

Achilles Colombo Prudêncio

Computer Aided Translation

Florianópolis, Santa Catarina

Agosto de 2006

Achilles Colombo Prudêncio

Computer Aided Translation

Trabalho de Conclusão de Curso apresentado como exigência para a obtenção do título de Bacharel em Ciências da Computação à Universidade Federal de Santa Catarina.

Orientador:

José Eduardo De Lucca

Banca:

Djali Avelino Valois
Helion Cardozo Junior

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Florianópolis, Santa Catarina

Agosto de 2006

Trabalho de Conclusão de Curso sob o título *Computer Aided Translation* apresentado para a obtenção do título de Bacharel em Ciências da Computação pela Universidade Federal de Santa Catarina, defendido por Achilles Colombo Prudêncio e aprovado em 23 de agosto de 2006, em Florianópolis, Santa Catarina, pela banca examinadora:

José Eduardo De Lucca
Orientador
Universidade Federal de Santa Catarina

Djali Avelino Valois
LIS Tecnologia

Helion Cardozo Junior
Directa Automação

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Abstract

| | | |
|----------|---|-------|
| 1 | Introdução | p. 12 |
| 1.1 | Sobre as tecnologias de tradução | p. 12 |
| 1.2 | Sobre a colaboração através da Internet | p. 13 |
| 1.3 | Justificativa | p. 13 |
| 1.4 | Objetivo geral | p. 14 |
| 1.5 | Objetivos específicos | p. 15 |
| 1.6 | Metodologia | p. 15 |
| 2 | Estudo da teoria de Internacionalização e Localização | p. 18 |
| 2.1 | Um pouco da história da relação entre tradução e computador | p. 18 |
| 2.1.1 | Onde se situa Internacionalização e Localização em tudo isso? | p. 19 |
| 2.2 | Internacionalização | p. 20 |
| 2.2.1 | História | p. 20 |
| 2.2.2 | Inspiração | p. 22 |
| 2.2.3 | Abstração | p. 23 |
| 2.3 | Localização | p. 24 |

| | | |
|----------|--|-------|
| 2.3.1 | Contexto | p. 24 |
| 2.3.2 | Software | p. 25 |
| 2.3.3 | Linguagem controlada | p. 25 |
| 2.3.4 | Localidades específicas | p. 26 |
| 2.3.5 | <i>Outsourcing</i> | p. 27 |
| 3 | Estudo do Modelo Colaborativo de Desenvolvimento | p. 28 |
| 3.1 | Software Livre | p. 28 |
| 3.1.1 | Breve conceituação do modelo de software proprietário | p. 29 |
| 3.1.2 | O licenciamento do SL | p. 30 |
| 3.1.3 | O Movimento SL | p. 31 |
| 3.1.4 | Comunicando-se e colaborando através da rede | p. 32 |
| 3.2 | A relação entre I18n, L10n e Software livre | p. 34 |
| 4 | Estudo da tecnologia de I18n e L10n | p. 36 |
| 4.1 | Memórias de Tradução | p. 36 |
| 4.1.1 | Interface de tradução | p. 37 |
| 4.1.2 | Formatação interna das frases | p. 42 |
| 4.2 | Banco de dados de termos | p. 43 |
| 5 | Estudo da tecnologia web | p. 44 |
| 5.1 | História breve | p. 44 |
| 5.2 | Novas tecnologias | p. 45 |
| 5.2.1 | <i>Rich Internet Applications</i> | p. 45 |
| 5.2.2 | Web 2.0 | p. 47 |
| 5.2.2.1 | Web Services APIs | p. 48 |
| 5.2.2.2 | Novas técnicas de desenvolvimento de <i>Rich Internet Applications</i> | p. 49 |

| | | |
|----------|---|-------|
| 5.3 | Sistemas web para tradução | p. 50 |
| 5.3.1 | Rosetta | p. 50 |
| 5.3.1.1 | Análise da interface de Tradução do Rosetta | p. 51 |
| 5.3.2 | Pootle | p. 52 |
| 5.3.2.1 | Análise da interface de Tradução do Pootle | p. 52 |
| 6 | Proposta e Implementação do Trabalho | p. 54 |
| 6.1 | Python | p. 55 |
| 6.2 | TMX | p. 56 |
| 6.3 | TBX | p. 59 |
| 6.4 | SAX | p. 61 |
| 6.4.1 | A especialização do SAX para realizar o <i>parse</i> de arquivos TMX | p. 62 |
| 6.4.2 | A especialização do SAX para realizar o <i>parse</i> de arquivos TBX | p. 64 |
| 6.5 | A interface da estrutura de Memória de Tradução com o sistema Pootle | p. 66 |
| 6.5.1 | Sobre a Correspondência Parcial de Strings | p. 68 |
| 6.6 | A interface da estrutura do banco de dados de termos com o sistema Pootle | p. 69 |
| 6.7 | A implementação de uma forma de mostrar o contexto da frase ao tradutor | p. 70 |
| 6.8 | Implementação da comunicação assíncrona com o servidor | p. 72 |
| 6.9 | Nota sobre o código-fonte contido nos anexos | p. 74 |
| 7 | Teste da Implementação e Análise dos Resultados | p. 75 |
| 7.1 | Testes do <i>parser</i> TMX | p. 75 |
| 7.2 | Testes da interface de memória de tradução no Pootle | p. 76 |
| 7.2.1 | Sobre a interface de mostra de contexto | p. 76 |
| 7.3 | Testes do <i>parser</i> TBX | p. 77 |
| 7.4 | Testes da interface de banco de dados de termos no Pootle | p. 77 |
| 7.5 | Testes da implementação como um todo | p. 78 |

| | |
|--------------------------------------|--------|
| 8 Conclusões | p. 80 |
| 8.1 Trabalhos Futuros | p. 81 |
| Anexo A - Artigo | p. 82 |
| A.1 Abstract | p. 82 |
| A.2 Introdução | p. 82 |
| A.2.1 Objetivo do trabalho | p. 83 |
| A.3 Internacionalização | p. 84 |
| A.4 Localização | p. 84 |
| A.5 Tecnologias Web | p. 85 |
| A.6 O trabalho realizado | p. 86 |
| A.7 Conclusões | p. 87 |
| Anexo B - tmx1.py | p. 89 |
| Anexo C - tmx_test.py | p. 103 |
| Anexo D - tbx1.py | p. 104 |
| Anexo E - tbx_test.py | p. 111 |
| Anexo F - pootle.py | p. 112 |
| Anexo G - projects.py | p. 120 |
| Anexo H - translatepage.py | p. 140 |
| Anexo I - translatepage.html | p. 152 |
| Anexo J - tranlate.js | p. 156 |
| Anexo K - content.js | p. 161 |

| | |
|---------------------------------------|--------|
| Anexo L - translatepagetb.html | p. 164 |
| Anexo M - translatepagetm.html | p. 165 |
| Referências | p. 166 |

Lista de Figuras

| | | |
|----|--|-------|
| 1 | Interface gráfica do SDLX | p. 38 |
| 2 | Interface gráfica do OmegaT | p. 38 |
| 3 | Interface gráfica do Transolution | p. 39 |
| 4 | Interface gráfica do Trados, rodando sobre o Microsoft Word | p. 40 |
| 5 | Interface gráfica do OOXlate, rodando sobre o OpenOffice | p. 41 |
| 6 | Interface gráfica do Lingobit Localizer | p. 41 |
| 7 | Interface de tradução do Rosetta | p. 51 |
| 8 | Interface de tradução do Pootle | p. 53 |
| 9 | Autômato finito que descreve o funcionamento do <i>parser</i> TMX. | p. 63 |
| 10 | Autômato que define o <i>parser</i> TBX. | p. 65 |
| 11 | Memória de tradução do Pootle em uso. | p. 67 |
| 12 | Interface da busca de termos do Pootle. | p. 69 |
| 13 | Interface do Pootle, mostrando informações adicionais | p. 71 |

Lista de Tabelas

| | | |
|---|--|-------|
| 1 | Alguns projetos de Software Livre e suas traduções | p. 35 |
|---|--|-------|

Resumo

Este trabalho reúne conceitos sobre Internacionalização e Localização de Software e sua importância no processo de Desenvolvimento de Software, tanto do Software Comercial quanto do Software Livre, dando ênfase ao último citado. Apresenta também a importância do Software Livre para o desenvolvimento de software em geral, e as tecnologias que contribuíram para o seu crescimento, principalmente a Internet. Fala também das novas tecnologias que utilizam a internet e a World Wide Web como plataforma de desenvolvimento, e a informação como motivação.

Finalmente, procura utilizar todos estes conceitos no aprimoramento de uma ferramenta de localização baseada em tecnologias web livres, visando desenvolver um exemplo prático dos conceitos da tecnologia de localização estudados, principalmente os de Memória de Tradução.

Palavras-chave: Internacionalização, Localização, Software Livre, World Wide Web, Memória de Tradução.

Abstract

This work approaches concepts about Software Internationalization and Localization and their importance in the process of Software Development, both Comercial Software and Free Software, emphasizing the last one. It presents also the importance of Free Software to the development of software in general, and the tecnologies that had contributed to the growth of Free Software, mainly the Internet. It also discourses the new tecnologies that use the internet and the World Wide Web as a development plataform, and information as a motivation.

Finally, it uses all those concepts in the improvement of a web-based free software localization tool, aiming the development of a practical example of localization tecnology concepts, mainly the Translation Memory concepts.

Keywords: Internationalization, Localization, Free Software, World Wide Web, Translation Memory.

1 *Introdução*

1.1 Sobre as tecnologias de tradução

Durante muito tempo, traduzir um texto era uma tarefa puramente artesanal. Com a criação do computador, logo se viu neste um meio de automatizar o processo. Após muito tempo e uma história de avanços, sucessos e fracassos, hoje a tecnologia de tradução baseia-se principalmente em duas tecnologias (JUNIOR, 2004):

- Tradução por Máquina - também chamada de Tradução Automática, cujo propósito é a tradução de um texto de uma linguagem para outra, a princípio sem o auxílio humano.
- Tradução Assistida por Computador - tecnologia criada para auxiliar um tradutor humano em seu trabalho, facilitando e agilizando o processo.

A tecnologia de tradução assistida por computador (em inglês *Computer Aided/Assisted Translation*, abreviada CAT), também possui ramificações. Uma delas é a memória de tradução, parte do foco deste trabalho.

Uma memória de tradução consiste basicamente de um banco de dados de frases ou termos já traduzidos, que permanece na memória do computador para consulta durante todo o processo de tradução, e que geralmente é persistido em arquivo (este, na maioria das vezes em formato XML) (JUNIOR, 2004).

Usando memórias de tradução, um tradutor pode reaproveitar (total ou parcialmente) em novos trabalhos traduções já feitas anteriormente, o que aumenta a produtividade do profissional e a qualidade do trabalho final.

E através do intercâmbio dos arquivos usados pelas memórias de tradução (feito hoje principalmente através da internet), tradutores podem se auxiliar mutuamente. Segundo Craciunescu, Gerding-Salas e Stringer-O’Keeffe (2004), ”a Internet com seu acesso uni-

versal à informação e comunicação instantânea entre usuários criou uma liberdade física e geográfica para tradutores que era inconcebível no passado.¹”

1.2 Sobre a colaboração através da Internet

Antigamente existia muito mais colaboração na produção científica mundial, especialmente no meio acadêmico e na comunidade ligada à informática. A criação de trabalhos derivados de outros era comum e incentivada, pois representava a melhora contínua das idéias e a autoria original em geral era respeitada.

Hoje, a produção científica em geral deixou um pouco de lado esse sentimento colaborativo (talvez devido ao fato de existir maior envolvimento e investimento de empresas privadas nesse processo) e passou a restringir o acesso a esse conhecimento, numa tentativa dos produtores manterem para si a maior parte do lucro que ele gera.

Felizmente, a idéia de produção derivada e aprimoramento de trabalhos existentes não desapareceu. Com a globalização da informação, fortemente apoiada no crescimento e expansão da Internet, as pessoas encontraram um meio de criar e compartilhar informação de forma independente. Qualquer idéia de qualquer pessoa estava disponível mundialmente.

Isso tomou forma já nos anos 80, na área da informática, através do projeto GNU², que incentiva a criação de software e a distribuição deste sob licenças que permitam a criação de softwares derivados, protegendo a autoria original. E a internet é usada como principal meio de intercâmbio e disponibilização de informações.

1.3 Justificativa

Um porém no uso das memórias de tradução é que, quando do início de um projeto, torna-se muito difícil montar um banco destes trabalhando sozinho, a não ser que já se disponha de um, criado e povoado anteriormente.

Em um projeto em conjunto, onde a possibilidade de povoamento do banco de dados é maior, quanto maior a quantidade de pessoas trabalhando na tradução, outro problema surge: a dificuldade de compartilhamento de dados entre tradutores. É necessário realizar um levantamento terminológico antes do início do trabalho, para garantir a consistência

¹Tradução do autor.

²<http://www.gnu.org/>

dos termos usados por toda a equipe, de modo a diminuir os casos de termos traduzidos de forma desigual.

A princípio, esta troca de dados é realizada "manualmente", ou seja, durante um projeto, todo o pessoal aloca um tempo adicional para simplesmente trocar arquivos de memória de tradução, sem qualquer controle de diferenças entre arquivos, pelo menos nenhum controle automatizado. Faz-se necessário um ambiente em que os tradutores possam trocar dados de forma transparente e se possível em tempo real.

Dessa forma, em um ambiente de trabalho em conjunto, ocorrerá uma povoação mais rápida dos bancos de dados, que passarão a ser automaticamente compartilhados entre todos os membros do projeto. Assim, a tradução feita por uns beneficiará outros mais rapidamente.

Um sistema web é um exemplo de ambiente de trabalho em conjunto, onde todas as informações estão centralizadas em um servidor e disponíveis para um grupo através de uma rede local (ou para várias pessoas através da internet). Ele permite a atualização e a troca de informações em tempo real, o que atende as necessidades expostas anteriormente.

Com o crescimento da tecnologia web, novos conceitos foram introduzidos e a tecnologia evoluiu muito: os serviços online deixaram de ser páginas estáticas, e o conteúdo passou a depender da interação com o usuário. Os aplicativos da web se aproximam cada vez mais dos aplicativos desktop.

Desse modo, um tradutor pode ter um programa de tradução especializado através do browser com senão todas, quase todas as funcionalidades de um aplicativo desktop e atuar em um projeto em conjunto com outros tradutores, tudo através da internet, sem a necessidade de instalar novos programas.

A disponibilidade do sistema na internet pode inclusive permitir a participação de pessoas que não são necessariamente tradutores profissionais, voluntários que podem contribuir para a tradução com seu esforço.

1.4 Objetivo geral

Com base no exposto, o objetivo geral deste trabalho será a adaptação de um sistema web de tradução baseado em software livre já existente, disponibilizando na medida do possível todas as tecnologias existentes em aplicativos desktop de tradução, como memórias de tradução e glossários, utilizando-se das novas tecnologias web para oferecer

ao usuário uma experiência de tradução rica e produtiva.

Desse modo, todos os tradutores poderão se beneficiar do trabalho feito por outros membros do projeto, ao mesmo tempo que haverá muito mais coerência no trabalho final, já que frases semelhantes sempre serão traduzidas da mesma maneira e utilizando a mesma terminologia, o que é justamente um dos maiores benefícios das memórias de tradução e o que é mais difícil de administrar em um ambiente de trabalho em conjunto.

1.5 Objetivos específicos

Seguem os objetivos específicos deste trabalho:

- Adaptação de um sistema web de tradução já existente, adicionando funcionalidades básicas de um programa de memória de tradução, e utilizando as novas tecnologias de desenvolvimento web para proporcionar uma aplicação dinâmica.
- Utilização, na base do desenvolvimento do aplicativo, de ferramentas baseadas em software livre e código aberto, aproveitando-se assim de todo o conhecimento inerente à esta filosofia de desenvolvimento de software.
- Utilização neste aplicativo a ser desenvolvido de formatos de arquivos de memória de tradução também abertos e padronizados, facilitando o intercâmbio de bancos de dados com outros aplicativos diferentes deste.
- Documentação de todo o processo, reunindo todo o conhecimento teórico que foi agregado para compreensão dos conceitos inerentes ao programa original, que vão desde as linguagens de programação utilizadas até os conceitos por trás da necessidade de um programa como este.
- Apresentação do aplicativo modificado, após sua conclusão, à comunidade científica, em especial à comunidade de software livre, para compartilhar esse novo conhecimento e criar oportunidades de expansão para o mesmo.

1.6 Metodologia

Para atingir os objetivos apresentados anteriormente, foi definida uma metodologia, que servirá de guia não só para o desenvolvimento do aplicativo mas também para a estrutura deste trabalho.

Estudo dos conceitos de tecnologia de Internacionalização e Localização Os conceitos inerentes e correlatos à Internacionalização e Localização serão estudados, aprofundados, e posteriormente será apresentada uma síntese destes, com o objetivo de demonstrar o que foi aprendido e explicar como estes conceitos serão utilizados no aplicativo a ser desenvolvido.

Estudo do modelo de desenvolvimento colaborativo O modelo de desenvolvimento colaborativo, como pode ser chamado o desenvolvimento e aprimoração de programas e outros trabalhos através da contribuição de voluntários através da internet, também será estudado e aprofundado, tentando mostrar que um modelo adaptado pode ser aplicado à tradução de software, com os mesmos benefícios.

Levantamento do estado da arte Serão analisadas algumas ferramentas diferentes de memória de tradução existentes (baseadas ou não em software livre), com o intuito de realizar um levantamento do "Estado da Arte": os requisitos básicos do projeto, os padrões aplicados e presentes em todo aplicativo de memória de tradução, que indiscutivelmente deverão fazer parte do aplicativo a ser desenvolvido.

Serão analisados também diferentes aplicativos web (de tradução ou não), visando mostrar como a tecnologia da web evoluiu, e o impacto disso na entrega de serviços ao usuário final.

Proposição e implementação de integração entre ferramentas disponíveis As ferramentas baseadas em software livre serão estudadas com especial atenção e finalmente será proposto uma solução que agrega as funcionalidades das memórias de tradução a um sistema web, que também será adaptado aos novos paradigmas de desenvolvimento.

Para tal, a tecnologia destes programas será estudada tendo em vista a linguagem de programação utilizada, a integração com outras plataformas, a facilidade de uso e aprendizagem e o paradigma de programação a ser utilizado.

Testes e avaliação dos resultados obtidos Um processo iterativo de planejamento, desenvolvimento e testes será aplicado no aplicativo a ser desenvolvido, visando buscar a qualidade técnica (referindo-se aos métodos de programação utilizados) e não-técnica (referindo-se ao atendimento de requisitos) desta. Os resultados obtidos, serão devidamente documentados e apresentados com vista ao desempenho do aplicativo desenvolvido, não só no sentido de documentar quais os objetivos atingidos pelo trabalho, mas também quais requisitos foram atendidos pelo aplicativo.

Documentação Esta parte da metodologia visa apresentar não somente os resultados obtidos (como dito anteriormente), mas também todos os detalhes relevantes do aplicativo desenvolvido e do processo de desenvolvimento desse. Além disso, a documentação visa ser uma prova escrita dos conceitos assimilados na criação do aplicativo.

2 Estudo da teoria de Internacionalização e Localização

2.1 Um pouco da história da relação entre tradução e computador

Há algum tempo, a tradução era, ainda, um processo artesanal. Traduziam-se documentos à mão, e muitas vezes uma única pessoa fazia todo o trabalho, em um processo que levava meses para ficar pronto e que era altamente propenso a erros e falhas (ou desvios) de interpretação.

Com a invenção do computador percebeu-se o potencial que a máquina tinha para automatizar este processo. Com o fim da Segunda Guerra Mundial e o início da Guerra Fria, os estudos intensificaram-se, principalmente do lado dos norte-americanos e ingleses, no sentido de traduzir documentos e informações russas da maneira mais rápida e eficaz possível.

A princípio, os estudos baseavam-se na tradução das palavras das frases, para se ter uma idéia do seu conteúdo. Ainda assim, a tradução era um processo consideravelmente lento. Então, o foco passou a ser as regras sintáticas e gramaticais e logo em seguida, o contexto das palavras, tentando resolver os problemas de natureza semântica (ALFARO, 1998).

Logo se percebeu que os bancos de dados e a lógica aplicados eram insuficientes para armazenar tamanha quantidade de informação referente ao sentido das frases e ocorreu um certo descrédito na área de lingüística computacional. Os textos passaram a ter uma revisão, realizada por humanos.

Mais estudos trouxeram novas abordagens, baseadas muitas vezes em estatísticas aplicadas à língua em que estava escrito o texto e a língua para a qual o texto seria

traduzido, chamados de **língua fonte** (em inglês, *source language*) e **língua destino** (em inglês, *target language*¹), respectivamente. Outra técnica, utilizada até hoje, baseia-se na utilização de uma linguagem neutra, intermediária, para a tradução entre as línguas (ALFARO, 1998).

2.1.1 Onde se situa Internacionalização e Localização em tudo isso?

O processo de Internacionalização está vinculado à Engenharia de Software, no sentido de garantir a qualidade de um produto. Com a globalização da economia e o advento da Internet, hoje é muito mais fácil atingir mercados externos e um produto de boa qualidade deve necessariamente estar adaptado à língua e aos costumes da pessoa que o utilizará. O software pode até ser desenvolvido em outra língua, mas deve levar em conta que terá vários de seus elementos (principalmente de sua interface) traduzidos.

Pensando nisso, deve-se priorizar o uso de técnicas e bibliotecas de software que facilitem a separação do que é traduzível e adaptável dentro de um programa (elementos da interface como texto, data, hora, moeda, etc.) das variáveis (elementos de um programa que guardam informações importantes para o funcionamento deste, sejam elas temporárias ou não) e do **código-fonte**.

O código-fonte de um software é o resultado do trabalho do desenvolvedor: um bom software passa por um processo de planejamento antes do início do desenvolvimento. O resultado do planejamento e da implementação resultam no código-fonte: a "receita do bolo"; um texto escrito em linguagem de programação que é processado por um programa especial, um compilador, que traduz esse código-fonte no programa executável, o software a ser usado.

Todo esse texto traduzível deve ser separado em arquivos especiais, chamados muitas vezes de arquivos de recursos, que contém somente texto (ESSELINK, 2000). A tradução do arquivo binário final de um programa também é possível, mas este é um processo muito mais propenso a erros que podem ser evitados simplesmente pela adoção de uma técnica simples de programação.

Todo o processo de Internacionalização² visa apoiar e tornar mais simples e eficaz o

¹Na medida do possível, será evitado o uso de estrangeirismos neste trabalho. Estes serão aplicados apenas quando não houver termo com a mesma expressividade em português ou para associar o termo em português ao correspondente, quanto o termo estrangeiro é mais conhecido

²Em jargão técnico, "i18n", pois existem dezoito letras entre o "i" e o "n" da palavra em inglês, *Internationalization*

processo de Localização³. Uma vez que o software tenha sido internacionalizado, o processo de localização torna-se a simples tradução dos termos utilizados neste, que já estão devidamente separados em um arquivo diferente do binário do programa. A tecnologia de hoje permite que existam simplesmente o(s) arquivo(s) binário(s) do programa e seus arquivos de tradução. A língua em que o programa será apresentado pode ser escolhida em tempo de execução, sem necessidade de recompilação.

Para dar maior sentido às vantagens da aplicação da Internacionalização e posteriormente da Localização no processo de Engenharia de Software, é necessário definir o que é exatamente i18n e l10n de software e mostrar como ocorreu a evolução para o que possuímos hoje. Isso é tema das seções a seguir.

2.2 Internacionalização

2.2.1 História

Quando da criação do computador, as interfaces dos softwares desenvolvidos eram extremamente simples (embora grande parte fosse de softwares matemáticos bastante complexos para a época). As limitações de hardware na época restringiam bastante o que se podia fazer em termos de software.

Havia sim a "importação de software", mas isso geralmente acontecia entre universidades e outros centros de pesquisa (STALLMAN, 1998). A grande maioria dos softwares eram desenvolvidos em língua inglesa, o que não representava grande empecilho: ainda hoje o inglês é uma das línguas mais faladas no mundo, e "pré-requisito" para o intercâmbio de pesquisas internacionais. Outro fator atenuante era que, como o software era geralmente fruto de pesquisa, era distribuído junto com seu código-fonte.

Com acesso ao código-fonte, era perfeitamente realizável o processo de modificação do software. Se realmente necessário, seria realizada a tradução das mensagens de erro e interface simples para a língua local, mesmo que levasse algum tempo para aprender a lógica de funcionamento do software.

Nos anos 80, teve início a era dos computadores pessoais (ALBUQUERQUE, 2005), que então podiam fazer parte do dia-a-dia das pessoas. Mais e mais pessoas foram descobrindo a computação e mais empresas de software surgiram.

A competição entre as empresas de software fez com que estas sentissem a necessi-

³Analogamente ao termo para internacionalização, *localization* é abreviado "l10n"

dade de fecharem o código-fonte de seus programas, evitando assim que seus algoritmos fossem plagiados por empresas concorrentes. Como cada nova geração de computadores pessoais era compatível com as mais antigas, com o acréscimo de algumas funcionalidades, os programas eram compilados para arquiteturas (de computadores) mais antigas, com pouquíssima otimização específica, e assim podiam rodar em todas as arquiteturas posteriores.

Com a exportação do hardware, veio a exportação de software e as empresas de software começaram a atingir diferentes mercados. O hardware também precisou adaptar-se aos novos usuários que possuíam, por exemplo, alfabetos diferentes, e portanto precisavam de adaptações nos meios de entrada de caracteres (teclados, etc.), o software também precisou se adaptar às necessidades dos novos locais de utilização (ALBUQUERQUE, 2005).

Quando do início da exportação, ainda era possível manter versões localizadas do software exportado. Tomava-se o código-fonte original, fazia-se as modificações necessárias, compilava-se novamente o software e então tinha-se um novo programa, agora localizado, pronto para ser exportado.

Óbvio que o processo não era tão simples assim: era necessário identificar dentro do código-fonte do programa todo o texto que podia ser traduzido. Isso não poderia ser feito por um tradutor, a menos que este tivesse grande conhecimento de programação e um sério contrato de sigilo industrial com a empresa. Por outro lado, uma vez identificados os textos, a tradução não poderia ser feita por um programador, a menos que este fosse capacitado para tal atividade.

Com o surgimento da Internet, a complexidade da situação foi aumentando cada vez mais: a capacidade de exportação tornou-se muito maior e muitos países passaram a impor a localização de todo software importado como condição para a aquisição (ALBUQUERQUE, 2005).

O que antes era feito por equipes de tradutores e programadores tornou-se muito grande para ser administrado: os prazos de entrega encurtaram e cada vez mais se exigia maior qualidade tanto do programa quanto da localização deste. Um programa mal localizado pode até exercer sua função, mas tem menos valor que um programa inteiramente em língua estrangeira.

A complexidade da administração estava principalmente no controle de versões do software: para cada versão principal, havia tantas versões localizadas quantos eram os países de importação. Era necessário assegurar, na medida do possível, que todas as

versões possuísem as mesmas funcionalidades.

Isso era extremamente difícil, pois a localização podia introduzir erros no código-fonte. A mesma versão do software, localizada, podia ter erros que comprometessem a integridade da interface gráfica, fazendo o software visualmente "travar".

A exportação de produtos a países de língua não-latina trouxe outro problema: eles deviam oferecer maior suporte a representação gráfica de dados. Os caracteres que aparecem na tela, os números, e muitos outros aspectos deviam ser levados em conta na criação da interface.

Foram criados esquemas de codificação de caracteres⁴ que pudessem representar todos os caracteres latinos e não-latinos, como asiáticos, arábicos e muitos outros.

Sentiu-se então a necessidade ferrenha de permitir que o processo de localização corresse em paralelo ao desenvolvimento do software, mas sem que um interferisse no outro. Em princípio, pensou-se em resolver esta necessidade na mesma parte do desenvolvimento em que se fazia primeiramente: em tempo de compilação.

2.2.2 Inspiração

Tomando o exemplo dos aspectos funcionais do software, para os quais foram criadas funções que mudavam as regras de funcionamento do software de acordo com o novo local de distribuição, por exemplo, em um software contábil, as regras relativas à base de cálculo de funções monetárias deviam mudar de acordo com o novo destino de exportação, devido à mudança de moeda corrente e etc. (LOMMEL; FRY, 2005)). Estas novas funções, passaram a lidar com o texto da interface do programa.

Estes textos passariam a ter como referência dentro do código-fonte do programa identificadores especiais, passados como parâmetros para as novas funções citadas anteriormente. Os textos a serem traduzidos agora ficariam em arquivos separados do código-fonte (como citado anteriormente, estes podem ser chamados de arquivos de recursos) e cada texto levaria consigo, dentro desse arquivo, o identificador correspondente.

Funções especiais exerceriam uma tarefa simples (lembrando, em tempo de compilação): substituir os identificadores no código pelos textos correspondentes, retirados dos arquivos de recursos.

Isso trazia uma vantagem enorme, tanto para os desenvolvedores quanto para as equi-

⁴Forma computacional como são representados os caracteres e números na tela de um computador.

pes de localização: a tradução podia ser feita em paralelo ao processo de desenvolvimento; tendo os arquivos de recursos, o processo de localização era feito rapidamente e os arquivos de recursos traduzidos eram enviados de volta à equipe de desenvolvimento, que os usava para recompilar o programa e obter a versão localizada.

Assim, os novos membros de equipes de localização não precisavam mais ter conhecimentos avançados de linguagens de programação, apenas conhecimento básico e um bom banco de dados de terminologia de informática. E os desenvolvedores não precisavam mais se preocupar com erros que antes surgiram na localização, já que a equipe de localização não manipulava mais o código-fonte.

2.2.3 Abstração

”A internacionalização consiste basicamente na abstração da funcionalidade de um produto sem associá-la a um idioma, de forma que posteriormente seja possível incluir com facilidade o suporte a outro idioma, sem a preocupação de que recursos especificamente lingüísticos constituam um problema quando o produto for localizado”. (LOMMEL; FRY, 2005)

A afirmação anterior resume bem a essência do processo de Internacionalização de Software. A abstração (separação) das funcionalidades de um programa (como internamente o programa trabalha para exercer sua finalidade) de sua interface visível (como o programa se apresenta para que o usuário interaja com ele) é não só um bom padrão de projeto, mas o fundamento da internacionalização.

Como a função independe da representação (esta sim é específica de localidade), esta pode ser desenvolvida, testada e aprimorada com total segurança. Existem milhares de bibliotecas de funções de diferentes criadores que fazem o trabalho de internacionalização no programa.

Inclusive, com a popularização do processo de internacionalização, as novas linguagens de programação preocuparam-se em inserir bibliotecas de i18n em suas Interfaces de Programação de Aplicativos padrão⁵.

O crescente aumento do poder de processamento dos computadores proporciona inclusive que os novos programas carreguem o idioma da interface em tempo de execução: não é mais necessário recompilar o programa quando chega um novo arquivo de recurso.

⁵Em inglês, *Application Program Interface* ou API: consiste na biblioteca de funcionalidades que faz parte da linguagem de programação e são suportadas por padrão pelo compilador desta

Este arquivo pode ser simplesmente colocado em um diretório padrão, onde o programa busca todos os arquivos de recurso durante a inicialização, e o novo arquivo incluído passa a fazer parte da lista de idiomas suportados. O mesmo acontece com as funcionalidades que dizem respeito a outros aspectos da interface, como representação de caracteres, moeda, data, hora.

Todo este "esforço" de internacionalizar um programa visa então aumentar a produtividade tanto da equipe de desenvolvimento como da equipe de localização, que agora pode trabalhar em paralelo com desenvolvedores. Como será visto mais adiante, o processo de localização pode inclusive ser terceirizado, gerando um menor custo de tempo (e às vezes de dinheiro) por parte dos desenvolvedores.

2.3 Localização

2.3.1 Contexto

Muitas pessoas confundem o conceito de localização com o de tradução. Na realidade, localização envolve muito mais do que a simples tradução de um texto para uma nova língua.

Localização envolve a adaptação de todo o aspecto cultural (e, às vezes, também do significado), de um texto escrito, propaganda, livro, manual, ou em nosso caso, de um software, para as necessidades de um dado local ou região específica.

Tomando um exemplo mais marcante: a "localização" de uma propaganda. Quando uma propaganda é criada, ela carrega uma série de elementos da cultura para a qual se destina: seja o ambiente em que se passa, os valores morais que a sociedade preza e que irão aparecer em cena, até as características do produto em si.

Quando se pensa em passar esta propaganda em outros países, outras regiões, há uma série de cuidados a se tomar: exibir as cenas como foram feitas, sem nenhum tipo de adaptação, pode levar a uma "crise internacional": a dita propaganda pode estar valorizando um costume que, no país estrangeiro onde ela será exibida, pode significar algo completamente diferente ou na pior das hipóteses, até ofensivo. Ou, na melhor das hipóteses, mas igualmente ruim para o produto, a piada pode simplesmente deixar de ter graça.

O mesmo acontece com todos os outros exemplos citados, dentre os quais discutiremos mais o caso do software: a "tradução" de um software envolve muito mais do que a simples

tradução. Na realidade, tradução é apenas uma das etapas que fazem parte do processo de localização. Localização, em termos de software, envolve o gerenciamento de projeto, engenharia de software, engenharia de testes e muitos outros aspectos (ESSELINK, 2000).

Localização na área de software utiliza os conceitos e as atividades ligadas à tradução e os especializa, incluindo a terminologia e os aspectos de tecnologia da informação. Acrescenta também muitas facilidades, consequência da internacionalização utilizada no projeto de software. Foi a partir de conceitos e ferramentas usadas em localização de software que surgiram muitas ótimas ferramentas de tradução assistida.

2.3.2 Software

A partir desse ponto deve-se explicar a importância da internacionalização na localização de software: como visto anteriormente, a idéia de abstração das funcionalidades do programa, com a separação de sua interface visível proporcionou a definição concreta das tarefas do desenvolvedor e do localizador, quando o assunto é exportação de software e permitiu que as equipes trabalhassem em harmonia.

O desenvolvedor passou a trabalhar mais com **o que** o software deveria fazer e **como** deveria fazê-lo, além de assegurar que uma mudança de idioma se torne transparente ao localizador.

Ao localizador coube a tarefa de adequar a **linguagem** do software ao novo mercado consumidor, assegurando que os **termos** sejam utilizados corretamente, e que o mesmo termo seja sempre usado dentro do mesmo contexto. Mais adiante veremos que muitas dessas tarefas repetitivas foram amenizadas pelos programas de tradução assistida.

Os membros da equipe de localização não mais precisavam ter conhecimentos específicos da linguagem de programação na qual o programa era desenvolvido, mas sim do vocabulário técnico necessário, vocabulário este que também passou a ser controlado, com a introdução da internacionalização no desenvolvimento do software. A **Linguagem Controlada** é uma forma de assegurar a coerência da tradução dos termos e também de que esta seja a mais próxima possível do "equivalente" na língua destino.

2.3.3 Linguagem controlada

A Linguagem Controlada utiliza-se de um vocabulário mais restrito, um subconjunto do vocabulário da língua destino. Este usa somente termos dentro do contexto do pro-

grama, sem gírias, talvez no máximo alguns jargões da área.

Este mesmo tipo de recurso é utilizado para a criação dos manuais de software, bem como de manuais em geral. Neste caso, procura-se escrever claramente, com frases completas na voz ativa, evitando as gírias e siglas (ALBUQUERQUE, 2005), a menos que estas estejam explicadas em algum momento do texto. O vocabulário restrito assegura que as frases não sejam ambíguas, pois para cada contexto, uma variedade mínima de termos (idealmente um único) pode ser usada.

Escrevendo desta forma, pode-se ter certeza de que a localização do texto se dará de forma consistente, significando na língua destino exatamente o mesmo que significa na língua fonte, salvo adaptações culturais. Quando se inicia um projeto de localização, uma das atividades primeiras é a criação e/ou utilização de um vocabulário restrito na língua destino. As grandes empresas de software inclusive dispõem estes vocabulários publicamente, para que os desenvolvedores os utilizem ao criar programas para suas plataformas. Para cada localidade, é utilizado um vocabulário específico.

2.3.4 Localidades específicas

E como classificar estes vocabulários? A primeira resposta óbvia é: por país. Nada mais natural de dividir os vocabulários de acordo com a língua do país na qual são utilizados. E logo em seguida percebe-se a falta: e nos países em que dentro de seu território é falado mais de um idioma?

Pode-se então pensar em fazer a classificação por língua. E mais uma vez nos induzimos ao erro. Uma língua pode possuir vários dialetos, e se estes variarem muito dentro do próprio território nacional, imagine quando se trata de países diferentes que falam em princípio a mesma língua.

Por essa e outras razões convencionou-se designar localidades específicas⁶ através do par *língua-região*. Desse modo, temos o Português Brasileiro (simbolizado por *pt-br*) e o Português de Portugal (*pt-pt*), o Francês da França (*fr-fr*) e o Francês Canadense (*fr-ca*), e muitos outros.

Mas esta convenção usada em i18n e l10n não se restringe apenas à língua utilizada no software, em seu manual ou *website*. Abrange todos os aspectos culturais (relevantes ao contexto usado) do país onde esta língua é falada.

⁶os *locales*

Eis então que um dado locale representa não só a **língua** falada numa determinada **região**, mas também o formato da data e da hora utilizadas, o calendário, o sistema numérico e de pesos e medidas. Por exemplo, o *locale* **pt-br** representa não só que no Brasil falamos um dialeto do português, mas que utilizamos o sistema decimal para números, o sistema internacional para pesos e medidas, o calendário gregoriano para contagem do tempo e que contamos as horas do dia de 0 a 24.

2.3.5 *Outsourcing*

A prática de *outsourcing*⁷ é uma das mais utilizadas hoje em localização para o **controle de qualidade**⁸ do software.

Consiste em contratar serviços de localização no país para onde será exportado o software. Isso garante a qualidade, coerência e correção da terminologia usada nos textos escritos, das imagens e outros tipos de mídia usados no programa.

Mesmo com a distribuição pública de vocabulários por partes das grandes empresas desenvolvedoras de sistemas operacionais, é extremamente necessário que a localização seja feita por um nativo da língua destino, pois somente ele conhece o contexto real das palavras e faz a associação correta entre o termo local e o termo na língua fonte.

Este também pode apontar o contexto correto das imagens e diversas outras mídias usadas no programa, evitando atitudes incorretas por parte dos desenvolvedores, e garantindo um *feedback* correto nos testes.

Novamente, graças à separação entre funcionalidade e interface fornecida pela internacionalização, pode-se contar com este tipo de estratégia na localização de software, pois a equipe de l10n não terá acesso ao código-fonte do programa, assegurando os segredos industriais da empresa.

⁷Infelizmente, não existe nenhuma palavra de conhecimento do autor que possa servir de tradução a este termo.

⁸Termo conhecido na área de software pelo inglês *Quality Assurance*.

3 *Estudo do Modelo Colaborativo de Desenvolvimento*

Assim como a internacionalização e localização fazem parte da história do modelo de softwares proprietários criados por *softwarehouses*¹, quando foram introduzidos seus conceitos e técnicas para a produção de software que pudesse atingir mercados estrangeiros, a i18n e l10n também foram peça chave para o crescimento de um outro modelo de criação de software, descrito a seguir.

3.1 Software Livre

Software Livre, segundo a definição da Free Software Foundation², corresponde a "qualquer programa de computador que pode ser usado, copiado, estudado, modificado e redistribuído sem restrições" (WIKIPEDIA, 2006l). Para tal, o **código-fonte** do software (cuja definição foi apresentada anteriormente neste trabalho) deve estar disponível para que qualquer pessoa possa, como diz a definição, usá-lo, copiá-lo, modificá-lo e redistribuí-lo sem nenhuma restrição legal.

Isso se opõe diretamente ao modelo de software proprietário, que proíbe (ou restringe) o acesso do cliente ao código-fonte, alegando que o algoritmos usados e o modo como o seu software trabalha internamente é parte de sua propriedade intelectual, e deve ser alterado somente sob sua autorização e com sua revisão.

O Software Livre como modelo de criação de software já existia antes deste termo ser cunhado (FOUNDATION, 1999). Como já citado anteriormente, no meio acadêmico é comum a criação de trabalhos derivados de outros, e isso tem bastante força na área de informática.

¹Denominação dada às empresas que se dedicam a criar software, programas sob demanda - sem termo equivalente em português, mas podendo ser traduzido como "Empresas de Software".

²<http://www.fsf.org/>

Só que com a popularização da informática nos anos 80 e o surgimento das grandes empresas de software (fenômeno já descrito anteriormente), os programas proprietários que tinham seu código-fonte fechado passaram a ter mais destaque no mundo da informática.

3.1.1 Breve conceituação do modelo de software proprietário

Para entender as razões da existência do SL³, é importante entender também as razões da existência do modelo oposto, o software proprietário.

O modelo de negócio do software proprietário está baseado na venda de licenças sobre o uso do software (LUCCA et al., 2005). A licença deve ser adquirida a um determinado custo para que se possa utilizar uma cópia do software. Muitas vezes a licença só é válida para uma cópia ou um número finito de cópias do software, obrigando o cliente a adquirir uma nova licença se desejar utilizar mais cópias. Em softwares que permitam a instalação de extensões⁴, a licença pode restringir a instalação destas, proibindo o uso de extensões de terceiros (WIKIPEDIA, 2006p).

As empresas de software proprietário utilizam este tipo de licenciamento para se estabelecer na condição de único fornecedor de soluções para seus clientes, tornando difícil ou economicamente inviável a mudança de fornecedor, numa tática conhecida como *vendor lock-in* (WIKIPEDIA, 2006p). Todas as restrições criadas pelas empresas de software proprietário estão apoiadas no sistema de direito autoral (STALLMAN, 1994).

