve um lugar com respeito a um evento completo, enquanto estas outras relações descrevem a posição com respeito às partes de um evento.

Um lugar é diferente do **opl** em que o plc não é visto como sendo modificado por um evento mas meramente como um ponto de referência caracterizando-o, enquanto que opl é visto como sendo modificado.

plf (initial place) lugar inicial

Plf define um lugar onde um evento começa ou um estado que se torne verdadeiro.

```
plf (ocorrência, coisa)
plf (fazer, coisa)
plf ((aoj>coisa), coisa)
```

Sintaxe

plf [“:”<Compound UW-ID>] “(“ {< UW1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Um “lugar inicial" (ou "lugar-de") é definido como a relação entre:
UW1 - um evento ou um estado, e
UW2 - um lugar ou uma coisa que define um lugar,
onde:
UW2 é o lugar específico onde UW1 começou, ou
UW2 é o lugar específico de onde UW1 é verdadeiro.

Exemplos e Leituras

`plf(come(icl>do), home(icl>place))`	… come from home
`plf(deep(aoj>thing), there(icl>how))`	The sea is deep from there to here.

Relações Relacionadas

Um lugar inicial é diferente do **plc** em que plc descreve eventos ou estados levados como um todo, enquanto o plf descreve somente a parte inicial de um evento ou de um estado.

Um lugar inicial é diferente de **plt** em que plt descreve a parte final de um evento ou de um estado, enquanto o plf descreve a parte inicial de um evento ou de um estado.

Um lugar inicial é diferente de **src** em que plf descreve o lugar onde o evento começou, enquanto o src descreve o estado inicial do objeto.

plt (final place) lugar final

Plt define um lugar onde um evento termina ou um estado que se torna falso.

plt (ocorrência, coisa)
plt (fazer, coisa)
plt ((aoj>coisa), coisa)

Sintaxe

plt [“:”<Compound UW-ID>] “(“ {< UW1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Um lugar final é definido como a relação entre:
UW1 - um evento ou um estado, e
UW2 - um lugar ou uma coisa que define um lugar,
onde:
UW2 é o lugar específico onde UW1 terminado, ou
UW2 é o lugar específico onde UW2 se torna falso.

Exemplos e Leituras

`plt(travel(icl>do), Boston(icl>city))`	I'm travelling up to Boston
`plf(deep(aoj>thing), there(icl>how))`	The sea is deep from there to here

Relações Relacionadas

Um lugar final é diferente do **plc** em que plc descreve eventos ou estados levados como um todo, enquanto o plt descreve somente a parte final de um evento.

Um lugar final é diferente de **plf** em que plt descreve a parte final de um evento ou de um estado, enquanto o plf descreve a parte inicial de um evento.

Um lugar final é diferente de **gol** em que plt descreve o lugar onde um evento ou um estado terminaram, enquanto o gol descreve o estado final do objeto.

pof(part-of) parte-de

Pof define um conceito de que uma coisa focalizada é uma parte.

`pof (coisa, coisa)`

Sintaxe

pof [":"<Compound UW-ID>] "(" {<UW1 >|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Parte-de é definida como a relação entre:
UW1 - uma coisa parcial, e
UW2 - uma coisa inteira,
onde:
UW1 é uma parte de UW2.

Exemplos e Leituras

`Pof(wing(icl>limb), bird(icl>animal))` Bird's wing.

pos (possessor) possuidor

Pos define o possuidor de uma coisa.

`pos (coisa, coisa)`

Sintaxe

pos [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Um possessor é definido como a relação entre:
UW1 - uma coisa ou um lugar, e
UW2 - um humano ou não, visto como uma coisa de vontade onde:
UW2 é o possuidor de UW1.

Exemplos e Leituras

`pos(dog(icl>aminal), John(icl>person))` John's dog
`pos(book(icl>concrete thing), I)` my book

ptn (partner) parceiro

Ptn define um iniciador não-focalizado indispensável de uma ação

`ptn (fazer, coisa)`

Sintaxe

ptn [":"<Compound UW-ID>] "(" {<UW1 >|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Um parceiro é definido como a relação entre:
UW1 - uma ação, e
UW2 - um humano ou não, visto como uma coisa de vontade
onde:
UW2 é visto como tendo um papel direto em fazer uma parte indispensável de UW1 suceder, e
UW1 é o mesmo, o evento colaborativo como aquele iniciado pelo agente, e
UW2 é visto como não estando no foco (em comparação ao agente).

Exemplos e Leituras

```
ptn(compete(icl>do), John(icl>person))      … compete with John
ptn(share(icl>do(obj>thing)),               share … with the poor
poor(icl>person))
ptn(collaborate(icl>do), he)                … collaborate with him …
```

Relações Relacionadas

Um parceiro é diferente do **agt** que um agente e seu evento estão no foco, quando um parceiro e seu evento não estiverem no foco.

Um parceiro é diferente do **cag** em que um co-agente inicia um evento que seja independente do evento de um agente, visto que um parceiro inicia o mesmo evento junto com um agente.

Um parceiro é diferente de **con** em que um parceiro inicia o mesmo evento que um agente , visto que uma circunstância tem somente uma influência indireta nesse evento.

pur (purpose or objective) finalidade ou objetivo

Pur define a finalidade ou o objetivo de um agente de um evento ou uma finalidade de uma coisa que exista.

```
pur (ocorrência, ocorrência)
pur (ocorrência, fazer)
pur (fazer, ocorrência)
pur (fazer, fazer)
pur (ocorrência, coisa)
pur (fazer, coisa)
pur (coisa, ocorrência)
pur (coisa, fazer)
pur (coisa, coisa)
```

Sintaxe

pur [“:”<Compound UW-ID>] “(“ {< UW1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Uma finalidade ou um objetivo são definidos como a relação entre:
UW1 - uma coisa ou um evento, e
UW2 - uma coisa ou um evento,
onde:
As UWs são diferentes, e
uando UW1 for um evento:
UW2 especificam a finalidade ou objetivo do agente, ou

UW2 especifica a coisa (objeto, estado, evento) que o agente deseja realizar conduzindo UW1 para fora, ou
uando UW1 não for um evento:
UW2 é o que UW1 deve ser usado para.

Exemplos e Leituras

`pur(come(icl>do), see(icl>do(obj>coisa)))`	… come to see you
`pur(work(icl>do), money(icl>do))`	... work for money
`pur(budget(icl>expense), research(icl>do))`	our budget for research

Relações Relacionadas

Uma finalidade ou um objetivo são diferente do **gol** em que pur descreve os desejos de um agente, visto que gol descreve o estado do objeto no fim do evento.

Uma finalidade ou um objetivo são diferentes de **man** e **met** em que pur descreve a razão porque o evento está sendo realizado, enquanto man e met descrevem como está sendo realizado.

qua (quantity) quantidade

Qua define a quantidade de uma coisa ou de uma unidade.

`qua (coisa, quantidade)`

Sintaxe

qua [“:”<Compound UW-ID>] “(“ {<U W1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Uma quantidade é definida como a relação entre:
UW1 - uma coisa, e
UW2 - quantidade,
onde:
UW2 é o número ou a quantidade de UW1.

Exemplos e Leituras

`qua(cup(icl>tabelware), 2))`	Two cups of coffee
`mod(coffee(icl>beverage), cup(icl>tableware))`	
`qua(kilogram(icl>unit), many(aoj>thing))`	many kilograms
`qua(dog(icl>animal), 2)`	two dogs

Relações Relacionadas

Uma quantidade é diferente de **per** em que uma quantidade é um número ou uma quantia absoluta, visto que per é um número ou uma quantidade relativa a alguma unidade da referência (tempo, distância, etc.). Uma quantidade é usada também expressar a iteração, ou o número de ocorrências de um evento ou um estado.

rsn (reason) razão

Rsn define uma razão porque um evento ou um estado acontece.

```
rsn (ocorrência, coisa)
rsn (fazer, coisa)
rsn (ocorrência, ocorrência)
rsn (ocorrência, fazer)
rsn (fazer, ocorrência)
rsn (fazer, fazer)
rsn (ocorrência, (aoj>coisa))
rsn (fazer, (aoj>coisa))
rsn ((aoj>coisa), ocorrência)
rsn ((aoj>coisa), fazer)
rsn ((aoj>coisa), coisa)
rsn ((aoj>coisa), (aoj>coisa))
```

Sintaxe

rsn [“:”<Compound UW-ID>] “(“ {<U W1>|“:”<Compound UW-ID>} “,” {<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Uma razão é definida como a relação entre:
UW1 - um evento ou um estado, e
UW2 - uma razão para um evento ou um estado,
onde:
UW2 é uma razão porque UW1 acontece.