O sistema de direito autoral foi criado no tempo da imprensa escrita. A mídia impressa é relativamente difícil de copiar em larga escala, e o direito autoral protegia os autores dos grandes produtores de cópias (STALLMAN, 1994).

Isso não acontece da mesma maneira com a mídia digital. Uma vez em formato digital, qualquer informação pode ser copiada e compartilhada facilmente, e essa flexibilidade não se encaixa muito bem no sistema de direito autoral (STALLMAN, 1994).

O direito autoral basicamente defende o direito que um autor tem de controlar o uso que se faz de sua obra - seja ela um software, uma imagem, uma música (WIKIPEDIA, 2006a). A confusão ocorre quando o direito autoral é associado à ideia de propriedade, na tentativa de controlar o monopólio sobre a distribuição da obra (WIKIPEDIA, 2006a). Essa associação é usada por grandes empresas e autores para vetar a cópia sem controle das obras em sua propriedade, independente da finalidade para a qual a cópia se destina.

³Abreviação comumente usada para Software Livre.

⁴Extensões ou *plug-ins* de software acrescentam novas funcionalidades a um determinado aplicativo.

O problema de aplicar o direito autoral como propriedade à mídia digital é que é virtualmente impossível manter o controle, já que uma cópia pode ser feita e distribuída facilmente. Uma empresa tem todo direito de exigir indenização de alguém que lucre ilegalmente com suas obras, mas e quando a obra é utilizada sem fins lucrativos?

3.1.2 O licenciamento do SL

A legislação que rege o SL tenta basicamente estabelecer as liberdades descritas na definição citada anteriormente:

- O direito de uso: uma pessoa que adquirir um software deve ter o direito de utilizar este como bem entender, assumindo toda a responsabilidade sobre as conseqüências do uso incorreto;
- O direito de cópia: essa pessoa pode copiar o seu exemplar do software para quaisquer fins que achar necessário, seja para instalar em outras máquinas de sua propriedade, ou presentear um amigo, novamente assumindo a responsabilidade sobre seus atos;
- O direito de estudar e modificar o software: a curiosidade sobre o funcionamento das coisas é inata do ser humano. Se alguém possui a propriedade de uma cópia de um software, o que deveria impedir ela de estudar como este funciona internamente, e possivelmente corrigir erros que a impedem de utilizá-lo corretamente?
- O direito de redistribuir: e uma vez corrigidos os problemas, porque ela não poderia distribuir essas correções (ou o software corrigido), assim ajudando a outros que também sofrem com estes problemas? Ou talvez de utilizar as partes que lhe são úteis do software em outro software diferente?

Todo software que utiliza uma licença livre (existem várias; no entanto, a mais conhecida e difundida é a GNU *General Public License*), assegura ao seu usuário todas as liberdades acima. Assegura inclusive, que um software licenciado como livre seja sempre licenciado como livre, mesmo que sofra alterações no seu código, e mesmo que estas alterações mudem completamente o funcionamento do software.

É importante frisar que nenhuma destas licenças impede uma pessoa de comercializar o software. É por isso que é feita a distinção entre "software proprietário" e "software comercial" (FOUNDATION, 2000). É perfeitamente válido que um software livre seja vendido a um preço, desde que não obstrua nenhuma das liberdades descritas.

O que nos leva a razão da existência do Movimento Software Livre.

3.1.3 O Movimento SL

O movimento em favor do Software Livre surgido nos anos 80, assim como o próprio termo Software Livre, foram criados por Richard Stallman. No início dos anos 80, quase todo software era proprietário, e Stallman se viu frustrado por todos os vendedores de software estarem limitado-o a utilizar software, sem poder corrigir problemas dos softwares que ele utilizava, apesar de ter o conhecimento técnico para isso (STALLMAN, 1998).

Ele começou com a idéia de criar um Sistema Operacional livre. Um sistema operacional, segundo a definição de Wikipedia (2006k):

”é um conjunto de ferramentas necessárias para que um computador possa ser utilizado de forma adequada, pois consiste de intermediário entre o aplicativo e a camada física do hardware. Este conjunto é constituído por um kernel, ou núcleo, e um conjunto de softwares básicos que executam operações simples.”

Para completar esta definição, é suficiente dizer que um computador não pode ser utilizado se não houver um sistema operacional instalado. Ou seja, um sistema operacional é indispensável para o uso de um computador. A idéia que podemos ter, voltando aos questionamentos sobre software proprietário, é a seguinte: é justo colocar um preço em algo que é essencial para todos? A idéia de Stallman era de que a pessoa devia ser livre para pagar - ou não - pelo que é essencial.

Criando um novo sistema operacional e todas as ferramentas necessárias para utilizá-lo, e disponibilizando o sistema e seu código-fonte sob uma licença livre, qualquer pessoa interessada poderia ajudar no desenvolvimento de software, seja corrigindo erros no código, ou simplesmente apontando estes erros (STALLMAN, 1998).

Surgia o projeto GNU, cuja intenção era criar não só um sistema operacional livre baseado no Unix⁵, mas uma comunidade de desenvolvedores e usuários em volta deste, participando ativamente do projeto através de desenvolvimento, divulgação ou simplesmente utilizando o software, de modo a ”trazer de volta o espírito cooperativo que prevalecia na comunidade de informática nos seus primórdios”(FOUNDATION, 1999).

O projeto GNU cresceu primeiramente criando software livre que rodava no Unix.

⁵Sistema operacional proprietário largamente usado nos laboratórios de informática nos anos 80.

O editor de texto GNU Emacs⁶ tornou-se especialmente popular, e começou a atrair usuários. A comunidade começou a se formar. Outro programa bem conhecido do mundo GNU e usado mundialmente é a coleção de compiladores GCC (STALLMAN, 1998).

Quando Linus Torvalds aderiu ao projeto, uma grande gama das ferramentas de interação com o sistema já havia sido desenvolvida e distribuída. Mas sua colaboração, o núcleo (kernel) do sistema, que faz a interação com o hardware, foi essencial para a criação de um sistema operacional completo. A distribuição de ferramentas GNU rodando sobre o kernel Linux foi então denominada, obviamente, GNU/Linux (STALLMAN, 1998).

É importantíssimo frisar que muito provavelmente o movimento SL talvez não tivesse atingido a notoriedade e magnitude que tem hoje sem um componente fundamental: a Internet.

3.1.4 Comunicando-se e colaborando através da rede

A Internet interliga computadores do mundo inteiro trazendo informação, entretenimento, educação. Uma vez na Internet, um texto pode ser acessado em qualquer outra parte do globo. Esse acesso livre à informação encurtou a distância entre pessoas e proporcionou a criação de comunidades em torno dos mais variados assuntos.

E foi através da Internet que o projeto GNU cresceu. Milhares de desenvolvedores e curiosos ao redor do mundo viram naquele software que vinha com o código-fonte uma chance de aprimorar seus conhecimentos na área da informática e ao mesmo tempo contribuir para a melhora dos softwares que eles utilizavam.

E esse é o espírito do que pode ser chamado Desenvolvimento Colaborativo. Milhares de pessoas ao redor do mundo interessadas no uso, nesse caso do software, contribuindo para o crescimento e melhoramento deste, reunidos em forma de comunidades.

Mas como é possível que milhares de contribuições de partes diferentes do mundo se encaixem em um projeto, sem que este se torne um completo caos ou seja sabotado por um grupo de mal-intencionados?

No caso do kernel Linux, a atuação de seu criador Linus Torvalds teve papel fundamental no controle do projeto. Ele abriu o desenvolvimento para todos que tivessem interessados, mas filtrava as contribuições realmente importantes. Mantinha todos os membros informados dos progressos, e de quem havia sido responsável pela melhora.

⁶<http://www.gnu.org/software/emacs/emacs.html>

Incentivava a todos o teste do código executável, e a procurar falhas (LUCCA et al., 2005).

Até agora todos os exemplos usados para demonstrar o desenvolvimento colaborativo foram de projetos de software. O projeto GNU também criou licenças para documentação, que abrangem todos os textos que podem acompanhar um software (manuais e demais textos explicativos) e que podem ser usados para licenciar livros. O próprio Stallman já publicou vários textos sob a GNU *Free Documentation License*.

A GNU *Free Documentation License* permitiu várias iniciativas de criação de textos para a difusão da informação. A mais recente e conhecida atualmente é a Wikipédia⁷, uma enciclopédia livre. A Wikipédia funciona através de um Wiki: um Wiki é um sistema web de criação de páginas que permite que qualquer pessoa possa criar ou editar uma página, sem que seja necessário qualquer tipo de cadastro.

A Wikipédia reúne milhares de artigos sobre os mais variados tópicos, criados por pessoas interessadas sobre aquele determinado assunto. E como a veracidade dos artigos é garantida? De certa forma, não é, mas assim como acontece no software livre, a contribuição de pessoas comprometidas com seus assuntos de interesse e com vontade de participar mantém uma quantidade enorme de verbetes. Como realmente qualquer pessoa pode participar, é altamente encorajado para que falhas encontradas nos artigos sejam corrigidas (WIKIPEDIA, 2006r).

A própria Wikipédia incentivou outros projetos: o Wikiquote⁸, fonte de citações dos mais diversos autores; o Wikcionário⁹, um dicionário online; o Wikilivros¹⁰, dedicado a divulgação e criação de livros de conteúdo aberto; e vários outros - todos estes projetos podem ser encontrados na página principal da Wikipédia.

A Wikipédia foi usada como fonte para várias informações usadas neste trabalho.

Outra iniciativa recente que aumentou ainda mais a gama de conteúdo aberto disponível na rede foi o conjunto de licenças da projeto Creative Commons¹¹. Este projeto nasceu inspirado na Free Software Foundation e distribui um conjunto de licenças que se aplica a vários tipos de conteúdo, de software a livros, de imagens a música.

As licenças Creative Commons buscam, segundo sua própria definição, o "melhor dos dois mundos", referindo-se ao mundo proprietário e ao mundo livre. O que acontece é

⁷Site oficial - <http://en.wikipedia.org/> ; Site em português - <http://pt.wikipedia.org/>

⁸<http://pt.wikiquote.org/>

⁹<http://pt.wiktionary.org/>

¹⁰<http://pt.wikibooks.org/>

¹¹<http://creativecommons.org/>

que com uma licença Creative Commons, você pode restringir certos usos de uma obra, enquanto permite outros (COMMONS, 2006).

3.2 A relação entre I18n, L10n e Software livre

Dadas as liberdades que o uso do Software Livre propõe, faz sentido ele ter sido adotado no mundo inteiro. Assim como houve hackers¹² que queriam melhorar o software e adaptá-lo às suas necessidades, houve aqueles que simplesmente queriam utilizar o software em sua língua nativa.

Para se comunicar e trabalhar através da Internet, os desenvolvedores e outros usuários utilizam o inglês como língua comum. Embora muitos não vejam problema em utilizar software em inglês, outros gostam de utilizar software em sua língua nativa nos seus trabalhos diários (FOUNDATION, 1998).

Enquanto desenvolvedores podem traduzir os textos que aparecem para o usuário dentro do código-fonte do software, já vimos que isso não é uma prática recomendável. Mas provavelmente foi graças a essa prática que boa parte do software livre foi traduzido para outras línguas, nos primeiros anos.

Felizmente as boas práticas de software são seguidas também pelos desenvolvedores de Software Livre. Vendo que o software localizado tem maior chance de conquistar mais usuários, os desenvolvedores trataram de adicionar suporte à internacionalização aos seus softwares aplicativos e softwares básicos.

Embora a maioria das linguagens de programação hoje tenha suporte nativo à internacionalização, a linguagem de programação usada na maioria dos programas no início do movimento SL e do projeto GNU, a linguagem C, não possuía uma biblioteca de internacionalização, pelo menos não uma biblioteca que tivesse licença livre. O projeto GNU tratou de resolver isso, criando a biblioteca Gettext.

O Gettext foi um grande passo para a popularização dos programas GNU: adicionou o suporte à internacionalização, e agora as mensagens dos programas residiam em arquivos separados (os chamados "catálogos de mensagens", basicamente arquivos de texto com um sintaxe especial indicando o texto em inglês e logo abaixo o texto traduzido, distribuídos como arquivos de extensão .PO), e uma pessoa sem conhecimentos avançados de programação podia traduzir estas mensagens.

¹²Leia-se hacker como um entusiasta da área da informática, seja profissionalmente ou não.

A tradução dos arquivos foi facilitada ainda mais com a criação de ferramentas especiais que abstraem a sintaxe do arquivo, permitindo que a pessoa traduza o arquivo sem precisar obter muito conhecimento desta sintaxe.

Os usuários podiam agora contribuir de mais uma forma: traduzindo seus programas de preferência. E mais uma vez a força do desenvolvimento colaborativo se faz presente: enquanto softwares proprietários só traduzem seus programas para mercados de interesse, softwares livre possuem uma quantidade enorme de traduções disponíveis¹³:

Tabela 1: Alguns projetos de Software Livre e suas traduções

| Projeto | Quantidade de idiomas suportados |
|----------------|---|
| KDE | 70 |
| GNOME | 59 |
| OpenOffice.org | 94 |
| Mozilla | 115 |

¹³As fontes para estes números são seus respectivos sites oficiais, e a consulta foi feita em 01/07/2006

4 *Estudo da tecnologia de I18n e L10n*

Neste capítulo serão estudados alguns aplicativos de memória de tradução e as principais funcionalidades implementadas nestes serão ponderadas. O objetivo deste estudo é, como descrito anteriormente, levantar o estado da arte da tecnologia e criar um conjunto mínimo de funcionalidades que deverão ser implementadas no aplicativo final.

4.1 Memórias de Tradução

As memórias de tradução são aplicativos CAT criados para tradução de textos repetitivos, como manuais técnicos, documentação de software, arquivos de recursos. Nestes textos existe alta incidência de palavras e trechos de texto iguais, em diferentes partes do documento.

Também são úteis para traduzir textos com pequenas mudanças incrementais, quando se dispõe da tradução da versão anterior. MTs¹ não são consideradas muito eficientes na tradução de textos literários ou criativos, por que a taxa de repetição é obviamente menor (WIKIPEDIA, 2006n).

A tradução manual destes documentos sem qualquer auxílio é um trabalho tedioso e cansativo, pois a quantidade de termos repetidos pode ser muito grande e um meio de reaproveitar as traduções já feitas pode aumentar consideravelmente a produtividade do tradutor.

As MTs trabalham guardando todas as traduções feitas pelo tradutor, e quando eventualmente este chega a uma frase semelhante, a tradução é automaticamente sugerida. Essa razão de semelhança é calculada de acordo com funções especiais, descritas em detalhes mais adiante neste trabalho.

¹Memórias de Tradução

O benefício destes aplicativos é óbvio: quando o tradutor já dispõe de traduções prontas (sejam elas parciais ou completas), o reaproveitamento destas torna o trabalho mais ágil. O reuso também torna a tradução mais consistente, pois os termos são traduzidos da mesma forma, quando dentro do mesmo contexto. Como a base de dados de frases geralmente pode ser salva em arquivo, uma tradução anterior pode ser usada em novos textos, aumentando ainda mais a probabilidade de reuso.

As desvantagens também são evidentes: como já citado, em textos com pouca repetição o reuso pode ser de pouca ajuda. A má escolha de um arquivo de MT pode resultar em baixo índice de reaproveitamento, ou no pior dos casos, sugerir traduções errôneas para o contexto que a frase trata. É provável que estas traduções sejam reusadas mais de uma vez, de modo a propagar o erro (WIKIPEDIA, 2006n).

4.1.1 Interface de tradução

Os primeiros aplicativos de memória de tradução eram proprietários e chegaram ao mercado no final dos anos 80 (WIKIPEDIA, 2006n), e estes ditaram basicamente três padrões de interface de programa:

- A interface de tabela;
- A interface que trabalha dentro de um documento de texto;
- A interface que trabalha diretamente com textos dentro de programas binários.

Explicando mais em detalhes estas interfaces: a interface de tabela basicamente consiste de duas ou mais colunas com frases alinhadas, sendo que as linhas da primeira coluna contém frases do texto original, e as colunas seguintes devem ser preenchidas com as respectivas traduções. Entre exemplos de aplicativos que trabalham dessa maneira, estão o SDLX, da SDL International², o DéjàVu, da Atril³, o OmegaT⁴ e o Transolution⁵, estes dois últimos livres.

Nota-se na figura 1 a memória de tradução do aplicativo sugerindo uma tradução baseada numa frase anteriormente traduzida. As sugestões podem ser chamadas de correspondências⁶, e podem ser correspondências totais (quando a frase sendo traduzida é

²<http://www.sdl.com/>

³<http://www.atril.com/>

⁴<http://www.omegat.org/omegat/omegat.html>

⁵<http://transolution.python-hosting.com/>

⁶Em inglês, o termo usado seria *match*.

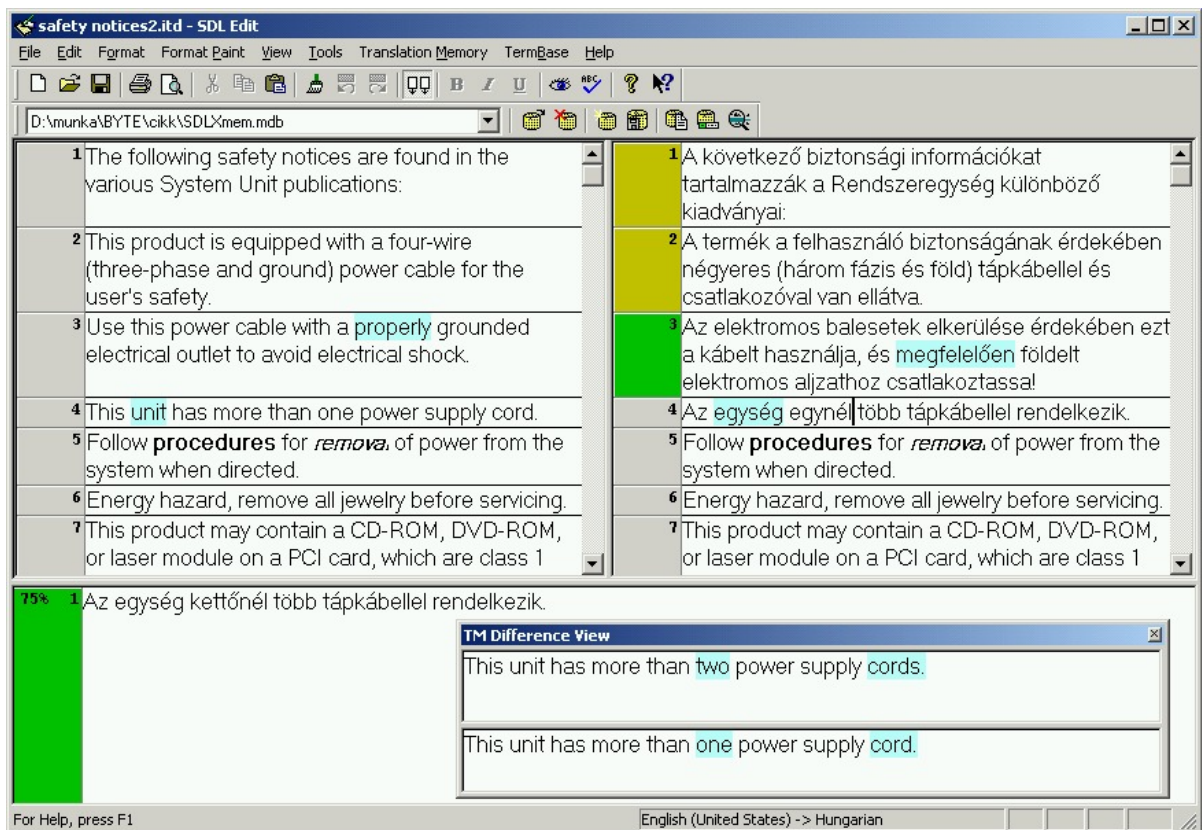


Figura 1: Interface gráfica do SDLX

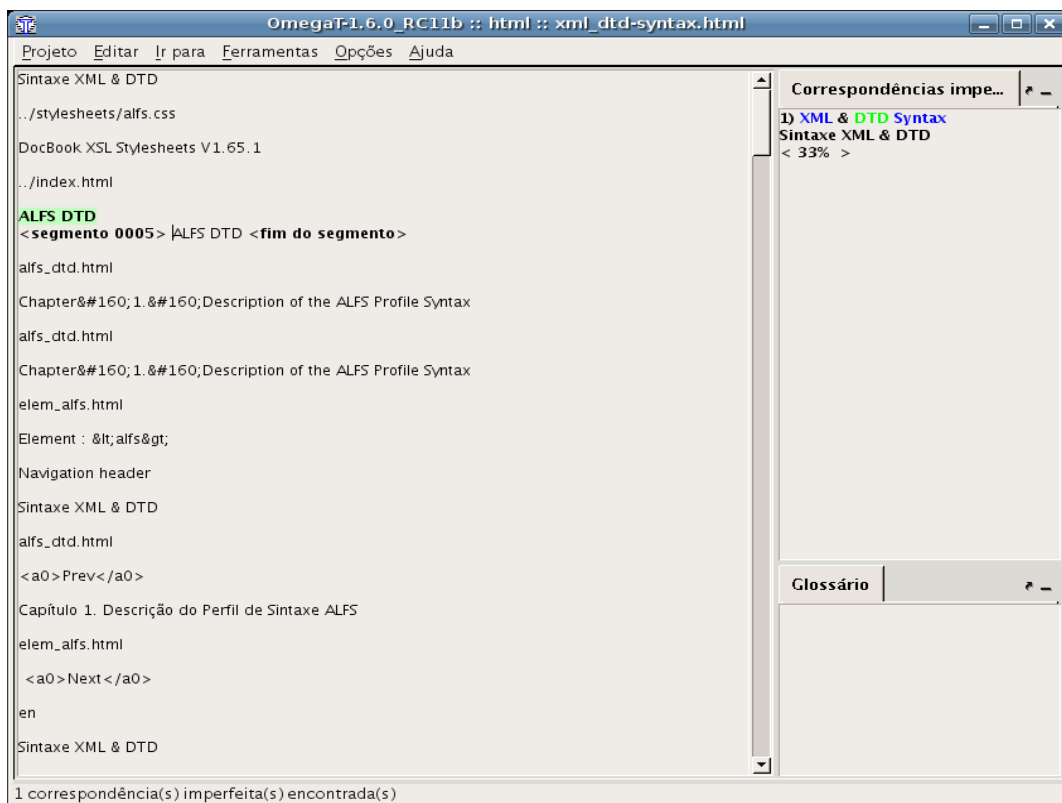


Figura 2: Interface gráfica do OmegaT

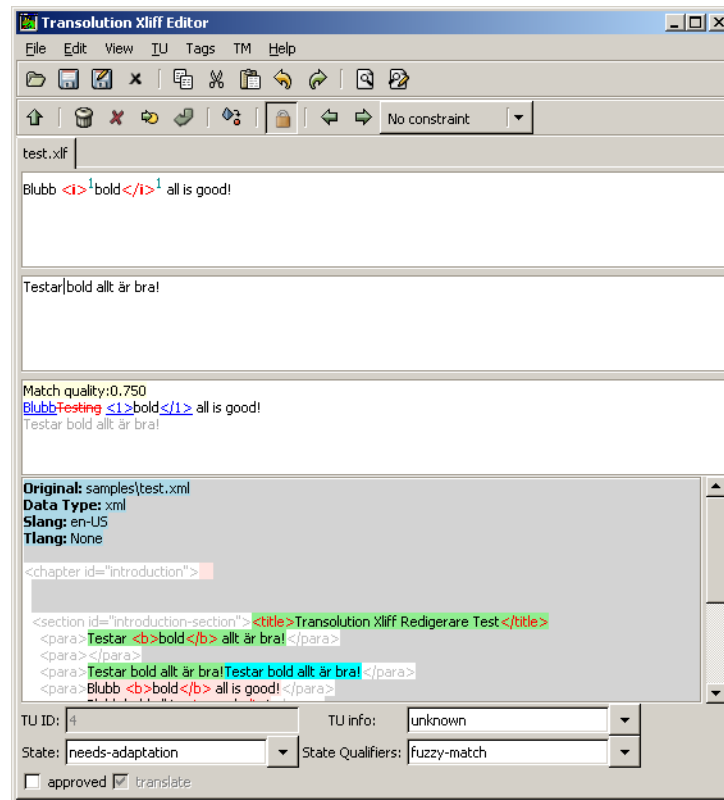


Figura 3: Interface gráfica do Transolution

exatamente igual a outra já traduzida) ou parciais (quando a frase sendo traduzida "se parece" com outra frase já traduzida anteriormente, numa proporção definida).

Na interface de documento de texto, a memória de tradução geralmente trabalha em conjunto com um processador de textos. Em um processador de textos, um documento é editado com auxílio visual, onde todas as alterações visuais feitas (espaçamentos, tamanho da fonte, negrito, itálico, etc.) refletirão no documento impresso⁷. O aplicativo de MT se associa ao processador de modo que cada frase ou parágrafo do texto é editado dentro do contexto visual original. A frase original também é mostrada, para que o tradutor mantenha a coerência visual. Exemplos desse tipo de interface são o Trados, recentemente comprado pela SDL, o Wordfast⁸ e o OOXlate⁹, livre.

Nota-se nas figuras 2 e 3, que a interface destes programas assemelha-se mais à interface de edição de documento de texto. Estes aplicativos foram encaixados na primeira categoria justamente porque não funcionam em conjunto com editores de texto, somente imitam este tipo de interface, dispondo as frases e suas traduções em seqüência, em uma

⁷Esse tipo de edição é comumente chamada de WYSIWYG (What You See Is What You Get - "O que você vê é o que você terá").

⁸<http://www.wordfast.net/>

⁹Infelizmente, o projeto OOXlate não existe mais.

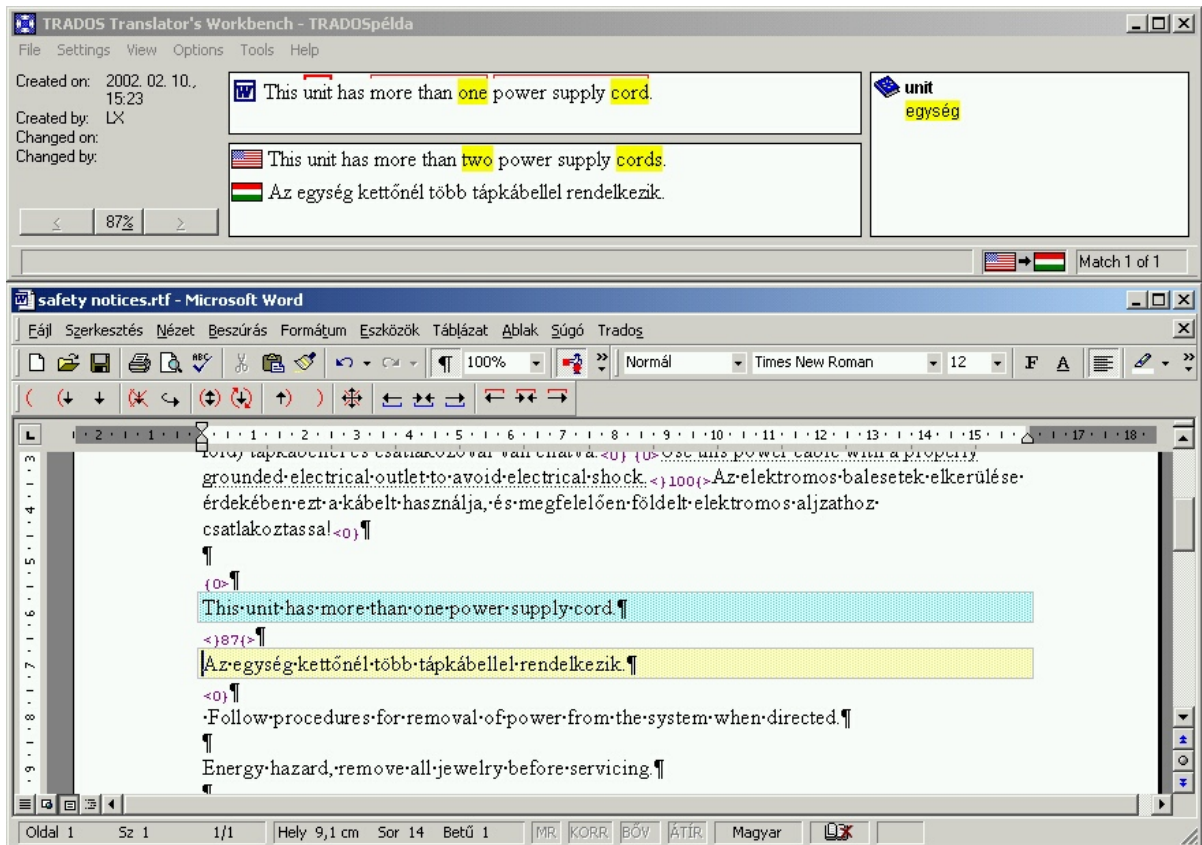


Figura 4: Interface gráfica do Trados, rodando sobre o Microsoft Word

tabela de uma coluna só.

Na interface de tradução de programas binários é o paralelo da tradução visual de documentos para a tradução de software. Geralmente, o aplicativo de tradução "roda" o programa original dentro de sua interface e o tradutor faz seu trabalho também traduzindo dentro do contexto visual original: menus, caixas de diálogo, janelas. Entre aplicativos que funcionam nesse modelo estão o Lingobit Localizer¹⁰, da Lingobit e o Passolo Software Localizer¹¹, da PASS. Infelizmente, o autor falhou em encontrar um equivalente em software livre para esta categoria.

A vantagem óbvia dos últimos dois modelos de interface é que o contexto da tradução é mantido. Dependendo do contexto, mesmo palavras iguais no idioma fonte podem ser traduzidas de modo diferente.

A vantagem do primeiro modelo é justamente a independência de contexto. Uma ferramenta desse modelo pode trabalhar com vários tipos de arquivos de entrada distintos, bastando que possa extrair as frases do documento original, e depois das frases traduzidas,

¹⁰<http://www.lingobit.com/>

¹¹<http://www.passolo.com/>

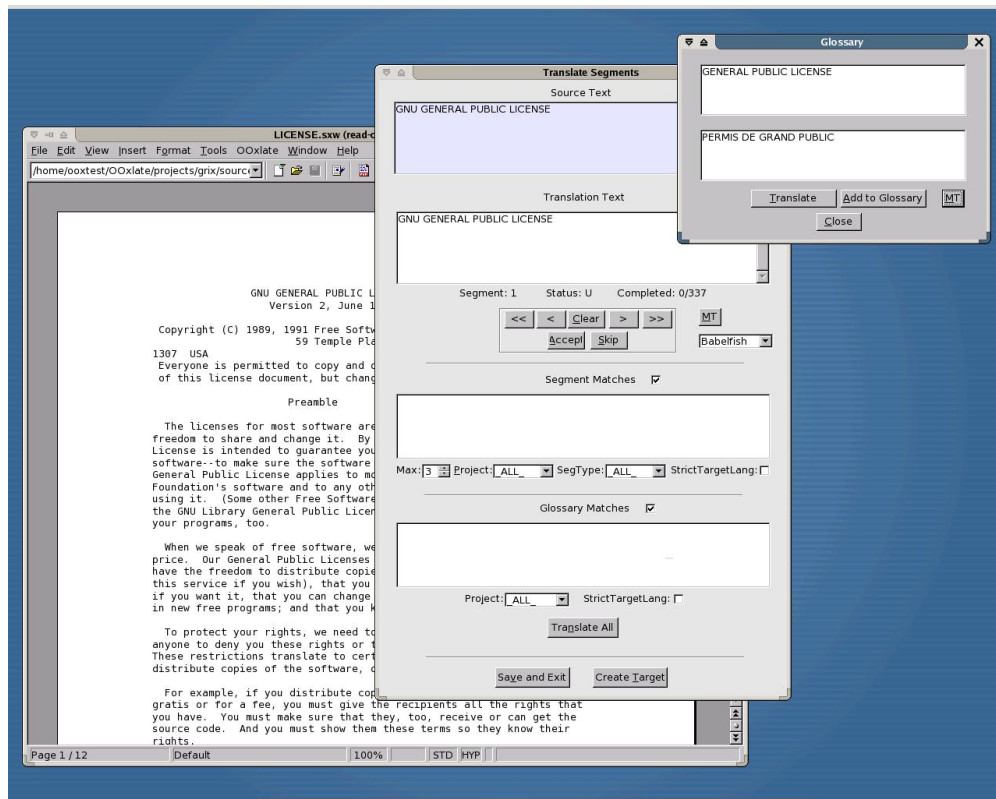


Figura 5: Interface gráfica do OOXlate, rodando sobre o OpenOffice

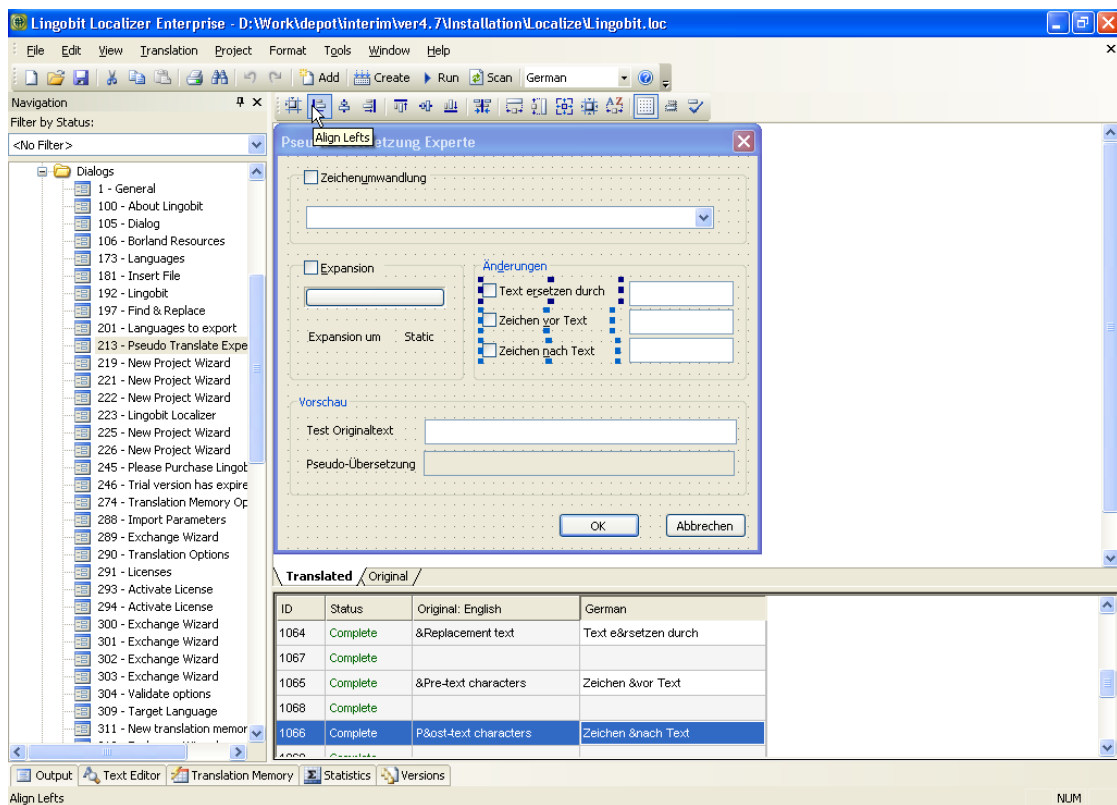


Figura 6: Interface gráfica do Lingobit Localizer

saber reconstruir o mesmo.

Comentando as desvantagens, o segundo modelo tem a desvantagem óbvia de que depende de programas de terceiros; o terceiro tem uma desvantagem técnica: a tradução de arquivos binários pode introduzir erros que podem terminar por impedir um programa de executar corretamente, como já dito anteriormente. A desvantagem do primeiro modelo reside justamente na falta de informação do contexto, bastante necessária para a localização, já que esta não reside na simples tradução das palavras.

Nesta análise rápida, a opinião do autor é de que uma abordagem híbrida pode ser trazer bons resultados: a interface de tabela é importante para que o tradutor trabalhe do mesmo modo independente do documento fonte. Mas como estamos falando de tradução de software, é necessário um meio de mostrar ao tradutor o contexto em que a frase é usada, para que este tome decisões corretas no processo de tradução.

4.1.2 Formatação interna das frases

Nota-se, na figura 1, a presença de cores em certas partes do texto mostrado (na tabela onde é feita a tradução). Algo parecido pode-se notar nas figuras 3, e 2, com a presença de rótulos (palavras entre "<" e ">") entre certos trechos do texto.

Isso acontece pois na interface desses arquivos a tradução não é feita baseada no contexto visual. Os rótulos (e cores) representam a formatação interna das frases dentro dos documentos, que varia desde negrito e itálico até *hyperlinks* em documentos html.

Esse tipo de informação é extremamente importante para o tradutor entender o contexto da tradução. O auxílio visual também pode ajudar quando o tradutor ainda não está familiarizado com o tipo de formatação que o documento fonte usa.

Isso é comum na área de desenvolvimento de software: as linguagens de programação podem embutir variáveis¹² nos strings¹³ que o programa usa. Diferentes linguagens tratam strings e variáveis de modos distintos, e quando estes strings vão para os arquivos de recursos, estas variáveis vão junto.

Os rótulos e as cores estão lá, então, para avisar ao tradutor que aquela informação é importante e deve ser tratada de maneira diferenciada, não podendo ser apagada ou, no caso de variáveis de programas, substituída.

¹²Uma variável, em programação, representa dados usados pelos programas que podem assumir valores diferentes durante a execução do programa.

¹³O string é um tipo de dado que nas linguagens de programação representa palavras e frases.

Sendo assim, é de opinião do autor que uma forma de abstrair a informação ao mostrá-la para o tradutor é extremamente importante para a tradução de software, exigindo deste menos conhecimentos específicos da linguagem de programas.

4.2 Banco de dados de termos

Apesar de se tratar de uma tipo de aplicativo a parte das memórias de tradução, é freqüente o uso de banco de dados de termos como glossários em conjunto com MTs. Um banco de dados de termos funciona de modo análogo a uma MT, sugerindo ao tradutor traduções do um determinado termo escolhido.

Citando como exemplo os aplicativos de MT já mencionados, o OmegaT vêm com um visualizador de termos embutido (como podemos ver na figura 2, em "Match/Glossary View"), assim como o OOXlate (figura 5). O SDLX e o Trados em suas versões mais recentes foram integrados em uma suíte de aplicativos de tradução que possui um banco de dados de termos, chamado MultiTerm¹⁴.

Em conjunto com uma MT, um banco de dados de termos pode melhorar ainda mais a consistência da tradução. Estes bancos são freqüentemente utilizados por tradutores em projetos. O único empecilho é que o banco **deve** ter sido povoado anteriormente ao processo de tradução, ou deve poder ser editado manualmente, para que sejam adicionados termos gradativamente.

Com base no exposto, é de opinião do autor que um banco de dados de termos deve acompanhar um aplicativo de MT, sendo que este deve ser editado apenas quando necessário, num processo separado do processo de tradução em si.

¹⁴Mais detalhes do site da SDL.

5 *Estudo da tecnologia web*

A *World Wide Web*¹ é um serviço que funciona sobre a Internet e provê um meio de publicar informação através desta. Esta informação se encontra na forma de texto, áudio, vídeo e vários outros tipos de informação, genericamente chamada de recursos (WIKIPEDIA, 2006s).

Embora largamente usados no mesmo sentido, Internet e Web (WWW) são conceitos distintos, sendo a **Internet** o **meio** por onde vários serviços funcionam (e-mail, compartilhamento de arquivos, etc.), e a **Web**, como já exposto, mais um **serviço** que roda através da Internet.

5.1 História breve

No final dos anos 50, várias pesquisas estavam sendo feitas pelo governo dos Estados Unidos no sentido de criar redes de grande escala entre computadores para o compartilhamento da informação. No fim dos anos 60, as primeiras redes já estavam ativas interligando os grandes centros de pesquisa do país (WIKIPEDIA, 2006e).

As tecnologias de rede foram evoluindo e no início dos anos 90 já estavam espalhadas pelo mundo inteiro, e ganharam conhecimento público. Nesse ponto, várias tecnologias que fariam a base de serviços da rede já haviam sido desenvolvidos, com destaque para o HTTP.

O HTTP (*Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto) foi peça chave para popularização da Web. Este protocolo rege o intercâmbio de arquivos de Hipertexto, arquivos HTML (*Hypertext Markup Language* - Linguagem de Marcação de Hipertexto) entre clientes HTTP, os navegadores web.

As primeiras páginas web nasceram como documentos estáticos. Os dados eram armazenados nos arquivos HTML, que eram editados através de simples editores de texto,

¹Abreviada WWW.

ou editores WYSIWYG especializados. Estes documentos ficavam em Servidores Web, computadores ligados na internet especialmente com o propósito de servir páginas. Esses servidores são identificados na rede através de seus URI (*Uniform Resource Identifier* - Identificador de Recurso Uniforme). O URI é o sistema de endereçamento usado para referenciar recursos dentro da Web (WIKIPEDIA, 2006s).

5.2 Novas tecnologias

O sucesso da Web e o crescimento da era "pontocom" marcou uma evolução da tecnologia web. A necessidade de páginas mais dinâmicas, que mostrassem seu conteúdo de acordo com a interação com um usuário, marcou o uso de linguagens de programação para tratar a entrada do usuário e baseado nele retornar uma página diferenciada.

O conteúdo destas páginas residia em bancos de dados, em contraste com as primeiras páginas HTML estáticas. Como dito antes, um programa interagiu com o usuário, apresentando uma página inicial e então, baseado nas entradas que este fazia na página (textos digitados, apertar de botões, etc), buscava o conteúdo requisitado no banco e criava uma página resposta de acordo (WIKIPEDIA, 2006q).