Exemplos e Leituras

`rsn(go(icl>do), rain(icl>weather))`	... didn't go <u>because of the rain</u>
`agt:01(arrive(icl>do), Mary(icl>person))` `rsn(start(icl>do(obj>thing)), :01)`	They can <u>start because Mary arrived.</u>

`rsn(known(aoj>thing), beauty(icl>abstract thing))`
`mod(city(icl>region), known(aoj>thing))`
`mod(beauty(icl>abstract thing), city(icl>region))`
	a city <u>known for its beauty</u>

scn (scene) cena

Scn define um mundo virtual onde um evento ocorra, ou o estado é verdadeiro, ou uma coisa existe.

```
scn (fazer, coisa)
scn (ocorrência, coisa)
scn ((aoj>coisa), coisa)
scn (coisa, coisa)
```

Sintaxe

scn [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Uma cena é definida como a relação entre:
UW1 - um evento ou um estado ou coisa, e
UW2 - uma coisa abstrata ou metafórica compreendida como um lugar,
onde:
As UWs são diferentes, e
UW1 está ou acontece em um lugar caracterizado por UW2.

Exemplos e Leituras

`scn(win(icl>do(obj>thing)), contest(icl>event))` … win a prize in a contest
`scn(appear(icl>ocorrência), program(icl>thing))` … appear on a TV program
`scn(play(icl>do), movie(icl>entertainment))` … play in movie

Relações Relacionadas

Uma cena é diferente do **plc** em que o lugar da referência para plc está no mundo real, visto que para scn é um mundo abstrato ou metafórico.

seq (sequence) seqüência

Seq define um evento ou um estado prévio de um evento ou de um estado focalizado.

```
seq (ocorrência, ocorrência)
seq (ocorrência, fazer)
seq (fazer, ocorrência)
seq (fazer, fazer)
seq (ocorrência, (aoj>coisa))
seq (fazer, estado)
seq ((aoj>coisa), ocorrência)
seq ((aoj>coisa), fazer)
```

Sintaxe

seq [“:”<Compound UW-ID>] “(“ {<U W1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Uma "seqüência" é definida como a relação entre:
UW1 - um evento ou um estado focalizado,
UW2 - um evento ou um estado prévio,
onde:
As UWs são diferentes, e
UW1 ocorre ou é verdadeira após UW2.

Exemplos e Leituras

`seq(leap(icl>do), look(icl>do))`	Look before you leap.
`seq(red(aoj>thing), green(aoj>thing))`	It was green and then red.

Relações Relacionadas

Uma seqüência é diferente do **coo** em que seq descreve os eventos ou os estados que não ocorrem ao mesmo tempo, mas um após o outro, visto que coo descreve os eventos que ocorrem simultaneamente.

Uma seqüência é diferente de **bas** em que seq descreve eventos ou estados nos termos da ordem de tempo, enquanto bas descreve coisas ou estados nos termos de diferenças ou de similaridades qualitativas.

src (source: initial state) fonte: estado inicial

Src define o estado inicial de um objeto ou de uma coisa associado inicialmente com o objeto de um evento.

```
src (ocorrência, coisa)
src (fazer, coisa)
```

Sintaxe

src [“:”<Compound UW-ID>] “(“ {<U W1>|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

Um estado inicial é definido como a relação entre:

UW1 - um evento, e
UW2 - um estado ou uma coisa,
onde:
UW2 é o estado específico que descreve o objeto de UW1 no começo de UW1, ou
UW2 é uma coisa que seja associada com o objeto de UW1 no começo de UW1.

Exemplos e Leituras

`src(change(icl>occor), red(aoj>thing))`	The lights changed from green to red.
`src(withdraw(icl>do(obj>thing)), stove(icl>furniture))`	I quickly withdrew my hand from the stove

Relações Relacionadas

Um estado inicial é diferente de **tmf** e de **plf** em que src descreve características e não o tempo ou o lugar de forma qualitativa.

Um estado inicial é diferente de **gol** em que gol descreve as características do objeto no estado final do evento.

tim (time) tempo

Tim define o tempo onde um evento ocorre ou um estado é verdadeiro.

`tim (ocorrência, tempo)`
`tim (fazer, tempo)`
`tim ((aoj>thing), tempo`)

Sintaxe

tim [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

O tempo é definido como a relação entre:
UW1 - um evento ou um estado, e
UW2 - um momento,
onde:
UW1, de um modo geral, ocorre indicado naquele tempo por UW2.

Exemplos e Leituras

`tim(leave(icl>do), Tuesday(icl>time))`	… leave on Tuesday
`tim(do(obj>thing), o'clock(icl>time))`	… do … at … o'clock
`tim(start(icl>do), come(icl>do))`	Let's start when … come

Relações Relacionadas

O tempo é diferente de **tmf** e de **tmt** em que o tempo caracteriza o evento de modo geral, enquanto tmf e tmt descrevem somente partes do evento.

O tempo é diferente de **coo** e de **seq** em que o tempo não descreve estados e eventos relativamente, com respeito um a outro, mas com respeito a determinados pontos no tempo.

tmf (initial time) tempo inicial

Tmf define o tempo inicial de um evento ou um estado torna-se verdadeiro.

```
tmf (ocorrência, tempo)
tmf (do, tempo)
tmf ((aoj>coisa), tempo)
```

Sintaxe

tmf [“:”<Compound UW-ID>] “(“ {<UW1 >|“:”<Compound UW-ID>} “,”
{<UW2>|”:”<Compound UW-ID>} “)”

Definição Detalhada

O tempo inicial é definido como a relação entre:
UW1 - um evento ou um estado, e
UW2 - um momento,
onde:
UW2 especifica o tempo em que UW1 começa, ou
UW2 especifica o tempo em que UW1 torna-se verdadeiro.

Exemplos e Leituras

`tmf(work(icl>do), morning(icl>time))` … work from morning to [till] night
`tmf(change(icl>ocorrência), live(icl>do))` changed … since I have lived here.

Relações Relacionadas

O tempo inicial é diferente de **tim** em que tmf expressa o tempo no começo do evento ou do estado enquanto tim expressa o momento para o evento de modo geral.

O tempo inicial é diferente de **src** em que tmf expressa o tempo no começo do evento ou do estado visto que src expressa características do objeto no começo do evento.

O tempo inicial é diferente de **tmt** em que tmf expressa o tempo no começo do evento ou estado enquanto tmt expressa o tempo em sua extremidade.

tmt (final time) tempo final

Tmt define um momento final de um evento ou um estado torna-se falso.

```
tmt (ocorrência, tempo)
tmt (fazer, tempo)
tmt ((aoj>coisa), tempo)
```

Sintaxe

tmt [":"<Compound UW-ID>] "(" {<UW1 >|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

O tempo final é definido como a relação entre:
UW1 - um evento ou um estado, e
UW2 - um momento,
onde:
UW2 especifica o tempo em que UW1 termina, ou
UW2 especifica o tempo em que UW1 torna-se falso.

Exemplos e Leituras

```
tmt(work(icl>do), night(icl>time))          … work from moning to [till] night
tmt(full(aoj>thing), tomorrow(icl>time))    … be full till tomorrow
```

Relações Relacionadas

O tempo final é diferente de **tim** em que tmt expressa o tempo na extremidade do evento ou do estado, visto que tim expressa o momento para o evento de modo geral.

O tempo final é diferente de **gol** em que tmt expressa o tempo na extremidade do evento ou do estado, visto que gol expressa características do objeto no fim do evento.

O tempo final é diferente de **tmf** em que tmt expressa o tempo na extremidade do evento ou do estado, visto que tmt expressa o tempo no começo do evento.

to (destination) destinação

To define o destino de uma coisa.

```
to (coisa, coisa)
```

Sintaxe
to [":"<Compound UW-ID>] "(" {< UW1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Um destino é definido como a relação entre:
UW1 - uma coisa, e
UW2 - um destino da coisa,
onde:
UW2 descreve o destino tal como a posição final de UW1.

Exemplos e Leituras

`to(train(icl>vehicle), London(icl>city))`	a train for London
`to(letter(icl>message), you)`	a letter to you

via (intermediate place or state) lugar ou estado intermediário

Via define um lugar ou um estado intermediário de um evento.

`via (ocorrência(gol>coisa,src>coisa), coisa)`
`via (fazer(gol>coisa,src>coisa), coisa)`

Sintaxe

via [":"<Compound UW-ID>] "(" {<U W1>|":"<Compound UW-ID>} ","
{<UW2>|":"<Compound UW-ID>} ")"

Definição Detalhada

Um lugar ou um estado intermediário são definidos como a relação entre:
UW1 - um evento, e
UW2 - um lugar ou um estado,
onde:
UW2 é o lugar ou o estado específico que descreve o objeto de UW1 de vez em quando no meio de UW1,
UW2 é uma coisa que descreva um lugar ou indica que o objeto de UW1 passou perto ou dentro durante UW1.