Estes programas rodam em conjunto com um servidor web e podem ser chamados de Sistemas de Gerenciamento de Conteúdo Web. Usados com os mais variados propósitos, os sistemas de gerenciamento foram um dos primeiros passos do crescimento da tecnologia web. Agora, as páginas web eram dinâmicas: podiam ser atualizadas mais freqüentemente, já que o único software necessário para isso seria um navegador.

Os aplicativos web estavam começando a se aproximar mais dos aplicativos desktop em funcionalidade. A interação com o usuário proporcionava uma experiência mais rica, e logo novas formas de interação foram criadas.

5.2.1 *Rich Internet Applications*

À medida que tráfego na Internet foi crescendo, o modelo cliente-servidor utilizado nos sistemas web foi se tornando um gargalo. Um modelo cliente-servidor consiste em um modelo de aplicação onde um programa cliente envia pedidos ao programa servidor, que processa estes pedidos e retorna uma resposta adequada, basicamente como foi descrito um sistema web.

O que acontecia é que nos primeiros sistemas web, praticamente toda a "inteligência" do

aplicativo estava no servidor, e o cliente basicamente recebia conteúdo HTML estático. As formas de interação com este conteúdo, os *widgets*² HTML padrões, não podiam proporcionar recursos típicos de aplicativos desktop (como "arrastar e soltar", por exemplo) (WIKIPEDIA, 2006j).

Então, foram criadas novas formas para interação com o usuário, tentando transferir parte da inteligência da aplicação para o programa cliente. Existem basicamente duas formas de fazer essa transferência: embutindo código interpretável pelo navegador dentro da estrutura do documento HTML (usando as chamadas linguagem de *scripting*), ou embutindo código executável, a ser interpretado por um programa a parte, que possui uma interface de comunicação com o navegador.

Um exemplo muito conhecido de linguagem de programação script pelos navegadores é o JavaScript. O JavaScript provê vários recursos de programação que uma linguagem de programação padrão provê, com a diferença de que é dinamicamente interpretada pelo navegador quando do carregamento de uma página. O navegador "lê" os comandos JavaScript e os executa, seqüencialmente ou sob demanda, de acordo com como o programa em JavaScript foi criado. A união de código JavaScript com HTML tornou-se o padrão chamado *Dynamic HTML*, ou DHTML.

O termo JavaScript, na verdade, refere-se à implementação da empresa Netscape³ do padrão ECMAScript para funcionar no seu produto Netscape Navigator. Existem outras implementações, como o JScript da Microsoft, que roda sobre o navegador Internet Explorer, da mesma empresa. Como pode ocorrer em implementações de diferentes companhias, o JavaScript e o JScript são, em vários aspectos, incompatíveis, o que dificulta a criação de aplicações web multiplataforma (WIKIPEDIA, 2006f).

Aplicações que rodam como código executável dentro de páginas HTML são chamadas *applets*⁴. Os dois exemplos de *applets* mais conhecidos são os *applets* Java, da empresa Sun Microsystems⁵ e os *applets* Flash, da empresa Macromedia⁶.

Um parêntese a ser feito: não se deve confundir Java com JavaScript. Java é uma linguagem de programação que, assim como JavaScript, têm boa parte de sua sintaxe derivada da linguagem de programação C. A grande diferença é que código JavaScript é interpretado diretamente pelo navegador, enquanto código Java é compilado e inter-

²Um *widget* (cujo nome não possui tradução para português), consitui um componente de uma interface gráfica com a qual o usuário interage.

³Que em 1998 foi comprada pela America Online.

⁴Sem tradução para português.

⁵<http://www.sun.com/>

⁶Recentemente comprada pela Adobe. Site: <http://www.adobe.com/>

pretado pela Máquina Virtual Java. Uma máquina virtual é um programa especial que interpreta comandos compilados para uma forma binária especial (chama bytecode). Uma MV (máquina virtual) funciona como um intermediário entre o aplicativo e o sistema operacional, proporcionando que aplicativos sejam rodados em plataformas diferentes sem a necessidade de recompilação.

5.2.2 Web 2.0

As *Rich Internet Applications* evoluíram então de modo a proporcionar uma experiência ainda mais rica ao usuário. O uso de técnicas nascidas no final dos anos 90, como APIs públicas de web services, Ajax e *web syndications* proporcionaram uma forma de publicação de conteúdo em massa (WIKIPEDIA, 2006q). Como os dois primeiros conceitos serão importantes para este trabalho, serão aprofundados mais adiante.

O termo em si foi popularizado por uma série de conferências sobre desenvolvimento web iniciadas em outubro de 2004 pela O'Reilly Media⁷ e pela MediaLive International⁸ (WIKIPEDIA, 2006q).

Os participantes das conferências nomearam alguns princípios chave que eles acreditam caracterizar a Web 2.0, como citado em Wikipedia (2006q). Aqui, é feita uma tentativa de explicá-los mais a fundo, já que estes têm relação com conceitos expostos anteriormente.

- Web como plataforma: o novo dinamismo de interação que os aplicativos Web exigiam e o surgimento das *rich internet applications* mudaram a visão original que se tinha da web, permitindo aos usuários ir além de apenas publicar conteúdo e dando a chance de poder fazê-lo crescer dentro da própria estrutura da web;
- A informação como a força motivadora: praticamente tudo na web passou a girar em torno da informação, e como as pessoas podiam contribuir para que ela fosse precisa e chegasse a todos, o que nos leva ao próximo item;
- Efeitos na rede criados por uma "arquitetura de participação": como as novos aplicativos web permitiam que qualquer um tivesse o potencial de colaborar através da rede, nasceu uma espécie de desenvolvimento colaborativo de conteúdo, que teve influência direta do modelo de comunidades que nasceu do movimento SL;

⁷<http://www.oreilly.com/>

⁸Mais tarde comprada pela CMP Media - site: <http://www.cmp.com/>

- Inovação pela criação de novos sistemas e sites compostos de fragmentos de sistemas de desenvolvedores diferentes: mais uma vez, vemos aqui a influência do modelo de desenvolvimento do SL. Essa idéia vem justamente a prática feita no SL - utilizar de aplicações ou partes de aplicações já existentes para melhorar algo que já existe ou combiná-las criando algo novo. A diferença principal é que essa composição não é feita utilizando exclusivamente sistemas abertos;
- Modelos de negócio "leves" proporcionados por *syndication*⁹ de serviços e conteúdo. A *Syndication* de serviço ou conteúdo é feita quando se disponibiliza este conteúdo de alguma maneira para que outros o usem;
- Fim do ciclo de adoção de software: com os serviços disponíveis através de um servidor centralizado, mudanças poderiam ser implementadas de modo transparente, e isso marcaria o fim dos ciclos de atualização por parte do usuário;
- O software além do nível de um único dispositivo: como está disponível em um meio, ao que parece, perpétuo, as chances de se tornar útil para alguma finalidade são maiores em longo prazo.

Como citado anteriormente, tanto APIs de web services quanto novas técnicas de desenvolvimento de *rich internet applications* foram importantes para o crescimento dos novos serviços web. Estas técnicas serão agora descritas.

5.2.2.1 Web Services APIs

O conceito de API já foi apresentado. A criação de APIs especiais para interagir com aplicações web é uma funcionalidade que permite que funções dos aplicativos web sejam chamadas diretamente e de modo seguro por parte do cliente, numa técnica conhecida como chamada remota de procedimento. Os exemplos mais conhecidos de RPCs¹⁰ são o XML-RPC, que obviamente usa XML para encapsular chamadas de funções, e o SOAP (*Simple Object Access Protocol*, ou Protocolo de Acesso Simples a Objetos.) que alia o conceito de RPC ao uso de objetos (também encapsulados em XML) como resposta à chamada.

O conceito de programação orientada a objetos remete ao uso de uma sintaxe diferenciada de programação, que encapsula dados, funções e lógica em estruturas chamadas,

⁹Infelizmente, o autor falhou em encontrar um termo apropriado em língua portuguesa.

¹⁰*Remote Procedure Call*, como é melhor conhecida esta técnica.

obviamente, de objetos. Uma linguagem de programação que utiliza esse conceito é dita, então, orientada a objetos. A linguagem Java, citada anteriormente, é um exemplo comum de linguagem orientada a objetos.

Já XML (*eXtensible Markup Language*, ou Linguagem de Marcação Extensível), é, assim como HTML, uma **linguagem de marcação**, que se utiliza de estruturas textuais denominadas marcações para indicar como o texto deve ser mostrado (WIKTIONARY, 2006). Ambas são derivadas do SGML (*Standard Generalized Markup Language*, ou Linguagem de Marcação Generalizada Padrão), e a diferença principal entre as duas é que HTML foi criada especificamente para a o uso com Hipertexto (texto que é interligado com outros, através dos conhecidos *hyperlinks*, ou simplesmente links), enquanto XML, como o nome sugere, foi criada para ser estendida e usada com propósitos variados.

5.2.2.2 Novas técnicas de desenvolvimento de *Rich Internet Applications*

O desenvolvimento da plataforma JavaScript permitiu que este se combinasse com outras formas de desenvolvimento de RIAs (*Rich Internet Applications*), criando novos usos para si e atraindo a atenção dos programadores.

Uma técnica teve destaque especial e também faz parte da estratégia de desenvolvimento deste trabalho: o Ajax. O *Asynchronous JavaScript and XML* (JavaScript e XML assíncrono) remete do uso de um objeto JavaScript que se comunica com o servidor HTTP de forma assíncrona, o `XMLHttpRequest`. Esse objeto foi primeiramente introduzido na versão da Microsoft do ECMAScript, o JScript, e posteriormente implementado em outras versões de outros navegadores devido a sua grande utilidade (WIKIPEDIA, 2006t).

Como o objeto faz uma requisição HTTP de modo assíncrono, a página que o navegador está exibindo não é afetada até que os dados tenham chegado do servidor e o processo de manipulação destes dados, que podem ser interpretados como texto puro ou XML, é transparente ao usuário.

Isso foi combinado ao fato de que JavaScript torna possível a atualização de apenas partes de um documento HTML sem que seja necessário a atualização de toda a página, acessando cada segmento do documento através de uma estrutura de árvore, chamada DOM ou *Document Object Model* (Modelo de Documento-Objeto). Uma árvore como uma estrutura de dados computacional refere-se a um conjunto de objetos denominados nodos, que guardam qualquer tipo de informação e estão ligados entre si numa estrutura que se assemelha aos galhos de uma árvore (WIKIPEDIA, 2006o).

Acessando e retornando conteúdo de forma assíncrona, atualizando o HTML somente quando necessário, quando combinados com vários outros recursos já existentes da linguagem JavaScript, faz um aplicativo web desenvolvido usando Ajax poder trabalhar de maneira mais semelhante a um aplicativo desktop, oferecendo rapidez de resposta aos comandos e transparência no uso. Do lado do servidor, a complexidade é consideravelmente diminuída, sendo que muito código pode ser movido para o cliente, mantendo a segurança dos dados no servidor.

O Ajax ganhou notoriedade recente devido a nova estratégia do portal de buscas Google¹¹ de entregar novos serviços ao usuário de maneira rápida e eficiente. Essa abordagem começou com a criação do serviço de webmail Gmail¹² em 2004 (WIKIPEDIA, 2006h). Outros serviços do Google muito conhecidos que utilizam a tecnologia Ajax são o Google Maps¹³, o Google Suggest¹⁴ e vários outros, listados na referência anterior. Outros aplicativos que usam Ajax são a nova interface do Hotmail¹⁵, que passa a se integrar com vários serviços similares aos do Google oferecidos pela Microsoft através do Windows Live¹⁶.

5.3 Sistemas web para tradução

Nesta seção serão analisados dois aplicativos web de tradução, um proprietário e outro livre. Será feita uma comparação com aplicativos desktop de tradução e proposta uma maneira de aproximar a usabilidade do desktop à interface web do aplicativo livre.

5.3.1 Rosetta

O Rosetta¹⁷, de acordo com a descrição do próprio site, é um "sistema baseado em web para traduzir software de código aberto para qualquer idioma¹⁸". Embora o próprio sistema em si não seja livre, ele se baseia no processo de internacionalização e localização dos projetos livres utilizando o Gettext como base.

O Rosetta é o sistema de tradução usado pela Canonical Ltd.¹⁹ para a tradução de seus projetos baseados em código livre, dentre os quais a ascendente distribuição do

¹¹<http://www.google.com/>

¹²<http://mail.google.com/>

¹³<http://maps.google.com/>

¹⁴<http://www.google.com/webhp?complete=1&hl=en>

¹⁵<http://www.hotmail.com/>

¹⁶<http://ideas.live.com/>

¹⁷<https://launchpad.net/rosetta>

¹⁸Tradução do autor.

¹⁹<http://canonical.com/>

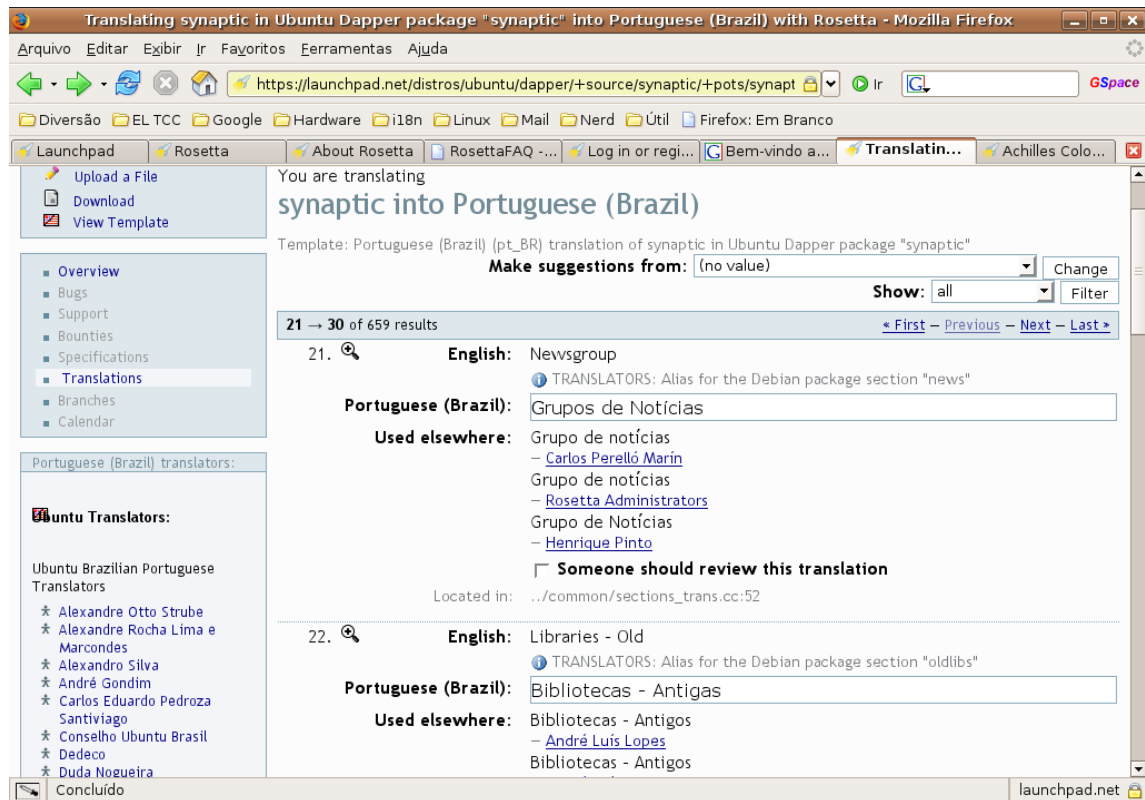


Figura 7: Interface de tradução do Rosetta

GNU/Linux Ubuntu²⁰ e o sistema de apoio à projetos de software livre Launchpad²¹.

Ele atua como um subsistema do Launchpad, focalizando na área de localização de software. O processo de desenvolvimento de toda a gama de aplicativos do Ubuntu está centralizado no Launchpad, e conseqüentemente a parte de localização no Rosetta.

5.3.1.1 Análise da interface de Tradução do Rosetta

Na figura 7, é mostrada uma captura de tela da interface do Rosetta. Nessa interface, como podemos ver na figura, os itens a serem traduzidos estão dispostos numa tabela de apenas uma coluna. Nela, podemos ver as palavras e frases a serem traduzidas (onde se pode ler a palavra "English" em negrito, no início de cada célula da tabela), e um campo de entrada onde deve ser feita a tradução (logo abaixo da palavra a ser traduzida, na mesma linha dos dizeres "Portuguese (Brazil)").

É possível notar uma série de sugestões de tradução, vindas possivelmente de traduções de outros programas (logo abaixo de onde se entra a tradução da palavra, onde se lê "Used elsewhere"). Estas sugestões podem ser tratadas como uma espécie de memória de

²⁰<http://www.ubuntu.com/>

²¹<https://launchpad.net/>

tradução, que utiliza os próprios arquivos de recursos gettext (os catálogos de mensagens, arquivos .PO) de outros programas como bancos de dados para mostrar correspondências totais (onde o termo ou frase achado no banco é exatamente igual ao termo sendo traduzido).

Estas sugestões podem ser feitas em traduções de outros idiomas. Onde lê-se ”*Make suggestions from*”, existe um menu onde o usuário pode escolher entre vários idiomas registrados. Uma vez feita a escolha e apertado o botão ”*Change*”, a página é recarregada e os campos de sugestões estarão carregados com palavras do idioma escolhido.

É possível notar outras informações úteis para o tradutor (nesse caso, onde se lê ”*TRANSLATORS: Alias for (...)*” e ”*Located in: (...)*”). Estas informações provêm do próprio arquivo PO, e dependem dos desenvolvedores para serem preenchidas.

Apesar de ser um bom sistema e já apresentar uma forma simples de memória de tradução, o Rosetta não possui nenhum aspecto de RIA, pelo menos não nos testes realizados. Uma abordagem assim poderia melhorar significativamente o processo de tradução e reduzir o tempo de carga das páginas.

5.3.2 Pootle

O Pootle²², é um sistema de tradução nos mesmos moldes do Rosetta. Desenvolvido em Python²³, propõe um tipo de tradução simples e direta de arquivo PO gettext, abstraindo o máximo necessário da sintaxe do arquivo e mostrando ao usuário o texto a ser traduzido.

Internamente, o sistema utiliza-se dos próprios arquivos PO para persistência dos dados. Quando se deseja criar um projeto, isto é feito através da interface de administração da ferramenta, onde se pode entrar com todas as informações necessárias e realizar o carregamentos dos arquivos PO usados pelo software a ser traduzido para o servidor.

Por ser uma ferramenta simples, a interface de administração do sistema é bem crua, sendo necessário que algumas coisas sejam editadas diretamente no servidor. Mas como a administração não é foco de estudo deste trabalho, trataremos apenas da interface de tradução.

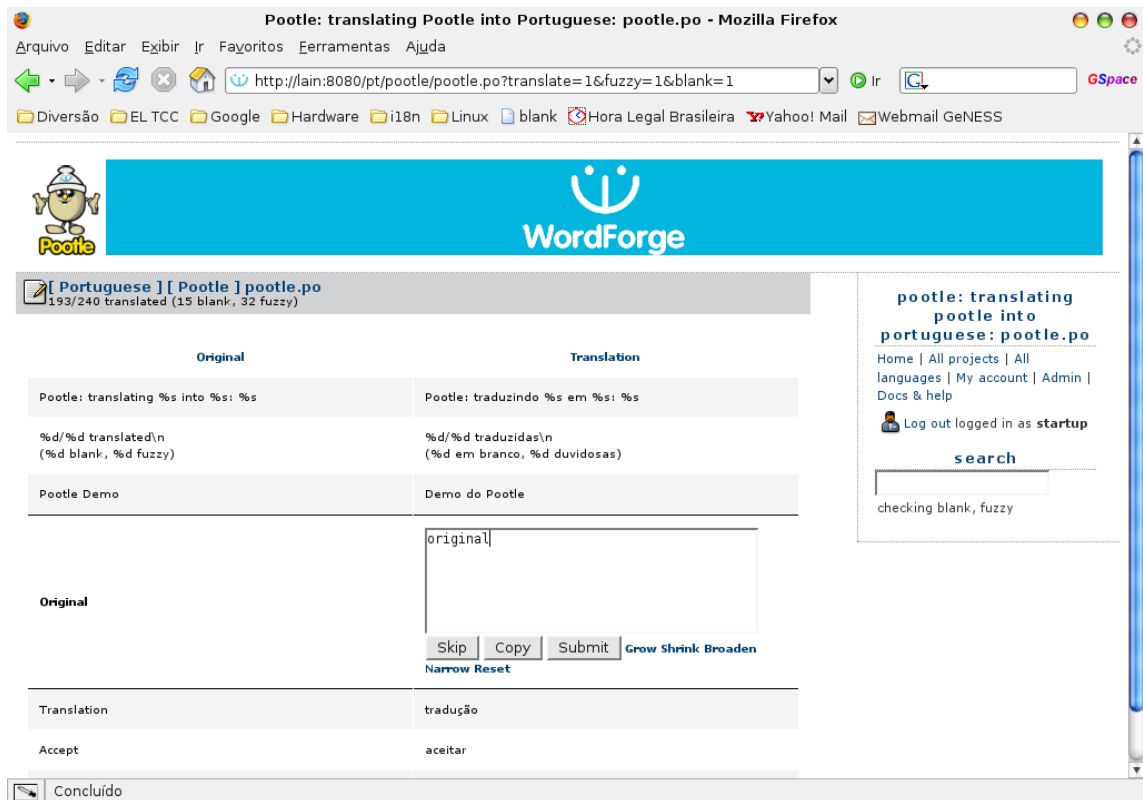


Figura 8: Interface de tradução do Pootle

5.3.2.1 Análise da interface de Tradução do Pootle

Na figura 8, nota-se a simplicidade da interface do Pootle. Uma tabela de duas colunas, onde na primeira da esquerda para a direita é mostrado o texto a se traduzir e na coluna ao lado, existe uma caixa de edição de texto onde o usuário faz a tradução.

Note que, e aqui se faz uma comparação com o Rosetta, somente um texto é editado de cada vez. Como se nota observando a figura, existe somente uma caixa de edição ativa em toda coluna de traduções. Embora não citado anteriormente, na interface do Rosetta aparece um número definido de frases a se traduzir e no fim da página existe um botão usado para enviar as traduções. No Pootle, a tradução é feita, submetida (através do botão "Submit", como mostra a figura), e a página toda é carregada novamente, e a caixa de edição da próxima frase é habilitada.

É possível observar também a inexistência de qualquer tipo de sugestão de tradução, como existe no Rosetta. Existe um meio de sugerir traduções (um botão "Suggest", que infelizmente não aparece na figura 8), mas estas sugestões somente aparecerão no processo de revisão da tradução do arquivo, feito pelo usuário administrador do projeto.

²²<http://pootle.wordforge.org/>

²³<http://www.python.org/>

6 Proposta e Implementação do Trabalho

Gerenciar projetos de localização é uma tarefa particularmente complexa, pois o processo envolve várias etapas, que vão desde o levantamento terminológico até a revisão geral da localização feita.

Na localização de projetos de software livre todo esse processo é praticamente inexistente, pois geralmente a localização é feita por voluntários, que muitas vezes não têm qualquer experiência de localização, o que pode às vezes resultar em uma localização de baixa qualidade.

Como já dito antes, assim como no mundo de software comercial no mundo do software livre a localização é um fator chave para aceitação. Projetos como o Gettext oferecem uma opção de padronização para a internacionalização do software, e as ferramentas associadas para tradução de arquivos PO oferecem meios para localização.

Ferramentas como o Rosetta e o Pootle incentivam ainda mais a localização do software livre, estando disponíveis para o uso através da internet para que os voluntários possam realizar sua colaboração, centralizando as informações e automatizando uma parte do processo. Mesmo o Rosetta sendo uma ferramenta comercial, ela é baseada no processo de localização de software livre e qualquer pessoa pode cadastrar-se e colaborar com a localização da lista de softwares que se encontra na página oficial.

O Pootle é uma alternativa totalmente livre, que oferece grande parte dos recursos do Rosetta, mas que possui uma interface menos elaborada, o que pode dificultar um pouco o processo de localização.

A proposta deste trabalho então é que a interface de tradução do Pootle pode ser melhorada, mantendo seu aspecto simples, através da adição de recursos RIA e de uma memória de tradução e banco de dados de termos simples.

Esta memória de tradução e banco de termos serão implementados nos mesmos moldes

da própria implementação do Pootle, que utiliza diretamente arquivos PO como forma de persistência, ao invés de utilizar bancos de dados relacionais. Serão utilizados então formatos de arquivos padrões da indústria da localização para realizar a persistência.

A interface web do sistema será modificada da seguinte forma: a troca de dados assíncrona será utilizada para diminuir a carga de informações do lado servidor, passando esse processamento para o lado cliente, que atualizará as informações disponíveis na página de maneira mais dinâmica, dando ao usuário a impressão de uma interface mais ágil.

Apesar de ser acrescentada a interface dinâmica utilizando AJAX, o aplicativo web em si deverá continuar funcionando independente da existência do suporte a JavaScript. Ou seja: se o navegador que acessa o sistema não possui suporte a JavaScript, o aplicativo deve funcionar sem impedir o usuário de realizar seu trabalho.

É óbvio que a ausência de suporte JavaScript impossibilita o uso de AJAX, logo o sistema web deve estar preparado para responder aos dois tipos de pedidos: síncrono e assíncrono. Quando o pedido é síncrono, o sistema devolve todas as informações necessárias para construir uma página completa. Mas quando identificado um pedido assíncrono, o sistema processa e devolve apenas as informações correspondentes a esse pedido.

Adicionando estes recursos, espera-se que a tradução utilizando o Pootle torne-se mais rápida e fácil, encorajando a adoção dessa ferramenta como um administrador totalmente livre de projetos de localização de software.

Espera-se também poder desenvolver um exemplo prático, ainda que simples, dos conhecimentos expostos até agora.

Neste capítulo, serão abordados em detalhes os conhecimentos adquiridos e as estratégias de programação criadas para implementar o módulo de memória de tradução e de banco de termos a serem acrescentados ao Pootle. A versão 0.9 do Pootle foi utilizada na implementação.

6.1 Python

Como o Pootle é um sistema desenvolvido em Python e essa linguagem de programação não era conhecida do autor, foi feito um estudo e aprendizagem da linguagem, cujos conceitos principais são apresentados a partir de agora.

Python é uma linguagem de programação orientada a objetos multiplataforma criada

por Guido van Rossum¹. Segundo o site oficial (já mostrado anteriormente) seus programas rodam nos sistemas operacionais Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, telefones móveis da Nokia e Windows. Neste trabalho, o desenvolvimento foi feito em Python 2.4 no sistema operacional GNU/Linux, distribuição Ubuntu.

Sendo uma linguagem interpretada, ela executa através de um **programa interpretador**. Um programa interpretador (ou simplesmente interpretador) lê o código fonte python e o executa, resolvendo símbolos (verificando o tipo dos dados sendo usados) dinamicamente em tempo de execução, razão pela qual Python é também dita uma linguagem **dinamicamente tipada**.

Um dos pontos fortes da linguagem Python está na enorme quantidade de aplicações para a qual sua biblioteca padrão oferece suporte. A biblioteca padrão (*standard library*) de uma linguagem de programação é um conjunto mínimo (nesse caso, bem extenso!) de funções que os desenvolvedores da linguagem se comprometem a suportar nativamente, e não através de módulos de terceiros. Outro ponto forte está na interoperabilidade com várias outras linguagens de programação muito usadas no mercado, como C e C++.

A título de curiosidade, é interessante dizer que Python utiliza o *escopo por indentação*: o escopo (o "tempo de vida" pelo qual o dado estará disponível para a realização de alguma lógica com este) de uma variável ou função está preso à indentação em que esta está. Uma variável está disponível para todo nível de indentação igual ou maior que o seu, mas não para um nível menor.

6.2 TMX

O TMX (*Translation Memory eXchange*, ou Intercâmbio de Memória de Tradução) é um padrão aberto de arquivo proposto pela LISA² (*Localisation Industry Standards Association* ou Associação para Padrões da Indústria da Localização, principal referência da área da indústria da localização) para o intercâmbio de dados entre aplicativos de tradução assistida por computador.

Baseado em XML, o padrão define uma estrutura de documento multilíngue que guarda em seu corpo vários pares frase-traduições, que podem ser usados por uma ferramenta CAT para povoar uma memória de tradução durante a sua execução e também para fazer persistência destes mesmos dados.

¹<http://www.python.org/~guido/>

²<http://www.lisa.org/>

A estrutura de um arquivo TMX é explicada através de um exemplo retirado de sua página de especificação³, mostrado a seguir. Propositalmente, não serão comentados todos os elementos e atributos descritos na especificação, só os que foram levados em conta no desenvolvimento do módulo TMX.

```
<?xml version="1.0"?>
<!-- Example of TMX document -->
<tmx version="1.4">
  <header
    creationtool="XYZTool"
    creationtoolversion="1.01-023"
    datatype="PlainText"
    segtype="sentence"
    adminlang="en-us"
    srclang="EN"
    o-tmf="ABCTransMem"
    creationdate="20020101T163812Z"
    creationid="ThomasJ"
    changedate="20020413T023401Z"
    changeid="Amity"
    o-encoding="iso-8859-1"
  >
  <note>This is a note at document level.</note>
  <prop type="RTFPreamble">{\rtf1\ansi\tag etc...{\fonttbl}</prop>
  <ude name="MacRoman" base="Macintosh">
    <map unicode="#xF8FF" code="#xF0" ent="Apple_logo" subst="[Apple]"/>
  </ude>
</header>
<body>
  <tu
    tuid="0001"
    datatype="Text"
    usagecount="2"
    lastusedate="19970314T023401Z"
  >
  <note>Text of a note at the TU level.</note>
  <prop type="x-Domain">Computing</prop>
  <prop type="x-Project">P&#x00E6;gasus</prop>
  <tuv
    xml:lang="EN"
    creationdate="19970212T153400Z"
```

³<http://www.lisa.org/standards/tmx/tmx.html>

```

    creationid="BobW"
  >
  <seg>data (with a non-standard character: &#xF8FF;).</seg>
</tuv>
<tuv
  xml:lang="FR-CA"
  creationdate="19970309T021145Z"
  creationid="BobW"
  changedate="19970314T023401Z"
  changeid="ManonD"
>
  <prop type="Origin">MT</prop>
  <seg>donn&#xE9;es (avec un caract&#xE8;re non standard: &#xF8FF;).</seg>
</tuv>
</tu>
<tu
  tuid="0002"
  srclang="*all*"
>
  <prop type="Domain">Cooking</prop>
  <tuv xml:lang="EN">
    <seg>menu</seg>
  </tuv>
  <tuv xml:lang="FR-CA">
    <seg>menu</seg>
  </tuv>
  <tuv xml:lang="FR-FR">
    <seg>menu</seg>
  </tuv>
</tu>
</body>
</tmx>

```

O cabeçalho do documento, envolto pelos rótulos `<header>` e `</header>`, guarda informações importantes sobre o conteúdo do arquivo e sobre o programa que o gerou. Estas informações podem ser usadas pelos aplicativos que interpretam o TMX, aplicando rotinas distintas se o arquivo foi gerado pela própria ferramenta ou não.

O corpo do documento, envolto pelos rótulos `<body>` e `</body>`, é a parte principal do arquivo e guarda os elementos que dizem respeito à frase original e suas traduções.

O corpo abriga vários elementos `<tu>`, as unidades de tradução ou *translation units*,

que por sua vez abrigam os elementos <tuv> (variantes de unidade de tradução ou *translation unit variants*), que contém a informação sobre o idioma da frase e a frase em si. O idioma da frase é representado segundo o padrão ISO 639, definindo dois caracteres para representar o idioma em si e mais dois caracteres para representar a região onde tal idioma é utilizado ("pt-br", por exemplo).

Como elementos opcionais, existem os elementos <note>, cuja função sugerida pela especificação é realmente guardar notas de tradução em todos os níveis do documento, possivelmente para facilitar a comunicação entre o desenvolvedor e o tradutor. E também os elementos <prop>, cuja sugestão da especificação é guardar propriedades das frases e mesmo do cabeçalho e das TUs (*translation units*) como um todo. Essas informações podem ser processadas pela aplicação e atuar na decisão sobre carregar aquela frase ou mesmo o documento inteiro para a memória (falando aqui da memória interna do aplicativo).

6.3 TBX

O TBX⁴ é um formato de arquivo criado para o intercâmbio de banco de dados de termos. Assim como o TMX, foi proposto pela LISA para se tornar o padrão da indústria de localização.

O TBX é um formato bastante completo no que diz respeito a guardar informações. Sua estrutura permite guardar vários pares termo-tradução, analogamente ao TMX. Mais do que o TMX, TBX pode também guardar sinônimos dos respectivos termos. Os termos podem ter mais de uma palavra e cada palavra pode ser classificada separadamente. Claro, o termo como um todo também pode ser classificado e definido.

O padrão define em sua especificação um conjunto de categorias em que um termo e seus componentes podem ser classificados, e também define normas para estender estas categorias ou criar novas.

Assim como foi feito com o TMX, a estrutura básica do TBX é explicada a partir de um exemplo, retirado do documento de especificação do padrão, encontrada em OSCAR Group (2005). Novamente, serão comentados apenas os elementos levados em consideração ao criar o módulo TBX.

<?xml version='1.0'?>

⁴*TermBase eXchange*, ou Intercâmbio de Bases de Termos

```

<!DOCTYPE martif SYSTEM "./TBXcoreStructureDTD-v-1-0.DTD">
<martif type='TBX' xml:lang='en' >
  <martifHeader>
    <fileDesc>
      <sourceDesc>
        <p>from an Oracle corporation termBase</p>
      </sourceDesc>
    </fileDesc>
    <encodingDesc>
      <p type='DCSName'>TBXdefaultXCS-v-1-0.XML</p>
    </encodingDesc>
  </martifHeader>
  <text> <body>
    <termEntry id='eid-Oracle-67'>
      <descrip type='subjectField'>manufacturing</descrip>
      <descrip type='definition'>A value between 0 and 1 used in ...</descrip>
      <langSet xml:lang='en'>
        <tig>
          <term tid='tid-Oracle-67-en1'>alpha smoothing factor</term>
          <termNote type='termType'>fullForm</termNote>
        </tig>
      </langSet>
      <langSet xml:lang='hu'>
        <tig>
          <term tid='tid-Oracle-67-hu1'>Alfa sim&#x00ED;t&#x00E1;si
            t&#x00E9;nyez&#x00F5; </term>
        </tig>
      </langSet>
    </termEntry>
  </body> </text>
</martif>

```

Do elemento principal do documento TBX, `<martif>`, é retirado o idioma fonte de todo o documento. A especificação determina que todos os textos deverão ser escritos no mesmo idioma que se encontra nesse elemento, a menos que explicitamente tenham um atributo `xml:lang` que defina outro idioma.

Dentro do elemento de cabeçalho (`<martifHeader>`), encontram-se informações que descrevem o documento TBX. O elemento `<fileDesc>` é obrigatório, e seus subelementos guardam descrições textuais, das quais somente `<sourceDesc>` é obrigatória e portanto é guardado na memória pelo *parser*, descrito em detalhes nas próximas seções. O elemento `<encodingDesc>`, apesar de ser bastante útil, apontando para uma descrição de

codificações usadas no arquivo, não é obrigatório e portanto não foi considerado.

O corpo do arquivo (envolto pelo elemento `<body>`, por sua vez envolto pelo elemento `<text>`), guarda os elementos que definem os termos que fazem parte da base. A título de informação, o elemento `<body>` é envolto por `<text>` porque existem dois elementos não obrigatórios, denominados `<front>` e `<back>`, que trazem mais informações ao arquivo e podem ocorrer antes e depois de `<body>`, respectivamente.

Os termos que fazem parte da base são envoltos pelo elemento `<termEntry>`, que por sua vez possui um conjunto de elementos `<langSet>`, que define o termo em cada idioma. Cada `<langSet>` pode possuir um ou mais elementos `<tig>` (ou `<ntig>`, que possui o mesmo propósito, mas é usado para guardar ainda mais informações, não consideradas na implementação), e cada elemento `<tig>` define o termo ou um sinônimo.

6.4 SAX

O SAX (*Simple API for XML* ou API Simples para XML), é um conjunto de interfaces que propõem um padrão para o *parsing* de arquivos XML.

Um *parser*, conforme descrito em Furtado (2006), é um algoritmo (uma seqüência de instruções) que, recebendo como entrada uma sentença x , emite como saída o *parse*⁵ de x , se x pertence a linguagem, ou um erro sintático, e x não pertence à linguagem. Simplificando este conceito, um *parser* compreende um algoritmo criado para verificar se uma entrada (que pode ser o conteúdo de um arquivo, ou simplesmente uma frase) obedece a uma sintaxe previamente estabelecida (chamada linguagem).

O *parser* DOM foi criado para verificar a sintaxe de um arquivo XML e durante esse processo, guardar as informações numa estrutura de dados baseada em árvore, como já descrito. O *parser* SAX define uma interface de verificação para arquivos XML que pode ser estendida para realizar qualquer função. Em princípio o *parser* SAX não guarda qualquer informação sobre o arquivo XML, sendo necessária a criação de uma especialização para tratar informações relevantes.

Baseando-se somente na explicação anterior, alguém pode afirmar que um *parser* DOM é uma especialização do SAX. De certa forma sim, mas DOM é mais antigo que SAX e utiliza algoritmos diferentes, logo é incorreto dizer que DOM deriva de SAX. Mas esses detalhes de implementação fogem ao escopo do trabalho.

⁵Sem tradução conhecida, nesse contexto podendo ser sinônimo de aceitação.

Outra diferença significativa é que a estrutura de árvore existente no *parser* DOM permite que quaisquer nodos sejam atualizados de forma transparente e rápida, desse modo atualizando elementos da estrutura de um documento derivado SGML igualmente de modo transparente.

6.4.1 A especialização do SAX para realizar o *parse* de arquivos TMX

Quando do início do desenvolvimento, o sistema Pootle já tinha uma implementação incompleta de *parser* TMX que utilizava uma especialização do *parser* DOM para tratar os elementos de um arquivo TMX.

Infelizmente, o *parser* DOM é conhecido por seu alto consumo da memória do computador, sendo viável apenas para documentos SGML (ou derivados) relativamente pequenos, pois ele guarda virtualmente todos os elementos de uma estrutura SGML ou derivada em memória.

Um arquivo de memória de tradução pode conter centenas ou até mesmo milhares de frases e traduções, e carregar todas estas informações na memória é custoso, uma vez que nem tudo será processado e/ou usado para alguma finalidade. Testes com o *parser* padrão TMX do Pootle confirmaram sua ineficiência.

Na última versão do Pootle estudada (versão 0.9, como já citado anteriormente), uma nova estratégia de implementação para o *parser* TMX foi criada, mas esta não foi considerada pelo autor em princípio, porque este já havia feito uma outra implementação, baseada em *parsers* SAX, que é descrita a partir de agora.

Como dito antes, a estrutura de árvore é custosa demais para arquivos XML razoavelmente grandes, DOM foi descartado para a implementação do *parser* TMX. Outra abordagem foi utilizada, derivada da abordagem proposta em Ogbuji (2004).

Essa abordagem defende o uso do *parser* SAX para criar dicionários a partir de estruturas XML. Dicionários, na linguagem Python, são uma estrutura de dados nativa que basicamente é uma lista de objetos que utiliza como índice outros tipos de objetos, em contraste com listas simples, indexadas por números inteiros.

Através da criação de uma máquina de estados finitos é possível definir a sintaxe do arquivo XML sendo verificado. Uma máquina de estados finitos, é um tipo de autômato finito. Segundo Furtado (2006), um autômato finito corresponde a um sistema formal que possui um conjunto finito de estados, um conjunto finito de entradas (alfabeto), um

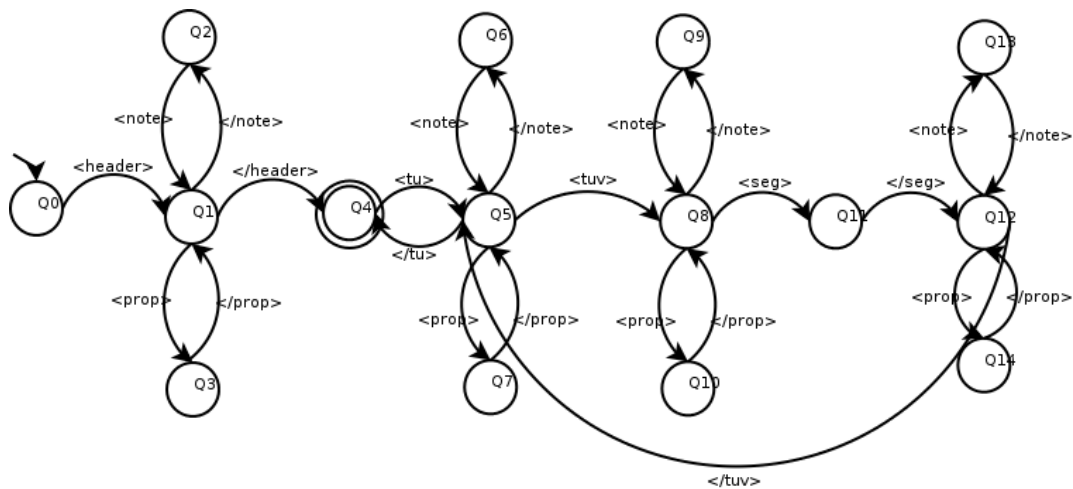


Figura 9: Autômato finito que descreve o funcionamento do *parser* TMX.

conjunto de regras de transição de estados, um estado inicial e um conjunto de estados finais. Os últimos estados citados devem pertencer ao conjunto inicial.