Exemplos e Leituras

`via(go(icl>do), New York(icl>city))`	… go … via New York
`via(bike(icl>do), Alps(icl>place))`	… bike … through the Alps

Relações Relacionadas

Um lugar ou um estado intermediário é diferente de **src**, **plf** e de **tmf** em que estes todos referem-se ao princípio de um evento, enquanto que via descreve o meio de um evento.

Um lugar ou um estado é diferente de **gol**, **plt** e de **tmt** em que estes todos referem-se ao fim de um evento, enquanto que via descreve o meio de um evento.

Capitulo 3: Universal Words

Uma UW (Palavra Universal) representa conceitos simples ou compostos. Há duas classes de UWs:

- simples, unidades conceituais chamadas "UWs" (palavras universais), e
- compostas de relações binárias agrupadas são chamadas de "UW composta". Estes são indicados com os UW-IDs compostos, como descritos abaixo.

3.1 UWs

3.1.1 Sintaxe da UW

Uma UW é composta de uma sequëncia de caráteres (uma palavra no idioma inglês) seguida por uma lista de regr as. O sentido e a função de cada uma destas peças são descritas na seção seguinte, na interpretação. As seguintes especificações fornecem uma indicação mais formal da Sintaxe de UWs.

```
<UW>              ::= <Head Word> [<Constraint List>]
<Head Word>       ::= <character>…
<Constraint List> := "(" <Constraint> [ "," <Constraint>]… ")"
<Constraint>      ::= <Relation Label> { ">" | "<" } <UW> [<Constraint List>] |
                      <Relation Label> { ">" | "<" } <UW> [<Constraint List>]
                      [ { ">" | "<" } <UW> [<Constraint List>] ] …
<Relation Label>::= "agt" | and" | "aoj" | "obj" | "icl" | ...
<character>       ::= "A" | ... | "Z" | "a" | ... | "z" | 0 | 1 | 2 | ... | 9 | "_" | " " | "#" | "!" | "$"
                    | "%" | "=" | "^" | "~" | "|" | "@" | "+" | "-" | "<" | ">" | "?"
```

3.1.2 Interpretação

Head Word

A Head Word é uma palavra composta em inglês (palavra/frase/sentença) que seja nterpretada como uma rótulo para um conjunto de conceitos: o conjunto formado por todos os conceitos que pode corresponder àquele em inglês. Uma UW básica (com nenhuma

restrição ou lista de limites) denota este conjunto. Cada UW restrita denota um subconjunto deste conjunto que é definido por sua lista de limites. UWs extras denotam novos conjuntos de conceitos que não têm os rótulos no idioma inglês. Assim, a palavra principal serve para organizar conceitos e fazê-los mais fáceis de recordar qual é qual.

Embaraços ou Limitações

A lista de limites restringe a interpretação de uma UW a um subconjunto ou a um conceito específico incluído dentro da UW básica, assim o termo "UWs restritas ". Da UW básica "bebida", sem uma lista de limites, inclui os conceitos de "pôr líquidos na boca", os "líquidos que são postos na boca", "líquidos com álcool", "absorva" e outros. A UW restrita "drink(icl>do, obj>liquid)" denota o subconjunto destes conceitos que inclui "pôr líquidos na boca", que corresponde por sua vez aos verbos tais como “drink”, “gulp”, “chug” e “slurp” em inglês. As limitações das UWs restritas, suas listas de limite, são limites. Os limites que usam os rótulos de relação definidos acima podem ser vistos como uma notação abreviada para relações binárias cheias: o drink(icl>do,obj>liquid) é o mesmo que o obj(drink(icl>do), o liquid) que significa algo como "casos de beber onde 'o obj ' é um líquido". Cada limite na lista de limites deve usar os rótulos da relação alistadas no Apêndice 2 e cada uma delas deve ser classificado na ordem alfabética. O rótulo da relação "icl" pode ser omitida quando se repete para restringir o conceito superior. Por exemplo, uma UW como "xxx(icl>change(icl>occur))" pode simplesmente ser definida como "xxx(icl>change>occur)".

3.1.3 Tipos de UW

UWs, consequentemente, são seqüencias de caracteres (palavras ou expressões) que podem ter determinadas especificações, atributos e instância-IDs. Sua função no sistema UNL é representar conceitos simples. Os três tipos de UWs, em ordem de importância prática, são:

- UWs básicas, são as palavras principais sem nenhuma lista de limite, por exemplo:
```
go
take
house
state
```

- UWs restritas, são as palavras principais com uma lista de limite, por exemplo:
```
state(icl>express)
state(icl>country)
state(icl>abstract thing)
state(icl>government)
```

- UW extra é um tipo especial de UW restrita, por exemplo:
```
ikebana(icl>flower arrangement)
samba(icl>dance)
soufflé(icl>food)
```

UWs Básicas

UWs básicas são seqüências de caracteres que correspondem a uma palavra inglesa. Uma UW básica denota todos os conceitos que podem corresponder àqueles em inglês. São usados para estruturar a base de conhecimento como um método de recuo para estabelecer correspondências entre palavras diferentes da língua quando suas correspondências mais específicas não podem ser encontradas.

UWs Restritas

UWs restritas são de longe de as mais importantes. Cada UW restrita representa um conceito mais específico, ou subconjunto dos conceitos. A lista de limites restringe a escala do conceito que uma UW básica representa. Da UW básica "bebida", sem nenhuma lista de limites, inclui os conceitos de "pôr líquidos na boca", os "líquidos que são postos a da boca", "líquidos com álcool", "absorva" e outros. A UW restrita "drink(icl>do(obj>liquid))" denota o subconjunto destes conceitos que inclui" pôr líquidos na boca ", que corresponde por sua vez aos verbos tais como “drink”, “gulp”, “chug” e “slurp” em inglês.

Considere outra vez os exemplos de UWs restritas citadas acima:

state(icl>express) é um conceito mais específico (associado arbitrariamente com a palavra inglesa “state”) isso denota uma ação em que os seres humanos expressam algo.

state(icl>country) é um sentido mais específico do "estado" que denota uma nação ou um país.

state(icl>abstract thing) é um sentido mais específico do "estado" que denota um tipo da circunstância em que as pessoas ou as coisas estejam. Esta UW é definida como um conceito mais geral e que possa ser consultada quando definindo outro sinônimo para a UW, tal como "situação" ou "circunstância".

state(icl>government) é um sentido mais específico de "estado" que denota um tipo de governo.

A informação nos parênteses é a lista de limites e descreve algumas limitações conceituais; por isso são chamadas UWs Restritas. Informalmente, as limitações que o meio "restringe sua atenção a este sentido articular da palavra". Assim, o foco é claramente a idéia e não a palavra inglesa específica. Gira frequentemente para fora daquele em uma língua dada lá é uma grande variedade de palavras diferentes para estes conceitos e não, coincidentemente, todas as palavras, como em inglês. Deve-se anotar que organizando estes sentidos em torno das palavras inglesas, a tarefa de fazer um dicionário novo da UW/língua específica está simplificada. Um dicionário bilíngüe de inglês/língua específica pode ser usado, e proseguindo de lá, o número dos conceitos diferentes necessários para cada palavra inglesa pode ser especificado.

Isto, naturalmente, não significa que as palavras inglesas estão traduzidas; o dicionário inglês é usado simplesmente como um lembrete dos conceitos que serão tratados de modo que o trabalho possa ser organizado mais eficientemente.

UWs Extras

UWs extras denotam os conceitos que não são encontrados em inglês e não têm que conseqüentemente ser introduzidos como categorias extras. As palavras da lingua estrangeira serão usadas enquanto as palavras principais estiverem usando caracteres (alfabéticos) ingleses. Considere outra vez os exemplos dados acima:

ikebana(icl>flower arrangement) é "um tipo de arranjo de flores" para o sentido de "algo que você faz com flores",

samba(icl>dance) é "um tipo da dança"

soufflé(icl>food) é "um tipo de alimento".

Até o ponto em que estes conceitos existem para o inglês, são expressos com palavras importadas de outra lingua e não aparecem sempre em dicionários ingleses. Assim simplesmente têm que ser adicionados para ser possível usar estes conceitos específicos no sistema UNL. A lista de limites ou as restrições dão a idéia de que tipo de conceito é associado com estas UWs extras e os limites fornecem as relações binárias entr e este conceito e outro, mais geral, conceitos já atuais (ação, dança, alimento, etc.).

3.2 UWs Compostas

UWs compostas são um conjunto das relações binárias que são agrupadas para expressar um conceito complexo. Um a sentença própria é considerada como uma UW composta. Isto torna possível tratar de situações como:

As mulheres que colocam chapéus grandes em casas de cinema deveriam ser convidadas a sair.

Sem UWs compostas, seria impossível construir as idéias complexas acima como as "mulheres que colocam chapéus grandes em casas de cinema" e os relacionam então a outros conceitos.

UWs compostas denotam os conceitos complexos que devem ser interpretados como conceitos da unidade, compreendidos ao todo de modo que se possa falar sobre todas suas peças ao mesmo tempo. Considere outra vez o exemplo dado acima.

[Mulheres que colocam chapéus grandes em casas de cinema] deveriam ser convidadas [a sair].

A parte da sentença dentro dos colchetes é o que deve ser perguntado. Somente quando são agrupadas em conjunto e consideradas como um todo a interpretação correta será obtida. Assim como unidades complexas podem ser relacionadas com outros conceitos relacionais, atributos como negação, negação, às atitudes do orador, entonação, etc, também podem ser associadas, que são interpretados geralmente para modificar o predicado principal dentro da UW composta.

3.2.1 A maneira de definir uma UW composta

Uma UW composta é definida colocando uma UW-ID composta imediatamente depois do rótulo da relação em todas as relações binárias que devem ser agrupadas. Assim, no xemplo abaixo, ":01" indica todos os elementos que devem ser agrupados para definir a UW composta número 01.

```
agt:01(wear(icl>do(obj>thing)), woman(icl>person).@pl)
obj:01(wear(icl>do(obj>thing)), hat(icl>clothes))
aoj:01(big(aoj>thing), hat(icl>clothes))
plc:01(wear(icl>do(obj>thing), theater(icl>facilities))
mod:01(theater(icl>facilities), movie(icl>entertainment))
agt:01(leave(icl>do).@entry, woman(icl>person).@pl)
```

Depois que este grupo foi definido, onde quer que a UW-ID composta esteja, por exemplo "01" no exemplo acima, pode ser usada para citar a UW composta. A maneira de citar uma UW composta é explicada na seção seguinte.

Uma UW composta é considerada como uma sentença ou uma subsentença, se na definição de uma UW composta um nó de entrada marcado por @entry for necessário.

3.2.2 A maneira de citar uma UW composta

Uma vez definida, uma UW composta pode citar ou se referir simplesmente e usar a UW-ID composta como uma UW. O método deve indicar a UW-ID composta depois de dois pontos ":". A referência a uma UW composta é chamada também de nó-escopo (Scope-Node). O Nó-escopo tem a seguinte sintaxe:

<Scope-Node> ::= ":" <Compound UW-ID> [<Attribute List>]
<Compound UW-ID> ::= two digits of a number 00 – 99
<Attribute List> ::= { "." <Attribute Label> } …
<Attribute Label> ::= "@entry" | "@may" | "@past" | ...

Para terminar, o exemplo acima poderia ser continuado com:

```
obj(ask(icl>do(obj>thing)).@should, :01)
gol(ask(icl>do(obj>thing)).@should, woman.@pl)
```

Outra vez, ":01 "é interpretado como um conjunto completo das relações binárias definidas acima. Significa que ":01" deve ser compreendidos para englobar estas relações binárias. UWs compostas podem ser citadas dentro de outras UWs compostas.

Capítulo 4: Atributos

Os atributos das UWs são usados para descrever a subjetividade das sentenças. Eles mostram o que é dito do ponto de vista do locutor: como o locutor vê o que é dito. Isto inclui tecnicamente os fenômeno chamados "ações por palavra", "atitudes", "honestidade", etc. As relações e as UWs são usadas para descrever a objetividade das sentenças. Os atributos das UWs enriquecem esta descrição com mais informação sobre como o locutor vê estes estados emocionais e suas reações diante destes. Tais atributos representam o papel de construir uma ponte sobre o mundo conceitual representado pelas UWs e pelas relações, com o mundo real. Ou seja, tais atributos trazem o conceito definido para as UWs e as relações no mundo real.

4.1 Locutor a respeito do Tempo

Onde o locutor situa sua descrição no tempo, capturando o momento da oração como um ponto de referência? No passo? Depois? Aproximadamente ao mesmo tempo? Esta é a informação que define "tempo narrativo" como passado, presente ou futuro. Estes atributos são ligados ao predicado principal. Embora em muitos idiomas esta informação é sinalizada através de sinais carregados em verbos, o conceito não está carregado. O exemplo mais claro é o presente simples em inglês que não é interpretado como o tempo presente mas como "independentemente de tempos específicos".

Considere o exemplo: "A Terra é redonda". Esta sentença é verdadeira no passado, presente e futuro, independentemente do tempo do locutor. Embora o tempo seja "presente" a sentença não é interpretada como o tempo atual.

@past	Acontecido no passado	ex) It <u>was</u> snowing yesterday
@present	Acontecido no presente	ex) It <u>is raining</u> hard.
@future	Acontecerá no futuro	ex) He <u>will arrive</u> tomorrow

4.2 O ponto de vista do Locutor

Um locutor pode enfatizar ou focalizar em parte um evento ou tratá-lo como um todo. Isto é relacionado acordo como o locutor emprega o evento no tempo. Estes atributos são unidos ao predicado principal.

O locutor pode focalizar no começo (@begin) do evento, olhando para a frente dele (@begin.@soon), ou para atrás dele (@begin.@just). Ele pode também focalizar no fim (@ end) ou na conclusão (@complete) do evento, olhando para a frente dele (@end.@soon ou @complete.@soon), ou para atrás dele (@end.@just ou @complete.@just). Ele pode focalizar no meio (@progress) ou na continuação (@continue) do evento.

O locutor pode escolher focalizar nos efeitos duráveis ou no estado final do evento (@state) ou no evento como uma unidade repetindo (@repeat), a experiência (@experience) ou o costume (@custom).

Ele pode também focalizar no que está incompleto ou no fato que não aconteceu ainda, usando o @yet.

@begin	Começo de um evento ou de um estado	Ex) It began to work again.
@complete	Conclusão/término do evento (inteiro).	Ex) I've looked through the script. look.@entry.@complete
@continue	Continuação de um evento	Ex) He went on talking. talk.@continue.@past
@custom	Ação habitual ou repetitiva	Ex) I used to visit [I would often go] there when I was a boy. visit.@custom.@past
@end	Fim/término de um evento ou de um estado	Ex) I have done it. do.@end.@present
@experience	Experiência	Ex) Have you ever visited Japan? visit.@experience.@interrogation
@progress	Um evento está em andamento	Ex) I am working now. work.@progress.@present
@repeat	Repetição de um evento	Ex) It is so windy that the tree branches are knocking against the roof. knock.@entry.@present.@repeat

@state	Estado final do objeto no qual uma ação foi realizada	Ex) It is broken. break.@state

Estes atributos são usados para modificar os atributos acima, para expressar uma variedade dos aspectos das línguas naturais.

@just	Expressa um evento ou um estado que apenas começou ou terminou	Ex) He has just come. come.@complete.@just
@soon	Expressa um evento ou um estado que deva começar ou terminar	Ex) The train is about to leave. leave.@begin.@soon
@yet	Expressa o sentimento de algo ainda não começado, terminado ou completado, ou expressa um evento ou um estado que ainda não começo, com o @not.	Ex) I have not yet done it. do.@complete.@not.@yet

4.3 A Visão da Referência do Locutor

Se uma expressão se refere a um único indivíduo, um grupo pequeno ou um conjunto inteiro, geralmente isto não está freqüentemente claro. A expressão "o leão" não está suficientemente explícita para sabermos se o locutor quer dizer "um leão" particular ou "todos os leões". Considere os exemplos seguintes:

O leão é um mamífero felino.

O leão está comendo um antílope.

No primeiro exemplo, parece razoável supor que o locutor entendeu "o leão" como "todos os leões" considerando que no segundo exemplo como "um leão" particular.

Os seguintes atributos são usados para tornar explícito o que parece ser na visão da referência do locutor.

@generic	Conceito genérico	Ex) The dog is a faithful animal.
@def	Já consultado	Ex) the book you lost
@indef	Classe não especificada	Ex) There is a book on the desk.
@not	Posição do complemento	Ex) Don't be late!
@ordinal	Número ordinal	Ex) the 2nd door.

Estes atributos são unidos geralmente a UWs que denotam coisas.