Simplificando essa definição, um autômato finito (ou uma máquina de estados finitos), corresponde a um sistema que dado um estado inicial e a inserção de uma entrada, utiliza uma função de transição para passar a um outro estado. Essa entrada pode ser também um conjunto de entradas, subconjunto do alfabeto do autômato, que é "consumido" unidade por unidade até que se atinja um estado final. Se ao final da entrada um estado final é atingido, esta é aceita. Caso contrário, é apontado um erro. A sintaxe de um arquivo XML pode ser facilmente descrita através de um autômato.

Uma vez definido, o *parser* implementado guarda todas as informações do arquivo TMX consideradas importantes pelo autor numa estrutura composta por vários níveis de dicionários, indexadas em primeiro nível pela frase na língua original, e próximo nível pelos códigos que representam os idiomas. Também são guardados os elementos de notas e propriedades do TMX, embora eles não seja proposto um uso para eles.

Foram criados dois *parsers* distintos, um para carregar as informações do arquivo TMX numa estrutura de dicionário para ser usado pela ferramenta de MT (no caso o Pootle), e outro para atualizar um arquivo TMX já existente.

A proposta do *parser* atualizador é guardar as informações originais do arquivo TMX que não são processadas pelo *parser* carregador, de modo que, se o arquivo TMX provém de um outro aplicativo de MT, estas sejam preservadas e o arquivo possa ser reutilizado pela aplicação original.

Infelizmente a figura 9 pode mostrar somente a estrutura básica do *parser* criado,

como ele reconhece um arquivo TMX, não dando qualquer idéia de como as informações são guardadas na memória. Mesmo assim, complementa a descrição feita anteriormente de como um arquivo TMX é estruturado. Para mais detalhes sobre a estrutura de dados usada e sobre o *parser* em si, consulte os anexos.

6.4.2 A especialização do SAX para realizar o *parse* de arquivos TBX

Assim como no *parser* TMX, a última versão estudada do Pootle possui um *parser* TBX, que usa uma abordagem híbrida de estruturas de dados padrões de Python e árvores XML para guardar os dados.

Quando do estudo da estrutura de um arquivo TBX, descobriu-se com a leitura da especificação a enorme quantidade de informações que esse formato pode guardar, e as infinitas possibilidades de uso destes dados.

Infelizmente, o uso proposto do banco de termos como uma ferramenta simples para consulta de termos usados em um projeto de tradução não utiliza sequer uma porção mínima dos recursos que o formato propõe.

Foi considerada numa solução utilizando um arquivo simples com termos separados por colunas e tabulações (ou vírgulas), mas essa proposta foi descartada pois fugiria ao escopo desse trabalho criar uma sintaxe, ainda que minimalista, para um arquivo de banco de termos.

Criou-se então um *parser* nos mesmos moldes do *parser* TMX: somente os elementos obrigatórios descritos anteriormente são guardados em memória, para consulta posterior. Por conveniência, não foi criado um *parser* atualizador.

Ao contrário do TMX, cuja implementação de um *parser* mínimo permitiu aproveitar a maioria dos recursos do formato, a implementação de um *parser* TBX mais elaborado acaba fugindo do escopo deste trabalho, devido as inúmeras possibilidades que o formato propõe.

Com base então no exposto, pode-se afirmar que o *parser* TBX implementado aproveita somente os recursos obrigatórios de um arquivo TBX, criando um estrutura de dados que guarda um conjunto de termos e seus respectivos sinônimos e traduções para consulta pela interface do sistema Pootle.

Na figura 10 podemos ver o autômato que define o *parser* TBX.

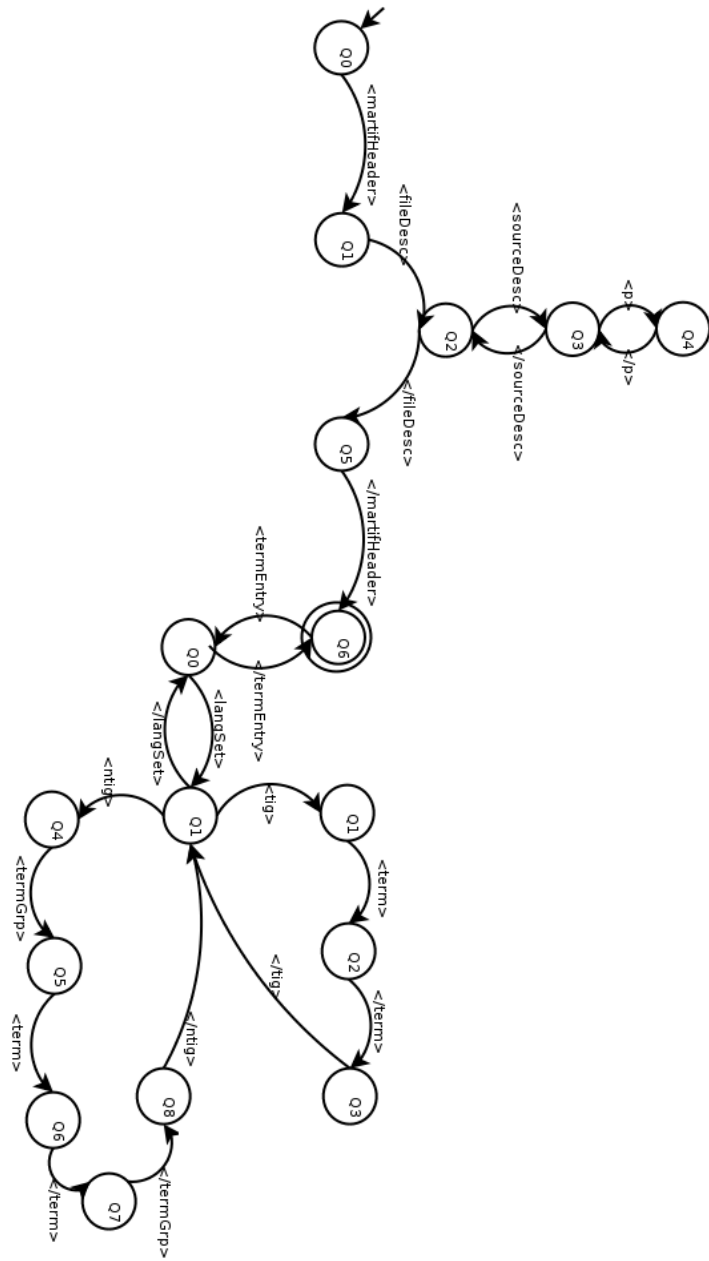


Figura 10: Autômato que define o *parser* TBX.

6.5 A interface da estrutura de Memória de Tradução com o sistema Pootle

Como a estrutura do dicionário criada é razoavelmente complexa (mas consideravelmente menos complexa que uma estrutura DOM), foi criada uma classe especialmente para lidar com o dicionário.

Esta classe permite que a estrutura seja consultada e atualizada de maneira correta. As consultas e atualizações são feitas utilizando como índice a frase no idioma fonte do arquivo. Quando uma consulta é feita, se a frase não se encontra no arquivo, ou se não possui tradução no idioma requerido, é retornado um string vazio.

Quando é feita uma atualização, a classe checa se a frase já existia anteriormente e se for o caso, simplesmente faz a atualização. Caso contrário, é criada uma nova entrada na estrutura TMX. Por conveniência, é possível retornar todos os índices da estrutura sob a forma de lista.

Tendo terminado a classe que manipula o arquivo de memória de tradução, partiu-se para a criação de um widget que utiliza essa classe para consultar um arquivo de memória de tradução e mostra as frases que possuem semelhança com a frase sendo atualmente traduzida. Para tal, foi necessário compreender como é organizado um projeto de tradução dentro do sistema Pootle.

Consultando exclusivamente o código-fonte do Pootle, chegou-se a seguinte conclusão de como funcionam os projetos de tradução existentes:

- Os projetos de tradução no Pootle estão dentro de uma estrutura de diretórios pré-definida, que faz parte da estrutura de diretórios principal do sistema;
- Um mesmo projeto de tradução pode possuir vários arquivos PO de recursos, organizados de duas maneiras distintas: cada arquivo pode encontrar-se dentro de um diretório cujo nome reflete o idioma para o qual ele está sendo traduzido (padrão), ou todos os arquivos podem se encontrar dentro do diretório principal do projeto, e nesse caso o nome reflete o idioma;
- Como já citado antes, a persistência é feita diretamente nos arquivos PO. Eles são carregados na memória quando acessados. Estatísticas e outros atributos do arquivo também são salvos (em arquivos próprios, separadamente) para consulta posterior, mas a compreensão desse sistema não acrescentará nada à implementação;

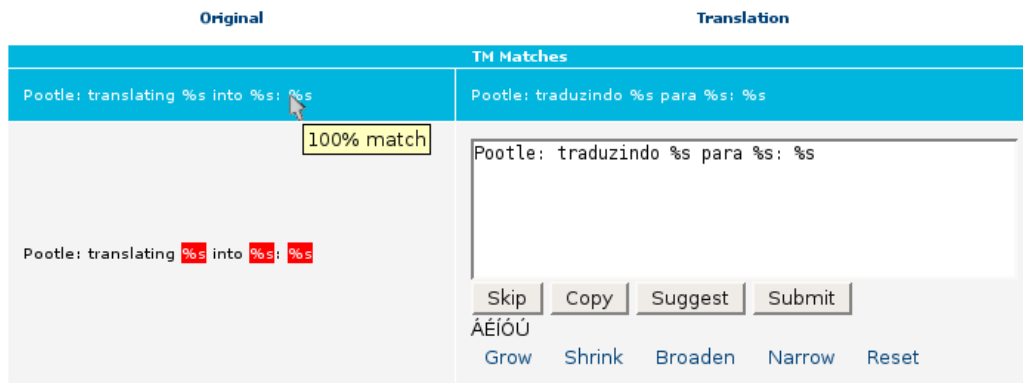


Figura 11: Memória de tradução do Pootle em uso.

- A classe que manipula um projeto de tradução está diretamente associada ao software e ao idioma para o qual está sendo feita a tradução, de modo que podem existir várias instâncias da classe com o mesmo projeto em comum, mas todas elas com idiomas diferentes;
- Quando um projeto é acessado, a classe é instanciada e armazenada num cache interno. Este cache serve não só para proporcionar acesso rápido ao projeto, mas também para retornar a mesma instância do projeto a cada usuário diferente que acessá-lo, prevenindo a inconsistência das informações (como todos trabalham com a mesma instância, alterações feita por um usuário refletem-se a todos os outros);
- Também quando um projeto é acessado (seja para visualização ou tradução), é criada uma instância da classe que representa a interface de tradução usada pelo usuário, e essa classe faz uso da instância da classe projeto para atualizar o arquivo de recursos PO. Quando o usuário sai da interface de tradução, a classe de tradução é descartada, mas a de projeto não.

Com base nisso, estabeleceu-se que uma memória de tradução deveria estar associada a um projeto, ou seja, cada instância de uma classe relacionada a projeto possui uma instância da classe de memória de tradução associada. Desse modo, o sistema de cache é aproveitado e, como acontece na edição de um arquivo, as mudanças feitas por um usuário refletem-se a todos os outros que estão trabalhando no mesmo projeto.

Assim como a classe de interface de tradução faz uso da classe de projeto, ela passará então a fazer uso da classe de memória de tradução associada ao projeto para realizar consultas e atualizações.

As consultas são feitas em conjunto com a carga da frase a ser traduzida, ou seja:

quando a interface carrega a próxima frase, ela também consulta a memória de tradução, e os resultados encontrados são mostrados acima da frase sendo traduzida, como mostra a figura 11. Dessa maneira, o tradutor tem um auxílio visual para comparar, entre as frases que a MT trouxe, a que melhor lhe serve. Uma vez escolhida a frase, esta pode ser selecionada e copiada através do mouse.

Quando o tradutor submete uma nova frase através da interface, esta frase é atualizada agora também no arquivo de memória de tradução. Outra situação em que a memória de tradução é atualizada é durante processo de revisão da tradução de um arquivo PO (já citado anteriormente): quando uma sugestão é aceita pelo usuário que está fazendo a revisão, a nova tradução é inserida na memória de tradução.

E como em qualquer outra memória de tradução, a busca também abrange frases com semelhança parcial com a frase sendo buscada. Para tal, foi feita uma pesquisa sobre como fazer esse tipo de comparação de frases, chamado em inglês de *string fuzzy matching* (podendo ser traduzido como correspondência parcial de strings).

6.5.1 Sobre a Correspondência Parcial de Strings

Para atender o requisito de procura por strings parcialmente semelhantes ao string de busca, foram considerados basicamente dois métodos de busca parcial, a distância de Levenshtein (ou distância de edição) e a distância de N-Gram.

Ambos os métodos diferem basicamente na função de similaridade (WIKIPEDIA, 2006c). Enquanto Levenshtein utiliza como medida de comparação apenas operações básicas acerca de um string (inserir, remover e substituir um caractere), a distância de N-Gram utiliza-se de cálculos baseados em estatística (WIKIPEDIA, 2006i) que introduzem uma complexidade matemática não necessária e que pode tomar um tempo considerável para realizar uma busca, sendo por esta razão descartado.

O funcionamento do algoritmo de distância de edição foi então mais aprofundado e foi considerado como adaptá-lo para o uso no aplicativo de MT. O funcionamento do algoritmo é descrito em detalhes em (WIKIPEDIA, 2006g).

Basicamente, o algoritmo constrói uma matriz cujas dimensões são os tamanhos dos dois strings sendo comparados. A partir dessa matriz, um cálculo é feito e é atribuído um peso a cada operação feita para transformar o primeiro string no segundo. Como dito anteriormente, as operações envolvem inserção, remoção e substituição de um caractere a cada iteração do algoritmo. O resultado final é o número mínimo de operações necessário



Figura 12: Interface da busca de termos do Pootle.

para transformar um string em outro.

Com base neste resultado, tomando o número de operações e o tamanho do string de procura, é possível estabelecer uma proporção utilizando regra de três simples. Essa proporção pode ser considerada a semelhança entre os strings e é o que foi adotado no algoritmo de correspondência usado neste trabalho.

6.6 A interface da estrutura do banco de dados de termos com o sistema Pootle

Como já citado anteriormente, o *parser* TBX implementado não abrange um conjunto considerável das possibilidades do formato, limitando-se apenas à consulta de termos. Pela mesma razão, tanto a estrutura de dados associada ao *parser* quanto a classe que a abstrai são mais simples.

Mesmo assim, ela serve ao seu propósito. Tendo um conjunto de termos previamente criado e exportado para TBX, o *parser* traz os termos à memória do aplicativo e estes podem ser consultados a qualquer momento.

Desse modo, toda vez que um tradutor está em dúvida quanto à tradução de um termo, e o resultado apontado pela memória de tradução não esclarece a dúvida, uma consulta pode ser realizada no banco de dados de termos.

A busca de termos em si também é simplificada, uma vez que a distância de edição não mostrou-se eficiente para termos pequenos. O termo buscado é comparado com os termos da base, e se é um substring de algum destes termos, as respectivas traduções são retornadas.

Tendo a classe tratadora de arquivos TBX, partiu-se para a criação da classe que abstrai o banco de dados de termos e do widget que mostra os resultados ao usuário. Felizmente, os conceitos aprendidos anteriormente com a criação da MT foram de grande valia para a rápida criação das classes referenciadas, que foram criadas e inseridas no

sistemas de modo análogo. Na figura 12, podemos ver a interface de busca funcionando.

Não foi desenvolvida uma maneira de editar os bancos de termos, pois o autor considera que isso foge ao escopo do trabalho, que trata-se principalmente de memórias de tradução. Mas a edição da base pode ser feita através de um aplicativo de edição de planilhas. A planilha de termos pode ser salva em formato de texto separado por vírgulas (extensão .csv) e posteriormente exportada para TBX por uma ferramenta de linha de comando que acompanha o Pootle.

Essa decisão em princípio pode parecer fora da proposta do trabalho, que é facilitar o processo de tradução. Mas como o processo de levantamento terminológico deveria ser realizado anteriormente ao processo de tradução, e os termos levantados não devem sofrer atualizações a não ser em caso de falha explícita de tradução, é possível notar que esta decisão afeta apenas o trabalho do gerenciador de projetos de tradução. O trabalho do tradutor não é afetado.

6.7 A implementação de uma forma de mostrar o contexto da frase ao tradutor

Como citado anteriormente, é extremamente importante para o tradutor saber o contexto em que a frase sendo traduzida se encontra. No caso da tradução de um software, a mesma palavra pode ser traduzida de maneira diferente se aparecer em um menu ou em um diálogo, por exemplo.

Como também citado anteriormente, o autor sugere uma maneira de híbrida de interface de tradução, onde o contexto da frase não aparece visualmente para o tradutor, mas sim textualmente na forma de uma explicação.

A implementação desta sugestão é feita levando em conta a estrutura das entradas de um arquivo de recursos PO, retirada de (FOUNDATION, 1998), e mostrada a seguir:

```
white-space
# translator-comments
#. automatic-comments
#: reference...
#, flag...
msgid untranslated-string
msgstr translated-string
```



Figura 13: Interface do Pootle, mostrando informações adicionais

Todas as linhas que começam com ”#” correspondem a comentários. Mas o próprio manual do Gettext sugere que um software que interprete os arquivos PO para tradução tenha consciência destes comentários, pois eles podem guardar informações relevantes à tradução.

A compreensão do autor do manual do Gettext é de que os dois primeiros tipos de comentários (”*translator-comments*” e ”*automatic-comments*”) são os que podem guardar esta informação relevante. Os ”*automatic-comments*”, como o nome sugere, são extraídos automaticamente do código-fonte do programa e são usados pelos programadores para deixarem notas aos tradutores do arquivo. Os ”*translator-comments*” podem ser utilizados pelos próprios tradutores para guardarem notas próprias, como comentários para outros tradutores, por exemplo. O uso destes comentários foi sugerido pela interface do Rosetta, já mostrada anteriormente.

A implementação do *parser* PO do sistema Pootle guarda corretamente as informações de comentários, e estas podem ser acessadas e modificadas livremente. Sendo assim, a classe de interface de tradução do Pootle foi modificada de modo a, quando estes comentários estiverem presentes, serem mostrados também no canto direito da tela, como mostra a figura 13.

É claro que, para a real eficiência deste sistema de contexto, é necessário planejamento por parte do desenvolvedor. Quando do desenvolvimento, todas as frases que serão exportadas para arquivos PO deverão ter seu comentário explicativo correspondente, que será então extraído pelo Gettext. Desse modo, o tradutor poderá ter informações adicionais de como traduzir o referido texto.

Tanto a busca por termos quanto a mostragem de contexto foram implementados de modo a ficarem sempre visíveis no canto inferior direito da interface da página de tradução, de modo a serem consultados com facilidade. Para tal, foi necessária uma pesquisa mais refinada, para manter a compatibilidade da interface de tradução com os browser mais utilizados.

O padrão W3C⁶ (World Wide Web Consortium, o Consórcio internacional que normatiza os padrões de protocolos, formatos de arquivos e muitos outros aspectos presentes na web), desenvolveu o padrão de arquivo CSS (*Cascading Style Sheets*, ou Folha de Estilo em Cascata) para realizar a diferenciação da apresentação dos dados dos dados em si em um arquivo HTML, já descrito anteriormente. O CSS representa a formatação dos dados apresentados, enquanto o HTML encerra os dados em si.

No entanto, assim como aconteceu com o ECMAScript, a implementação do CSS é diferente através dos diferentes navegadores. Para implementar os widgets de mostra de contexto e busca de termos descritos de modo que estes funcionassem corretamente no navegador mais utilizado no mercado, o Internet Explorer, foi necessário se voltar a *hacks*, códigos que exploram falhas na implementação do aplicativo.

6.8 Implementação da comunicação assíncrona com o servidor

A comunicação assíncrona utilizando JavaScript é feita principalmente através da classe `XMLHttpRequest`, como citado anteriormente. Para cada requisição assíncrona que é feita, uma nova instância da classe é criada.

Ao objeto `XMLHttpRequest` é então associado uma função JavaScript que trata o conteúdo retornado pelo servidor, assim que este se encontra disponível. Este conteúdo pode ser texto simples ou XML, que pode ser manipulado pela função tratadora (Na realidade, a resposta do servidor pode ser tratada como outros tipos de conteúdo, mas estes não são abordadas neste trabalho).

No entanto, instanciar com sucesso um objeto `XMLHttpRequest` é uma tarefa indigna, facilmente passível de erro, como outras tantas operações usando JavaScript. Principalmente de não se tratar de um padrão ECMAScript, o `XMLHttpRequest` é possui formas confusamente distintas de ser instanciado em browsers diferentes.

Como JavaScript não era também uma linguagem de conhecimento do autor, ocorreu um estudo para melhor utilizá-la e compreendê-la. A impressão deixada pelo uso da linguagem é de que ela é pobre, com muitos poucos recursos disponíveis em sua API padrão, em constraite a Python.

Muitas das limitações de JavaScript dizem respeito ao ambiente onde seus scripts

⁶<http://www.w3.org/>

executam: um interpretador dentro do programa navegador. Ao contrário de um interpretador que roda separadamente, o interpretador JavaScript possui acesso extremamente limitado aos recursos do sistema operacional, até mesmo por razões de segurança. Essas e outras limitações impedem o desenvolvimento de funções mais elaboradas.

Mas, e isso é uma opinião do autor, estas limitações acabaram impedindo que a API padrão se desenvolvesse, o que muitas vezes faz o programador voltar-se a APIs de terceiros, como foi o caso neste trabalho.

Utilizando a biblioteca MochiKit⁷, o autor encontrou uma maneira de instanciar objetos JavaScript que funciona em diversos browsers diferentes. Tendo em mãos uma maneira de realizar a comunicação assíncrona do lado do cliente, passou-se a visionar as mudanças necessárias no servidor.

Como uma das metas do projeto é que a memória de tradução e o banco de termos funcionem independentemente da presença de suporte JavaScript, ou melhor, que o sistema como um todo funcione dessa maneira, a solução encontrada para que o servidor distingüia requisições síncronas de assíncronas leva em conta a presença de um parâmetro específico na URL requisitada: na presença desse parâmetro, a chamada é assíncrona, caso contrário é síncrona.

Quando a página requisitada chega ao usuário, código JavaScript é executado de modo a atribuir uma função especial a determinados elementos da página. Quando o usuário interage com um destes elementos a função é chamada, e o parâmetro é inserido automaticamente na URL que será chamada.

O sistema, quando detecta a presença deste parâmetro, instancia a classe que representa a página requisitada com o mesmo parâmetro. A classe então processa as informações de acordo, e devolve o conteúdo correspondente.

Na interface da página de tradução do Pootle, essa função é atribuída a dois elementos específicos, na verdade a um grupo de elementos e a um elemento específico: aos formulários de submissão de novas traduções, e ao formulário de busca de termos.

A interface da página de tradução do Pootle foi modificada de modo a conter vários formulários, um para cada frase sendo traduzida. Ao submeter uma tradução, a função assíncrona envia os dados do formulário junto com o parâmetro especial que remete a busca assíncrona na memória de tradução.

Do lado do servidor, a implementação mostrou que é interessante devolver não só o

⁷<http://mochikit.org/>

resultado da consulta na MT como também as informações adicionais daquela entrada do arquivo PO. Desse modo, toda vez que submete uma tradução, ou escolhe uma nova frase para traduzir (a interface permite que se "pule" frase que não se quer traduzir), uma única consulta é feita para retornar ambas as informações, que são então atualizadas na página de tradução.

No formulário de busca de termos o princípio aplicado é o mesmo: o termo buscado é submetido e a função assíncrona anexa o parâmetro especial na consulta. O servidor, ao detectar o parâmetro, devolve apenas os dados pertinentes à consulta no banco de dados de termos.

É importante frisar que a submissão assíncrona de dados não é feita em todas as frases da interface de tradução. Na última frase da lista foi mantida uma consulta padrão, atualizando toda a página, pois novas frases devem ser carregadas na interface e o algoritmo de carregá-las é um tanto complexo e exigiria uma reformulação completa da interface de tradução, o que infelizmente não pôde ser abordado na implementação das mudanças.

A busca assíncrona também não foi implementada no processo de revisão de traduções, pois este processo se encontra a parte do processo de tradução em si, e a modificação também necessitaria uma reformulação elaborada da interface de tradução.

6.9 Nota sobre o código-fonte contido nos anexos

Foram feitas várias modificações em código-fonte já existente durante a implementação deste trabalho. Os arquivos que possuem somente modificações serão reproduzidos integralmente, mas **com um texto indicando que o arquivo é modificado antes da apresentação do código e com as modificações destacadas por comentários descritivos dentro do código**. É importante frisar **que todo o código que não faz parte das modificações feitas pelo autor é de autoria dos desenvolvedores do projeto Pootle**.

7 *Teste da Implementação e Análise dos Resultados*

A implementação descrita anteriormente foi feita de modo iterativo, e cada iteração compunha-se das seguintes fases: codificação da idéia proposta, uma série de testes sobre o código escrito, aí incluindo testes de unidade com as classes codificadas, visando encontrar erros e achar possíveis falhas na idéia proposta, e testes de integração com o sistema Pootle. A última parte da iteração consistia na correção dos erros e outros problemas encontrados.

Seguindo o estilo usado em todo este trabalho, este capítulo conterà uma descrição teórica e basicamente textual dos testes realizados, dos problemas encontrados e das providências tomadas. Todo o código escrito, incluindo os testes realizados, encontram-se nos anexos.

7.1 Testes do *parser* TMX

A codificação do *parser* TMX foi realizada primeiramente utilizando como base um autômato finito mais simples do que o mostrado anteriormente. Mesmo assim, a primeira versão do *parser* era estável e realizava com sucesso o *parse* de arquivos TMX bem formados.

Utilizando um teste simples, que primeiro carrega um arquivo TMX já existente na memória, faz atualizações aleatórias, grava o arquivo e depois o lê novamente apontou falhas principalmente na codificação da montagem da estrutura de dados visionada, principalmente erros com os índices usados.

Uma vez corrigidos estes erros, realizou-se um segundo teste, que criava um arquivo TMX totalmente novo, o povoava com frases também aleatórias, e finalmente gravava o arquivo em disco. Felizmente os erros encontrados anteriormente refletiram-se na realização com sucesso deste teste.

Mas uma reflexão foi feita em cima do autômato em si revelou alguns erros conceituais. O autômato garantia que as estruturas principais de um arquivo TMX fossem reconhecidas corretamente, como um elemento `tuv` e seus subelementos, mas não garantia que a ordem em que estes elementos no arquivo fosse correta. Um elemento `header` poderia aparecer no final do arquivo, por exemplo.

Com base nisto, foi feita uma reformulação do autômato, cuja forma final é a apresentada anteriormente. Basicamente adicionou-se novos estados para garantir a ordem correta dos elementos. Na implementação, bastou também a adição de novos estados, e a garantia correta da transição entre eles.

Os testes realizados anteriormente foram novamente aplicados, e revelaram pequenas falhas no algoritmo, rapidamente corrigidas.

7.2 Testes da interface de memória de tradução no Pootle

Uma vez analisado como o sistema Pootle funciona internamente (como descrito anteriormente), e criado uma classe que abstrai o conceito de memória de tradução para um projeto e um widget para mostrar as frases que a MT retorna, partiu-se para o teste do widget.

O teste foi feito da maneira mais óbvia: utilizando um projeto de exemplo do próprio Pootle, fez-se a tradução de várias frases, e freqüentemente voltava-se a uma frase anterior para constatar que a tradução havia mesmo sido guardada na memória. Voltando à tradução de uma frase anterior, forçava-se a MT a fazer uma busca e mostrar um resultado com 100% de correspondência (a tradução já feita).

Quanto à relevância das traduções anteriores nas novas traduções, o próprio projeto de teste mostrou a eficácia do conceito de MT: com poucas frases traduzidas a MT já mostrava correspondências significativas, com porcentagem superior à 60% de correspondência, o limite mínimo configurado.

7.2.1 Sobre a interface de mostra de contexto

Ao mesmo tempo que realizou-se o teste da memória de tradução, realizou-se o teste do widget de mostra de contexto. Apesar do projeto teste do Pootle ter poucos comentários com informação relevante, a inserção manual de mais comentários provou que as modi-

ficações na classe de projeto para retornar os requeridos comentários estava funcionando corretamente, e que os comentários estavam sendo corretamente visualizados na interface de tradução.

Estes testes constataram a dependência desse módulo de boas práticas de internacionalização do código. Infelizmente, um código com poucos comentários explicativos (aqueles que se encontram imediatamente antes da função de internacionalização `Gettext`, e que são extraídos pelas ferramentas `Gettext` para o arquivo PO) por parte do desenvolvedor ajuda pouco o tradutor nesse quesito.

7.3 Testes do *parser* TBX

Como já citado anteriormente, as funcionalidades do formato TBX utilizadas no projeto foram muito aquém das possibilidades que o formato propõe. Nem por isso, sua utilização tornou-se menos importante no projeto.

Assim como a implementação do *parser* TBX baseou-se em grande parte na implementação do *parser* TMX, os testes realizados no *parser* TBX também foram baseados nos testes do TMX. Basicamente, foram os mesmos: a atualização de um arquivo já existente, e a criação de um novo arquivo.

Como não foi implementado um *parser* atualizador no caso do TBX, a grande maioria das informações de um arquivo TBX mais completo foi perdida. Mas é necessário frisar novamente que todas as informações descartadas pelo *parser* TBX provêm de elementos não obrigatórios, cuja presença não é necessária em um arquivo.

Os testes de atualização mostraram necessário criar um valor padrão para uma informação obrigatória, a descrição do arquivo. Como a classe que trata arquivos TBX não obriga o usuário a inserir essa informação, valeu-se desse valor padrão para criar um arquivo totalmente novo, se nenhuma descrição é inserida.

7.4 Testes da interface de banco de dados de termos no Pootle

A exemplo do *parser* TBX, a implementação da interface de busca de termos também foi amplamente baseada na interface de memória de tradução, como dito anteriormente.

Como não se dispunha de um arquivo TBX com os termos utilizados no projeto, criou-

se um com alguns termos aleatórios, todos traduzidos corretamente, óbvio. A utilização da busca provou-se eficaz quando esta era realizada utilizando-se até uma palavra. Com duas ou mais, não foram obtidos resultados. Isso se deve a dois fatores:

- O arquivo TBX criado foi povoado com termos proveniente do próprio arquivo de recursos exemplo do Pootle, e todos os termos eram compostos por apenas uma palavra;
- O algoritmo de busca utilizado, que considera um termo como relevante apenas se este é um substring do termo sendo buscado, só trará resultados se o string de busca for menor que o string analisado, proveniente do banco de termos.

Baseado nisso, fez-se uma pequena modificação no algoritmo de busca, que compara agora sempre o string maior com o string menor, independente do string menor ser um termo do banco de dados e não o string de busca.

Essa modificação conseguiu aumentar um pouco a eficácia da busca, embora talvez fosse necessário um outro critério para maximizar os resultados (infelizmente não abordado pela implementação), uma vez que o tipo de busca usado depende largamente do tamanho dos strings, sendo que a quantidade de resultados obtidos é inversamente proporcional ao tamanho do string de busca.

7.5 Testes da implementação como um todo

Uma vez tendo testado todos os módulos individualmente, foi realizado um teste com todos os módulos sendo utilizados em conjunto. Novamente foi utilizado o projeto padrão que vêm junto com o Pootle, que na verdade é o projeto de tradução do próprio Pootle.

A presença da memória de tradução mostrou-se eficaz como o esperado, agilizando a tradução realizada. A interface dinâmica feita em Javascript também possibilitou uma tradução ágil, mesmo sendo necessário que no final da lista de strings contida na página, que toda a página fosse recarregada. Obviamente, a tradução de uma dezena de strings sem que seja necessária a recarga da página é muito mais eficaz que a tradução da mesma dezena recarregando a página a cada string.

O widget de mostra de contexto e busca de termos também mostraram-se eficazes, uma vez que estavam sempre visíveis durante o processo de tradução, podendo ser prontamente consultados.

Baseando-se nos testes descritos, a opinião do autor é de que a implementação realizada serviu ao seu propósito, por mais que este seja simples. Com as pequenas mudanças na interface do Pootle, este mostrou-se bem mais amigável, e a tradução tornou-se mais eficiente.

8 *Conclusões*

Neste capítulo, serão apresentadas as conclusões do autor, acerca das pesquisas feitas e da implementação realizada. Muitas conclusões reafirmam o que foi trazido na parte teórica deste trabalho, mostrando a consistência das teorias apresentadas.

Muito se falou neste trabalho sobre a importância da internacionalização e localização para a o desenvolvimento de software.

De tudo que foi estudado, chega-se a conclusão de que realmente a internacionalização é uma peça chave, que sustenta todo o processo de localização de software em si. Mais do que isso, a boa prática durante o processo de internacionalização garante o trabalho eficaz do tradutor, que tem acesso então a muito mais informação para realizá-lo.

Sobre localização é possível afirmar que a especialização do processo de tradução para realizar a localização de software trouxe muitos benefícios não só para os desenvolvedores como também para os tradutores. A interação com os desenvolvedores fez surgir uma gama muito rica de ferramentas que passaram a auxiliar o tradução não só na localização de software como na localização em geral.

Sobre o Software Livre pode-se afirmar que este trouxe de volta o espírito de colaboração que tinha fraquejado com a cada vez maior busca por lucros, promovida pelas empresas de software no início da década de 80.

Sobre as novas tecnologias presentes na web, conclui-se que elas promoveram ainda mais a colaboração, utilizando a rede como plataforma e a manipulação da informação como a motivação.

Sobre a implementação realizada, conclui-se que apesar das limitações impostas por falhas no planejamento em certas partes do projeto, o resultado final é satisfatório, alcançando seus objetivos.

8.1 Trabalhos Futuros

Como trabalhos futuros o autor sugere a total reformulação da interface de tradução do Pootle utilizando uma biblioteca que abstraia o uso de JavaScript pelo programador. Bibliotecas nesses moldes começaram a surgir com o lançamento do Google Web Toolkit¹, mas infelizmente o autor não encontrou nenhum exemplo de biblioteca que utilizasse Python como linguagem de programação durante sua pesquisa.

A criação de uma nova interface para a base de dados de termos também seria uma ótima proposta de trabalho posterior. Como citado várias vezes no trabalho, esse padrão de formato de arquivo têm muitas possibilidades interessantes que podem ser exploradas.

¹<http://code.google.com/webtoolkit/>

ANEXO A – Artigo

A.1 Abstract

This work approaches concepts about Software Internationalization and Localization and their importance in the process of Software Development. It presents also the importance of Free Software to the development of software in general, and the technologies that had contributed to the growth of Free Software, mainly the Internet. It also discourses the new technologies that use the internet and the World Wide Web as a development platform, and information as a motivation.

Finally, it explains the use of those concepts in the improvement of a web-based free software localization tool, aiming the development of a practical example of localization technology concepts, mainly the Translation Memory concepts.

Keywords: Internationalization, Localization, Free Software, World Wide Web, Translation Memory.

A.2 Introdução

A tecnologia de tradução, hoje, baseia-se principalmente em duas tecnologias (JUNIOR, 2004):

- Tradução por Máquina - também chamada de Tradução Automática, cujo propósito é a tradução de um texto de uma linguagem para outra, a princípio sem o auxílio humano.
- Tradução Assistida por Computador - tecnologia criada para auxiliar um tradutor humano em seu trabalho, facilitando e agilizando o processo.

A tradução assistida por computador (em inglês *Computer Aided/Assisted Translation*, abreviada CAT), também possui ramificações. Uma delas é a memória de tradução.

Memórias de tradução são basicamente banco de dados de frases ou termos já traduzidos, que permanecem na memória do computador para consulta durante todo o processo de tradução.

Usando memórias de tradução, um tradutor pode reaproveitar (total ou parcialmente) em novos trabalhos traduções já feitas anteriormente, o que aumenta a produtividade do profissional e a qualidade do trabalho final. Realizando o intercâmbio de memórias de tradução, tradutores promovem ainda mais o reuso e auxiliam-se mutuamente. Esse intercâmbio é realizado principalmente através da Internet.

A Internet é hoje o principal meio de intercâmbio de qualquer tipo de informação, seja esta de qualquer natureza. Ela incentivou e facilitou a colaboração entre grupos geograficamente distantes em prol de um bem comum. Foi nesse contexto que nasceu o movimento do Software Livre.

Um grupo de desenvolvedores liderados por Richard Stallman promovia o desenvolvimento de software e a disponibilização do seu código fonte (o texto em linguagem de programação que descreve o funcionamento de um software) livremente através da internet.

E assim como a internet colaborou para o desenvolvimento do software livre, o inverso também ocorreu. Hoje a grande maioria dos servidores existentes na rede rodam o sistema operacional GNU/Linux, o principal fruto do software livre.

A globalização da informação foi incentivada também por novas tecnologias aplicadas à *World Wide Web*, que permitiam que qualquer pessoa pudesse participar da disseminação da informação, através de sistemas de gerenciamento de informação baseados em web.

A.2.1 Objetivo do trabalho

Com base no exposto, o objetivo deste trabalho é a adaptação de um sistema web de tradução baseado em software livre já existente, disponibilizando na medida do possível todas as tecnologias existentes em aplicativos desktop de tradução, como memórias de tradução e glossários, utilizando-se das novas tecnologias web para oferecer ao usuário uma experiência de tradução rica e produtiva.

Através dessa adaptação, será possível obter um exemplo prático de como a utilização da tecnologia de memória de tradução auxilia o processo de localização. Sendo uma ferramenta livre, espera-se também contribuir indiretamente com a localização de outros

programas, proporcionando uma ferramenta melhor e mais amigável ao tradutor.

A.3 Internacionalização

Quando da criação do computador, as interfaces dos softwares desenvolvidos eram extremamente simples (embora grande parte fosse de softwares matemáticos bastante complexos para a época). As limitações de hardware na época restringiam bastante o que se podia fazer em termos de software. Grande parte do software desenvolvido era fruto de pesquisa científica e distribuído com seu código fonte.

Com acesso ao código-fonte, era perfeitamente realizável o processo de modificação do software. Se realmente necessário, seria realizada a tradução das mensagens de erro e interface simples para a língua local, mesmo que levasse algum tempo para aprender a lógica de funcionamento do software (STALLMAN, 1998).

A competição entre o crescente número de empresas de software nascidas nos anos 80 fez com que estas sentissem a necessidade de fecharem o código-fonte de seus programas, evitando assim que seus algoritmos fossem plagiados por empresas concorrentes.

E com a exportação dos softwares para outros países, sentiu-se a necessidade de criar código cuja lógica funcional fosse separada da apresentação das informações, enfim, da interface visível. Desse modo, o software poderia ser traduzido sem precisar dar ao tradutor o acesso ao código fonte, e esse tradutor não precisaria ter conhecimentos específicos de linguagens de programação.

Dessa necessidade surgiram as bibliotecas de internacionalização. Utilizadas pelos programadores, permitiam acessar o texto dos elementos da interfaces dos programas em arquivos separados do código fonte, os chamados arquivos de recursos (ESSELINK, 2000).

A relação da internacionalização com o software livre se dá principalmente da necessidade de que o software pode ser utilizado na língua nativa dos usuários. Utilizando bibliotecas de internacionalização livres, os desenvolvedores incentivaram os usuários a traduzirem seus software preferidos, num processo que é chamado de localização.

A.4 Localização

A Localização envolve a adaptação de todo o aspecto cultural (e, às vezes, também do significado), de um texto escrito, propaganda, livro, manual, ou em nosso caso, de um

software, para as necessidades de um dado local ou região específica.

A localização de software não internacionalizado é um processo muito difícil e propenso a erros, e exige do tradutor um conhecimento técnico maior do que o necessário. A internacionalização veio para corrigir este problema.

O software internacionalizado pode ser localizado em paralelo ao desenvolvimento. O tradutor tem mais liberdade para trabalhar e pode inclusive não ter muito conhecimento da linguagem de programação na qual o software é desenvolvido.

Os tradutores inclusive podem contar com o auxílio de outros softwares para realizar a localização, como as memórias de tradução, citadas na introdução deste artigo. As memórias de tradução são softwares de edição de texto que se valem da grande quantidade de repetição de frases e termos em textos técnicos para agilizar a tradução.

Utilizando funções para realizar busca aproximada de strings (um string é um tipo de dado que representa uma frase, nas linguagens de programação), que retornam como resultado strings que têm uma semelhança parcial com o string sendo buscado, numa porcentagem pré-definida.

Desse modo, é possível aproveitar traduções anteriores total ou parcialmente. Isso é especialmente útil quando se trata de localização de software, pois muitos termos dos elementos da interface aparecem em mais de uma vez, em locais diferentes.

Memórias de tradução e outros softwares semelhantes facilitaram em muitos aspectos o trabalho de localização de software. No mundo do software livre, permitiram que os próprios usuários pudessem traduzir seus programas preferidos, independente de saberem programar.

A.5 Tecnologias Web

Como já afirmado anteriormente, as tecnologias em uso na web reafirmaram seu status como fonte global de informação. Os protocolos e outros padrões disponíveis permitem o rápido intercâmbio e edição de informações.

O dinamismo da web começou quando as páginas de informações estáticas feitas em HTML foram substituídas por páginas criadas dinamicamente nos servidores. O usuário interagia através de elementos HTML padrões e o servidor "montava" uma página de resposta segundo as informações enviadas por esta interação (WIKIPEDIA, 2006q).

À medida que tráfego na Internet foi crescendo, o modelo cliente-servidor utilizado nos sistemas web foi se tornando um gargalo. Então, foram criadas novas formas para interação com o usuário, tentando transferir parte da inteligência da aplicação para o programa cliente.

Existem basicamente duas formas de fazer essa transferência: embutindo código interpretável pelo navegador dentro da estrutura do documento HTML (usando as chamadas linguagem de *scripting*), ou embutindo código executável, a ser interpretado por um programa a parte, que possui uma interface de comunicação com o navegador. Ambas as formas caracterizam uma nova modalidade de aplicações web chamada de *Rich Internet Applications* (WIKIPEDIA, 2006j).

A utilização de linguagens de *scripting*, como por exemplo JavaScript, é particularmente interessante pois elas permitem a criação de aplicações mais integradas ao navegador web, e conseqüentemente com uma execução mais ágil.

JavaScript propociona uma comunicação assíncrona com o servidor web através do objeto `XMLHttpRequest`. Fazendo uma requisição HTTP de modo assíncrono, a página que o navegador está exibindo não é afetada até que os dados tenham chegado do servidor e o processo de manipulação destes dados, que podem ser interpretados como texto puro ou XML, é transparente ao usuário.

Desse modo a aplicação web pode atualizar apenas partes do documento HTML com informações que obteve assincronamente do servidor, tornando a experiência web mais ágil.

A.6 O trabalho realizado

Com base na pesquisa feita sobre internacionalização, localização e tecnologias web, obteve-se uma idéia de aplicação para todos estes conceitos: uma aplicação web de tradução. E tomando como base uma aplicação web de tradução livre, partiu-se para aprimoramentos na sua interface de tradução utilizando a tecnologia das *Rich Internet Applications*.

O software de tradução web Pootle¹, escrito em Python² é uma alternativa totalmente livre ao gerenciamento de projetos de localização baseadas na biblioteca de internaciona-

¹<http://pootle.wordforge.org/>

²<http://www.python.org/>

lização Gettext³, largamente utilizada em projetos de software livre.

A interface de tradução do Pootle é derivada da primeira geração de aplicativos web, onde a interação do usuário com o aplicativo era realizada totalmente de modo síncrono.

Através da adição de alguns recursos, como uma memória de tradução, um banco de dados de termos, uma forma de mostrar mais informações do arquivo de recursos Gettext e a troca de informações assíncrona através de JavaScript conseguiu-se uma pequena melhora na usabilidade da interface do programa.

A memória de tradução, que utiliza como formato de persistência arquivos TMX⁴, que são baseados em XML e largamente usados na indústria da localização para a persistência de bancos de memórias de tradução, permite o intercâmbio de informações com outros aplicativos de localização. O banco de termos utiliza persistência o padrão TBX⁵, que é análogo ao TMX.

A mostra de informações contidas no arquivo de recursos Gettext visa trazer ao tradutor mais informações sobre o contexto da tradução sendo realizada. As informações que são trazidas ao tradutor são extraídas do próprio código fonte do software pelas ferramentas Gettext, e são uma maneira encontrada pelos desenvolvedores do Gettext para que os programadores possam dar mais informações aos tradutores.

A interface da página de tradução do Pootle foi então incrementada com recursos de comunicação assíncrona, que buscam as informações da memória de tradução e do banco de dados de termos sob demanda do usuário. Tudo isso de forma dinâmica, sem realizar a recarga total da página de tradução.

A.7 Conclusões

Da realização da pesquisa e implementações feitas neste trabalho reafirma-se a importância dos processos de internacionalização e localização no processo de desenvolvimento de software. Eles permitem que um software esteja disponível para outros usuários em seus idiomas nativos, facilitando o seu uso e aumentando sua aceitação.

Da utilização das novas tecnologias de aplicações web conclui-se que o dinamismo que elas trazem proporciona uma experiência muito melhor para o usuário, e comprova-se o papel fundamental que elas tiveram na disseminação da informação através da internet.

³www.gnu.org/software/gettext/

⁴<http://www.lisa.org/tmx>

⁵<http://www.lisa.org/tbx>

Aliando os conceitos de memórias de tradução e de novas tecnologias web conseguiu-se renovar a interface de uma aplicação já existente, proporcionando uma experiência de tradução muito mais rica e produtiva por parte dos tradutores.

ANEXO B – tmx1.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# Copyright 2006 Achilles Colombo Prudêncio
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with translate; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
#
# author: Achilles Colombo Prudêncio

"""module for parsing TMX translation memory files"""

from copy import deepcopy
from shutil import move
from xml.sax import make_parser
from xml.sax import parseString
from xml.sax.handler import ContentHandler
from xml.sax.handler import EntityResolver
from xml.sax.saxutils import escape
from xml.sax.saxutils import unescape
from xml.sax.saxutils import XMLGenerator
from xml.sax.xmlreader import AttributesImpl
from translate import __version__
import codecs
import os
import sys

class TMXHandler:
    """Class to handle TMX files."""
    # It actually does not handle the file itself, but it retrieves all things
    # that are (or supposed to be) useful and stores them in a dict in memory. The
    # structure of the dict itself is not regular, so I made a class to work with
    # it. I'm not sure right now how much memory it occupies, but it is lightning
    # fast. =D
    def __init__(self, filename, srclang=None):
        """Initialize a new TMXHandler, reading from filename if it exists."""
        self._filename = filename
        self._srclang = 'en'
        if srclang is not None:
            self._srclang = srclang
        self._tmxdict = {}
        if os.path.exists(filename):
            parser = make_parser()
            contenthandler = TMXReader()
            parser.setContentHandler(contenthandler)
            parser.setEntityResolver(FakeEntityResolver())
```

```

    parser.parse(filename)
    self._tmxdict = contenthandler.tmxdict
    # note that if the TMX file exists, the srclang param end up ignored...
    self._srclang = self._tmxdict['__header__']['_srclang_']

def savefile(self):
    """Save the internal TM data structure to a TMX file."""
    if os.path.exists(self._filename):
        # Standard algorithm to update a file: the original is renamed
        backupfilename = self._filename + '.bak'
        move(self._filename, backupfilename)
        # A new (updated) file is constructed based on the original
        try:
            parser = make_parser()
            contenthandler = TMXUpdater(self._filename, deepcopy(self._tmxdict))
            parser.setContentHandler(contenthandler)
            parser.setEntityResolver(FakeEntityResolver())
            parser.parse(backupfilename)
            # Once the new copy is finished, the old one is removed
            os.remove(backupfilename)
        except:
            print "An error occured, reversing changes. Details: " + str(sys.exc_info()[0]) + " " + str(sys.exc_info()[1])
            move(backupfilename, self._filename)
    else:
        writer = TMXWriter(self._filename)
        writer.writetmxdoc(self._tmxdict)

# -----Administration methods-----
# Some not needed/useful (IMO), but it's good to have them.
def getsrclang(self):
    """Returns TMX's source language."""
    return self._srclang

def setsrclang(self, newsrclang):
    """Sets TMX's source language.

Keyword arguments:
    newsrclang -- a (unicode) string defining the new srclang (iso) code

    """
    self._srclang = newsrclang.lower()
    if '__header__' in self._tmxdict.keys():
        self._tmxdict['__header__']['_srclang_'] = self._srclang
    else:
        self._tmxdict['__header__'] = { '_note_': [],
                                         '_prop_': {},
                                         '_srclang_': self._srclang }

def addheadernote(self, note):
    """Adds a new note to TMX's header.

Keyword arguments:
    note -- the note (unicode) string

    """
    if '__header__' in self._tmxdict.keys():
        if '_note_' in self._tmxdict['__header__'].keys():
            self._tmxdict['__header__']['_note_'].append(note)
        else:
            self._tmxdict['__header__']['_note_'] = [note]
    else:
        self._tmxdict['__header__'] = { '_note_': [note],
                                         '_prop_': {},
                                         '_srclang_': self._srclang }

def addheaderproperty(self, property, value):
    """Adds a new property to TMX's header.

Keyword arguments:
    property -- the (unicode) string property type
    value -- the (unicode) string property value

    """

```

```

if '__header__' in self._tmxdict.keys():
    if '__prop__' in self._tmxdict['__header__'].keys():
        self._tmxdict['__header__']['__prop__'][property] = value
    else:
        self._tmxdict['__header__']['__prop__'] = { property: value }
else:
    self._tmxdict['__header__'] = { '__prop__': { property: value },
                                    '__note__': [],
                                    '__srclang__': self._srclang }

def getheadernotes(self):
    """Returns a list with all TMX's header notes."""
    if '__header__' in self._tmxdict.keys():
        if '__note__' not in self._tmxdict['__header__'].keys():
            self._tmxdict['__header__']['__note__'] = []
        else:
            self._tmxdict['__header__'] = { '__note__': [],
                                            '__prop__': {},
                                            '__srclang__': self._srclang }
    return self._tmxdict['__header__']['__note__']

def getheaderproperties(self):
    """Returns a dict with all TMX's header properties."""
    if '__header__' in self._tmxdict.keys():
        if '__note__' not in self._tmxdict['__header__'].keys():
            self._tmxdict['__header__']['__prop__'] = {}
        else:
            self._tmxdict['__header__'] = { '__note__': [],
                                            '__prop__': {},
                                            '__srclang__': self._srclang }
    return self._tmxdict['__header__']['__prop__']

# Something to be aware off. The 'source string' I'm talking about it's
# literally the source string, generally in english, equivalent to the po's
# 'msgid'. Now, the 'target string' it's the translated string, equivalent to
# po's msgstr. A TMX file can have more than one target string.
# The tricky part here is that you can have notes at tu level (higher), and
# notes at tuv level (lower). When you add a note (or a property) on a tu, you
# probably want to add something that is true to the source string AND it's
# translations.
def addtunote(self, tuid, note):
    """Adds a new note to a Translation Unit.

    Keyword arguments:
        tuid -- the source (unicode) string
        note -- the note (unicode) string

    Raises ValueError if tuid not in tmxdict.

    """
    if tuid in self._tmxdict.keys():
        if '__note__' in self._tmxdict[tuid].keys():
            self._tmxdict[tuid]['__note__'].append(note)
        else:
            self._tmxdict[tuid]['__note__'] = [note]
        else:
            raise ValueError('TU inexistant: \'%s\'' % tuid)

def addtuproperty(self, tuid, property, value):
    """Adds a new property to a Translation Unit.

    Keyword arguments:
        tuid -- the source (unicode) string
        property -- the (unicode) string property type
        value -- the (unicode) string property value

    Raises ValueError if tuid not in tmxdict.

    """
    if tuid in self._tmxdict.keys():
        if '__prop__' in self._tmxdict[tuid].keys():
            self._tmxdict[tuid]['__prop__'][property] = value
        else:

```

```

        self._tmxdict[tuid]['__prop__'] = { property : value }
    else:
        raise ValueError('TU inexistant: \'%s\'' % tuid)

def gettunotes(self, tuid):
    """Returns a list with all the Translation Unit notes.

    Keyword arguments:
        tuid -- the source (unicode) string

    Raises ValueError if tuid not in tmxdict.

    """
    if tuid in self._tmxdict.keys():
        if '__note__' not in self._tmxdict[tuid].keys():
            self._tmxdict[tuid]['__note__'] = []
        else:
            raise ValueError('TU inexistant: \'%s\'' % tuid)
    return self._tmxdict[tuid]['__note__']

def gettuproperties(self, tuid):
    """Returns a dict with all the Translation Unit properties.

    Keyword arguments:
        tuid -- the source (unicode) string

    Raises ValueError if tuid not in tmxdict.

    """
    if tuid in self._tmxdict.keys():
        if '__prop__' not in self._tmxdict[tuid].keys():
            self._tmxdict[tuid]['__prop__'] = {}
        else:
            raise ValueError('TU inexistant: \'%s\'' % tuid)
    return self._tmxdict[tuid]['__prop__']

# When you add a tuv note or property, you wan't to add something particular
# to the source string OR one of it's translations. What can cause confusion
# here is that I'm indexing the tu by it's source string, but the source
# string itself is really stored inside a tuv. In TMX, a tu element does
# have a tuid attribute which can be used for indexing, but it's content is
# not defined (could be a word, or just a number, or whatever). Also that
# attribute is not required, so I just can't count on it.
def addtuvnote(self, tuid, translang, note):
    """Adds a new note to a Translation Unit Variant.

    Keyword arguments:
        tuid -- the source (unicode) string
        translang -- the language (iso) code (unicode) string
        note -- the note (unicode) string

    Raises ValueError if tuid not in tmxdict or if translang not in tmxdict[tuid].

    """
    if tuid in self._tmxdict.keys():
        if translang in self._tmxdict[tuid].keys():
            if '__note__' in self._tmxdict[tuid][translang].keys():
                self._tmxdict[tuid][translang]['__note__'].append(note)
            else:
                self._tmxdict[tuid][translang]['__note__'] = [note]
        else:
            raise ValueError('TUV inexistant: \'%s\'' % translang)
    else:
        raise ValueError('TU inexistant: \'%s\'' % tuid)

def addtuvproperty(self, tuid, translang, property, value):
    """Adds a new property to a Translation Unit Variant.

    Keyword arguments:
        tuid -- the source (unicode) string
        translang -- the language (iso) code (unicode) string
        property -- the (unicode) string property type
        value -- the (unicode) string property value

```

```

Raises ValueError if tuid not in tmxdict or if translang not in tmxdict[tuid].

"""
if tuid in self._tmxdict.keys():
    if translang in self._tmxdict[tuid].keys():
        if '__prop__' in self._tmxdict[tuid][translang].keys():
            self._tmxdict[tuid][translang]['__prop__'][property] = value
        else:
            self._tmxdict[tuid][translang]['__prop__'] = { property: value }
        else:
            raise ValueError('TUV inexistant: \'%s\'' % translang)
    else:
        raise ValueError('TU inexistant: \'%s\'' % tuid)

def gettuvnotes(self, tuid, translang):
    """Returns a list with all the Translation Unit Variant notes.

    Keyword arguments:
        tuid -- the source (unicode) string
        translang -- the language (iso) code (unicode) string

    Raises ValueError if tuid not in tmxdict or if translang not in tmxdict[tuid].

    """
    if tuid in self._tmxdict.keys():
        if translang in self._tmxdict[tuid].keys():
            if '__note__' not in self._tmxdict[tuid][translang].keys():
                self._tmxdict[tuid][translang]['__note__'] = []
            else:
                raise ValueError('TUV inexistant: \'%s\'' % translang)
        else:
            raise ValueError('TU inexistant: \'%s\'' % tuid)
    return self._tmxdict[tuid][translang]['__note__']

def gettuvproperties(self, tuid, translang):
    """Returns a dict with all the Translation Unit Variant properties.

    Keyword arguments:
        tuid -- the source (unicode) string
        translang -- the language (iso) code (unicode) string

    Raises ValueError if tuid not in tmxdict or if translang not in tmxdict[tuid].

    """
    if tuid in self._tmxdict.keys():
        if translang in self._tmxdict[tuid].keys():
            if '__prop__' not in self._tmxdict[tuid][translang].keys():
                self._tmxdict[tuid][translang]['__prop__'] = {}
            else:
                raise ValueError('TUV inexistant: \'%s\'' % translang)
        else:
            raise ValueError('TU inexistant: \'%s\'' % tuid)
    return self._tmxdict[tuid][translang]['__prop__']

# "OK, I can add things and retrieve all things I've added. How can I *delete*
# things?" Easy tiger, I'm working on it. But you can always hack.

# -----Translation methods-----
# TODO: Review this ugly algorithm.
def addtranslation(self, source, srclang, translation, translang):
    """Adds a new translation to the memory, or updates and existing one.

    Keyword arguments:
        source -- the source (unicode) string
        srclang -- the source language (iso) code (unicode) string
        translation -- the translation (unicode) string
        translang -- the translation language (iso) code (unicode) string

    """
    srclang = srclang.lower()
    translang = translang.lower()
    if source in self._tmxdict.keys():

```

```

    if srclang in self._tmxdict[source].keys(): # This *shouldn't* be needed
        self._tmxdict[source][srclang]['_str_'] = source
    else:
        self._tmxdict[source][srclang] = { '_note_': [],
                                             '_prop_': {},
                                             '_str_': source }

    if translang in self._tmxdict[source].keys():
        self._tmxdict[source][translang]['_str_'] = translation
    else:
        self._tmxdict[source][translang] = { '_note_': [],
                                             '_prop_': {},
                                             '_str_': translation }

    else:
        self._tmxdict[source] = { '_note_': [],
                                   '_prop_': {},
                                   srclang : { '_str_': source,
                                               '_note_': [],
                                               '_prop_': {} },
                                   translang : { '_str_': translation,
                                                '_note_': [],
                                                '_prop_': {} } }

def gettranslation(self, source, translang):
    """Retrieves the translation of the inputted segment.

    Keyword arguments:
        source -- the source (unicode) string
        translang -- the translation language (iso) code (unicode) string

    Raises ValueError if source not in tmxdict. If translang not in
    tmxdict[source], returns empty (unicode) string.

    """
    translang = translang.lower()
    if source in self._tmxdict.keys():
        if translang not in self._tmxdict[source].keys():
            self._tmxdict[source][translang] = { '_note_': [],
                                                  '_prop_': {},
                                                  '_str_': u'' }
        else:
            raise ValueError('TU inexistant: \'%s\'' % source)
        return self._tmxdict[source][translang]['_str_']

def getsources(self):
    """Returns a list with all source segments."""
    return self._tmxdict.keys()

def debug(self):
    from pprint import pprint
    pprint(self._tmxdict, indent=2)

class FakeEntityResolver(EntityResolver):
    """Class to bypass resolving of External Entities"""
    # As I'm NOT using the DTD information, I came up with this class to skip the
    # resolving of external entities.
    def resolveEntity(self, publicId, systemId):
        return os.tmpfile()

class TMXReader(ContentHandler):
    """Utility class to parse TMX files and return convenient data structure."""
    # important states
    HEADER = 1
    BODY = 2
    TU = 3
    TUV = 4
    SEG = 5
    HEADERNOTE = 6
    HEADERPROP = 7
    TUNOTE = 8
    TUPROP = 9
    TUVNOTE = 10
    TUVPROP = 11

```

```

def __init__(self):
    self.tmxdict = {}
    self.srclang = None
    self.state = None

def startElement(self, name, attributes):
    if name == 'header':
        if self.state != None:
            raise ValueError('Malformed TMX file: header element starts on invalid location.')
        srclang = attributes.get('srclang', 'en')
        self.srclang = srclang.lower()
        self._headernotes = []
        self._headerprops = {}
        self.state = self.HEADER
    if name == 'tu':
        if self.state != self.BODY:
            raise ValueError('Malformed TMX file: tu element starts on invalid location.')
        self._tuvdict = {}
        self._tunotes = []
        self._tuprops = {}
        self._curruid = None
        self.state = self.TU
    if name == 'tuv':
        if self.state != self.TU:
            raise ValueError('Malformed TMX file: tuv element starts on invalid location.')
        tuvlang = attributes.get('xml:lang', None)
        if not tuvlang:
            tuvlang = attributes.get('lang', None) # Compatibility with TMX 1.1
        if not tuvlang:
            raise ValueError('Malformed TMX file: no \'xml:lang\' attribute in tuv element.')
        self._tuvlang = tuvlang.lower()
        self._tuvnotes = []
        self._tuvprops = {}
        self.state = self.TUV
    if name == 'seg':
        if self.state != self.TUV:
            raise ValueError('Malformed TMX file: seg element starts on invalid location.')
        self._segtext = ''
        self.state = self.SEG
    if name == 'note':
        if self.state == self.HEADER:
            self.state = self.HEADERNOTE
        if self.state == self.TU:
            self.state = self.TUNOTE
        if self.state == self.TUV:
            self.state = self.TUVNOTE
    if name == 'prop':
        self._proptype = attributes.get('type', None)
        if self._proptype == None:
            raise ValueError('Malformed TMX file: no \'type\' attribute in prop element.')
        if self.state == self.HEADER:
            self.state = self.HEADERPROP
        if self.state == self.TU:
            self.state = self.TUPROP
        if self.state == self.TUV:
            self.state = self.TUVPROP
    if name in ['bpt', 'ept', 'it', 'ph', 'ut']:
        self._segtext += '<' + name + ' '
        for key, value in attributes.items():
            self._segtext += key + '=' + value + '\n '
        self._segtext = self._segtext[:-1]
        self._segtext += '>'

def endElement(self, name):
    if name == 'header':
        self.tmxdict['_header_'] = { '_note_': self._headernotes,
                                   '_prop_': self._headerprops,
                                   '_srclang_': self.srclang }
        self.state = self.BODY
    if name == 'tu':
        if self._curruid == None:
            raise ValueError('Malformed TMX file: no tuv were xml:lang equals srclang.')
        self.tmxdict[self._curruid] = self._tuvdict

```



```

        self.tmxdict[self._curruid]['__note__'] = self._tunotes
        self.tmxdict[self._curruid]['__prop__'] = self._tuprops
        self.state = self.BODY
    if name == 'tuv':
        self._tuvdict[self._tuvlang] = { '__note__': self._tunotes,
                                         '__prop__': self._tuprops,
                                         '__str__': self._segtext }

        if self._tuvlang == self.srclang:
            self._curruid = self._segtext
            self.state = self.TU
    if name == 'seg':
        self.state = self.TUV
    if name == 'note':
        if self.state == self.HEADERNOTE:
            self.state = self.HEADER
        if self.state == self.TUNOTE:
            self.state = self.TU
        if self.state == self.TUVNOTE:
            self.state = self.TUV
    if name == 'prop':
        if self.state == self.HEADERPROP:
            self.state = self.HEADER
        if self.state == self.TUPROP:
            self.state = self.TU
        if self.state == self.TUVPROP:
            self.state = self.TUV
    if name in ['bpt', 'ept', 'it', 'ph', 'ut']:
        self._segtext += '</' + name + '>'

def characters(self, text):
    if self.state == self.SEG:
        self._segtext += escape(text)
    if self.state == self.HEADERNOTE:
        self._headernotes.append(text)
    if self.state == self.HEADERPROP:
        self._headerprops[self._proptype] = text
    if self.state == self.TUNOTE:
        self._tunotes.append(text)
    if self.state == self.TUPROP:
        self._tuprops[self._proptype] = text
    if self.state == self.TUVNOTE:
        self._tuvnotes.append(text)
    if self.state == self.TUVPROP:
        self._tuvprops[self._proptype] = text

class TMXUpdater(ContentHandler):
    """Utility class to update TMX files based on the convenient data structure.

    It is assumed that the structure passed corresponds to an updated version of
    original file.

    """
    # important states
    HEADER = 1
    TU = 2
    TUV = 3
    SEG = 4
    HEADERNOTE = 5
    HEADERPROP = 6
    TUNOTE = 7
    TUPROP = 8
    TUVNOTE = 9
    TUVPROP = 10

    def __init__(self, tmxfile, tmxdict):
        self.tmxdict = tmxdict
        self.srclang = self.tmxdict['__header__']['__srclang__']
        del self.tmxdict['__header__']['__srclang__'] # I delete stuff from the dict, trying to lower memory usage
        self.outfile = open(tmxfile, 'w')
        self._generator = XMLGenerator(self.outfile, 'utf-8')
        self._tmxwriter = TMXWriter(self.outfile)
        self.state = None

```

```

def startDocument(self):
    self._generator.startDocument()
    self.outfile.write('<!DOCTYPE tmx SYSTEM "tmx14.dtd">\n') # Reaaaly forced solution...

def endDocument(self):
    self._generator.endDocument()

def startElement(self, name, attributes):
    if name == 'tmx':
        self._generator.startElement(name, attributes)
        self._generator.characters('\n')
    if name == 'body':
        self._generator.characters(' ')
        self._generator.startElement(name, attributes)
        self._generator.characters('\n')
    if name == 'header':
        self._headerattrs = attributes
        self._headernotes = {} # { notetext : AttributesImpl }
        self._headerprops = {} # { prop : ( value, AttributesImpl ) }
        self.state = self.HEADER
    if name == 'tu':
        self._tuattrs = attributes
        self._tunotes = {} # { notetext : AttributesImpl }
        self._tuprops = {} # { prop : ( value, AttributesImpl ) }
        self._tuvdict = {} # { lang : list( str, self._tuvnotes, self._tuvprops, AttributesImpl ) }
        self.state = self.TU
    if name == 'tuv':
        self._tuvattrs = attributes
        self._tuvnotes = {} # { notetext : AttributesImpl }
        self._tuvprops = {} # { prop : ( value, AttributesImpl ) }
        self.state = self.TUV
    if name == 'seg':
        self._segtext = u''
        self.state = self.SEG
    if name == 'note':
        self._noteattrs = attributes
        if self.state == self.HEADER:
            self.state = self.HEADERNOTE
        if self.state == self.TU:
            self.state = self.TUNOTE
        if self.state == self.TUV:
            self.state = self.TUVNOTE
    if name == 'prop':
        self._propattrs = attributes
        if self.state == self.HEADER:
            self.state = self.HEADERPROP
        if self.state == self.TU:
            self.state = self.TUPROP
        if self.state == self.TUV:
            self.state = self.TUVPROP
    if name in ['bpt', 'ept', 'it', 'ph', 'ut']:
        self._segtext += '<' + name + ' '
        for key, value in attributes.items():
            self._segtext += key + '=' + value + '\n'
        self._segtext = self._segtext[:-1]
        self._segtext += '>'

def endElement(self, name):
    if name == 'tmx':
        self._generator.endElement(name)
        self._generator.characters('\n')
    if name == 'header':
        self._headernotes = self._updatenotes(self._headernotes, self.tmxdict['__header__']['__note__'])
        self._headerprops = self._updateprops(self._headerprops, self.tmxdict['__header__']['__prop__'])
        self._tmxwriter.writeheader(self._headernotes, self._headerprops, self._headerattrs, 1)
        del self.tmxdict['__header__']
    if name == 'body':
        for tu in self.tmxdict.keys(): # writing new tus added
            tuu = self._tmxwriter.buildtuforwriting(self.tmxdict[tu])
            self._tmxwriter.writetu(tuu[0], tuu[1], tuu[2], tuu[3], 2)
            del self.tmxdict[tu]
        self._generator.characters(' ')
        self._generator.endElement('body')

```

```

self._generator.characters('\n')
if name == 'tu':
    tuindex = self._tuvdict[self.srclang][0]
    self._tunotes = self._updatenotes(self._tunotes, self.tmxdict[tuindex]['__note__'])
    self._tuprops = self._updateprops(self._tuprops, self.tmxdict[tuindex]['__prop__'])
    for lang in self._tuvdict:
        self._tuvdict[lang][1] = self._updatenotes( self._tuvdict[lang][1],
                                                    self.tmxdict[tuindex][lang]['__note__'])
        self._tuvdict[lang][2] = self._updateprops( self._tuvdict[lang][2],
                                                    self.tmxdict[tuindex][lang]['__prop__'])
    del self.tmxdict[tuindex][lang]
for lang in self.tmxdict[tuindex]: # new tuvs added
    if lang == '__note__' or lang == '__prop__':
        continue
    tuvnotes = {}
    for note in self.tmxdict[tuindex][lang]['__note__']:
        tuvnotes[note] = AttributesImpl({})
    tuvprops = {}
    for prop, value in self.tmxdict[tuindex][lang]['__prop__']:
        tuvprops[prop] = AttributesImpl({'type': value})
    tuvattrs = AttributesImpl({'xml:lang': lang})
    self._tuvdict[lang.lower()] = [self.tmxdict[tuindex][lang]['__str__'], tuvnotes, tuvprops, tuvattrs]
self._tmxwriter.writetu(self._tuvdict, self._tunotes, self._tuprops, self._tuattrs, 2)
del self.tmxdict[tuindex]
if name == 'tuv':
    tuvlang = self._tuvattrs.get('xml:lang', None)
    if tuvlang == None:
        tuvlang = self._tuvattrs.get('lang', None)
    if tuvlang == None:
        raise ValueError('Malformed TMX file: no \'xml:lang\' attribute in tuv element.')
    self._tuvdict[tuvlang.lower()] = [self._segtext, self._tuvnotes, self._tuvprops, self._tuvattrs]
if name == 'seg':
    self.state = self.TUV
if name in ['bpt', 'ept', 'it', 'ph', 'ut']:
    self._segtext += '</' + name + '>'

def characters(self, text):
    if self.state == self.SEG:
        self._segtext += escape(text)
    if self.state == self.HEADERNOTE:
        self._headernotes[text] = self._noteattrs
        self.state = self.HEADER
    if self.state == self.HEADERPROP:
        prop = self._propattrs.get('type', None)
        if prop == None:
            raise ValueError('Malformed TMX file: no \'type\' attribute in prop element.')
        self._headerprops[prop] = (text, self._propattrs)
        self.state = self.HEADER
    if self.state == self.TUNOTE:
        self._tunotes[text] = self._noteattrs
        self.state = self.TU
    if self.state == self.TUPROP:
        prop = self._propattrs.get('type', None)
        if prop == None:
            raise ValueError('Malformed TMX file: no \'type\' attribute in prop element.')
        self._tuprops[prop] = (text, self._propattrs)
        self.state = self.TU
    if self.state == self.TUVNOTE:
        self._tuvnotes[text] = self._noteattrs
        self.state = self.TUV
    if self.state == self.TUVPROP:
        prop = self._propattrs.get('type', None)
        if prop == None:
            raise ValueError('Malformed TMX file: no \'type\' attribute in prop element.')
        self._tuvprops[prop] = (text, self._propattrs)
        self.state = self.TUV

def _updatenotes(self, oldnotes, newnotes):
    """Updates a set of notes.

    Keyword arguments:
    oldnotes -- a dict( notetext : AttributesImpl )
    newnotes -- a list() of notes

```

```

Returns:
    a dict( notetext : AttributesImpl )

"""
finalnotes = {}
for note in newnotes:
    if note in oldnotes.keys():
        finalnotes[note] = oldnotes[note] # preserving old note attributes
    else:
        finalnotes[note] = AttributesImpl({}) # a new note has been added
return finalnotes

def _updateprops(self, oldprops, newprops):
    """Updates a set of props.

    Keyword arguments:
        oldprops -- a dict( prop : ( value, AttributesImpl ) )
        newprops -- a dict( prop : value )
    Returns:
        a dict( prop : ( value, AttributesImpl ) )

    """
    finalprops = {}
    for prop, value in newprops.items():
        if prop in oldprops.keys():
            finalprops[prop] = (value, oldprops[prop][1])
        else:
            finalprops[prop] = (value, AttributesImpl({'type': prop}))
    return finalprops

class TMXWriter:
    """Class to write a plain new TMX file, based on the convenient data structure.

    It also provide methods to write TMX elements, used by TMXUpdate. In case of
    TMX level 2, the segments must already have the inline tags. This class does
    not know how to process internal formatting, only how to write it.

    """
    def __init__(self, tmxfile): # Maybe not the best approach... Any ideas?
        """Initialize the class.

        Keyword arguments:
            tmxfile -- a file object or string with object filename

        """
        if isinstance(tmxfile, file):
            self.outfile = tmxfile
            self._generator = XMLGenerator(self.outfile, 'utf-8')
        elif isinstance(tmxfile, str):
            self.outfile = open(tmxfile, 'w')
            self._generator = XMLGenerator(self.outfile, 'utf-8')
        else:
            raise ValueError('tmx arg should be a file with writing mode on or a string object.')
        self._segwriter = TMXSegWriter(self._generator)

    def writeheader(self, headernotes, headerprops, headerattrs, indent):
        """Writes the header element properly.

        Keyword arguments:
            headernotes -- a dict( note: AttributesImpl )
            headerprops -- a dict( prop: ( value, AttributesImpl ) )
            headerattrs -- AttributesImpl

        """
        self._generator.characters(' ' * indent)
        self._generator.startElement('header', headerattrs)
        self._generator.characters('\n')
        newindent = indent + 1
        for note in headernotes:
            self.writeelement('note', note, headernotes[note], newindent)
            self._generator.characters('\n')
        for prop in headerprops:
            self.writeelement('prop', headerprops[prop][0], headerprops[prop][1], newindent)

```

```

        self._generator.characters('\n')
self._generator.characters(' ' * indent)
self._generator.endElement('header')
self._generator.characters('\n')

def writetu(self, tuvdict, tunotes, tuprops, tuattrs, indent):
    """Writes a tu element properly.

    Keyword arguments:
        tuvdict -- a dict( lang: (str, tuvnotes, tuvprops, tuvattrs) )
        tunotes -- a dict( note: AttributesImpl )
        tuprops -- a dict( prop: ( value, AttributesImpl ) )
        tuattrs -- AttributesImpl
        indent -- indentation level

    """
    self._generator.characters(' ' * indent)
    self._generator.startElement('tu', tuattrs)
    self._generator.characters('\n')
    newindent = indent + 1
    for note in tunotes:
        self.writeelement('note', note, tunotes[note], newindent)
        self._generator.characters('\n')
    for prop in tuprops:
        self.writeelement('prop', tuprops[prop][0], tuprops[prop][1], newindent)
        self._generator.characters('\n')
    for tuv in tuvdict.values():
        self.writetuv(tuv[0], tuv[1], tuv[2], tuv[3], newindent)
    self._generator.characters(' ' * indent)
    self._generator.endElement('tu')
    self._generator.characters('\n')

def writetuv(self, str, tuvnotes, tuvprops, tuvattrs, indent):
    """Writes a tuv element properly.

    Keyword arguments:
        str -- the segment (unicode) string
        tuvnotes -- a dict( note: AttributesImpl )
        tuvprops -- a dict( prop: ( value, AttributesImpl ) )
        tuvattrs -- AttributesImpl
        indent -- indentation level

    """
    self._generator.characters(' ' * indent)
    self._generator.startElement('tuv', tuvattrs)
    self._generator.characters('\n')
    newindent = indent + 1
    for note in tuvnotes:
        self.writeelement('note', note, tuvnotes[note], newindent)
        self._generator.characters('\n')
    for prop in tuvprops:
        self.writeelement('prop', tuvprops[prop][0], tuvprops[prop][1], newindent)
        self._generator.characters('\n')
    self._generator.characters(' ' * newindent)
    seg = u'<seg>' + str + u'</seg>'
    parseString(seg.encode('utf-8'), self._segwriter) # maybe calling encode is dangerous??!
    self._generator.characters('\n')
    self._generator.characters(' ' * indent)
    self._generator.endElement('tuv')
    self._generator.characters('\n')

def writeelement(self, name, text, attributes, indent):
    """Writes an TMX element properly.

    Keyword arguments:
        name -- element name
        text -- element text
        attributes -- an AttributesImpl object, with all element attributes.
        indent -- indentation level

    """
    self._generator.characters(' ' * indent)
    self._generator.startElement(name, attributes)

```

```

self._generator.characters(text)
self._generator.endElement(name)

def writetmxdoc(self, tmxdict):
    self._generator.startDocument()
    self._generator.startElement('tmx', AttributesImpl({'version' : '1.4'}))
    self._generator.characters('\n')
    self.outfile.write('<!DOCTYPE tmx SYSTEM "tmx14.dtd">\n') # Reaaaly forced solution...
    headerattrs = AttributesImpl({ 'creationtool' : 'translate',
                                   'creationtoolversion' : __version__.ver,
                                   'datatype' : 'unknown',
                                   'segtype' : 'sentence',
                                   'adminlang' : 'en-us',
                                   'srclang' : tmxdict['__header__']['__srclang__'],
                                   'o-tmf' : 'Translate TMX convenient data structure'})

    headernotes = {}
    for note in tmxdict['__header__']['__note__']:
        headernotes[note] = AttributesImpl({})
    headerprops = {}
    for prop, value in tmxdict['__header__']['__prop__'].items():
        headernotes[prop] = (value, AttributesImpl({'type': prop}))
    self.writeheader(headernotes, headerprops, headerattrs, 1)
    del tmxdict['__header__']
    self._generator.characters(' ')
    self._generator.startElement('body', AttributesImpl({}))
    self._generator.characters('\n')
    for tu in tmxdict.values():
        tu = self.buildtuforwriting(tu)
        self.writetu(tu[0], tu[1], tu[2], tu[3], 2)
    self._generator.characters(' ')
    self._generator.endElement('body')
    self._generator.characters('\n')
    self._generator.endElement('tmx')
    self._generator.endDocument()

def buildtuforwriting(self, tu):
    """Build the attributes necessary for writing a new (added) tu.

    Keyword arguments:
        tu -- a dict( __note__ - tu notes
                    __prop__ - tu props
                    lang-1 : __note__ - tuv notes
                    __prop__ - tuv props
                    __str__ - tuv string
                    ...
                    lang-n : __note__ - tuv notes
                    __prop__ - tuv props
                    __str__ - tuv string
        )

    Returns:
        a tuple( dict( str,
                    dict( tuvnote : AttributesImpl ),
                    dict( tuvprop : ( value, AttributesImpl ),
                        AttributesImpl ),
                    dict( tunote : AttributesImpl ),
                    dict( tuprop : ( value, AttributesImpl ),
                        AttributesImpl )

    This function must be called for new tus added in a tmxdict.

    """
    tuattrs = AttributesImpl({'creationtool' : 'translate',
                              'creationtoolversion' : __version__.ver}) # resetting tuattrs
    tunotes = {}
    for note in tu['__note__']:
        tunotes[note] = AttributesImpl({})
    del tu['__note__']
    tuprops = {}
    for prop, value in tu['__prop__'].items():
        tuprops[prop] = (value, AttributesImpl({'type': prop}))
    del tu['__prop__']
    tuvdict = {}
    for lang in tu: # as I deleted __note__ and __prop__, only langs are left.

```

```

tuvnotes = {}
for note in tu[lang]['__note__']:
    tuvnotes[note] = AttributesImpl({})
tuvprops = {}
for prop, value in tu[lang]['__prop__'].items():
    tuvprops[prop] = (value, AttributesImpl({'type': prop}))
tuvdict[lang] = [ tu[lang]['__str__'],
                 tuvnotes,
                 tuvprops,
                 AttributesImpl({'xml:lang' : lang,
                                 'creationtool' : 'translate',
                                 'creationtoolversion' : __version__})]
return (tuvdict, tunotes, tuprops, tuattrs)

class TMXSegWriter(ContentHandler):
    """Small parser to write TMX seg element (and its subelements) properly.

    You can see in the code that it really doesn't do anything unusual. What
    happent is that I fail to see a way to put this logic inside the "big"
    parsers.

    """
    def __init__(self, tmxgenerator):
        """Initialize the class.

        Keyword arguments:
        tmxgenerator -- a xml.sax.saxutils.XMLGenerator object

        """
        if isinstance(tmxgenerator, XMLGenerator):
            self._generator = tmxgenerator
        else:
            raise ValueError('tmxgenerator arg should be a XMLGenerator object.')

    def startElement(self, name, attributes):
        if name in ['seg', 'bpt', 'ept', 'it', 'ph', 'ut', 'hi', 'foreign', 'p', 'term']:
            self._generator.startElement(name, attributes)

    def endElement(self, name):
        if name in ['seg', 'bpt', 'ept', 'it', 'ph', 'ut', 'hi', 'foreign', 'p', 'term']:
            self._generator.endElement(name)

    def characters(self, text):
        self._generator.characters(text)

```

ANEXO C – tmx_test.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#
# Copyright 2006 Achilles Colombo Prudêncio
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with translate; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
#
# author: Achilles Colombo Prudêncio

from translate.storage import tmx1

print "Instanciando tratador TMX usando um arquivo existente como parâmetro..."

tmxh = tmx1.TMXHandler("./tmxsample1.tmx")

print "\nPronto. Fazendo atualizações aleatórias...\n"

tmxh.addheadernote(u"Testando header...")

print tmxh.getheadernotes()

for srckey in tmxh.getsources():
    tmxh.addtranslation(srckey, tmxh.getsrclang(), u"Testing.", u"pt-pt")
    tmxh.addtuvnote(srckey, u"pt-pt", u"Nota")

tmxh.addtranslation(u"Testing", tmxh.getsrclang(), u"Testando.", u"pt-pt")

print "\nPronto. Salvando arquivo..."

tmxh.savefile()

print "\nPronto. Recarregando arquivo..."

tmxh = tmx1.TMXHandler("tmxsample1.tmx")

print "\nPronto. Teste finalizado."
```


ANEXO D – tbx1.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# Copyright 2006 Achilles Colombo Prudêncio
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with translate; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
#
# author: Achilles Colombo Prudêncio

# allota copy&paste from tmx1.py; Oh dear...

"""module for parsing TBX term base exchange files"""

from copy import deepcopy
from shutil import move
from xml.sax import make_parser
from xml.sax import parseString
from xml.sax.handler import ContentHandler
from xml.sax.saxutils import escape
from xml.sax.saxutils import unescape
from xml.sax.saxutils import XMLGenerator
from xml.sax.xmlreader import AttributesImpl
from translate import __version__
from translate.storage.tmx1 import FakeEntityResolver
from translate.storage.tmx1 import TMXSegWriter # Modified and used to write TBX inline elements
import codecs
import os
import sys

class TBXHandler:
    """Class to handle TMX files."""
    # Please read comments in tmx1.py first =D

    def __init__(self, filename, srclang=None):
        """Initialize a new TBXHandler, reading from filename if it exists."""
        self._filename = filename
        self._srclang = 'en'
        if srclang is not None:
            self._srclang = srclang
        self._tbxdict = {}
        if os.path.exists(filename):
            parser = make_parser()
            contenthandler = TBXReader()
            parser.setContentHandler(contenthandler)
            parser.setEntityResolver(FakeEntityResolver())
```

```

    parser.parse(filename)
    self._tbxdict = contenthandler.tbxdict
    # note that if the TMX file exists, the srclang param end up ignored...
    self._srclang = self._tbxdict['_srclang_']

def getsrclang(self):
    return self._srclang

def setsrclang(self, srclang):
    self._srclang = srclang
    self._tbxdict['_srclang_'] = srclang

def savefile(self):
    """Write file to disk."""
    if '_srclang_' not in self._tbxdict:
        self._tbxdict['_srclang_'] = self._srclang
    if '_header_' not in self._tbxdict:
        self.addsourcedesc(u'No description given.')
    writer = TBXWriter(self._filename)
    writer.writelbdoc(self._tbxdict)

def addsourcedesc(self, desc):
    """Adds a new description to header.