4.4 A Ênfase, Enfoque e Tópico do Locutor

O locutor pode escolher enfocar ou enfatizar partes de uma oração para mostrar quão importante ele pensa que elas são na situação descrita. Isto está relacionado frequentemente à estrutura da sentença.

@contrast	Contrastar UWs	Por exemplo, o "but" nos exemplos abaixo é usado para introduzir uma palavra ou frase que contrasta com o que foi dito antes. Ex) It wasn't the red one but the blue one. Ex) He's poor but happy.
@emphasis	Enfatizar UWs	Ex) I do like it.
@entry	Entrada ou UW principal em uma sentença ou escopo	Ex) He promised (entrada da sentença) that he would come (entrada do escopo)
@qfocus	UW enfocada em uma pergunta	Ex) Are you painting the bathroom blue? To this question, the answer will be "No, I'm painting the LIVING-ROOM blue"
@theme	Instanciar um objeto de uma classe diferente	Ex)
@title	Título	Ex)
@topic	Tópico	Ex) He(@topic) was killed by her. Ex) The girl(@topic) was given a doll. Ex) This doll(@topic) was given to the girl.

Um UW marcada com "@entry" é essencial para toda expressão UNL ou em uma UW Composta.

4.5 As Atitudes do Locutor

O locutor também pode expressar, diretamente ou indiretamente, suas atitudes ou emoções no que está sendo dito ou para quem está sendo dito. Isto inclui respeito e cortesia para o ouvinte ou pode mostrar surpresa no que está sendo dito.

@affirmative	Afirmação	Ex)
@confirmation	Confirmação	Ex) You won't say that, will you? Ex) It's red, isn't it? Ex) Then you won't come, right?
@exclamation	Sentimento de exclamação	Ex) kirei na! ("How beautiful (it is)!" in Japanese) Ex) Oh, look out!
@imperative	Imperativo	Ex) Get up! Ex) You will please leave the room.
@interrogative	Interrogação	Ex) Who is it?
@invitation	Convidar a fazer algo	Ex) Will / Won't you have some tea? Ex) Let's go, shall we?
@polite	Sentimento de polidez. Põe ênfase em um modo de falar	Ex) Could you (please)... Ex) If you could … I would
@request	Pedido	Ex) Please don't forget.
@respect	Sentimento respeitoso. Em muitos casos, algumas palavras especiais são usadas.	Ex) o taku ("(sua) casa" em Japonês)
@vocative	Vocativo	Ex) Boys, be ambitious!

4.6 Os sentimentos, o julgamento e o ponto de vista do locutor

Estes atributos expressam os sentimentos do locutor ou como o locutor vê ou julga o que é dito. Este tipo de informação subjetiva é muito dependente do tipo de idioma. Deveria ser possível expressar cada tipo de informação subjetiva de todos os idiomas. Assim, o desenvolvimento dos atributos está aberto aos colaboradores de cada idioma e podem introduzir um atributo novo quando nenhum atributo atual expressar seu significado. O atributo novo também deve ser introduzido da mesma maneira. Os seguintes atributos são usados para esclarecer a informação do ponto de vista do locutor:

Habilidade		
@ability	Habilidade ou capacidade fazer algo	Ex) The child can't walk yet. Ex) He can speak English but he can't write it very well.
Admiração		
@admire	Sentimento de admiração em relação a algo	Ex)

Conclusão		
@conclusion	Conclusão lógica	Ex) He is her husband; she is his wife.
@consequence	Consequência lógica	Ex) He is angry, wherefore I left him alone.
Culpa		
@blame	Sentimento de culpa em relação a algo	Ex) A sailor, and afraid of the sea!
Consentimento		
@dissent	Adversidade	Ex) But that's not true.
@grant	Dar/pedir permissão acerca de algo.	Ex) Can I smoke in here? Ex) You may borrow my car if you like.
@grant-not	Negar permissão acerca de algo.	Ex) You {mustn't/are not allowed to/may not} borrow my car.
Expectativa		
@although	Algo segue de encontro [contrário a] ou além da xpectativa	Ex) Although he didn't speak, I felt a certain warmth in his manner.
@discontented	Sentimento de descontentamento do locutor sobre algo	Ex) (I'll tip you 10 pence.) But that's not enough!
@expectation	Expectativa de algo	Ex) Children ought to be able to read by the age of 7. Ex) If you leave now, you should get there by five o'clock.
@wish	Sentimento de desejo acerca de algo	Ex) If only I could remember his name! (~I do wish I could remember his name!) Ex) You might have just let me know.
Intenção		
@insistence	Forte determinação a fazer algo	Ex) He will do it, whatever you say.
@intention	Intenção sobre alguma coisa	Ex) He shall get this money. (Speaker's intention) Ex) We shall let you know our decision.
@will	Determinação a fazer alguma coisa	Ex) I'll write as soon as I can. Ex) We won't stay longer than two hours

Necessidade		
@need	Necessidade de fazer alguma coisa	Ex) You need to finish thit work today.
@obligation	Obrigação de fazer alguma coisa juridicamente	Ex) The vendor shall maintain the equipment in good repair
@obligation-not	Desobrigação de fazer alguma coisa juridicamente	Ex) Cars must not park in front of the entrance. Ex) No smoking
@should	Sentimento de que se deve fazer alguma coisa	Ex) You should do as he says. Ex) You ought to start at once.
Possibilidade		
@certain	Certeza de que algo é verdade ou acontece	Ex) If Peter had the money, he would have bought a car.
@inevitable	Inevitabilidade lógica de que algo é verdadeiro	Ex) There must be a mistake. Ex) They should be home by now.
@may	Possibilidade prática de algo acontecer	Ex) It may be true. Ex) It could be.
@possible	Possibilidade lógica de algo acontecer	Ex) Anybody can make mistakes. Ex) If Peter had the money, he would buy a car.
@probable	Probabilidade de algo acontecer	Ex) That would be his mother. Ex) He must be lying.
@rare	Possibilidade rara de algo acontecer	Ex) If such a thing should happen, what shall we do? Ex) If I should fail, I will [would] try again.
@unreal	Irrealidade de algo acontecer	Ex) If we had enough money, we could buy a car. Ex) If Peter had the money, he could buy a car.
Pesar		
@regret	Sentimento de pesar acerca de algo	Ex) It's a pity that he should miss such a golden pportunity
Surpresa		
@surprise	Sentimento de surpresa acerca de algo	Ex) (He has succeeded!) But that's great!

4.7 Conversão

As estruturas típicas da UNL podem ser expressas por atributos para evitar a complexidade de conversão e desconversão. Estes atributos não expressam a informação dos locutores.

@pl	Plural	Ex) These (this.@p l) are the wrong size.
@angle_bracket	< > é usado	
@double_parenthesis	(( )) é usado	
@double_quotation	“ ” é usado	
@parenthesis	( ) é usado	Ex) UNL (Universal Networking Language) cnt(UNL, Universal Networking Language.@parenthesis)
@single_quotation	‘ ’ é usado	
@square_bracket	[ ] é usado	

Capítulo 5: Formato da UNL

5.1 Documento UNL

A informação é fornecida em originais UNL. O original em UNL tem o seguinte formato.

```
<UNL document>          ::= "[D:" <dinf> "]" { "[P]" { "[S:" <number> "]" <sentence> "[/S]"
                            }... "[/P]" }... "[/D]"
<dinf>                  ::= <document name> "," <owner name> [ "," <document id> "," <date> ","
                            <mail address> ]
<document name>         ::= "dn=" <character string>
<owner name>            ::= "on=" <character string>
<document id>           ::= "did=" <character string> /* definido pelo sistema */
<date>                  ::= "dt=" <character string> /* definido pelo sistema */
<mail address>          ::= "mid=" <character string> /* definido pelo sistema */
<sentence>              ::= "{org:" <l-tag> [ "=" <code> ] "}" <source sentence> "{/org}" "{unl" [ ":"
                            <uinf> ] "}" <UNL expression> "{/unl}" "{" <l-tag> [ "=" <code> ] [ ":" <sinf> "]"
                            <generated sentence> "{/" <l-tag> "}"
                            /* informação necessária sobre uma sentença */
<l-tag>                 ::= "ab" | "cn" | "de" | "el" | "es" | "fr" | "id" | "hd" | "it" | "jp" | "lv" | "mg" | "pg" |
                            "ru" | "sh" | "th" /*  indicador da língua */
<code>                  ::= <character code name>
<character code name>   ::= <character string>
<source sentence>       ::= <character string>
<generated sentence>    ::= <character string>
<uinf>                  ::= <system name> "," <post editor name> "," <reliability> [ "," <date> ","
                            <mail address> ]
<sinf>                  ::= <system name> "," <post editor name> "," <reliability> [ "," <date> ","
                            <mail address> ]
<system name>           ::= "sn=" <character string>
<post editor name>      ::= "pn=" <character string>
<reliability>           ::= "rel=" <digit>
<number>                ::= <digit> /* número da sentença */
```

Os caracteres usados na definição acima são os seguintes:

[D:<dinf>]	indica o começo de um original e da informação necessária
[/D]	indica a extremidade de um original
[P:<number>]	indica o começo de um parágrafo
[/P]	indica o fim de um parágrafo
[S:<number>]	indica o começo de uma sentença e o número da sentença
[/S]	indica o fim de uma sentença
{org:<l-tag>=<code>}	indica o começo de uma sentença original, de uma língua e de um código de caracter,onde "=<code>" pode ser omitido.
{/org}	indica o fim de uma sentença original
{unl:<uinf>}	indica o começo das expressões UNL de uma sentença e de uma informação necessária, ":<uinf >" pode ser omitido.
{/unl}	indica o fim das expressões UNL de uma sentença

Veja a seguinte seção sobre <expressão UNL>.

5.2 Expressão UNL

Uma expressão UNL de uma sentença é identificada com os seguintes Tag: {unl} e {/unl}. Há duas formas para expressar expressões UNL, uma é a forma de tabela e a outra é a orma de lista. A forma de tabela de uma expressão UNL é mais legível do que a forma de lista, mas a forma de lista de uma expressão UNL é mais compacta do que a forma de tabela.

Todo o componente, tal como uma palavra, frase ou título e, certamente, uma sentença de uma língua natural pode ser representada com as expressões UNL. Uma expressão UNL consiste conseqüentemente em uma UW ou de uma (ajuste de) relação(s) binária . Em originais UNL, uma expressão UNL para uma sentença é incluída pelos Tag {unl} e {/unl} interno [S] e pelo [/S]. Se uma expressão UNL consistir em um a UW, esta UW deveria ser incluída para favorecer a ligação [W] e pelo [/W].

Assim, uma expressão UNL de uma sentença é a seguinte:

```
{unl}
<Binary Relation>
...
{/unl}

ou,

{unl}
[W]
<UW><Attribute List>
[/W]
{/unl}
```

Cada Tag e relação binária devem terminar com um código do retorno: “0x0a”.

5.2.1 A forma de tabela da expressão UNL

Sintaxe da relação binária

<Binary Relation>	::= <Relation Label> [":"<Compound UW-ID>] "("
	{{ <UW_1> [":" <$UW\text{-}ID_1$>]} | {":"<Compound $UW\text{-}ID_1$>}}[<Attribute List>]" ,"
	{{ <UW_2> [":" <$UW\text{-}ID_2$>]} | {":" Compound $UW\text{-}ID_2$> }}[<Attribute List>] ")"
<Relation Label>	::= uma relação, veja "Capítulo 2: Relações"
<UW>	::= uma UW, veja "Capítulo 3: Universal Words"
<Attribute List>	::= { "." <Attribute> } …
<Attribute>	::= um attributo, veja "Capítulo 4: Atributos"
<UW-ID>	::= dois caracteres de '0–9' e 'A–Z'
<Compound UW-ID>	::= número decimal de dois dígitos (00 – 99)
	00 é usado para representar a sentença principal, que pode ser omitida.

As UW-IDs compostas são seqüencias de dois dígitos usados para identificar cada exemplo especificado por UWs compostas. UWs compostas são grupos de relações binárias (também conhecida como – "Scope-Nodes") as quais podem ser consultadas como uma UW.

Por exemplo, a seguir é mostrado um exemplo de uma expressão UNL da sentença "Eu posso ouvir um cão latindo lá fora".

```
{unl}
aoj(hear(icl>perceive(agt>person,obj>thing)).@entry.@ability, I)
obj(hear(icl>perceive(agt>thing,obj>thing)).@entry.@ability, :01)
agt:01(bark(agt>dog).@entry, dog(icl>mammal))
plc:01(bark(agt>dog).@entry, outside(icl>place))
{/unl}
```

Na expressão UNL acima "**aoj**", "**agt**" e "**obj**" são os rótulos da relação "**I**", "**bark(agt>dog)**", "**dog(icl>mammal)**", "**hear(icl>perceive(agt>person,obj>thing)**)" são as UWs, e "**:01**", qual aparece três vezes no exemplo, mostra a UW-ID Composta. A UW-ID Composta aparece na posição de uma UW, o assim chamado "scope-node", e é usada para citar ou consultar uma UW composta definida previamente. As relações binárias indicadas pelas UWs-ID compostas definem os índices do escopo. Um scope-node começa sempre com ":" seguido pelos dois dígitos de uma UW-ID Composta.

As UW-IDs são omitidas da express ão UNL acima. Quando uma UW é única em uma expressão UNL, a UW-ID pode ser omitida.

A UW-ID é usada para indicar alguma informação referential: aquelas lá são duas ou mais diferentes ocorrências do mesmo conceito (não são co-referentes). Normalmente, se a mesma UW ocorrer mais de uma vez, está em todos os casos compreendidas para consultar à mesma entidade ou ocorrência. Por exemplo, se um homem cumprimentasse um outro homem, a mesma UW poderia ser usada duas vezes "man(icl>male person)" para distinguir um do outro com UW-IDs:

`man(icl>male person):01` para o primeiro
`man(icl>male person):02` para o outro, para deixar claro que o primeiro homem não cumprimentou a si próprio.

5.2.2 A forma de lista da expressão UNL

A forma de lista de uma expressão UNL consiste em um conjunto de UWs e de relações binárias codificadas de uma sentença.

```
{unl}
[W]
{<UW> | {":"<Compound UW-ID>}}":"<Node-ID> /* nó identificador */
…
[/W]
[R]
<Relação Binária Codificada>
…
[/R]
{/unl}
```

Os tags usados acima têm os seguintes sentido:

[W]	indica o começo do identificador do nó
[/W]	indica o fim do identificador do nó
[R]	indica o começo das relações binárias codificadas
[/R]	indica o fim das relações binárias codificadas

Cada Tag, relação binária codificada e identificador do nó devem terminar com um código do retorno: "0x0a".

Sintaxe de uma relação binária codificada

<Encoded Binary Relation> := <Node1-ID><Relation Label>[":"<Compound UW-ID>]<Node2-ID>
<Node-ID> := two characters of '0 – 9' and 'A – Z'

Por exemplo, a seguir é mostrado um exemplo de forma de lista de uma expressão UNL "I can hear a dog barking outside", ou seja, "Eu posso ouvir um cão latindo lá fora".

```
{unl}
[W]
I:01
hear(icl>perceive(agt>person,obj>thing)).@entry.@ability:02
dog(icl>mammal):03
bark(agt>dog).@entry:04
outside(icl>place):05
:01:06
[/W]
[R]
02aoj01
02obj06
04agt:0103
04plc:0105
[/R]
{/unl}
```

O Uso do Sistema UNL nos Processos de Localização e Internacionalização de Sofware

Diego Duarte de Aragão[1]

[1]Curso de Bacharelado em Ciências da Computação
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina – UFSC, Brasil, 88040-900
aragao@inf.ufsc.br

Resumo

O presente artigo tem como objetivo apresentar os fundamentos teóricos e práticos acerca da Internacionalização e Localização de Software, integrando ferramentas do sistema UNL (Universal Networking Language) a estes processos, essenciais à globalização do software desenvolvido em nosso país, visando um aumento de produtividade nos processos de desenvolvimento dos mesmos.

O artigo consiste de uma sólida revisão bibliográfica acerca da Internacionalização e Localização de Software, a Tradução por Máquinas e sobre o sistema UNL, dissecando o funcionamento das ferramentas de conversão e deconversão desse sistema. Por fim, desenvolveu-se um framework baseado no Ciclo de Vida de Desenvolvimento em Cascata, integrando aspectos cruciais da Internacionalização e Localização de Software a um processo formal de desenvolvimento, utilizando as ferramentas do Sistema UNL para este fim.

Palavras-chave: Internacionalização de Software, Localização de Software, Tradução por Máquinas, Sistema UNL, Linguagem UNL.

Abstract

The present article has the target to present the theorical and practical fundaments about the Software Internacionalization and Localization, integrating the tools of the UNL (Universal Networking Language) system to these processes, key points to the globalized software to be developed in our country, aiming a productivity increase in their software development processes.

The article consists in a solid bibliographic review about the Software Internacionalization and Localization, the Machine Translation and the UNL system, deploying the conversion and deconversion tools of this system. Thus, a framework based in the Waterfall Development Lifecycle was developed, integrating the key aspects of the Software Internacionalization and Localization to a formal development lifecycle, using the UNL system tools to accomplish this target.