    Keyword arguments:
        desc -- unicode text of the description, with (or without) inline formatting

    """
    if '_header_' in self._tbxdict:
        if '_filedesc_' in self._tbxdict['_header_']:
            if desc not in self._tbxdict['_header_']['_filedesc_']:
                self._tbxdict['_header_']['_filedesc_'].append(desc)
            else:
                self._tbxdict['_header_']['_filedesc_'] = [ desc ]
        else:
            self._tbxdict['_header_'] = { '_filedesc_': [ desc ] }

def getsourcedesc(self):
    """Returns source descriptions"""
    if '_header_' in self._tbxdict:
        if '_filedesc_' not in self._tbxdict['_header_']:
            self._tbxdict['_header_']['_filedesc_'] = []
        else:
            self._tbxdict['_header_'] = { '_filedesc_': [] }
    return self._tbxdict['_header_']['_filedesc_']

def addterm(self, source, srclang, translation, translang):
    """Adds a new term to the termbase.

    Keyword arguments:
        source -- the source (unicode) string
        srclang -- the source language (iso) code (unicode) string
        translation -- the translation (unicode) string
        translang -- the translation language (iso) code (unicode) string

    """
    srclang = srclang.lower()
    translang = translang.lower()
    for source in self._tbxdict.keys():
        if source == '_header_' or source == '_srclang_':
            continue
        if translang in self._tbxdict[source]:
            if translation not in self._tbxdict[source][translang]:
                self._tbxdict[source][translang].append(translation)
            else:
                self._tbxdict[source][translang] = [ translation ]
        else:
            self._tbxdict[source] = { srclang : [ source ], translang : [ translation ] }

def gettermtranslation(self, source, translang):
    """Get translations for a given term. If translang equals srclang, get synonyms.

    Keyword arguments:

```

```

source -- the source (unicode) string
translang -- the translation language (iso) code (unicode) string

"""
translang = translang.lower()
if source in self._tbxdict:
    if translang not in self._tbxdict[source]:
        self._tbxdict[source][translang] = []
    else:
        self._tbxdict[source] = { srclang : [ source ] , translang : [] }
return self._tbxdict[source][translang]

def getsources(self):
    """Returns a list with all source segments."""
    return self._tbxdict.keys()

def debug(self):
    from pprint import pprint
    pprint(self._tbxdict, indent=2)

class TBXReader(ContentHandler):
    """Utility class to parse TBX files and return a convenient data structure"""
    # (maybe not so) important states
    HEADER = 1
    HEADCONTENT = 2
    FILEDESC = 3
    SRCDESC = 4
    P = 5
    INLINE = 6
    PREBODY = 7
    BODY = 8
    TERMENTRY = 9
    TENTRYCONTENT = 10
    LANGSET = 11
    TIG = 12
    TIGEND = 13
    NTIG = 14
    NTIGEND = 15
    TERM = 16
    TERMEND = 17
    TERMGRP = 18
    TGRPENDING = 19

    def __init__(self):
        self.tbxdict = {}
        self.srclang = None
        self.state = None
        self._laststate = []

    def getsrclang(self):
        if '__srclang__' in self.tbxdict:
            return self.tbxdict['__srclang__']
        else:
            raise ValueError('srclang not declared yet.')
```

```

    def setsrclang(self, srclang):
        if '__srclang__' in self.tbxdict:
            self.tbxdict['__srclang__'] = srclang.lower()
        else:
            raise ValueError('srclang not declared yet.')
```

```

    def startElement(self, name, attributes):
        if name == 'martif':
            # lang we can guess, but if type is not TBX we just can't know if we really have a TBX file or not
            type = attributes.get('type', None)
            if type.lower() != 'tbx' or type == None:
                raise ValueError('Malformed TBX file: no (or invalid) \'type\' declaration in martif element.')
            srclang = attributes.get('xml:lang', 'en')
            self.tbxdict = { '__srclang__': srclang.lower() }
            self.state = self.HEADER
        if name == 'martifHeader':
            if self.state != self.HEADER:
                raise ValueError('Malformed TBX file: martifHeader declared on invalid location in the file.')
```

```

self.tbxdict['_header_'] = {}
self.state = self.HEADCONTENT
if name == 'fileDesc':
    if self.state == self.PREBODY:
        raise ValueError('Malformed TBX file: there should be only one fileDesc declared in the file.')
    if self.state != self.HEADCONTENT:
        raise ValueError('Malformed TBX file: fileDesc declared on invalid location in the file.')
    self.tbxdict['_header_']['_filedesc_'] = []
    self.state = self.FILEDESC
if name == 'sourceDesc':
    if self.state != self.FILEDESC:
        raise ValueError('Malformed TBX file: sourceDesc declared on invalid location in the file.')
    self._srcdesc = []
    self.state = self.SRCDESC
if name == 'p':
    self._text = ""
    self._laststate.append(self.state)
    self.state = self.P
if name == 'text':
    if self.state != self.PREBODY:
        raise ValueError('Malformed TBX file: text declared on invalid location in the file.')
    self.state = self.BODY
if name == 'body':
    if self.state != self.BODY:
        raise ValueError('Malformed TBX file: body declared on invalid location in the file.')
    self.state = self.TERMENTRY
if name == 'termEntry':
    if self.state != self.TERMENTRY:
        raise ValueError('Malformed TBX file: termEntry declared on invalid location in the file.')
    self._currtermid = None
    self._currtermdict = {}
    self.state = self.TENTRYCONTENT
if name == 'langSet':
    if self.state != self.TENTRYCONTENT:
        raise ValueError('Malformed TBX file: langSet declared on invalid location in the file.')
    lang = attributes.get('xml:lang', None)
    if lang == None:
        raise ValueError('Malformed TBX file: no \'xml:lang\' declared in langSet.')
    self._currlang = lang.lower()
    self._currterms = []
    self.state = self.LANGSET
if name == 'tig':
    if self.state != self.LANGSET and self.state != self.TIGEND:
        raise ValueError('Malformed TBX file: tig declared on invalid location in the file.')
    self._text = ""
    self.state = self.TIG
if name == 'ntig':
    if self.state != self.LANGSET and self.state != self.NTIGEND:
        raise ValueError('Malformed TBX file: ntig declared on invalid location in the file.')
    self._text = ""
    self.state = self.NTIG
if name == 'termGrp':
    if self.state != self.NTIG:
        raise ValueError('Malformed TBX file: termGrp declared on invalid location in the file.')
    self.state = self.TERMGRP
if name == 'term':
    if self.state != self.TIG and self.state != self.TERMGRP:
        raise ValueError('Malformed TBX file: term declared on invalid location in the file.')
    self._currtermtext = ''
    self.state = self.TERM
if name in ['foreign', 'hi', 'bpt', 'ept', 'it', 'ph', 'ut']:
    self._laststate.append(self.state)
    self.state = self.INLINE
    self._text += '<' + name + ' '
    for key, value in attributes.items():
        self._text += key + '=' + value + ' '
    self._text = self._text[:-1]
    self._text += '>'

def endElement(self, name):
    if name == 'fileDesc':
        self.state = self.PREBODY
    if name == 'sourceDesc':

```

```

    if len(self._srcdesc) < 1:
        raise ValueError('Malformed TBX file: sourceDesc is empty.')
    self.tbxdict['_header_']['_filedesc_'].extend(self._srcdesc)
    self.state = self.FILEDESC
if name == 'p':
    self.state = self._laststate.pop()
    if self.state == self.SRCDESC:
        self._srcdesc.append(self._text)
if name == 'body':
    if len(self.tbxdict) == 2 and '_header_' in self.tbxdict and '_srcclang_' in self.tbxdict:
        raise ValueError('Malformed TBX file: no termEntry declared in the file.')
if name == "termEntry":
    if self._currtermid == None:
        raise ValueError('Malformed TBX file: no langSet with file\'s source language.')
    self.tbxdict[self._currtermid] = self._currtermdict
    self.state = self.TERMENTRY
if name == 'langSet':
    self._currtermdict[self._currlang] = self._currterms
    self._currterms.sort()
    if self._currlang == self.getsrcclang():
        self._currtermid = self._currterms[0]
    self.state = self.TENTRYCONTENT
if name == 'tig':
    self._currterms.append(self._currtermtext)
    self.state = self.TIGEND
if name == 'ntig':
    self._currterms.append(self._currtermtext)
    self.state = self.NTIGEND
if name == 'termGrp':
    self.state = self.TGRPENDING
if name == 'term':
    self._currtermtext = self._text
    self.state = self.TERMENDING
if name in ['foreign', 'hi', 'bpt', 'ept', 'it', 'ph', 'ut']:
    self._text += '</' + name + '>'
    self.state = self._laststate.pop()

def characters(self, text):
    if self.state == self.P:
        self._text += text
    if self.state == self.INLINE:
        self._text += escape(text)
    if self.state == self.TERM:
        self._text += text

class TBXWriter:
    """Utility class to write TBX files based on the convenient data structure"""

    def __init__(self, tbxfile):
        """Initialize the class.

        Keyword arguments:
            tbxfile -- a file object or string with object filename

        """
        if isinstance(tbxfile, file):
            self.outfile = tbxfile
            self._generator = XMLGenerator(self.outfile, 'utf-8')
        elif isinstance(tbxfile, str):
            self.outfile = open(tbxfile, 'w')
            self._generator = XMLGenerator(self.outfile, 'utf-8')
        else:
            raise ValueError('tmx arg should be a file with writing mode on or a string object.')
        self._segwriter = TMXSegWriter(self._generator)

    def writetbxdoc(self, tbxdict):
        """Writes a TBX doc properly

        Keyword arguments:
            tbxdict -- convenient data structure

        """
        self._generator.startDocument()

```

```

self.outfile.write('<!DOCTYPE tmx SYSTEM "tbx.dtd">\n')
self._generator.startElement('martif', AttributesImpl({'type': 'TBX',
                                                       'xml:lang': tbdict['_srclang_'] }))

self.srclang = tbdict['_srclang_']
del tbdict['_srclang_']
self._generator.characters('\n')
self.writeheader(tbdict['_header_']['_filedesc_'], 1)
del tbdict['_header_']
self._generator.characters('\n')
self.writebody(tbdict, 1)
self._generator.characters('\n')
self._generator.endElement('martif')
self._generator.endDocument()

def writeheader(self, headlist, indent):
    """Writes the TBX header properly

    Keyword arguments:
        headlist -- a list of sourceDescs from header (see TBX spec for details)

    """
    self._generator.characters(' ' * indent)
    self._generator.startElement('martifHeader', AttributesImpl({}))
    self._generator.characters('\n')
    self._generator.characters(' ' * (indent+1))
    self._generator.startElement('fileDesc', AttributesImpl({}))
    self._generator.characters('\n')
    self._generator.characters(' ' * (indent+2))
    self._generator.startElement('sourceDesc', AttributesImpl({}))
    self._generator.characters('\n')
    for desc in headlist:
        self._generator.characters(' ' * (indent+3))
        p = '<p>' + desc + '</p>'
        parseString(p.encode('utf-8'), self._segwriter) # Error prooone!
        self._generator.characters('\n')
    self._generator.characters(' ' * (indent+2))
    self._generator.endElement('sourceDesc')
    self._generator.characters('\n')
    self._generator.characters(' ' * (indent+1))
    self._generator.endElement('fileDesc')
    self._generator.characters('\n')
    self._generator.characters(' ' * indent)
    self._generator.endElement('martifHeader')

def writebody(self, tbdict, indent):
    """Writes the TBX body properly

    Keyword arguments:
        tbdict -- convenient data structure, without '_header_' and '_srclang_' entries

    """
    self._generator.characters(' ' * indent)
    self._generator.startElement('text', AttributesImpl({}))
    self._generator.characters(' ')
    self._generator.startElement('body', AttributesImpl({}))
    self._generator.characters('\n')
    for term in tbdict:
        self.writetermentry(tbdict[term], (indent+1))
        self._generator.characters('\n')
    self._generator.characters(' ' * indent)
    self._generator.endElement('body')
    self._generator.characters(' ')
    self._generator.endElement('text')

def writetermentry(self, termdict, indent):
    """Writes a TBX term properly

    Keyword arguments:
        termdict -- a list of entries for lang

    """
    self._generator.characters(' ' * indent)
    self._generator.startElement('termEntry', AttributesImpl({}))

```

```

self._generator.characters('\n')
for lang, termlist in termdict.items():
    self._generator.characters(' ' * (indent+1))
    self._generator.startElement('langSet', AttributesImpl({ 'xml:lang': lang }))
    self._generator.characters('\n')
    for term in termlist:
        self._generator.characters(' ' * (indent+2))
        self._generator.startElement('ntig', AttributesImpl({}))
        self._generator.characters('\n')
        self._generator.characters(' ' * (indent+3))
        self._generator.startElement('termGrp', AttributesImpl({}))
        self._generator.characters('\n')
        self._generator.characters(' ' * (indent+4))
        term = '<term>' + term + '</term>'
        parseString(term.encode('utf-8'), self._segwriter)
        self._generator.characters('\n')
        self._generator.characters(' ' * (indent+3))
        self._generator.endElement('termGrp')
        self._generator.characters('\n')
        self._generator.characters(' ' * (indent+2))
        self._generator.endElement('ntig')
        self._generator.characters('\n')
    self._generator.characters(' ' * (indent+1))
    self._generator.endElement('langSet')
    self._generator.characters('\n')
self._generator.characters(' ' * indent)
self._generator.endElement('termEntry')

```

ANEXO E – tbx_test.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#
# Copyright 2006 Achilles Colombo Prudêncio
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with translate; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
#
# author: Achilles Colombo Prudêncio

from translate.storage import tbx1

print "Instanciando tratador TBX usando um arquivo existente como parâmetro..."

tbxh = tbx1.TBXHandler("tbxsample1.tbx")

print "\nPronto. Fazendo atualizacoes aleatorias...\n"

tbxh.addsourcedesc(u"Testando header...")

print tbxh.getsourcedesc()

#tbxh.debug()

for srckey in tbxh.getsources():
    tbxh.addterm(srckey, tbxh.getsrclang(), u"Testing.", u"pt-pt")

print "\nPronto. Salvando arquivo..."

tbxh.savefile()

print "\nPronto. Recarregando arquivo..."

tbxh = tbx1.TBXHandler("tbxsample1.tbx")

print "\nPronto. Teste finalizado."
```


ANEXO F – pootle.py

Este arquivo foi apenas **modificado** pelo autor, para implementar as mudanças propostas! Todos os trechos de código modificado são envoltos pelos comentários **##MODIFIED Achilles** e **##END MODIFICATION**. Todo o restante do código é de autoria dos desenvolvedores do Pootle.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# Copyright 2004-2006 Zuza Software Foundation
#
# This file is part of translate.
#
# translate is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# translate is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with translate; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

from jToolkit.web import server
from jToolkit.web import templateserver
from jToolkit.web import session
from jToolkit import prefs
from jToolkit import localize
from jToolkit.widgets import widgets
from jToolkit.widgets import spellui
from jToolkit.widgets import thumbgallery
from jToolkit.web import simplewebserver
from Pootle import indexpage
from Pootle import adminpages
from Pootle import translatepage
from Pootle import pagelayout
from Pootle import projects
from Pootle import potree
from Pootle import users
from Pootle import filelocations
from Pootle import __version__ as pootleversion
from translate import __version__ as toolkitversion
from jToolkit import __version__ as jtoolkitversion
import sys
import os
import random
import pprint
```

```

class PootleServer(users.OptionalLoginAppServer, templateserver.TemplateServer):
    """the Server that serves the Pootle Pages"""
    def __init__(self, instance, webserver, sessioncache=None, errorhandler=None, loginpageclass=users.LoginPage):
        if sessioncache is None:
            sessioncache = session.SessionCache(sessionclass=users.PootleSession)
        self.potree = potree.POTree(instance)
        super(PootleServer, self).__init__(instance, webserver, sessioncache, errorhandler, loginpageclass)
        self.templatedir = filelocations.templatedir
        self.setdefaultoptions()

    def loadurl(self, filename, context):
        """loads a url internally for overlay code"""
        # print "call to load %s with context:\n%s" % (filename, pprint.pformat(context))
        filename = os.path.join(self.templatedir, filename+os.extsep+"html")
        if os.path.exists(filename):
            return open(filename, "r").read()
        return None

    def saveprefs(self):
        """saves any changes made to the preferences"""
        # TODO: this is a hack, fix it up nicely :-)
        prefsfile = self.instance.__root__.__dict__["_setvalue"].im_self
        prefsfile.savefile()

    def.setdefaultoptions(self):
        """sets the default options in the preferences"""
        changed = False
        if not hasattr(self.instance, "title"):
            setattr(self.instance, "title", "Pootle Demo")
            changed = True
        if not hasattr(self.instance, "description"):
            defaultdescription = "This is a demo installation of pootle. The administrator can customize the description in the preferences."
            setattr(self.instance, "description", defaultdescription)
            changed = True
        if not hasattr(self.instance, "baseurl"):
            setattr(self.instance, "baseurl", "/")
            changed = True
        if changed:
            self.saveprefs()

    def changeoptions(self, argdict):
        """changes options on the instance"""
        for key, value in argdict.iteritems():
            if not key.startswith("option-"):
                continue
            optionname = key.replace("option-", "", 1)
            setattr(self.instance, optionname, value)
        self.saveprefs()

    def inittranslation(self, localedir=None, localedomains=None, defaultlanguage=None):
        """initializes live translations using the Pootle PO files"""
        self.localedomains = ['jToolkit', 'pootle']
        self.localedir = None
        self.languageelist = self.potree.getlanguagecodes('pootle')
        self.languagenames = self.potree.getlanguages()
        self.defaultlanguage = defaultlanguage
        if self.defaultlanguage is None:
            self.defaultlanguage = getattr(self.instance, "defaultlanguage", "en")
        if self.potree.hasproject(self.defaultlanguage, 'pootle'):
            try:
                self.translation = self.potree.getproject(self.defaultlanguage, 'pootle')
                return
            except:
                self.errorhandler.logerror("Could not initialize translation")
        # if no translation available, set up a blank translation
        super(PootleServer, self).inittranslation()

    def gettranslation(self, language):
        """returns a translation object for the given language (or default if language is None)"""
        if language is None:
            return self.translation
        else:
            try:

```

```

        return self.potree.getproject(language, 'pootle')
    except:
        self.errorhandler.logerror("Could not get translation for language %r" % language)
        return self.translation

def refreshstats(self, args):
    """refreshes all the available statistics..."""
    if args:
        def filtererrorhandler(functionname, str1, str2, e):
            print "error in filter %s: %r, %r, %s" % (functionname, str1, str2, e)
            return False
        checkerclasses = [projects.checks.StandardChecker, projects.pofilter.StandardPOChecker]
        stdchecker = projects.pofilter.POTeeChecker(checkerclasses=checkerclasses, errorhandler=filtererrorhandler)
        for arg in args:
            if not os.path.exists(arg):
                print "file not found:", arg
            if os.path.isdir(arg):
                if not arg.endswith(os.sep):
                    arg += os.sep
                projectcode, languagecode = self.potree.getcodesfordir(arg)
                dummyproject = projects.DummyStatsProject(arg, stdchecker, projectcode, languagecode)
                def refreshdir(dummy, dirname, fnames):
                    reldirname = dirname.replace(dummyproject.podir, "")
                    for fname in fnames:
                        fpath = os.path.join(reldirname, fname)
                        if fname.endswith(".po") and not os.path.isdir(os.path.join(dummyproject.podir, fpath)):
                            print "refreshing stats for", fpath
                            projects.pootlefile.pootlefile(dummyproject, fpath).updatequickstats()
                os.path.walk(arg, refreshdir, None)
                if projectcode and languagecode:
                    dummyproject.savequickstats()
            elif os.path.isfile(arg):
                dummyproject = projects.DummyStatsProject(".", stdchecker)
                print "refreshing stats for", arg
                projects.pootlefile.pootlefile(dummyproject, arg)
        else:
            print "refreshing stats for all files in all projects"
            self.potree.refreshstats()

def generateactivationcode(self):
    """generates a unique activation code"""
    return "".join(["%02x" % int(random.random()*0x100) for i in range(16)])

def getpage(self, pathwords, session, argdict):
    """return a page that will be sent to the user"""
    #Ensure we get unicode from argdict
    #TODO: remove when jToolkit does this
    ##MODIFIED Achilles
    print "\npathwords: " + str(pathwords)
    print "argdict:"
    ##END MODIFICATION
    import pprint
    pprint.pprint(argdict)
    newargdict = {}
    for key, value in argdict.iteritems():
        if isinstance(key, str):
            key = key.decode("utf-8")
        if isinstance(value, str):
            value = value.decode("utf-8")
        newargdict[key] = value
    argdict = newargdict
    # TODO: strip off the initial path properly
    while pathwords and pathwords[0] == "pootle":
        pathwords = pathwords[1:]
    if pathwords:
        top = pathwords[0]
    else:
        top = ""
    if top == 'js':
        pathwords = pathwords[1:]
        jsfile = os.path.join(filelocations.htmlmdir, 'js', *pathwords)
        if not os.path.exists(jsfile):
            jsfile = os.path.join(filelocations.jtoolkitdir, 'js', *pathwords)

```

```

        if not os.path.exists(jsfile):
            return None
        jspage = widgets.PlainContents(None)
        jspage.content_type = "application/x-javascript"
        jspage.sendfile_path = jsfile
        jspage.allowcaching = True
        return jspage
    elif pathwords and pathwords[-1].endswith(".css"):
        cssfile = os.path.join(filelocations.html_dir, *pathwords)
        if not os.path.exists(cssfile):
            cssfile = os.path.join(filelocations.jtoolkit_dir, *pathwords)
            if not os.path.exists(cssfile):
                return None
        csspage = widgets.PlainContents(None)
        csspage.content_type = "text/css"
        csspage.sendfile_path = cssfile
        csspage.allowcaching = True
        return csspage
    elif top == 'images':
        pathwords = pathwords[1:]
        picturefile = os.path.join(filelocations.html_dir, 'images', *pathwords)
        picture = widgets.SendFile(picturefile)
        picture.content_type = thumbgallery.getcontenttype(pathwords[-1])
        picture.allowcaching = True
        return picture
    elif top == "testtemplates.html":
        return templateserver.TemplateServer.getpage(self, pathwords, session, argdict)
    elif not top or top == "index.html":
        ##MODIFIED Achilles
        return indexpage.PootleIndex(self.potree, session, "contentonly" in argdict)
        ##END MODIFICATION
    elif top == 'about.html':
        ##MODIFIED Achilles
        return indexpage.AboutPage(session, "contentonly" in argdict)
        ##END MODIFICATION
    elif top == "login.html":
        if session.isopen:
            returnurl = argdict.get('returnurl', None) or getattr(self.instance, 'homepage', 'home/')
            return server.Redirect(returnurl)
        if 'username' in argdict:
            session.username = argdict["username"]
            return users.LoginPage(session, languagenames=self.languagenames)
    elif top == "register.html":
        return self.registerpage(session, argdict)
    elif top == "activate.html":
        return self.activatepage(session, argdict)
    elif top == "projects":
        pathwords = pathwords[1:]
        if pathwords:
            top = pathwords[0]
        else:
            top = ""
        if not top or top == "index.html":
            ##MODIFIED Achilles
            return indexpage.ProjectsIndex(self.potree, session, "contentonly" in argdict)
            ##END MODIFICATION
        else:
            projectcode = top
            if not self.potree.hasproject(None, projectcode):
                return None
            pathwords = pathwords[1:]
            if pathwords:
                top = pathwords[0]
            else:
                top = ""
            if not top or top == "index.html":
                return indexpage.ProjectLanguageIndex(self.potree, projectcode, session)
            elif top == "admin.html":
                return adminpages.ProjectAdminPage(self.potree, projectcode, session, argdict)
    elif top == "languages":
        pathwords = pathwords[1:]
        if pathwords:
            top = pathwords[0]

```

```

else:
    top = ""
if not top or top == "index.html":
    ##MODIFIED Achilles
    return indexpage.LanguagesIndex(self.potree, session, "contentonly" in argdict)
    ##END MODIFICATION
elif top == "home":
    pathwords = pathwords[1:]
    if pathwords:
        top = pathwords[0]
    else:
        top = ""
if not session.isopen:
    templatename = "redirect"
    templatevars = {
        "pagetitle": session.localize("Redirecting to login..."),
        "refresh": 1,
        "refreshurl": "login.html",
        "message": session.localize("Need to log in to access home page"),
    }
    pagelayout.completetemplatevars(templatevars, session)
    return server.Redirect("../login.html", withtemplate=(templatename, templatevars))
if not top or top == "index.html":
    ##MODIFIED Achilles
    return indexpage.UserIndex(self.potree, session, "contentonly" in argdict)
    ##END MODIFICATION
elif top == "options.html":
    if "changeoptions" in argdict:
        session.setoptions(argdict)
        ##MODIFIED Achilles
        if "contentonly" in argdict:
            return widgets.PlainContents(session.localize("Your options have been saved. "))
        ##END MODIFICATION
    if "changepersonal" in argdict:
        session.setpersonaloptions(argdict)
        ##MODIFIED Achilles
        if "contentonly" in argdict:
            return widgets.PlainContents(session.localize("Your options have been saved. "))
        ##END MODIFICATION
    if "changeinterface" in argdict:
        session.setinterfaceoptions(argdict)
        ##MODIFIED Achilles
        if "contentonly" in argdict:
            return widgets.PlainContents(session.localize("Your options have been saved. "))
    return users.UserOptions(self.potree, session, "contentonly" in argdict)
    ##END MODIFICATION
elif top == "admin":
    pathwords = pathwords[1:]
    if pathwords:
        top = pathwords[0]
    else:
        top = ""
if not session.isopen:
    templatename = "redirect"
    templatevars = {
        "pagetitle": session.localize("Redirecting to login..."),
        "refresh": 1,
        "refreshurl": "login.html",
        "message": session.localize("Need to log in to access admin page"),
    }
    pagelayout.completetemplatevars(templatevars, session)
    return server.Redirect("../login.html", withtemplate=(templatename, templatevars))
if not session.issiteadmin():
    templatename = "redirect"
    templatevars = {
        "pagetitle": session.localize("Redirecting to home..."),
        "refresh": 1,
        "refreshurl": "login.html",
        "message": self.localize("You do not have the rights to administer pootle."),
    }
    pagelayout.completetemplatevars(templatevars, session)
    return server.Redirect("../index.html", withtemplate=(templatename, templatevars))
if not top or top == "index.html":

```

```

if "changegeneral" in argdict:
    self.changeoptions(argdict)
    ##MODIFIED Achilles
    if "contentonly" in argdict:
        return widgets.PlainContents(session.localize("Your options have been saved.))
    return adminpages.AdminPage(self.potree, session, self.instance, "contentonly" in argdict)
##END MODIFICATION
elif top == "users.html":
    if "changeusers" in argdict:
        self.changeusers(session, argdict)
        ##MODIFIED Achilles
        if "contentonly" in argdict:
            return widgets.PlainContents(session.localize("Your options have been saved.))
    return adminpages.UsersAdminPage(self, session.loginchecker.users, session, self.instance, "contentonly" in argdict)
##END MODIFICATION
elif top == "languages.html":
    if "changelanguages" in argdict:
        self.potree.changelanguages(argdict)
        ##MODIFIED Achilles
        if "contentonly" in argdict:
            return widgets.PlainContents(session.localize("Your options have been saved.))
    return adminpages.LanguagesAdminPage(self.potree, session, self.instance, "contentonly" in argdict)
##END MODIFICATION
elif top == "projects.html":
    if "changeprojects" in argdict:
        self.potree.changeprojects(argdict)
        ##MODIFIED Achilles
        if "contentonly" in argdict:
            return widgets.PlainContents(session.localize("Your options have been saved.))
    return adminpages.ProjectsAdminPage(self.potree, session, self.instance, "contentonly" in argdict)
##END MODIFICATION
elif top == "templates" or self.potree.haslanguage(top):
    languagecode = top
    pathwords = pathwords[1:]
    if pathwords:
        top = pathwords[0]
        bottom = pathwords[-1]
    else:
        top = ""
        bottom = ""
    if not top or top == "index.html":
        return indexpage.LanguageIndex(self.potree, languagecode, session)
    if self.potree.hasproject(languagecode, top):
        projectcode = top
        project = self.potree.getproject(languagecode, projectcode)
        pathwords = pathwords[1:]
        if pathwords:
            top = pathwords[0]
        else:
            top = ""
    if not top or top == "index.html":
        return indexpage.ProjectIndex(project, session, argdict)
    elif top == "admin.html":
        return adminpages.TranslationProjectAdminPage(self.potree, project, session, argdict)
    elif bottom == "translate.html":
        if len(pathwords) > 1:
            dirfilter = os.path.join(*pathwords[:-1])
        else:
            dirfilter = ""
        try:
            return translatepage.TranslatePage(project, session, argdict, dirfilter)
        except projects.RightsError, stoppedby:
            argdict["message"] = str(stoppedby)
            return indexpage.ProjectIndex(project, session, argdict, dirfilter)
    elif bottom == "spellcheck.html":
        # the full review page
        argdict["spellchecklang"] = languagecode
        return spellui.SpellingReview(session, argdict, js_url="/js/spellui.js")
    elif bottom == "spellingstandby.html":
        # a simple 'loading' page
        return spellui.SpellingStandby()
    elif bottom.endswith(".") + project.fileext):
        pofilename = os.path.join(*pathwords)

```

```

if argdict.get("translate", 0):
    try:
        return translatepage.TranslatePage(project, session, argdict, dirfilter=pofilename)
    except projects.RightsError, stoppedby:
        argdict["message"] = str(stoppedby)
        return indexpage.ProjectIndex(project, session, argdict, dirfilter=pofilename)
elif argdict.get("index", 0):
    return indexpage.ProjectIndex(project, session, argdict, dirfilter=pofilename)
else:
    pofile = project.getpofile(pofilename, freshen=False)
    page = widgets.SendFile(pofile.filename)
    page.etag = str(pofile.pomtime)
    page.allowcaching = True
    encoding = pofile.encoding or "UTF-8"
    page.content_type = "text/plain; charset=%s" % encoding
    return page
elif bottom.endswith(".csv") or bottom.endswith(".xlf") or bottom.endswith(".ts") or bottom.endswith(".mo"):
    destfilename = os.path.join(*pathwords)
    basename, extension = os.path.splitext(destfilename)
    pofilename = basename + os.extsep + project.fileext
    extension = extension[1:]
    if extension == "mo":
        if not "pocompile" in project.gettrights(session):
            return None
    etag, filepath_or_contents = project.convert(pofilename, extension)
    if etag:
        page = widgets.SendFile(filepath_or_contents)
        page.etag = str(etag)
    else:
        page = widgets.PlainContents(filepath_or_contents)
    page.allowcaching = True
    if extension == "csv":
        page.content_type = "text/plain; charset=UTF-8"
    elif extension == "xlf" or extension == "ts":
        page.content_type = "text/xml; charset=UTF-8"
    elif extension == "mo":
        page.content_type = "application/octet-stream"
    return page
elif bottom.endswith(".zip"):
    if not "archive" in project.gettrights(session):
        return None
    if len(pathwords) > 1:
        dirfilter = os.path.join(*pathwords[:-1])
    else:
        dirfilter = None
    goal = argdict.get("goal", None)
    if goal:
        goalfiles = project.getgoalfiles(goal)
        pofilenames = []
        for goalfile in goalfiles:
            pofilenames.extend(project.browsefiles(goalfile))
    else:
        pofilenames = project.browsefiles(dirfilter)
    archivecontents = project.getarchive(pofilenames)
    page = widgets.PlainContents(archivecontents)
    page.content_type = "application/zip"
    return page
elif bottom.endswith(".sdf") or bottom.endswith(".sgi"):
    if not "pocompile" in project.gettrights(session):
        return None
    oocontents = project.getoo()
    page = widgets.PlainContents(oocontents)
    page.content_type = "text/tab-separated-values"
    return page
elif bottom == "index.html":
    if len(pathwords) > 1:
        dirfilter = os.path.join(*pathwords[:-1])
    else:
        dirfilter = None
    return indexpage.ProjectIndex(project, session, argdict, dirfilter)
else:
    return indexpage.ProjectIndex(project, session, argdict, os.path.join(*pathwords))
return None

```

```

class PootleOptionParser(simplewebservice.WebOptionParser):
    def __init__(self):
        versionstring = "%prog %s\njToolkit %s\nTranslate Toolkit %s" % (pootleversion.ver, jtoolkitversion.ver, toolkitversion.ver)
        simplewebservice.WebOptionParser.__init__(self, version=versionstring)
        self.set_default('prefsfile', filelocations.prefsfile)
        self.set_default('instance', 'Pootle')
        self.set_default('htmlmdir', filelocations.htmlmdir)
        self.add_option('', "--refreshstats", dest="action", action="store_const", const="refreshstats",
            default="runwebservice", help="refresh the stats files instead of running the webservice")

def checkversions():
    """Checks that version dependencies are met"""
    if not hasattr(toolkitversion, "build") or toolkitversion.build < 9000:
        raise RuntimeError("requires Translate Toolkit version >= 0.9. Current installed version is: %s" % toolkitversion.ver)

def main():
    # run the web server
    checkversions()
    parser = PootleOptionParser()
    options, args = parser.parse_args()
    if options.action != "runwebservice":
        options.servertype = "dummy"
    server = parser.getserver(options)
    if options.action == "runwebservice":
        simplewebservice.run(server, options)
    elif options.action == "refreshstats":
        server.refreshstats(args)

if __name__ == '__main__':
    main()

```


ANEXO G – projects.py

Este arquivo foi apenas **modificado** pelo autor, para implementar as mudanças propostas! Todos os trechos de código modificado são envoltos pelos comentários **##MODIFIED Achilles** e **##END MODIFICATION**. Todo o restante do código é de autoria dos desenvolvedores do Pootle.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# Copyright 2004-2006 Zuza Software Foundation
#
# This file is part of translate.
#
# translate is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# translate is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with translate; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

"""manages projects and files and translations"""

from translate.storage import po
##MODIFIED Achilles
from translate.storage import tmx1
from translate.storage import tbx1
##END MODIFICATION
from translate.filters import checks
from translate.filters import pofilter
from translate.convert import po2csv
from translate.convert import po2xliff
from translate.convert import po2ts
from translate.convert import pot2po
from translate.convert import po2oo
from translate.tools import pocompile
from translate.tools import pogrep
from Pootle import pootlefile
from Pootle import versioncontrol
from jToolkit import timecache
from jToolkit import prefs
import time
import os
import cStringIO
import gettext
from jToolkit.data import indexer
```

```

from jToolkit import glock

class RightsError(ValueError):
    pass

class InternalAdminSession:
    """A fake session used for doing internal admin jobs"""
    def __init__(self):
        self.username = "internal"
        self.isopen = True

    def localize(self, message):
        return message

    def issiteadmin(self):
        return True

class potimecache(timecache.timecache):
    """caches pootlefile objects, remembers time, and reverts back to statistics when necessary..."""
    def __init__(self, expiryperiod, project):
        """initialises the cache to keep objects for the given expiryperiod, and point back to the project"""
        timecache.timecache.__init__(self, expiryperiod)
        self.project = project

    def __getitem__(self, key):
        """[] access of items"""
        if key and not dict.__contains__(self, key):
            popath = os.path.join(self.project.podir, key)
            if os.path.exists(popath):
                # update the index to pofiles...
                self.project.scanpofiles()
            return timecache.timecache.__getitem__(self, key)

    def expire(self, pofilename):
        """expires the given pofilename by recreating it (holding only stats)"""
        timestamp, currentfile = dict.__getitem__(self, pofilename)
        if currentfile.pomtime is not None:
            # use the currentfile.pomtime as a timestamp as well, so any modifications will extend its life
            if time.time() - currentfile.pomtime > self.expiryperiod.seconds:
                self.__setitem__(pofilename, pootlefile.pootlefile(self.project, pofilename))

class TranslationProject(object):
    """Manages iterating through the translations in a particular project"""
    fileext = "po"
    ##MODIFIED Achilles
    tmext = "tmx"
    tbext = "tbx"
    ##END MODIFICATION
    def __init__(self, languagecode, projectcode, potree, create=False):
        self.languagecode = languagecode
        self.projectcode = projectcode
        self.potree = potree
        self.languagename = self.potree.getlanguagename(self.languagecode)
        self.projectname = self.potree.getprojectname(self.projectcode)
        self.projectdescription = self.potree.getprojectdescription(self.projectcode)
        self.pofiles = potimecache(15*60, self)
        self.projectcheckerstyle = self.potree.getprojectcheckerstyle(self.projectcode)
        checkerclasses = [checks.projectcheckers.get(self.projectcheckerstyle, checks.StandardChecker), pofilter.StandardPOChecker]
        self.checker = pofilter.POTeeChecker(checkerclasses=checkerclasses, errorhandler=self.filtererrorhandler)
        if create:
            self.quickstats = {}
            self.converttemplates(InternalAdminSession())
        self.podir = potree.getpodir(languagecode, projectcode)
        ##MODIFIED Achilles
        tmfilename = os.path.join(self.podir, projectcode + os.extsep + self.tmext)
        tbfilename = os.path.join(self.podir, projectcode + os.extsep + self.tbext)
        self.transmemory = TranslationMemory(tmfilename, languagecode.replace('_', '-'))
        self.termbase = Termbase(tbfilename, languagecode.replace('_', '-'))
        ##END MODIFICATION
        if self.potree.hasgnufiles(self.podir, self.languagecode) == "gnu":
            self.filestyle = "gnu"
        else:
            self.filestyle = "std"

```

```

self.readprefs()
self.readquickstats()
self.scanpofiles()
self.initindex()

def readprefs(self):
    """reads the project preferences"""
    self.prefs = prefs.PrefsParser()
    self.prefsfile = os.path.join(self.podir, "pootle-%s-%s.prefs" % (self.projectcode, self.languagecode))
    if not os.path.exists(self.prefsfile):
        prefsfile = open(self.prefsfile, "w")
        prefsfile.write("# Pootle preferences for project %s, language %s\n\n" % (self.projectcode, self.languagecode))
        prefsfile.close()
    self.prefs.parsefile(self.prefsfile)

def saveprefs(self):
    """saves the project preferences"""
    self.prefs.savefile()

def getrightnames(self, session):
    """gets the available rights and their localized names"""
    localize = session.localize
    return [("view", localize("View")),
            ("suggest", localize("Suggest")),
            ("translate", localize("Translate")),
            ("review", localize("Review")),
            ("archive", localize("Archive")),
            ("pocompile", localize("Compile PO files")),
            ("assign", localize("Assign")),
            ("admin", localize("Administrate")),
            ]

def getrights(self, session=None, username=None, usedefaults=True):
    """gets the rights for the given user (name or session, or not-logged-in if username is None)
    if usedefaults is False then None will be returned if no rights are defined (useful for editing rights)"""
    # internal admin sessions have all rights
    if isinstance(session, InternalAdminSession):
        return [right for right, localizedright in self.getrightnames(session)]
    if session is not None and session.isopen and username is None:
        username = session.username
    if username is None:
        username = "nobody"
    rights = None
    rightstree = getattr(self.prefs, "rights", None)
    if rightstree is not None:
        if rightstree.__hasattr__(username):
            rights = rightstree.__getattr__(username)
        else:
            rights = None
    if rights is None:
        if usedefaults:
            if username == "nobody":
                rights = "view"
            elif rightstree is None:
                if self.languagecode == "en":
                    rights = "view, archive, pocompile"
                else:
                    rights = "view, review, translate, archive, pocompile"
            else:
                rights = getattr(rightstree, "default", None)
        else:
            return rights
    rights = [right.strip() for right in rights.split(",")]
    if session is not None and session.issiteadmin():
        if "admin" not in rights:
            rights.append("admin")
    return rights

def getuserswithrights(self):
    """gets all users that have rights defined for this project"""
    return [username for username, user_rights in getattr(self.prefs, "rights", {}).iteritems()]

def setrights(self, username, rights):

```

```

"""sets the rights for the given username... (or not-logged-in if username is None)"""
if username is None: username = "nobody"
if isinstance(rights, list):
    rights = ", ".join(rights)
if not hasattr(self.prefs, "rights"):
    self.prefs.rights = prefs.PrefNode(self.prefs, "rights")
self.prefs.rights.__setattr__(username, rights)
self.saveprefs()

def delrights(self, username):
    """deletes teh rights for the given username"""
    if username == "nobody" or username == "default":
        raise RightsError(session.localize('You cannot remove the "nobody" or "default" user'))
    self.prefs.rights.__delattr__(username)
    self.saveprefs()

def getgoalnames(self):
    """gets the goals and associated files for the project"""
    goals = getattr(self.prefs, "goals", {})
    goallist = []
    for goalname, goalnode in goals.iteritems():
        goallist.append(goalname.decode("utf-8"))
    goallist.sort()
    return goallist

def getgoals(self):
    """gets the goal, goalnode tuples"""
    goals = getattr(self.prefs, "goals", {})
    newgoals = {}
    for goalname, goalnode in goals.iteritems():
        newgoals[goalname.decode("utf-8")] = goalnode
    return newgoals

def getgoalfiles(self, goalname, dirfilter=None, maxdepth=None, includedirs=True, expanddirs=False, includepartial=False):
    """gets the files for the given goal, with many options!
    dirfilter limits to files in a certain subdirectory
    maxdepth limits to files / directories of a certain depth
    includedirs specifies whether to return directory names
    expanddirs specifies whether to expand directories and return all files in them
    includepartial specifies whether to return directories that are not in the goal, but have files below maxdepth in the goal"""
    goals = getattr(self.prefs, "goals", {})
    poext = os.extsep + self.fileext
    pathsep = os.path.sep
    unique = lambda filelist: dict.fromkeys(filelist).keys()
    for testgoalname, goalnode in self.getgoals().iteritems():
        if goalname != testgoalname: continue
        goalmembers = getattr(goalnode, "files", "")
        goalmembers = [goalfile.strip() for goalfile in goalmembers.split(",") if goalfile.strip()]
        goaldirs = [goaldir for goaldir in goalmembers if goaldir.endswith(pathsep)]
        goalfiles = [goalfile for goalfile in goalmembers if not goalfile.endswith(pathsep)]
        if expanddirs:
            expandgoaldirs = []
            expandgoalfiles = []
            for goaldir in goaldirs:
                expandedfiles = self.browsefiles(dirfilter=goaldir, includedirs=includedirs, includefiles=True)
                expandgoalfiles.extend([expandfile for expandfile in expandedfiles if expandfile.endswith(poext)])
                expandgoaldirs.extend([expanddir + pathsep for expanddir in expandedfiles if not expanddir.endswith(poext)])
            goaldirs = unique(goaldirs + expandgoaldirs)
            goalfiles = unique(goalfiles + expandgoalfiles)
        if dirfilter:
            if not dirfilter.endswith(pathsep) and not dirfilter.endswith(poext):
                dirfilter += pathsep
            goalfiles = [goalfile for goalfile in goalfiles if goalfile.startswith(dirfilter)]
            goaldirs = [goaldir for goaldir in goaldirs if goaldir.startswith(dirfilter)]
        if maxdepth is not None:
            if includepartial:
                partialdirs = [goalfile for goalfile in goalfiles if goalfile.count(pathsep) > maxdepth]
                partialdirs += [goalfile for goalfile in goaldirs if goalfile.count(pathsep) > maxdepth]
                makepartial = lambda goalfile: pathsep.join(goalfile.split(pathsep)[:maxdepth+1])+pathsep
                partialdirs = [makepartial(goalfile) for goalfile in partialdirs]
            goalfiles = [goalfile for goalfile in goalfiles if goalfile.count(pathsep) <= maxdepth]
            goaldirs = [goaldir for goaldir in goaldirs if goaldir.count(pathsep) <= maxdepth+1]
            if includepartial:

```

```

        goaldirs += partialdirs
    if includedirs:
        return unique(goalfiles + goaldirs)
    else:
        return unique(goalfiles)
return []

def getancestry(self, filename):
    """returns parent directories of the file"""
    ancestry = []
    parts = filename.split(os.path.sep)
    for i in range(1, len(parts)):
        ancestor = os.path.join(*parts[:i]) + os.path.sep
        ancestry.append(ancestor)
    return ancestry

def getfilegoals(self, filename):
    """gets the goals the given file is part of"""
    goals = self.getgoals()
    filegoals = []
    ancestry = self.getancestry(filename)
    for goalname, goalnode in goals.iteritems():
        goalfiles = getattr(goalnode, "files", "")
        goalfiles = [goalfile.strip() for goalfile in goalfiles.split(",") if goalfile.strip()]
        if filename in goalfiles:
            filegoals.append(goalname)
            continue
        for ancestor in ancestry:
            if ancestor in goalfiles:
                filegoals.append(goalname)
                continue
    return filegoals

def setfilegoals(self, session, goalnames, filename):
    """sets the given file to belong to the given goals exactly"""
    filegoals = self.getfilegoals(filename)
    for othergoalname in filegoals:
        if othergoalname not in goalnames:
            self.removefilefromgoal(session, othergoalname, filename)
    for goalname in goalnames:
        goalfiles = self.getgoalfiles(goalname)
        if filename not in goalfiles:
            goalfiles.append(filename)
        self.setgoalfiles(session, goalname, goalfiles)

def removefilefromgoal(self, session, goalname, filename):
    """removes the given file from the goal"""
    goalfiles = self.getgoalfiles(goalname)
    if filename in goalfiles:
        goalfiles.remove(filename)
        self.setgoalfiles(session, goalname, goalfiles)
    else:
        unique = lambda filelist: dict.fromkeys(filelist).keys()
        ancestry = self.getancestry(filename)
        for ancestor in ancestry:
            if ancestor in goalfiles:
                filedepth = filename.count(os.path.sep)
                ancestordirs = self.getgoalfiles(goalname, ancestor, maxdepth=filedepth+1, includedirs=True, expanddirs=True)
                ancestordirs = [ancestorfile for ancestorfile in ancestordirs if ancestorfile.endswith(os.path.sep)]
                if filename.endswith(os.path.sep):
                    ancestorfiles = self.getgoalfiles(goalname, ancestor, maxdepth=filedepth-1, expanddirs=True)
                else:
                    ancestorfiles = self.getgoalfiles(goalname, ancestor, maxdepth=filedepth, expanddirs=True)
                ancestorfiles = unique(ancestordirs + ancestorfiles)
                if not filename in ancestorfiles:
                    raise KeyError("expected to find file %s in ancestor %s files %r" % (filename, ancestor, ancestorfiles))
                ancestorfiles.remove(filename)
                ancestorfiles.remove(ancestor)
            goalfiles.remove(ancestor)
            goalfiles.extend(ancestorfiles)
        self.setgoalfiles(session, goalname, goalfiles)
        continue

```

```

def setgoalfiles(self, session, goalname, goalfiles):
    """sets the goalfiles for the given goalname"""
    if "admin" not in self.getrights(session):
        raise RightsError(session.localize("You do not have rights to alter goals here"))
    if isinstance(goalfiles, list):
        goalfiles = [goalfile.strip() for goalfile in goalfiles if goalfile.strip()]
        goalfiles.sort()
        goalfiles = ", ".join(goalfiles)
    if not hasattr(self.prefs, "goals"):
        self.prefs.goals = prefs.PrefNode(self.prefs, "goals")
    goals = self.getgoals()
    goalname = goalname.encode("utf-8")
    if not goalname in goals:
        # TODO: check that its a valid goalname (alphanumeric etc)
        self.prefs.goals._setattr__(goalname, prefs.PrefNode(self.prefs.goals, goalname))
    goalnode = self.prefs.goals._getattr__(goalname)
    goalnode.files = goalfiles
    self.saveprefs()

def getgoalusers(self, goalname):
    """gets the users for the given goal"""
    goals = self.getgoals()
    for testgoalname, goalnode in goals.iteritems():
        if goalname != testgoalname: continue
        goalusers = getattr(goalnode, "users", "")
        goalusers = [goaluser.strip() for goaluser in goalusers.split(",") if goaluser.strip()]
    return goalusers
return []

def getusergoals(self, username):
    """gets the goals the given user is part of"""
    goals = getattr(self.prefs, "goals", {})
    usergoals = []
    for goalname, goalnode in goals.iteritems():
        goalusers = getattr(goalnode, "users", "")
        goalusers = [goaluser.strip() for goaluser in goalusers.split(",") if goaluser.strip()]
        if username in goalusers:
            usergoals.append(goalname)
        continue
    return usergoals

def addusertogoal(self, session, goalname, username, exclusive=False):
    """adds the given user to the goal"""
    if exclusive:
        usergoals = self.getusergoals(username)
        for othergoalname in usergoals:
            if othergoalname != goalname:
                self.removeuserfromgoal(session, othergoalname, username)
    goalusers = self.getgoalusers(goalname)
    if username not in goalusers:
        goalusers.append(username)
    self.setgoalusers(session, goalname, goalusers)

def removeuserfromgoal(self, session, goalname, username):
    """removes the given user from the goal"""
    goalusers = self.getgoalusers(goalname)
    if username in goalusers:
        goalusers.remove(username)
    self.setgoalusers(session, goalname, goalusers)

def setgoalusers(self, session, goalname, goalusers):
    """sets the goalusers for the given goalname"""
    if "admin" not in self.getrights(session):
        raise RightsError(session.localize("You do not have rights to alter goals here"))
    if isinstance(goalusers, list):
        goalusers = [goaluser.strip() for goaluser in goalusers if goaluser.strip()]
        goalusers = ", ".join(goalusers)
    if not hasattr(self.prefs, "goals"):
        self.prefs.goals = prefs.PrefNode(self.prefs, "goals")
    if not hasattr(self.prefs.goals, goalname):
        self.prefs.goals._setattr__(goalname, prefs.PrefNode(self.prefs.goals, goalname))
    goalnode = self.prefs.goals._getattr__(goalname)
    goalnode.users = goalusers

```

```

self.saveprefs()

def scanpofiles(self):
    """sets the list of pofilenames by scanning the project directory"""
    self.pofilenames = self.potree.getpofiles(self.languagecode, self.projectcode, poext=self.fileext)
    for pofilename in self.pofilenames:
        if not pofilename in self.pofiles:
            self.pofiles[pofilename] = pootlefile.pootlefile(self, pofilename)
    # remove any files that have been deleted since initialization
    for pofilename in self.pofiles.keys():
        if not pofilename in self.pofilenames:
            del self.pofiles[pofilename]

def getuploadpath(self, dirname, pofilename):
    """gets the path of a po file being uploaded securely, creating directories as neccessary"""
    if os.path.isabs(dirname) or dirname.startswith("."):
        raise ValueError("invalid/insecure file path: %s" % dirname)
    if os.path.basename(pofilename) != pofilename or pofilename.startswith("."):
        raise ValueError("invalid/insecure file name: %s" % pofilename)
    if self.filestyle == "gnu":
        if not self.potree.languagematch(self.languagecode, pofilename[:-len(".po")]):
            raise ValueError("invalid GNU-style file name %s: must match '%s.po' or '%s[_][A-Z]{2,3}.po'" % (pofilename, self.languagecode, self.languagecode))
    dircheck = self.podir
    for part in dirname.split(os.sep):
        dircheck = os.path.join(dircheck, part)
        if dircheck and not os.path.isdir(dircheck):
            os.mkdir(dircheck)
    return os.path.join(self.podir, dirname, pofilename)

def uploadpofile(self, session, dirname, pofilename, contents):
    """uploads an individual PO files"""
    pathname = self.getuploadpath(dirname, pofilename)
    if os.path.exists(pathname):
        origpofile = self.getpofile(os.path.join(dirname, pofilename))
        newpofile = po.pofile(elementclass=pootlefile.pootleelement)
        infile = cStringIO.StringIO(contents)
        newpofile.parse(infile)
        if "admin" in self.getrights(session):
            origpofile.mergefile(newpofile, session.username)
        elif "translate" in self.getrights(session):
            origpofile.mergefile(newpofile, session.username, allownewstrings=False)
        else:
            raise RightsError(session.localize("You do not have rights to upload files here"))
    else:
        if "admin" not in self.getrights(session):
            raise RightsError(session.localize("You do not have rights to upload new files here"))
        outfile = open(pathname, "wb")
        outfile.write(contents)
        outfile.close()
        self.scanpofiles()

def updatepofile(self, session, dirname, pofilename):
    """updates an individual PO file from version control"""
    if "admin" not in self.getrights(session):
        raise RightsError(session.localize("You do not have rights to update files here"))
    pathname = self.getuploadpath(dirname, pofilename)
    # read from version control
    if os.path.exists(pathname):
        popath = os.path.join(dirname, pofilename)
        currentpofile = self.getpofile(popath)
        # reading BASE version of file
        origcontents = versioncontrol.getcleanfile(pathname, "BASE")
        origpofile = pootlefile.pootlefile(self, popath)
        originfile = cStringIO.StringIO(origcontents)
        origpofile.parse(originfile)
        # matching current file with BASE version
        matches = origpofile.matchitems(currentpofile, useids=False)
        # TODO: add some locking here...
        # reading new version of file
        versioncontrol.updatefile(pathname)
        newpofile = pootlefile.pootlefile(self, popath)
        newpofile.pofreshen()
        if not hasattr(newpofile, "msgidindex"):

```

```

    newpofile.makeindex()
newmatches = []
# sorting through old matches
for origpo, localpo in matches:
    # we need to find the corresponding newpo to see what to merge
    if localpo is None:
        continue
    if origpo is None:
        # if it wasn't in the original, then use the addition for searching
        origpo = localpo
    else:
        origmsgstr = origpo.target
        localmsgstr = localpo.target
        if origmsgstr == localmsgstr:
            continue
    foundsource = False
    if usesources:
        for source in origpo.getlocations():
            if source in newpofile.sourceindex:
                newpo = newpofile.sourceindex[source]
                if newpo is not None:
                    foundsource = True
                    newmatches.append((newpo, localpo))
                continue
    if not foundsource:
        msgid = origpo.unquotedmsgid
        if msgid in newpofile.msgidindex:
            newpo = newpofile.msgidindex[msgid]
            newmatches.append((newpo, localpo))
        else:
            newmatches.append((None, localpo))
# finding new matches
for newpo, localpo in newmatches:
    if newpo is None:
        # TODO: include localpo as obsolete
        continue
    if localpo is None:
        continue
    newpofile.mergeitem(newpo, localpo, "versionmerge")
# saving
newpofile.savepofile()
self.pofiles[pofilename] = newpofile
# recalculate everything
newpofile.readpofile()
else:
    versioncontrol.updatefile(pathname)
    self.scanpofiles()

def converttemplates(self, session):
    """creates PO files from the templates"""
    projectdir = os.path.join(self.potree.podirectory, self.projectcode)
    if not os.path.exists(projectdir):
        os.mkdir(projectdir)
    templatesdir = os.path.join(projectdir, "templates")
    if not os.path.exists(templatesdir):
        templatesdir = os.path.join(projectdir, "pot")
    if not os.path.exists(templatesdir):
        templatesdir = projectdir
    if self.potree.isgnustyle(self.projectcode):
        self.filestyle = "gnu"
    else:
        self.filestyle = "std"
    templates = self.potree.gettemplates(self.projectcode)
    if self.filestyle == "gnu":
        self.podir = projectdir
        if not templates:
            raise NotImplementedError("Cannot create GNU-style translation project without templates")
    else:
        self.podir = os.path.join(projectdir, self.languagecode)
        if not os.path.exists(self.podir):
            os.mkdir(self.podir)
    for potfilename in templates:
        inputfile = open(os.path.join(templatesdir, potfilename), "rb")

```



```

outputfile = cStringIO.StringIO()
pot2po.convertpot(inputfile, outputfile, None)
dirname, potfilename = os.path.dirname(potfilename), os.path.basename(potfilename)
if self.filestyle == "gnu":
    pofilename = self.languagecode + os.extsep + "po"
else:
    pofilename = potfilename[:-len(os.extsep+"pot")] + os.extsep + "po"
self.uploadpofile(session, dirname, pofilename, outputfile.getvalue())

def filtererrorhandler(self, functionname, str1, str2, e):
    print "error in filter %s: %r, %r, %s" % (functionname, str1, str2, e)
    return False

def getarchive(self, pofilenames):
    """returns an archive of the given filenames"""
    tempzipfile = os.tmpnam()
    try:
        # using zip command line is fast
        os.system("cd %s ; zip -r - %s > %s" % (self.podir, " ".join(pofilenames), tempzipfile))
        return open(tempzipfile, "r").read()
    finally:
        if os.path.exists(tempzipfile):
            os.remove(tempzipfile)
    # but if it doesn't work, we can do it from python
    import zipfile
    archivecontents = cStringIO.StringIO()
    archive = zipfile.ZipFile(archivecontents, 'w', zipfile.ZIP_DEFLATED)
    for pofilename in pofilenames:
        pofile = self.getpofile(pofilename)
        archive.write(pofile.filename, pofilename)
    archive.close()
    return archivecontents.getvalue()

def uploadarchive(self, session, dirname, archivecontents):
    """uploads the files inside the archive"""
    try:
        tempzipfile = os.tmpnam()
        # using zip command line is fast
        # os.system("cd %s ; zip -r - %s > %s" % (self.podir, " ".join(pofilenames), tempzipfile))
        # return open(tempzipfile, "r").read()
        pass
    finally:
        if os.path.exists(tempzipfile):
            os.remove(tempzipfile)
    # but if it doesn't work, we can do it from python
    import zipfile
    archivefile = cStringIO.StringIO(archivecontents)
    archive = zipfile.ZipFile(archivefile, 'r')
    # TODO: find a better way to return errors...
    for filename in archive.namelist():
        if not filename.endswith(os.extsep + self.fileext):
            print "error adding %s: not a %s file" % (filename, os.extsep + self.fileext)
            continue
        contents = archive.read(filename)
        subdirname, pofilename = os.path.dirname(filename), os.path.basename(filename)
        try:
            # TODO: use zipfile info to set the time and date of the file
            self.uploadpofile(session, os.path.join(dirname, subdirname), pofilename, contents)
        except ValueError, e:
            print "error adding %s" % filename, e
            continue
    archive.close()

def ootemplate(self):
    """Tests whether this project has an OpenOffice.org template SDF file in
    the templates directory."""
    projectdir = os.path.join(self.potree.podirectory, self.projectcode)
    templatefilename = os.path.join(projectdir, "templates", "en-US.sdf")
    if os.path.exists(templatefilename):
        return templatefilename
    else:
        return None

```

```

def getoo(self):
    """Returns an OpenOffice.org gsi file"""
    #TODO: implement caching
    templateoo = self.oootemplate()
    if templateoo is None:
        return
    outputoo = os.path.join(self.podir, self.languagecode + ".sdf")
    inputdir = os.path.join(self.potree.podirectory, self.projectcode, self.languagecode)
    po2oo.main(["-i%s"%inputdir, "-t%s"%templateoo, "-o%s"%outputoo, "-l%s"%self.languagecode, "--progress=none"])
    return file(os.path.join(self.podir, self.languagecode + ".sdf"), "r").read()

def browsefiles(self, dirfilter=None, depth=None, maxdepth=None, includedirs=False, includefiles=True):
    """gets a list of pofilenames, optionally filtering with the parent directory"""
    if dirfilter is None:
        pofilenames = self.pofilenames
    else:
        if not dirfilter.endswith(os.path.sep) and not dirfilter.endswith(os.extsep + self.fileext):
            dirfilter += os.path.sep
        pofilenames = [pofilename for pofilename in self.pofilenames if pofilename.startswith(dirfilter)]
    if includedirs:
        podirs = {}
        for pofilename in pofilenames:
            dirname = os.path.dirname(pofilename)
            if not dirname:
                continue
            podirs[dirname] = True
            while dirname:
                dirname = os.path.dirname(dirname)
                if dirname:
                    podirs[dirname] = True
        podirs = podirs.keys()
    else:
        podirs = []
    if not includefiles:
        pofilenames = []
    if maxdepth is not None:
        pofilenames = [pofilename for pofilename in pofilenames if pofilename.count(os.path.sep) <= maxdepth]
        podirs = [podir for podir in podirs if podir.count(os.path.sep) <= maxdepth]
    if depth is not None:
        pofilenames = [pofilename for pofilename in pofilenames if pofilename.count(os.path.sep) == depth]
        podirs = [podir for podir in podirs if podir.count(os.path.sep) == depth]
    return pofilenames + podirs

def iterpofilenames(self, lastpofilename=None, includelast=False):
    """iterates through the pofilenames starting after the given pofilename"""
    if not lastpofilename:
        index = 0
    else:
        index = self.pofilenames.index(lastpofilename)
    if not includelast:
        index += 1
    while index < len(self.pofilenames):
        yield self.pofilenames[index]
        index += 1

def initindex(self):
    """initializes the search index"""
    if not indexer.HAVE_INDEXER:
        return
    self.indexdir = os.path.join(self.podir, ".poindex-%s-%s" % (self.projectcode, self.languagecode))
    class indexconfig:
        indexdir = self.indexdir
    self.analyzer = indexer.PerFieldAnalyzer(["pofilename", indexer.ExactAnalyzer()])
    self.indexer = indexer.Indexer(indexconfig, analyzer=self.analyzer)
    self.searcher = indexer.Searcher(self.indexdir, analyzer=self.analyzer)
    pofilenames = self.pofilenames.keys()
    pofilenames.sort()
    for pofilename in pofilenames:
        self.updateindex(pofilename, optimize=False)
    self.indexer.optimizeIndex()

def updateindex(self, pofilename, items=None, optimize=True):
    """updates the index with the contents of pofilename (limit to items if given)"""

```

```

if not indexer.HAVE_INDEXER:
    return
needsupdate = True
pofile = self.pofiles[pofilename]
# check if the pomtime in the index == the latest pomtime
pomtime = pootlefile.getmodtime(pofile.filename)
pofilenamequery = self.searcher.makeQuery([("pofilename", pofilename)], True)
pomtimequery = self.searcher.makeQuery([("pomtime", str(pomtime))], True)
if items is not None:
    itemsquery = self.searcher.makeQuery([("itemno", str(itemno)) for itemno in items], False)
gooditemsquery = self.searcher.makeQuery([pofilenamequery, pomtimequery], True)
gooditems = self.searcher.search(gooditemsquery, "itemno")
allitems = self.searcher.search(pofilenamequery, "itemno")
if items is None:
    if len(gooditems) == len(allitems) == pofile.getitemslen():
        return
    print "updating", self.projectcode, self.languagecode, "index for", pofilename
    self.searcher.deleteDoc({"pofilename": pofilename})
else:
    print "updating", self.languagecode, "index for", pofilename, "items", items
    self.searcher.deleteDoc([pofilenamequery, itemsquery])
pofile.pofreshen()
addlist = []
if items is None:
    items = range(len(pofile.transelements))
for itemno in items:
    thepo = pofile.transelements[itemno]
    doc = {"pofilename": pofilename, "pomtime": str(pomtime), "itemno": str(itemno)}
    if thepo.hasplural():
        orig = "\n".join(thepo.source.strings)
        trans = "\n".join(thepo.target.strings)
    else:
        orig = thepo.source
        trans = thepo.target
    doc["msgid"] = orig
    doc["msgstr"] = trans
    addlist.append(doc)
if addlist:
    self.indexer.startIndex()
    try:
        self.indexer.indexFields(addlist)
    finally:
        self.indexer.commitIndex(optimize=optimize)

def matchessearch(self, pofilename, search):
    """returns whether any items in the pofilename match the search (based on collected stats etc)"""
    if search.dirfilter is not None and not pofilename.startswith(search.dirfilter):
        return False
    # search.assignedto == [None] means assigned to nobody
    if search.assignedto or search.assignedaction:
        if search.assignedto == [None]:
            assigns = self.pofiles[pofilename].getunassigned(search.assignedaction)
        else:
            assigns = self.pofiles[pofilename].getassigns()
        if search.assignedto is not None:
            if search.assignedto not in assigns:
                return False
            assigns = assigns[search.assignedto]
        else:
            assigns = reduce(lambda x, y: x+y, [userassigns.keys() for userassigns in assigns.values()], [])
        if search.assignedaction is not None:
            if search.assignedaction not in assigns:
                return False
    if search.matchnames:
        postats = self.getpostats(pofilename)
        matches = False
        for name in search.matchnames:
            if postats[name]:
                matches = True
        if not matches:
            return False
    return True

```

```

def indexsearch(self, search, returnfields):
    """returns the results from searching the index with the given search"""
    if not indexer.HAVE_INDEXER:
        return False
    searchparts = []
    if search.searchtext:
        textquery = self.searcher.makeQuery(["msgid", search.searchtext], ("msgstr", search.searchtext), False)
        searchparts.append(textquery)
    if search.dirfilter:
        pofilenames = self.browsefiles(dirfilter=search.dirfilter)
        filequery = self.searcher.makeQuery(["pofilename", pofilename] for pofilename in pofilenames), False)
        searchparts.append(filequery)
    # TODO: add other search items
    limitedquery = self.searcher.makeQuery(searchparts, True)
    return self.searcher.search(limitedquery, returnfields)

def searchpofilenames(self, lastpofilename, search, includelast=False):
    """find the next pofilename that has items matching the given search"""
    if lastpofilename and not lastpofilename in self.pofiles:
        # accessing will autoload this file...
        self.pofiles[lastpofilename]
    if indexer.HAVE_INDEXER and search.searchtext:
        # TODO: move this up a level, use index to manage whole search, so we don't do this twice
        hits = self.indexsearch(search, "pofilename")
        print "ran search %s, got %d hits" % (search.searchtext, len(hits))
        searchpofilenames = dict.fromkeys([hit["pofilename"] for hit in hits])
    else:
        searchpofilenames = None
    for pofilename in self.iterpofilenames(lastpofilename, includelast):
        if searchpofilenames is not None:
            if pofilename not in searchpofilenames:
                continue
        if self.matchessearch(pofilename, search):
            yield pofilename

def searchpoitems(self, pofilename, item, search):
    """finds the next item matching the given search"""
    if search.searchtext:
        pogrepfilter = pogrep.pogrepfilter(search.searchtext, None, ignorecase=True)
    for pofilename in self.searchpofilenames(pofilename, search, includelast=True):
        pofile = self.getpofile(pofilename)
        if indexer.HAVE_INDEXER:
            filesearch = search.copy()
            filesearch.dirfilter = pofilename
            hits = self.indexsearch(filesearch, "itemno")
            items = [int(doc["itemno"]) for doc in hits]
            items = [searchitem for searchitem in items if searchitem > item]
            items.sort()
            notextsearch = search.copy()
            notextsearch.searchtext = None
            matchitems = list(pofile.iteritems(notextsearch, item))
        else:
            items = pofile.iteritems(search, item)
            matchitems = items
        for item in items:
            if items != matchitems:
                if item not in matchitems:
                    continue
            # TODO: move this to iteritems
            if search.searchtext:
                thepo = pofile.transelements[item]
                if pogrepfilter.filterelement(thepo):
                    yield pofilename, item
            else:
                yield pofilename, item
        item = None

def reassignpoitems(self, session, search, assignto, action):
    """reassign all the items matching the search to the assignto user(s) evenly, with the given action"""
    # remove all assignments for the given action
    self.unassignpoitems(session, search, None, action)
    assigncount = self.assignpoitems(session, search, assignto, action)
    return assigncount

```

```

def assignpoitems(self, session, search, assignto, action):
    """assign all the items matching the search to the assignto user(s) evenly, with the given action"""
    if not "assign" in self.getrights(session):
        raise RightsError(session.localize("You do not have rights to alter assignments here"))
    if search.searchtext:
        pogrepfilter = pogrep.pogrepfilter(search.searchtext, None, ignorecase=True)
    if not isinstance(assignto, list):
        assignto = [assignto]
    usercount = len(assignto)
    assigncount = 0
    if not usercount:
        return assigncount
    docountwords = lambda pofilename: self.countwords([(pofilename, item) for item in range(self.pofiles[pofilename].getitemslen())])
    pofilenames = [pofilename for pofilename in self.searchpofilenames(None, search, includelast=True)]
    wordcounts = [(pofilename, docountwords(pofilename)) for pofilename in pofilenames]
    totalwordcount = sum([wordcount for pofilename, wordcount in wordcounts])

    wordsperuser = totalwordcount / usercount
    print "assigning", totalwordcount, "words to", usercount, "user(s)", wordsperuser, "words per user"
    usernum = 0
    userwords = 0
    for pofilename, wordcount in wordcounts:
        pofile = self.getpofile(pofilename)
        for item in pofile.iteritems(search, None):
            # TODO: move this to iteritems
            if search.searchtext:
                validitem = False
                thepo = pofile.transelements[item]
                if pogrepfilter.filterelement(thepo):
                    validitem = True
                if not validitem:
                    continue
            itemwordcount = self.countwords([(pofilename, item)])
            if userwords + itemwordcount > wordsperuser:
                usernum = min(usernum+1, len(assignto)-1)
                userwords = 0
            userwords += itemwordcount
            pofile.assignto(item, assignto[usernum], action)
            assigncount += 1
    return assigncount

def unassignpoitems(self, session, search, assignedto, action=None):
    """unassigns all the items matching the search to the assignedto user"""
    if not "assign" in self.getrights(session):
        raise RightsError(session.localize("You do not have rights to alter assignments here"))
    if search.searchtext:
        pogrepfilter = pogrep.pogrepfilter(search.searchtext, None, ignorecase=True)
    assigncount = 0
    for pofilename in self.searchpofilenames(None, search, includelast=True):
        pofile = self.getpofile(pofilename)
        for item in pofile.iteritems(search, None):
            # TODO: move this to iteritems
            if search.searchtext:
                thepo = pofile.transelements[item]
                if pogrepfilter.filterelement(thepo):
                    pofile.unassign(item, assignedto, action)
                    assigncount += 1
            else:
                pofile.unassign(item, assignedto, action)
                assigncount += 1
    return assigncount

def updatequickstats(self, pofilename, translatedwords, translated, totalwords, total):
    """updates the quick stats on the given file"""
    self.quickstats[pofilename] = (translatedwords, translated, totalwords, total)
    self.savequickstats()

def savequickstats(self):
    """saves the quickstats"""
    self.quickstatsfilename = os.path.join(self.podir, "pootle-%s-%s.stats" % (self.projectcode, self.languagecode))
    quickstatsfile = open(self.quickstatsfilename, "w")
    sortedquickstats = self.quickstats.items()

```

```

sortedquickstats.sort()
for pofilename, (translatedwords, translated, totalwords, total) in sortedquickstats:
    quickstatsfile.write("%s, %d, %d, %d\n" % (pofilename, translatedwords, translated, totalwords, total))
quickstatsfile.close()

def readquickstats(self):
    """reads the quickstats from disk"""
    self.quickstats = {}
    self.quickstatsfilename = os.path.join(self.podir, "pootle-%s-%s.stats" % (self.projectcode, self.languagecode))
    if os.path.exists(self.quickstatsfilename):
        quickstatsfile = open(self.quickstatsfilename, "r")
        for line in quickstatsfile:
            pofilename, translatedwords, translated, totalwords, total = line.split(",")
            self.quickstats[pofilename] = tuple([int(a.strip()) for a in translatedwords, translated, totalwords, total])

def getquickstats(self, pofilenames=None):
    """gets translated and total stats and wordcounts without doing calculations returning dictionary"""
    if pofilenames is None:
        pofilenames = self.pofilenames
    alltranslatedwords, alltranslated, alltotalwords, alltotal = 0, 0, 0, 0
    slowfiles = []
    for pofilename in pofilenames:
        if pofilename not in self.quickstats:
            slowfiles.append(pofilename)
            continue
        translatedwords, translated, totalwords, total = self.quickstats[pofilename]
        alltranslatedwords += translatedwords
        alltranslated += translated
        alltotalwords += totalwords
        alltotal += total
    for pofilename in slowfiles:
        self.pofiles[pofilename].updatequickstats()
        translatedwords, translated, totalwords, total = self.quickstats[pofilename]
        alltranslatedwords += translatedwords
        alltranslated += translated
        alltotalwords += totalwords
        alltotal += total
    return {"translatedwords": alltranslatedwords, "translated": alltranslated, "totalwords": alltotalwords, "total": alltotal}

def combinestats(self, pofilenames=None):
    """combines translation statistics for the given po files (or all if None given)"""
    totalstats = {}
    if pofilenames is None:
        pofilenames = self.pofilenames
    for pofilename in pofilenames:
        if not pofilename or os.path.isdir(pofilename):
            continue
        postats = self.getpostats(pofilename)
        for name, items in postats.iteritems():
            totalstats[name] = totalstats.get(name, []) + [(pofilename, item) for item in items]
    assignstats = self.combineassignstats(pofilenames)
    totalstats.update(assignstats)
    return totalstats

def combineassignstats(self, pofilenames=None, action=None):
    """combines assign statistics for the given po files (or all if None given)"""
    totalstats = {}
    if pofilenames is None:
        pofilenames = self.pofilenames
    for pofilename in pofilenames:
        assignstats = self.getassignstats(pofilename, action)
        for name, items in assignstats.iteritems():
            totalstats["assign-"+name] = totalstats.get("assign-"+name, []) + [(pofilename, item) for item in items]
    return totalstats

def countwords(self, stats):
    """counts the number of words in the items represented by the stats list"""
    wordcount = 0
    for pofilename, item in stats:
        pofile = self.pofiles[pofilename]
        if 0 <= item < len(pofile.msgidwordcounts):
            wordcount += sum(pofile.msgidwordcounts[item])
    return wordcount

```

```

def track(self, pofilename, item, message):
    """sends a track message to the pofile"""
    self.pofiles[pofilename].track(item, message)

def gettracks(self, pofilenames=None):
    """calculates translation statistics for the given po files (or all if None given)"""
    alltracks = []
    if pofilenames is None:
        pofilenames = self.pofilenames
    for pofilename in pofilenames:
        if not pofilename or os.path.isdir(pofilename):
            continue
        tracker = self.pofiles[pofilename].tracker
        items = tracker.keys()
        items.sort()
        for item in items:
            alltracks.append("%s item %d: %s" % (pofilename, item, tracker[item]))
    return alltracks

def getpoststats(self, pofilename):
    """calculates translation statistics for the given po file"""
    return self.pofiles[pofilename].getstats()

def getassignstats(self, pofilename, action=None):
    """calculates translation statistics for the given po file (can filter by action if given)"""
    polen = len(self.getpoststats(pofilename)["total"])
    assigns = self.pofiles[pofilename].getassigns()
    assignstats = {}
    for username, userassigns in assigns.iteritems():
        allitems = []
        for assignaction, items in userassigns.iteritems():
            if action is None or assignaction == action:
                allitems += [item for item in items if 0 <= item < polen and item not in allitems]
        if allitems:
            assignstats[username] = allitems
    return assignstats

def getpofile(self, pofilename, freshen=True):
    """parses the file into a pofile object and stores in self.pofiles"""
    pofile = self.pofiles[pofilename]
    if freshen:
        pofile.pofreshen()
    return pofile

def getpofilelen(self, pofilename):
    """returns number of items in the given pofilename"""
    # TODO: needn't parse the file for this ...
    pofile = self.getpofile(pofilename)
    return len(pofile.transelements)

def getitem(self, pofilename, item):
    """returns a particular item from a particular po file's orig, trans strings as a tuple"""
    pofile = self.getpofile(pofilename)
    thepo = pofile.transelements[item]
    orig, trans = thepo.unquotedmsgid[0], thepo.unquotedmsgstr[0]
    return orig, trans

##MODIFIED Achilles
def getitemtranscomments(self, pofilename, item):
    """return comments of an item. those are used to display any additional translation information"""
    # Note: only translator comments and automatic comments are returned
    pofile = self.getpofile(pofilename)
    thepo = pofile.transelements[item]
    comments = [comment.strip("#") for comment in thepo.othercomments]
    comments.extend([comment.strip("#.") for comment in thepo.automaticcomments])
    return comments

def getitemtypecomments(self, pofilename, item):
    pofile = self.getpofile(pofilename)
    thepo = pofile.transelements[item]
    return thepo.typecomments
##END MODIFICATION

```

```

def getitemclasses(self, pofilename, item):
    """returns which classes this item belongs to"""
    # TODO: needn't parse the file for this ...
    pofile = self.getpofile(pofilename)
    return [classname for (classname, classitems) in pofile.classify.iteritems() if item in classitems]

def unquotefrompo(self, postr):
    """extracts a po-quoted string to normal text"""
    if isinstance(postr, dict):
        quotedpo = {}
        for pluralid in postr:
            quotedpo[pluralid] = self.unquotefrompo(postr[pluralid])
        return quotedpo
    else:
        return po.unquotefrompo(postr)

def getitems(self, pofilename, itemstart, itemstop):
    """returns a set of items from the pofile, converted to original and translation strings"""
    pofile = self.getpofile(pofilename)
    elements = pofile.transelements[max(itemstart,0):itemstop]
    return elements

def updatetranslation(self, pofilename, item, trans, session):
    """updates a translation with a new value..."""
    if "translate" not in self.getrights(session):
        raise RightsError(session.localize("You do not have rights to change translations here"))
    pofile = self.pofiles[pofilename]
    pofile.track(item, "edited by %s" % session.username)
    languageprefs = getattr(session.instance.languages, self.languagecode, None)
    pofile.setmsgstr(item, trans, session.prefs, languageprefs)
    self.updateindex(pofilename, [item])

def suggesttranslation(self, pofilename, item, trans, session):
    """stores a new suggestion for a translation..."""
    if "suggest" not in self.getrights(session):
        raise RightsError(session.localize("You do not have rights to suggest changes here"))
    pofile = self.getpofile(pofilename)
    pofile.track(item, "suggestion made by %s" % session.username)
    pofile.addsuggestion(item, trans, session.username)

def getsuggestions(self, pofile, item):
    """find all the suggestions submitted for the given (pofile or pofilename) and item"""
    if isinstance(pofile, (str, unicode)):
        pofilename = pofile
        pofile = self.getpofile(pofilename)
    suggestpos = pofile.getsuggestions(item)
    return suggestpos

def acceptsuggestion(self, pofile, item, suggitem, newtrans, session):
    """accepts the suggestion into the main pofile"""
    if not "review" in self.getrights(session):
        raise RightsError(session.localize("You do not have rights to review suggestions here"))
    if isinstance(pofile, (str, unicode)):
        pofilename = pofile
        pofile = self.getpofile(pofilename)
    pofile.track(item, "suggestion by %s accepted by %s" % (self.getsuggester(pofile, item, suggitem), session.username))
    pofile.deletesuggestion(item, suggitem)
    self.updatetranslation(pofilename, item, newtrans, session)

def getsuggester(self, pofile, item, suggitem):
    """returns who suggested the given item's suggitem if recorded, else None"""
    if isinstance(pofile, (str, unicode)):
        pofilename = pofile
        pofile = self.getpofile(pofilename)
    suggestionpo = pofile.getsuggestions(item)[suggitem]
    for msgidcomment in suggestionpo.msgidcomments:
        if msgidcomment.find("suggested by ") != -1:
            suggestedby = po.unquotefrompo([msgidcomment]).replace(":", " ", 1).replace("suggested by ", " ", 1).strip()
            return suggestedby
    return None

def rejectsuggestion(self, pofile, item, suggitem, newtrans, session):

```



```

"""rejects the suggestion and removes it from the pending file"""
if not "review" in self.getrights(session):
    raise RightsError(session.localize("You do not have rights to review suggestions here"))
if isinstance(pofile, (str, unicode)):
    pofilename = pofile
    pofile = self.getpofile(pofilename)
pofile.track(item, "suggestion by %s rejected by %s" % (self.getsuggester(pofile, item, suggitem), session.username))
pofile.deletesuggestion(item, suggitem)

def savepofile(self, pofilename):
    """saves changes to disk..."""
    pofile = self.getpofile(pofilename)
    pofile.savepofile()

def getoutput(self, pofilename):
    """returns pofile source"""
    pofile = self.getpofile(pofilename)
    return pofile.getoutput()

def convert(self, pofilename, destformat):
    """converts the pofile to the given format, returning (etag_if_filepath, filepath_or_contents)"""
    destfilename = pofilename[:-len(self.fileext)] + destformat
    pofile = self.getpofile(pofilename, freshen=False)
    destfilename = pofile.filename[:-len(self.fileext)] + destformat
    destmtime = pootlefile.getmodtime(destfilename)
    pomtime = pootlefile.getmodtime(pofile.filename)
    if pomtime and destmtime == pomtime:
        try:
            return pomtime, destfilename
        except Exception, e:
            print "error reading cached converted file %s: %s" % (destfilename, e)
    pofile.pofreshen()
    converters = {"csv": po2csv.po2csv, "xlf": po2xliff.po2xliff, "ts": po2ts.po2ts, "mo": pocompile.P0Compile}
    converterclass = converters.get(destformat, None)
    if converterclass is None:
        raise ValueError("No converter available for %s" % destfilename)
    contents = converterclass().convertfile(pofile)
    if not isinstance(contents, basestring):
        contents = str(contents)
    try:
        destfile = open(destfilename, "w")
        destfile.write(contents)
        destfile.close()
        currenttime, modtime = time.time(), pofile.pomtime
        os.utime(destfilename, (currenttime, modtime))
        return modtime, destfilename
    except Exception, e:
        print "error caching converted file %s: %s" % (destfilename, e)
    return False, contents

def gettext(self, message):
    """uses the project as a live translator for the given message"""
    for pofilename, pofile in self.pofiles.iteritems():
        if pofile.pomtime != pootlefile.getmodtime(pofile.filename):
            pofile.readpofile()
            pofile.makeindex()
        elif not hasattr(pofile, "msgidindex"):
            pofile.makeindex()
        thepo = pofile.msgidindex.get(message, None)
        if not thepo or thepo.isblankmsgstr():
            continue
        tmsg = thepo.target
        if tmsg is not None:
            return tmsg
    return message

def ugettext(self, message):
    """gets the translation of the message by searching through all the pofiles (unicode version)"""
    for pofilename, pofile in self.pofiles.iteritems():
        try:
            if pofile.pomtime != pootlefile.getmodtime(pofile.filename):
                pofile.readpofile()
                pofile.makeindex()