Keywords: Software Internacionalization, Software Localization, Machine Translation, UNL System, UNL Language.

1. Introducão

As indústrias de base tecnológica brasileiras em geral, encontram-se, atualmente, participando de um processo de globalização da economia. Esta situação induz mudanças profundas nos mercados de atuação dessas empresas, pois indústrias e empresas, que normalmente atuavam sobre mercados regionalmente restritos, vêem-se, agora, confrontados com concorrentes do mundo inteiro.

Além disso, com a evolução cada vez mais rápida de uma economia globalizada e sem fronteiras, muitas empresas não possuem a consciência de que softwares que fazem sucesso no mercado nacional podem não ser aceitos ou não obterem o mesmo êxito em mercados internacionais. Isso acontece devido ao fato de que em cada novo país, existem diversos fatores que poderão afetar a receptividade do software pelos clientes locais e, conseqüentemente, a sua decisão em comprar o produto. O que é necessário ser feito, para aumentar a competitividade e aceitabilidade do software em um determinado país é submetê-lo aos processos de internacionalização e localização, para que este software tenha maiores chances de ser aceito no mercado internacional. Uma conseqüência evidente é que a partir do momento em que o software está totalmente adaptado ao país que se deseja atingir é maior a sua competitividade através do aumento de suas vendas.

No entanto, para continuarem competitivas nesse mercado global, empresas desenvolvedoras de software que pretendem atuar globalmente, precisam adaptar seus produtos com o objetivo de atender às expectativas, requisitos e necessidades dos clientes potenciais. Todos esses fatores, que estão cada vez mais diversificados, se forem tratados por um processo adequado de adaptação, permitem que seus softwares possam ser facilmente compreendidos por todos os clientes, seja qual for seu país, idioma ou cultura e como conseqüência imediata a maior aceitação e venda de seus produtos.

O projeto visa utilizar ferramentas do Sistema UNL no processo de internacionalização e localização de software, através do levantamento de potenciais benefícios advindos do uso dessa tecnologia e criação de de um framework para o desenvolvimento de software utilizando essas ferramentas.

2. A Internacionalização e Localização de Software

Há muita confusão a respeito de como as expressões globalização, internacionalização, localização e tradução devem ser utilizadas. Estes termos são usados indiscriminadamente pelos desenvolvedores de software, autores de documentação, usuários, mídia, departamentos de marketing e pelos vendedores dos softwares de modos distintos. O correto entendimento destes termos é um passo crucial no processo de desenvolvimento de software. Esses termos são definidos da seguinte maneira:

Globalização: O processo de conceituação da linha de produtos para mercados globais de modo que possam ser vendidos em qualquer lugar do mundo com pequenas revisões.Está relacionado com a estratégia global de marketing da empresa e associado a conceitos de marketing (marcas, estabelecimento de market share, etc). A globalização é particularmente importante em mercados como os das indústrias de roupas e comidas. Qualquer um sabe o que é e pode beber Coca-Cola ou usar calças jeans da Levis, por exemplo.

Internacionalização: O processo de criação de um produto que possa ser facilmente e eficazmente localizado. Essa internacionalização pode ser algo tão básico quanto a formatação do layout de um documento, por exemplo, até a habilitação do software para lidar com character sets de dois bits, mudança muito mais complexa.

Localização: O processo de personalizar um produto para consumidores em um mercado alvo de forma que quando eles o usem tenham a impressão da que o produto foi projetado por um nativo daquele próprio país.

Tradução: O processo de conversão das escritas de uma linguagem de origem para as escritas de uma linguagem de destino. A tradução é um componente crucial da localização.

Estes quatro conceitos ajustam-se tal como um diagrama em forma de alvo. A globalização envolve o conceito inteiro de tornar uma linha de produto global. A internacionalização é executada de forma que o produto possa ser então localizado. Finalmente, a tradução é o componente-base para o processo completo, pois representa a transformação da linguagem.

Portar um software para um mercado internacional é um processo de duas partes: primeiro o software deve ser internacionalizado e só então localizado. A internacionalização de software envolve a preparação do código fonte original para o processo de localização. A localização de software transforma o idioma de origem do software em um ou mais idiomas de destino, dando para o produto o aspecto de ter sido criado no país designado. A internacionalização é usualmente referida pela abreviação I18N – I seguido de 18 letras, então a letra N. A localização é abreviada freqüentemente como L10N).

3. A Tradução por Máquinas (MT)

A Tradução por Máquinas (MT) é uma ferramenta poderosa, largamente utilizada por governos, indústrias e pelos consumidores individualmente. A tecnologia de ponta evolui constantemente e usos inovadores da MT aparecem com uma freqüência cada vez maior. Nos últimos anos, o uso da MT vem se diversificando: por ano, já são traduzidas mais palavras utilizando MT em relação ao número de palavras traduzidas por tradutores humanos, e a demanda continua crescente.

Os sistemas de MT modernos começaram surgiram nos anos 50 a partir de uma pesquisa em parceria entre a Georgetown University e a IBM. Mesmo com a frustração inicial gerada pelo estudo do governo americano de 1966 que indicava a MT como sendo muito cara, ineficiente e lenta para a tradução automática, as pesquisas e o desenvolvimento da tecnologia continuaram em outros países como Japão, Rússia e na Europa. Com a globalização e os aumentos dos projetos internacionais nos anos 80, o uso da MT voltou a surgir como uma excelente alternativa no compartilhamento de conhecimento em línguas distintas. Nos dias de hoje, a MT é um componente crítico para as demandas lingüísticas do século XXI, permitindo aplicações e usos onde os tradutores humanos não poderiam atuar e maximizando sua performance em outros aspectos.

A MT é um método de tradução onde um texto é traduzido de uma língua para outra automaticamente, sem a intervenção humana. Apesar de outras tecnologias aparentarem o mesmo uso, na realidade as mesmas diferem substancialmente. Os

Dicionários Eletrônicos Bilíngües, por exemplo, apesar de serem capazes de traduzir palavras e expressões automaticamente, são incapazes de traduzir sentenças completas, além de não serem capazes de escolher a melhor alternativa de tradução entre as possíveis. Por outro lado, existem os produtos que utilizam a Tradução por Memória, que utilizam um número fixo de sentenças e parágrafos armazenados em memória e fazem a tradução baseada na similaridade entre os mesmos e o texto inserido. Tal paradigma exige que um tradutor humano preencha a memória com tais sentenças e parágrafos, ficando sujeito à tendência das pessoas em dizer a mesma coisa de maneiras diferentes, minimizando a possibilidade do casamento correto entre o texto inserido e a tradução desejada.

Os sistemas de MT constroem automaticamente uma tradução para qualquer sentença, sendo independente de um número fixo de prévias traduções armazenadas em memória. A MT não provê uma tradução palavra-a-palavra: essa técnica processa o contexto da sentença a fim de determinar os significados das palavras e das sentenças. Logo, é muito mais flexível que os produtos que utilizam a Tradução por Memória e muito mais profundo que os que utilizam Dicionários Eletrônicos. Entretanto, pelo fato do conhecimento da gramática e das palavras das línguas envolvidas ser muito menor em um sistema de MT do que o conhecimento do tradutor humano, o texto traduzido não atinge total correção semântica.

4. O Projeto UNL

O Projeto Universal Networking Language (Projeto UNL) é um projeto financiado pela Universidade das Nações Unidas (The United Nations University - UNU), mais particularmente pelo Instituto de Estudos Avançados (Institute of Advanced Studies - IAS/UNU), com sede em Tóquio. Seu principal objetivo é promover e facilitar a comunicação internacional por meio do uso de sistemas computacionais de Tradução por Máquinas (MT).

A idéia central do Projeto UNL está no desenvolvimento da Linguagem UNL, que deverá ser usada em um ambiente Internet a fim de facilitar o fluxo de informação entre indivíduos de qualquer parte do mundo. Desse modo, a UNL deve servir de meio de troca de informação para que as pessoas se

comuniquem em suas próprias línguas nativas.

O problema central deste projeto se coloca a partir da necessidade de superar a barreira lingüística, com a utilização de uma representação intermediária das mensagens que cada usuário de computador, uma vez conectado a uma rede de computadores, planeja enviar a destinatários cujas línguas nativas não necessariamente sejam comuns.

O Projeto UNL também procura estimular e promover o desenvolvimento de tecnologias de engenharia lingüística e da MT, além de criar um fórum para discussões internacionais e colaborações em grande escala.

5. A Internacionalização e a UNL no Ciclo de Vida de Desenvolvimento de Software

O Projeto Universal Networking Language (Projeto UNL) é um projeto financiado pela Universidade das Nações Unidas (The United Nations University - UNU), mais particularmente pelo Instituto de Estudos Avançados (Institute of Advanced Studies - IAS/UNU), com sede em Tóquio. Seu principal objetivo é promover e facilitar a comunicação internacional por meio do uso de sistemas computacionais de Tradução por Máquinas (MT).

A idéia central do Projeto UNL está no desenvolvimento da Linguagem UNL, que deverá ser usada em um ambiente Internet a fim de facilitar o fluxo de informação entre indivíduos de qualquer parte do mundo. Desse modo, a UNL deve servir de meio de troca de informação para que as pessoas se comuniquem em suas próprias línguas nativas.

O problema central deste projeto se coloca a partir da necessidade de superar a barreira lingüística, com a utilização de uma representação intermediária das mensagens que cada usuário de computador, uma vez conectado a uma rede de computadores, planeja enviar a destinatários cujas línguas nativas não necessariamente sejam comuns.

O Projeto UNL também procura estimular e promover o desenvolvimento de tecnologias de engenharia lingüística e da MT, além de criar um fórum para discussões internacionais e colaborações em grande escala.

O objetivo é que o framework de internacionalização aja como um guia

pelo qual uma equipe de desenvolvimento de software possa começar a projetar software de modo completamente internacionalizado. O framework foi desenvolvido visando uma audiência sem uma experiência prévia do funcionamento e das técnicas de internacionalização.

O framework foi construído sobre o modelo em cascata do ciclo de vida de desenvolvimento de software. Há duas razões por trás desta escolha. Primeiro, este modelo é um dos mais difundidos atualmente, sendo um dos padrões da indústria de desenvolvimento de software nacional. Dessa maneira, fica mais fácil a integração da internacionalização com este ciclo de vida, bem como o foco nos aspectos fundamentais da internacionalização de software e do sistema UNL é mantido. A segunda razão para a escolha do modelo em cascata é que este modelo cobre sistematicamente todas as fases do processo de desenvolvimento de software [18], de tal sorte que desenvolvedores que queiram utilizar outros modelos para o ciclo de vida de desenvolvimento ainda poderiam analisar do princípio ao fim este framework e adaptar o mesmo às necessidades de cada modelo em específico.

O Sistema UNL aparece no framework na medida em que os resource files e os documentos contendo os textos e as especificações do software mantêm-se desacoplados do mesmo. Utilizando ferramentas como o EnCo e o DeCo em fases específicas do processo de desenvolvimento, o desenvolvimento do software fica facilitado e desvincula-se o produto de qualquer linguagem específica, ficando o mesmo facilmente localizável.

Ciclo de Vida do Desenvolvimento de Software - Atividades de Internacionalização

1. Fase de Análise de Requisitos

1.1. determinação do locales iniciais e futuros.

1.2. execução da análise de requisitos para cada locale inicial e/ou futuro.

1.3. determinação se é necessário ou lógico internacionalizar este produto de software, baseado nos requisitos para cada locale.

1.4. criação das diretrizes de estilo, glossários e listas de terminologias.

2. Fase de Especificação

2.1. decisão quais locales serão desenvolvidos agora e quais locales somente serão habilitados para desenvolvimento posterior (se aplicável).

2.2. decisão sobre um character set.

2.3. determinação de quais sistemas operacionais o produto deve rodar.

2.4. determinação de quais softwares de terceiros o produto deve ser compatível, conforme o locale.

2.5. finalização da determinação do nível de internacionalização que será atingido no desenvolvimento deste produto de software.

2.6. separação do texto localizável do código do software, através dos resource files.

2.7. identificação de aspectos regionais a serem localizados, tais como formato de numeração, moedas e formatos de data e hora.

2.8. identificação de quais processos, módulos ou objetos serão afetados pela internacionalização.

3. Fase de Design

3.1. determinação de como os arquivos de locale serão dispostos e acessados.

3.2. determinação das partes do software que serão deixadas desinternacionalizadas, no caso do produto não ser completamente internacionalizado (para que a internacionalização futura ser feita).

3.3. definições acerca de aspectos técnicos da internacionalização relacionados à implementação do programa.

4. Fase de Integração e Implementação

4.1. escolha de uma linguagem que forneça o apoio adequado para o nível de internacionalização almejado para este projeto.

4.2. teste completo do produto para todos as versões localizadas, incluindo os scripts, character sets utilizados e as interações com sistemas operacionais locais e software de terceiros locais.

5. Fase de Manutenção

5.1. executar os passos 1 - 4 num menor contexto.

6. Considerações Finais

A Internacionalização e Localização de software são fatores estratégicos tanto para o governo brasileiro, que almeja o aumento das relações comerciais

internacionais quanto para as empresas brasileiras que lutam para vencer as barreiras comerciais internacionais. Os mercados a serem explorados são imensuráveis e as oportunidades advindas desse cenário são infinitas.

De encontro a isso vem o crescimento do sistema UNL, criado a fim de diminuir os entraves causados pelas diferenças lingüísticas e culturais. A linguagem UNL é um bem da humanidade e mantida por uma fundação ligada às Nações Unidas. Esta foi homologada por uma rede de cientistas de renome internacional. O Sistema UNL conta com diversos utilitários que permitem à linguagem UNL ter "vida". Alguns destes sistemas já estão operacionais e outros ainda são objetos de pesquisa.

De tal sorte, tornam-se evidentes os benefícios advindos da integração desse promissor sistema com as metodologias de desenvolvimento para a Internacionalização e Localização de sofware.

Referências

[1] SOFTEX**.** Disponível em: http://www.softex.br/ Acesso no dia: 14/10/2004

[2] WATKINS, J. **The Guide to Translation and Localization: Preparing Products for the Global Marketplace**. Portland, Oregon: Lingo Systems, 2001.

[3] ESSELINK, B. **A Practical Guide to Localization**. 2nd Edition. Amsterdam/Philadelphia: John Benjamins, 2000.

[4] LISA. **Best Practices Guide – Implementing Machine Translation.** Disponível em: http://www.lisa.org/products/bestPractice/ . Acesso no dia 15/08/2004

[5] HUTCHINS, J.: **Towards a New Vision for MT**. Disponível em: http://www.mt-archive.info/MTS-2001-Hutchins.pdf . Acesso em 16/08/2004

[6] LANGLAIS, P., et al. **Computer-Aided Translation Typing System**.Disponível em: http://www.mt-archive.info/ACL-2000-Langlais.pdf. Acesso no dia: 16/08/2004

[7] TURIAN, J. P., **Evaluation of Machine Translation and its Evaluation** Disponível em: http://www.mt-archive.info/MTS-2003-Turian.pdf. Acesso no dia: 16/08/2004.

[8] GOUGH, N. **Example-based Controlled Translation**. Disponível em:

http://www.mt-archive.info/EAMT-2004-Gough.pdf. Acesso no dia: 16/08/2004,

[9] ALLEN, J., et al. **Text-to-speech: the MITalk System,** Cambridge University Press, 1997.

[10] PROJETO UNDL BRASIL. Disponível em: http://www.undl.org.br/. Acesso no dia 22/09/2004.

[11] HUTCHINS, J., et al . **An Introduction to Machine Translation**. Academic Press, 1992.

[12] DILINGER, M. **Notas sobre Semântica**. Departamento de Lingüística, UFMG. Minas Gerais, 1993.

[13] UNL FOUNDATION. UNL: **Universal Networking Language - An Electronic Language for Communication, Understanding and Collaboration**. UNU/IAS/UNL Center.Tóquio, Japão. 1997.

[14] UCHIDA, H., et al. **The UNL, a Gift for a Millenium**. UNU/IAS/UNL Center. Tóquio, Japão. 1999.

[15] UNL FOUNDATION. **DeConverter Specification**. Version 3.0 (Tech. Rep. UNL - TR1997 - 010). UNU/IAS/UNL Center. Tóquio, Japão. 2003.

[16] MARTINS, R., et al. **As Regras Gramaticais para a Decodificação UNL-Português no Projeto UNL**. Relatório Técnico 67. Instituto de Ciências Matemáticas e da Computação. Universidade de São Paulo, São Carlos.

[17] LARMAN, C. **Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process**. São Paulo: Prentice-Hall, 2002.

[18] JACOBSON, I., et al. Object-**Oriented Software Engineering – A Use Case Driven Approach**. Harlow: Addison-Wesley, 1992.

[19] PRESSMAN, R. **Engenharia de Software**. São Paulo: McGraw Hill, 2003.

[20] SOMMERVILLE, N. **Engenharia de Software.** São Paulo: Prentice-Hall, 2001.