```

```

        elif not hasattr(pofile, "msgidindex"):
            pofile.makeindex()
        thepo = pofile.msgidindex.get(message, None)
        if not thepo or thepo.isblankmsgstr() or thepo.isfuzzy():
            continue
        tmsg = thepo.target
        if tmsg is not None:
            if isinstance(tmsg, unicode):
                return tmsg
            else:
                return unicode(tmsg, pofile.encoding)
    except Exception, e:
        print "error reading translation from pofile %s: %s" % (pofile, e)
    return unicode(message)

def ungettext(self, singular, plural, n):
    """gets the plural translation of the message by searching through all the pofiles (unicode version)"""
    for pofile, pofile in self.pofiles.iteritems():
        try:
            if pofile.potime != pootlefile.getmtime(pofile.filename):
                pofile.readpofile()
                pofile.makeindex()
            elif not hasattr(pofile, "msgidindex"):
                pofile.makeindex()
            nplural, pluralequation = pofile.getheaderplural()
            if pluralequation:
                pluralfn = gettext.c2py(pluralequation)
                thepo = pofile.msgidindex.get(singular, None)
                if not thepo or thepo.isblankmsgstr() or thepo.isfuzzy():
                    continue
                tmsg = po.unquotefrompo(thepo.msgstr[pluralfn(n)])
                if tmsg is not None:
                    if isinstance(tmsg, unicode):
                        return tmsg
                    else:
                        return unicode(tmsg, pofile.encoding)
            except Exception, e:
                print "error reading translation from pofile %s: %s" % (pofile, e)
        if n == 1:
            return unicode(singular)
        else:
            return unicode(plural)

def hascreatemofiles(self, projectcode):
    """returns whether the project has createmofile set"""
    return self.potree.getprojectcreatemofiles(projectcode) == 1

class DummyProject(TranslationProject):
    """a project that is just being used for handling pootlefiles"""
    def __init__(self, podir, checker=None, projectcode=None, languagecode=None):
        """initializes the project with the given podir"""
        self.podir = podir
        if checker is None:
            self.checker = pofilter.POTeeChecker()
        else:
            self.checker = checker
        self.projectcode = projectcode
        self.languagecode = languagecode
        self.readquickstats()

    def scanpofiles(self):
        """A null operation if potree is not present"""
        pass

    def readquickstats(self):
        """dummy statistics are empty"""
        self.quickstats = {}

    def savequickstats(self):
        """saves quickstats if possible"""
        pass

class DummyStatsProject(DummyProject):

```

```

"""a project that is just being used for refresh of statistics"""
def __init__(self, podir, checker, projectcode=None, languagecode=None):
    """initializes the project with the given podir"""
    DummyProject.__init__(self, podir, checker, projectcode, languagecode)

def readquickstats(self):
    """reads statistics from whatever files are available"""
    self.quickstats = {}
    if self.projectcode is not None and self.languagecode is not None:
        TranslationProject.readquickstats(self)

def savequickstats(self):
    """saves quickstats if possible"""
    if self.projectcode is not None and self.languagecode is not None:
        TranslationProject.savequickstats(self)

class TemplatesProject(TranslationProject):
    """Manages Template files (.pot files) for a project"""
    fileext = "pot"
    def __init__(self, projectcode, potree):
        super(TemplatesProject, self).__init__("templates", projectcode, potree, create=False)

    def getrights(self, session=None, username=None, usedefaults=True):
        """gets the rights for the given user (name or session, or not-logged-in if username is None)"""
        # internal admin sessions have all rights
        rights = super(TemplatesProject, self).getrights(session=session, username=username)
        if rights is not None:
            rights = [right for right in rights if right not in ["translate", "suggest", "pocompile"]]
        return rights

##MODIFIED Achilles
# Code borrowed from Wikisource
# (http://en.wikisource.org/wiki/Levenshtein\_distance#Python)
# Author: Magnus Lie Hetland; Many, many thanks to him.
def distance(a, b):
    """Calculates the Levenshtein distance between a and b."""
    n, m = len(a), len(b)
    if n > m:
        # Make sure n <= m, to use O(min(n,m)) space
        a,b = b,a
        n,m = m,n

    current = range(n+1)
    for i in range(1,m+1):
        previous, current = current, [i]+[0]*n
        for j in range(1,n+1):
            add, delete = previous[j]+1, current[j-1]+1
            change = previous[j-1]
            if a[j-1] != b[i-1]:
                change = change + 1
            current[j] = min(add, delete, change)

    return current[n]

class TranslationMemory:
    """Translation Memory associated with a project."""
    # In my view, only one TM is necessary per project. It could match strings
    # for all po files the project contains.
    def __init__(self, tmfilename, translang, srclang="en"):
        self.tmfilename = tmfilename
        self.srclang = srclang
        self.translang = translang
        self.tmfile = tmx1.TMXHandler(self.tmfilename, self.srclang)

    def updatetranslation(self, source, trans):
        """Adds a new translation.

        Keyword arguments:
            source -- the source string
            trans -- its corresponding translation in translang set on __init__

        """
        self.tmfile.addtranslation(source, self.srclang, trans, self.translang)

```

```

self.tmfile.savefile()

def getmatches(self, source="", percent=60):
    """Returns a list of strings that match 'source' in 'percent' % or more.

    Keyword arguments:
        source -- the source string
        percent -- the minimum percent strings must be similar. default: 60

    'percent' should, obviously, be a number.

    """
    # TODO: strip punctuation from strings (tricky)
    # TODO: handle TMX internal tags (display them in a way the user understands)
    matches = []
    tmxsources = self.tmfile.getsources()
    for src in tmxsources:
        maxlen, ld = max(len(source), len(src)), distance(source.lower(), src.lower())
        similarity = (100 * (maxlen - ld)) / maxlen
        if similarity >= percent:
            matches.append((similarity, src, self.tmfile.gettranslation(src, self.translang)))
    matches.sort()
    matches.reverse()
    return matches

class Termbase:
    """Termbase associated with a project."""
    def __init__(self, tbfilename, translang, srclang="en"):
        self.tbfilename = tbfilename
        self.srclang = srclang
        self.translang = translang
        self.tbfile = tbx1.TBXHandler(self.tbfilename, self.srclang)

    def updatetermtranslation(self, term, trans):
        """Adds a new term translation.

        Keyword arguments:
            term -- the source string
            trans -- its corresponding translation in translang set on __init__

        """
        self.tbfile.addterm(term, self.srclang, trans, self.translang)
        self.tbfile.savefile()

    def getmatches(self, term=""):
        """Returns a list of strings that contain the term searched.

        Keyword arguments:
            source -- the source string
            percent -- the minimum percent strings must be similar. default: 20

        'percent' should, obviously, be a number.

        """
        matches = []
        tbxsources = self.tbfile.getsources()
        for srcterm in tbxsources: # Simpler search. Strings are too little to do Levenshtein
            if len(term) < len(srcterm):
                term1 = term
                term2 = srcterm
            else:
                term1 = srcterm
                term2 = term
            if term1 in term2:
                matches.extend(self.tbfile.gettermtranslation(srcterm, self.translang))
        return matches
##END MODIFICATION

```

ANEXO H – translatepage.py

Este arquivo foi apenas **modificado** pelo autor, para implementar as mudanças propostas! Todos os trechos de código modificado são envoltos pelos comentários **##MODIFIED Achilles** e **##END MODIFICATION**. Todo o restante do código é de autoria dos desenvolvedores do Pootle.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# Copyright 2004-2006 Zuza Software Foundation
#
# This file is part of translate.
#
# translate is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# translate is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with translate; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

import sre
from jToolkit import spellcheck
from Pootle import pagelayout
from Pootle import projects
from Pootle import pootlefile
import difflib
import urllib

def oddoreven(polarity):
    if polarity % 2 == 0:
        return "even"
    elif polarity % 2 == 1:
        return "odd"

##MODIFIED Achilles
def gethighlighter(syntax):
    if syntax.find("python-format") != -1:
        return sre.compile("(%[cd]uixspr}{1}|%l[du]{1}|%z[du]{1}")
    elif syntax.find("c-format") != -1:
        return sre.compile("%[cd]uixspoXfeEgGn}{1}") # really basic support, should be revised
    else:
        return None
##END MODIFICATION

class TranslatePage(pagelayout.PootleNavPage):
```

```

"""the page which lets people edit translations"""
def __init__(self, project, session, argdict, dirfilter=None):
    self.argdict = argdict
    self.dirfilter = dirfilter
    self.project = project
    ##MODIFIED Achilles
    # Really really special case that we don't need all that processing
    if 'gettmatches' in self.argdict and 'searchterm' in self.argdict:
        searchterm = self.argdict.get("searchterm", "")
        if isinstance(searchterm, str):
            searchterm = searchterm.decode("utf8")
        templatename = "translatepagetb"
        if searchterm:
            templatevars = { "tmatches": self.project.termbase.getmatches(searchterm) }
        else:
            templatevars = { "tmatches": [] }
        pagelayout.PootleNavPage.__init__(self, templatename, templatevars, session, bannerheight=0)
        return
    ##END MODIFICATION
    self.matchnames = self.getmatchnames(self.project.checker)
    self.searchtext = self.argdict.get("searchtext", "")
    # TODO: fix this in jToolkit
    if isinstance(self.searchtext, str):
        self.searchtext = self.searchtext.decode("utf8")
    self.showassigns = self.argdict.get("showassigns", 0)
    if isinstance(self.showassigns, (str, unicode)) and self.showassigns.isdigit():
        self.showassigns = int(self.showassigns)
    self.session = session
    self.localize = session.localize
    self.rights = self.project.getrights(self.session)
    self.instance = session.instance
    self.lastitem = None
    self.pofilename = self.argdict.pop("pofilename", None)
    if self.pofilename == "":
        self.pofilename = None
    if self.pofilename is None and self.dirfilter is not None and self.dirfilter.endswith(".po"):
        self.pofilename = self.dirfilter
    self.receivetranslations()
    # TODO: clean up modes to be one variable
    self.viewmode = self.argdict.get("view", 0) and "view" in self.rights
    self.reviewmode = self.argdict.get("review", 0)
    notice = {}
    try:
        self.finditem()
    except StopIteration, stoppedby:
        notice = self.getfinishedtext(stoppedby)
        self.item = None
    ##MODIFIED Achilles
    if "gettmatches" in argdict:
        templatename = "translatepagetm"
        templatevars = {}
        if self.item is not None:
            templatevars["tmatches"] = self.gettmatches(self.item)
            templatevars["additionalinfo"] = self.getadditionalinformation(self.item)
        pagelayout.PootleNavPage.__init__(self, templatename, templatevars, session, bannerheight=0)
        return
    ##END MODIFICATION
    items = self.maketable()
    # self.pofilename can change in search...
    givenpofilename = self.pofilename
    formation = self.makelink("")
    title = self.localize("Pootle: translating %s into %s: %s", self.project.projectname, self.project.languagename, self.pofilename)
    mainstats = ""
    if self.pofilename is not None:
        poststats = self.project.getpoststats(self.pofilename)
        blank, fuzzy = poststats["blank"], poststats["fuzzy"]
        translated, total = poststats["translated"], poststats["total"]
        mainstats = self.localize("%d/%d translated\n%d blank, %d fuzzy)", len(translated), len(total), len(blank), len(fuzzy))
    if self.viewmode:
        rows = self.getdisplayrows("view")
        pagelinks = self.getpagelinks("?translate=1&view=1", rows)
        icon="file"
    else:

```

```

pagelinks = []
icon="edit"
navbarpath_dict = self.makenavbarpath_dict(self.project, self.session, self.pofilename)
# templatising
templatename = "translatepage"
pagetitle = self.localize("Pootle: translating %s into %s: %s", self.project.projectname, self.project.languagecode, self.pofilename)
instancetitle = getattr(session.instance, "title", session.localize("Pootle Demo"))
sessionvars = {"status": session.status, "isopen": session.isopen, "issiteadmin": session.issiteadmin()}
stats = {"summary": mainstats, "checks": [], "tracks": [], "assigns": []}
templatevars = {"pagetitle": pagetitle,
                "project": {"code": self.project.projectcode, "name": self.project.projectname},
                "language": {"code": self.project.languagecode, "name": self.project.languagecode},
                "pofilename": self.pofilename,
                # navigation bar
                "navitems": [{"icon": "edit", "path": navbarpath_dict, "actions": {}, "stats": stats}],
                "pagelinks": pagelinks,
                # translation form
                "actionurl": formaction,
                "notice": notice,
                "original_title": self.localize("Original"),
                "translation_title": self.localize("Translation"),
                "items": items,
                "viewmode": self.viewmode,
                "reviewmode": self.reviewmode,
                "accept_button": self.localize("Accept"),
                "reject_button": self.localize("Reject"),
                # optional sections, will appear if these values are replaced
                "assign": None,
                "search": {"title": self.localize("Search")},
                # hidden widgets
                "searchtext": self.searchtext,
                "pofilename": givenpofilename,
                # general vars
                "session": sessionvars, "instancetitle": pagetitle,
                ##MODIFIED Achilles
                "additionalinfo": { "title": self.localize("Additional Information"),
                                   "information": self.getadditionalinformation(None) },
                "termsearch": { "title": self.localize("Search termbase"),
                                "usage": self.localize("Type a term in source language and press \enter"),
                                "answer": [] }
                ##END MODIFICATION
if self.showassigns and "assign" in self.rights:
    templatevars["assign"] = self.getassignbox()
##MODIFIED Achilles
searchterm = self.argdict.get("searchterm", "")
if isinstance(searchterm, str):
    searchterm = searchterm.decode("utf8")
if searchterm:
    matches = self.project.termbase.getmatches(searchterm)
    templatevars["termsearch"]["answer"] = matches
##END MODIFICATION
pagelayout.PootleNavPage.__init__(self, templatename, templatevars, session, bannerheight=81)
self.addfilelinks(self.pofilename, self.matchnames)

def getfinishedtext(self, stoppedby):
    """gets notice to display when the translation is finished"""
    title = self.localize("End of batch")
    finishedlink = "index.html?" + "&".join(["%s=%s" % (arg, value) for arg, value in self.argdict.iteritems() if arg.startswith("show")])
    returnlink = self.localize("Click here to return to the index")
    stoppedbytext = stoppedby.args[0]
    return {"title": title, "stoppedby": stoppedbytext, "finishedlink": finishedlink, "returnlink": returnlink}

##MODIFIED Achilles
def gettmatches(self, item):
    """returns matches from translation memory, if any"""
    if not self.viewmode:
        matches = {"orig": [], "trans": []}
        list = [ item ]
        if item is None:
            list = self.editable
        for item in list:
            for (similarity, orig, trans) in self.project.transmemory.getmatches(self.project.getitem(self.pofilename, item)[0]):
                sim = self.localize("%d%% match", similarity)

```

```

        matches["orig"].append({"match": orig, "similarity": sim})
        matches["trans"].append({"match": trans, "similarity": sim})
    return matches
else:
    return {}

def gettbmatches(self, term):
    """returns term matches from termbase, if any"""
    if not self.viewmode:
        matches = []
        list = [ item ]
        if item is None:
            list = self.editable
        for item in list:
            for translation in self.project.termbase.getmatches(term):
                matches.append(translation)
        return matches
    else:
        return {}

def getadditionalinformation(self, item):
    """get translator comments and automatic comments from the po item, to display useful information to the translator"""
    if not self.viewmode:
        info = []
        list = [ item ]
        if item is None:
            list = self.editable
        for item in list:
            info.extend(self.project.getitemtranscomments(self.pofilename, item))
        return info
    else:
        return []
##END MODIFICATION

def getpagelinks(self, baselink, pagesize):
    """gets links to other pages of items, based on the given baselink"""
    pagelinks = []
    pofilelen = self.project.getpofilelen(self.pofilename)
    if pofilelen <= pagesize or self.firstitem is None:
        return pagelinks
    lastitem = min(pofilelen-1, self.firstitem + pagesize - 1)
    if pofilelen > pagesize and not self.firstitem == 0:
        pagelinks.append({"href": baselink + "&item=0", "text": self.localize("Start")})
    else:
        pagelinks.append({"text": self.localize("Start")})
    if self.firstitem > 0:
        linkitem = max(self.firstitem - pagesize, 0)
        pagelinks.append({"href": baselink + "&item=%d" % linkitem, "text": self.localize("Previous %d", (self.firstitem - linkitem))})
    else:
        pagelinks.append({"text": self.localize("Previous %d", pagesize)})
    pagelinks.append({"text": self.localize("Items %d to %d of %d", self.firstitem+1, lastitem+1, pofilelen)})
    if self.firstitem + len(self.translations) < self.project.getpofilelen(self.pofilename):
        linkitem = self.firstitem + pagesize
        itemcount = min(pofilelen - linkitem, pagesize)
        pagelinks.append({"href": baselink + "&item=%d" % linkitem, "text": self.localize("Next %d", itemcount)})
    else:
        pagelinks.append({"text": self.localize("Next %d", pagesize)})
    if pofilelen > pagesize and (self.item + pagesize) < pofilelen:
        pagelinks.append({"href": baselink + "&item=%d" % max(pofilelen - pagesize, 0), "text": self.localize("End")})
    else:
        pagelinks.append({"text": self.localize("End")})
    for n, pagelink in enumerate(pagelinks):
        if n < len(pagelinks)-1:
            pagelink["sep"] = " | "
        else:
            pagelink["sep"] = ""
    return pagelinks

def addfilelinks(self, pofilename, matchnames):
    """adds a section on the current file, including any checks happening"""
    if self.showassigns and "assign" in self.rights:
        self.templatevars["assigns"] = self.getassignbox()
    if self.pofilename is not None:

```



```

if matchnames:
    checknames = [matchname.replace("check-", "", 1) for matchname in matchnames]
    self.templatevars["checking_text"] = self.localize("checking %s", ", ".join(checknames))

def getassignbox(self):
    """gets strings if the user can assign strings"""
    users = [username for username, userprefs in self.session.loginchecker.users.iteritems() if username != "__dummy__"]
    users.sort()
    return {
        "title": self.localize("Assign Strings"),
        "user_title": self.localize("Assign to User"),
        "action_title": self.localize("Assign Action"),
        "submit_text": self.localize("Assign Strings"),
        "users": users,
    }

def receivetranslations(self):
    """receive any translations submitted by the user"""
    if self.pofilename is None:
        return
    skips = []
    submitsuggests = []
    submits = []
    accepts = []
    rejects = []
    ##MODIFIED Achilles
    originals = {}
    ##END MODIFICATION
    translations = {}
    suggestions = {}
    pluralitems = {}
    keymatcher = sre.compile("(\\d+)([0-9.]*)")
    def parsekey(key):
        match = keymatcher.match(key)
        if match:
            keytype, itemcode = match.groups()
            return keytype, itemcode
        return None, None
    def pointsplit(item):
        dotcount = item.count(".")
        if dotcount == 2:
            item, pointitem, subpointitem = item.split(".", 2)
            return int(item), int(pointitem), int(subpointitem)
        elif dotcount == 1:
            item, pointitem = item.split(".", 1)
            return int(item), int(pointitem), None
        else:
            return int(item), None, None
    delkeys = []
    for key, value in self.argdict.iteritems():
        keytype, item = parsekey(key)
        if keytype is None:
            continue
        item, pointitem, subpointitem = pointsplit(item)
        if keytype == "skip":
            skips.append(item)
        elif keytype == "submitsuggest":
            submitsuggests.append(item)
        elif keytype == "submit":
            submits.append(item)
        elif keytype == "accept":
            accepts.append((item, pointitem))
        elif keytype == "reject":
            rejects.append((item, pointitem))
        elif keytype == "trans":
            value = self.unescapesubmission(value)
            if pointitem is not None:
                translations.setdefault(item, {})[pointitem] = value
            else:
                translations[item] = value
        elif keytype == "suggest":
            suggestions.setdefault((item, pointitem), {})[subpointitem] = value
        elif keytype == "orig-pure":

```

```

        # this is just to remove the hidden fields from the argdict
        ##MODIFIED Achilles
        # Not anymore =D
        value = self.unescapesubmission(value)
        originals[item] = value
        ##END MODIFICATION
    else:
        continue
    delkeys.append(key)
for key in delkeys:
    del self.argdict[key]
for item in skips:
    self.lastitem = item
for item in submitsuggests:
    if item in skips or item not in translations:
        continue
    value = translations[item]
    self.project.suggesttranslation(self.pofilename, item, value, self.session)
    self.lastitem = item
for item in submits:
    if item in skips or item not in translations:
        continue
    value = translations[item]
    if isinstance(value, dict) and len(value) == 1 and 0 in value:
        value = value[0]
    self.project.updatetranslation(self.pofilename, item, value, self.session)
    ##MODIFIED Achilles
    self.project.transmemory.updatetranslation(originals[item], value)
    ##END MODIFICATION
    self.lastitem = item
for item, suggid in rejects:
    value = suggestions[item, suggid]
    if isinstance(value, dict) and len(value) == 1 and 0 in value:
        value = value[0]
    self.project.rejectsuggestion(self.pofilename, item, suggid, value, self.session)
    self.lastitem = item
for item, suggid in accepts:
    if (item, suggid) in rejects or (item, suggid) not in suggestions:
        continue
    value = suggestions[item, suggid]
    if isinstance(value, dict) and len(value) == 1 and 0 in value:
        value = value[0]
    self.project.acceptsuggestion(self.pofilename, item, suggid, value, self.session)
    ##MODIFIED Achilles
    self.project.transmemory.updatetranslation(originals[item], value)
    ##END MODIFICATION
    self.lastitem = item

def getmatchnames(self, checker):
    """returns any checker filters the user has asked to match..."""
    matchnames = []
    for checkname in self.argdict:
        if checkname in ["fuzzy", "blank", "translated", "has-suggestion"]:
            matchnames.append(checkname)
        elif checkname in checker.getfilters():
            matchnames.append("check-" + checkname)
    matchnames.sort()
    return matchnames

def getusernode(self):
    """gets the user's prefs node"""
    if self.session.isopen:
        return getattr(self.session.loginchecker.users, self.session.username, None)
    else:
        return None

def finditem(self):
    """finds the focussed item for this page, searching as necessary"""
    item = self.argdict.pop("item", None)
    if item is None:
        try:
            search = pootlefile.Search(dirfilter=self.dirfilter, matchnames=self.matchnames, searchtext=self.searchtext)
            # TODO: find a nicer way to let people search stuff assigned to them (does it by default now)

```

```

    # search.assignedto = self.argdict.get("assignedto", self.session.username)
    search.assignedto = self.argdict.get("assignedto", None)
    search.assignedaction = self.argdict.get("assignedaction", None)
    self.pofilename, self.item = self.project.searchpoitems(self.pofilename, self.lastitem, search).next()
except StopIteration:
    if self.lastitem is None:
        raise StopIteration(self.localize("There are no items matching that search ('%s')", self.searchtext))
    else:
        raise StopIteration(self.localize("You have finished going through the items you selected"))
else:
    if not item.isdigit():
        raise ValueError("Invalid item given")
    self.item = int(item)
    if self.pofilename is None:
        raise ValueError("Received item argument but no pofilename argument")
    self.project.track(self.pofilename, self.item, "being edited by %s" % self.session.username)

def getdisplayrows(self, mode):
    """get the number of rows to display for the given mode"""
    if mode == "view":
        prefsfield = "viewrows"
        default = 10
        maximum = 100
    elif mode == "translate":
        prefsfield = "translaterows"
        default = 7
        maximum = 20
    else:
        raise ValueError("getdisplayrows has no mode '%s'" % mode)
    usernode = self.getusernode()
    rowsdesired = getattr(usernode, prefsfield, default)
    if isinstance(rowsdesired, str):
        if rowsdesired == "":
            rowsdesired = default
        else:
            rowsdesired = int(rowsdesired)
    rowsdesired = min(rowsdesired, maximum)
    return rowsdesired

def gettranslations(self):
    """gets the list of translations desired for the view, and sets editable and firstitem parameters"""
    if self.item is None:
        self.editable = []
        self.firstitem = self.item
        return []
    elif self.viewmode:
        self.editable = []
        self.firstitem = self.item
        rows = self.getdisplayrows("view")
        return self.project.getitems(self.pofilename, self.item, self.item+rows)
    else:
        self.editable = [self.item]
        rows = self.getdisplayrows("translate")
        before = rows / 2
        fromitem = self.item - before
        self.firstitem = max(self.item - before, 0)
        toitem = self.firstitem + rows
        return self.project.getitems(self.pofilename, fromitem, toitem)

def maketable(self):
    self.translations = self.gettranslations()
    if self.reviewmode and self.item is not None:
        suggestions = {self.item: self.project.getsuggestions(self.pofilename, self.item)}
    items = []
    for row, thepo in enumerate(self.translations):
        orig = thepo.unquotedmsgid
        trans = thepo.unquotedmsgstr
        item = self.firstitem + row
        origdict = self.getorigdict(item, orig, item in self.editable)
        transmerge = {}
        ##MODIFIED Achilles
        tmmatches = {"title": self.localize("TM Matches"), "matches": { "orig": {}, "trans": {}}}
        if item in self.editable:

```

```

tmmatches["matches"] = self.gettmmatches(item)
if self.reviewmode:
    itemsuggestions = [suggestion.unquotedmsgstr for suggestion in suggestions[item]]
    transmerge = self.gettransreview(item, trans, itemsuggestions)
else:
    transmerge = self.gettransedit(item, trans)
else:
    transmerge = self.gettransedit(item, trans)
##END MODIFICATION
transdict = {"itemid": "trans%d" % item,
            "focus_class": origdict["focus_class"],
            "isplural": len(trans) > 1,
            }
transdict.update(transmerge)
polarity = oddoreven(item)
origcell_class = "translate-original translate-original-%s" % polarity
transcell_class = "translate-translation translate-translation-%s" % polarity
if item in self.editable:
    ##MODIFIED Achilles
    visibility = "show"
    ##END MODIFICATION
    focus_class = "translate-focus"
else:
    ##MODIFIED Achilles
    visibility = "hide"
    ##END MODIFICATION
    focus_class = ""
itemdict = {
    "itemid": item,
    "orig": origdict,
    "trans": transdict,
    "polarity": polarity,
    "focus_class": focus_class,
    "editable": item in self.editable,
    ##MODIFIED Achilles
    "editlink": self.geteditlink(item),
    "visibility": visibility,
    "tmmatches": tmmatches,
    ##END MODIFICATION
}
items.append(itemdict)
return items

def escapetext(self, text):
    """Replace special characters &, <, >, add and handle quotes if asked"""
    text = text.replace("&", "&amp;") # Must be done first!
    text = text.replace("<", "&lt;").replace(">", "&gt;")
    #TODO:Show fancy spaces
    text = text.replace("\r\n", '\\r\\n<br />\\n')
    text = text.replace("\n", '\\n<br />\\n')
    text = text.replace("\r", '\\r<br />\\n')
    text = text.replace("\t", '\\t')
    return text

def escapefortextarea(self, text):
    text = text.replace("&", "&amp;") # Must be done first!
    text = text.replace("<", "&lt;").replace(">", "&gt;")
    text = text.replace("\r\n", '\\r\\n\\n')
    text = text.replace("\n", '\\n\\n')
    text = text.replace("\t", '\\t')
    return text

def unescapesubmission(self, text):
    text = text.replace("\t", "")
    text = text.replace("\n", "")
    text = text.replace("\r", "")
    text = text.replace("\\t", "\t")
    text = text.replace("\\n", "\n")
    text = text.replace("\\r", "\r")
    return text

##MODIFIED Achilles
def highlighttext(self, str, highlightsyntax):

```

```

        """'highlight' special parts of the text that contain variables or special tags"""
        highlighter = gethighlighter(highlightsyntax)
        repstr = r'<span class="highlight" title="' + self.localize(r"This is a special string that should not be edited in the translation.") + r'>\1</span>'
        return highlighter.sub(repstr, str)
    ##END MODIFICATION

def getorigdict(self, item, orig, editable):
    if editable:
        focus_class = "translate-original-focus"
    else:
        focus_class = "autoexpand"
    purefields = []
    for pluralid, pluraltext in enumerate(orig):
        pureid = "orig-pure%d.%d" % (item, pluralid)
        purefields.append({"pureid": pureid, "name": pureid, "value": pluraltext})
    origdict = {
        "focus_class": focus_class,
        "itemid": "orig%d" % item,
        "pure": purefields,
        "isplural": len(orig) > 1 or None,
        "singular_title": self.localize("Singular"),
        "plural_title": self.localize("Plural"),
    }
    ##MODIFIED Achilles
    typecomments = self.project.getitemtypecomments(self.pofilename, item)
    highlightsyntax = ""
    for comment in typecomments:
        if comment.lower().find("format") != -1:
            highlightsyntax = comment.lower()
    ##END MODIFICATION
    if len(orig) > 1:
        singular_text = self.escapetext(orig[0])
        plural_text = self.escapetext(orig[1])
        ##MODIFIED Achilles
        if highlightsyntax:
            singular_text = self.highlighttext(singular_text, highlightsyntax)
            plural_text = self.highlighttext(plural_text, highlightsyntax)
        ##END MODIFICATION
        origdict["singular_text"] = singular_text
        origdict["plural_text"] = plural_text
    else:
        text = self.escapetext(orig[0])
        ##MODIFIED Achilles
        if highlightsyntax:
            text = self.highlighttext(text, highlightsyntax)
        ##END MODIFICATION
        origdict["text"] = text
    return origdict

def geteditlink(self, item):
    """gets a link to edit the given item, if the user has permission"""
    if "translate" in self.rights or "suggest" in self.rights:
        translateurl = "?translate=1&item=%d&pofilename=%s" % (item, urllib.quote(self.pofilename, '/'))
        ##MODIFIED Achilles
        return {"href": translateurl, "text": self.localize("Edit"), "id": "edit%d" % item}
    ##END MODIFICATION
    else:
        return None

def gettransbuttons(self, item, desiredbuttons):
    """gets buttons for actions on translation"""
    if "suggest" in desiredbuttons and "suggest" not in self.rights:
        desiredbuttons.remove("suggest")
    if "translate" in desiredbuttons and "translate" not in self.rights:
        desiredbuttons.remove("translate")
    specialchars = getattr(getattr(self.session.instance.languages, self.project.languagecode, None), "specialchars", "")
    if isinstance(specialchars, str):
        specialchars = specialchars.decode("utf-8")
    usernode = self.getusernode()
    return {"desired": desiredbuttons,
            "item": item,
            "copy_text": self.localize("Copy"),
            "skip": self.localize("Skip"),
    }

```

```

        "suggest": self.localize("Suggest"),
        "submit": self.localize("Submit"),
        "specialchars": specialchars,
        "rows": getattr(usernode, "inputheight", 5),
        "cols": getattr(usernode, "inputwidth", 40),
        "grow": self.localize("Grow"),
        "shrink": self.localize("Shrink"),
        "broaden": self.localize("Broaden"),
        "narrow": self.localize("Narrow"),
        "reset": self.localize("Reset")
    }

def gettransedit(self, item, trans):
    """returns a widget for editing the given item and translation"""
    if "translate" in self.rights or "suggest" in self.rights:
        usernode = self.getusernode()
        transdict = {
            "isplural": len(trans) > 1 or None,
            "rows": getattr(usernode, "inputheight", 5),
            "cols": getattr(usernode, "inputwidth", 40),
        }
        focusbox = ""
        spellargs = {"standby_url": "spellingstandby.html", "js_url": "/js/spellui.js", "target_url": "spellcheck.html"}
        if len(trans) > 1:
            buttons = self.gettransbuttons(item, ["skip", "suggest", "translate"])
            forms = []
            for pluralitem, pluraltext in enumerate(trans):
                pluralform = self.localize("Plural Form %d", pluralitem)
                pluraltext = self.escapefortextarea(pluraltext)
                textid = "trans%d.%d" % (item, pluralitem)
                forms.append({"title": pluralform, "name": textid, "text": pluraltext, "n": pluralitem})
            if not focusbox:
                focusbox = textid
            transdict["forms"] = forms
        else:
            buttons = self.gettransbuttons(item, ["skip", "copy", "suggest", "translate", "resize"])
            transdict["text"] = self.escapefortextarea(trans[0])
            textid = "trans%d" % item
            focusbox = textid
            transdict["can_spell"] = spellcheck.can_check_lang(self.project.languagecode)
            transdict["spell_args"] = spellargs
            transdict["buttons"] = buttons
            transdict["focusbox"] = focusbox
        else:
            # TODO: work out how to handle this (move it up?)
            transdict = self.gettransview(item, trans, textarea=True)
            buttons = self.gettransbuttons(item, ["skip"])
            transdict["buttons"] = buttons
    return transdict

def highlightdiffs(self, text, diffs, issrc=True):
    """highlights the differences in diffs in the text.
    diffs should be list of diff opcodes
    issrc specifies whether to use the src or destination positions in reconstructing the text
    this escapes the text on the fly to prevent confusion in escaping the highlighting"""
    if issrc:
        diffstart = [(i1, 'start', tag) for (tag, i1, i2, j1, j2) in diffs if tag != 'equal']
        diffstop = [(i2, 'stop', tag) for (tag, i1, i2, j1, j2) in diffs if tag != 'equal']
    else:
        diffstart = [(j1, 'start', tag) for (tag, i1, i2, j1, j2) in diffs if tag != 'equal']
        diffstop = [(j2, 'stop', tag) for (tag, i1, i2, j1, j2) in diffs if tag != 'equal']
    diffswitches = diffstart + diffstop
    diffswitches.sort()
    textdiff = ""
    textnest = 0
    textpos = 0
    spanempty = False
    for i, switch, tag in diffswitches:
        textsection = self.escapetext(text[textpos:i])
        textdiff += textsection
        if textsection:
            spanempty = False
        if switch == 'start':

```

```

    textnest += 1
elif switch == 'stop':
    textnest -= 1
if switch == 'start' and textnest == 1:
    # start of a textition
    textdiff += "<span class='translate-diff-%s'>" % tag
    spanempty = True
elif switch == 'stop' and textnest == 0:
    # start of an equals block
    if spanempty:
        # FIXME: work out why kid swallows empty spans, and browsers display them horribly, then remove this
        textdiff += "("
        textdiff += "</span>"
    textpos = i
textdiff += self.escapetext(text[textpos:])
return textdiff

def getdiffcodes(self, cmp1, cmp2):
    """compares the two strings and returns opcodes"""
    return difflib.SequenceMatcher(None, cmp1, cmp2).get_opcodes()

def gettransreview(self, item, trans, suggestions):
    """returns a widget for reviewing the given item's suggestions"""
    hasplurals = len(trans) > 1
    diffcodes = {}
    for pluralitem, pluraltrans in enumerate(trans):
        if isinstance(pluraltrans, str):
            trans[pluralitem] = pluraltrans.decode("utf-8")
    for suggestion in suggestions:
        for pluralitem, pluralsugg in enumerate(suggestion):
            if isinstance(pluralsugg, str):
                suggestion[pluralitem] = pluralsugg.decode("utf-8")
    forms = []
    for pluralitem, pluraltrans in enumerate(trans):
        pluraldiffcodes = [self.getdiffcodes(pluraltrans, suggestion[pluralitem]) for suggestion in suggestions]
        diffcodes[pluralitem] = pluraldiffcodes
        combineddiffs = reduce(list.__add__, pluraldiffcodes, [])
        transdiff = self.highlightdiffs(pluraltrans, combineddiffs, issrc=True)
        form = {"n": pluralitem, "diff": transdiff, "title": None}
        if hasplurals:
            pluralform = self.localize("Plural Form %d", pluralitem)
            form["title"] = pluralform
        forms.append(form)
    transdict = {
        "current_title": self.localize("Current Translation:"),
        "editlink": self.geteditlink(item),
        "forms": forms,
        "isplural": hasplurals or None,
        "itemid": "trans%d" % item,
    }
    suggitems = []
    for suggid, msgstr in enumerate(suggestions):
        suggestedby = self.project.getsuggester(self.pofilename, item, suggid)
        if len(suggestions) > 1:
            if suggestedby:
                suggtitle = self.localize("Suggestion %d by %s:", suggid+1, suggestedby)
            else:
                suggtitle = self.localize("Suggestion %d:", suggid+1)
        else:
            if suggestedby:
                suggtitle = self.localize("Suggestion by %s:", suggestedby)
            else:
                suggtitle = self.localize("Suggestion:")
        forms = []
        for pluralitem, pluraltrans in enumerate(trans):
            pluralsuggestion = msgstr[pluralitem]
            suggdiffcodes = diffcodes[pluralitem][suggid]
            suggdiff = self.highlightdiffs(pluralsuggestion, suggdiffcodes, issrc=False)
            if isinstance(pluralsuggestion, str):
                pluralsuggestion = pluralsuggestion.decode("utf8")
            form = {"diff": suggdiff}
            form["suggid"] = "suggest%d.%d.%d" % (item, suggid, pluralitem)
            form["value"] = pluralsuggestion

```

```

    if hasplurals:
        form["title"] = self.localize("Plural Form %d", pluralitem)
    forms.append(form)
    suggdict = {"title": suggtitle,
               "forms": forms,
               "suggid": "%d.%d" % (item, suggid),
               "canreview": "review" in self.rights,
               "skip": None,
               }
    suggitems.append(suggdict)
    skipbutton = {"item": item, "text": self.localize("Skip")}
    if suggitems:
        suggitems[-1]["skip"] = skipbutton
    else:
        transdict["skip"] = skipbutton
    transdict["suggestions"] = suggitems
    return transdict

def gettransview(self, item, trans, textarea=False):
    """returns a widget for viewing the given item's translation"""
    if textarea:
        escapefunction = self.escapefortextarea
    else:
        escapefunction = self.escapetext
    editlink = self.geteditlink(item)
    transdict = {"editlink": editlink}
    if len(trans) > 1:
        forms = []
        for pluralitem, pluraltext in enumerate(trans):
            form = {"title": self.localize("Plural Form %d", pluralitem), "n": pluralitem, "text": escapefunction(pluraltext)}
            forms.append(form)
        transdict["forms"] = forms
    else:
        transdict["text"] = escapefunction(trans[0])
    return transdict

```


ANEXO I – translatepage.html

Este arquivo foi apenas **modificado** pelo autor, para implementar as mudanças propostas! Todos os trechos de código modificado são envoltos pelos comentários **##MODIFIED Achilles** e **##END MODIFICATION**. Todo o restante do código é de autoria dos desenvolvedores do Pootle.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns:py="http://purl.org/kid/ns#" xmlns:over="http://dev.sjsoft.com/ns/overlay" xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta name="generator" content="HTML Tidy for Linux/x86 (vers 1st September 2004), see www.w3.org" />

  <title py:content="pagetitle">Pootle: translating ${project.name} into ${language.name}: ${pofilename}</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <link rel="stylesheet" type="text/css" href="/pootle.css" />
  <link rel="shortcut icon" href="/favicon.ico" />
  ##MODIFIED Achilles
  <script src="/js/MochiKit/MochiKit.js" type="text/javascript"></script>
  <script src="/js/translate.js" type="text/javascript">
  ##END MODIFICATION
  <!--
  /-->
</script>
<!--[if IE]>
  <script defer type="text/javascript" src="/js/correctpng.js"></script>
  ##MODIFIED Achilles
  <style type="text/css">
    /* CSS hack to make "position: fixed" work on IE. */
    body
    {
      overflow: hidden;
      font-size: 100%;
    }
    #content
    {
      overflow: auto;
      height: 100%;
      width: 72%;
    }
    #fixedhelp
    {
      position: absolute;
      overflow: hidden;
      height: 100%;
      right: 15px;
      bottom: 15px;
      width: 20%;
    }
  </style>
  ##END MODIFICATION
  <![endif]-->
</head>

<body bgcolor="#FFFFFF">
  <div id="banner" over:include="pootlepage#banner"/>

  <div id="content">
    <div class="navbar" py:for="item in navitems">
```

```

<div over:include="navbar#item">
  <h3 over:replace="itemtitle" class="title">
    <span py:strip="True" py:if="item.path.language">[ <a href="{item.path.language.href}">{item.path.language.text}</a> ]</span>
    <span py:strip="True" py:if="item.path.project">[ <a href="{item.path.project.href}">{item.path.project.text}</a> ]</span>
    <span py:strip="True" py:for="pathlink in item.path.pathlinks">
      <a href="{pathlink.href}">{pathlink.text}</a>{pathlink.sep}
    </span>
  </h3>
</div>

<div over:include="navbar#itemstats"/>

<div class="item-statistics" py:if="pagelinks">
  <span py:strip="True" py:for="pagelink in pagelinks">
    <a href="{pagelink.href}">{pagelink.text}</a>{pagelink.sep}
  </span>
</div>
</div>

<div class="translate-form">
  <span py:if="notice">
    <h3 class="title">{notice.title}</h3>
    <div class="intro">{notice.stoppedby}</div>
    <a href="{notice.finishedlink}">{notice.returnlink}</a>
  </span>
  <table cellpadding="10" cellspacing="1" class="translate-table">
    <tr>
      <td colspan="1" rowspan="1" class="translate-table-title" py:content="original_title">original</td>

      <td colspan="1" rowspan="1" class="translate-table-title" py:content="translation_title">translation</td>
    </tr>

    ##MODIFIED Achilles
    <span py:for="item in items" py:strip="True">
      <tr py:if="not reviewmode and not viewmode">
        <td id="tmtitle${item.itemid}" name="tm${item.itemid}" class="tmtitle ${item.visibility}" colspan="2" rowspan="1" py:content="item.tmmatches.title">TM
      </tr>

      <tr py:if="not reviewmode and not viewmode">
        <td id="tmorig${item.itemid}" name="tm${item.itemid}" class="tmmatch ${item.visibility}" colspan="1" rowspan="1">
          <div py:for="match in item.tmmatches.matches.orig" title="{match.similarity}">{match.match}</div>
        </td>
        <td id="tmtrans${item.itemid}" name="tm${item.itemid}" class="tmmatch ${item.visibility}" colspan="1" rowspan="1">
          <div py:for="match in item.tmmatches.matches.trans" title="{match.similarity}">{match.match}</div>
        </td>
      </tr>

      <tr>
        <td colspan="1" rowspan="1" class="translate-translation translate-translation-${item.polarity}" ">
          <div id="{item.orig.itemid}" class="translate-original">
            ##END MODIFICATION
            <input py:for="field in item.orig.pure" type="hidden" name="{field.pureid}" value="{field.value}" id="{field.pureid}"/>
            <span py:if="not item.orig.isplural" py:replace="XML(item.orig.text)">
              </span>
            <span py:if="item.orig.isplural" py:strip="True">
              <span class="translation-text-headers" py:content="item.orig.singular_title">Singular</span><br />
              <span py:content="XML(item.orig.singular_text)">singular text</span><br />
              <span class="translation-text-headers" py:content="item.orig.plural_title">Plural</span><br />
              <span py:content="XML(item.orig.plural_text)">plural text</span><br />
            </span>
            ##MODIFIED Achilles
            <input py:if="not viewmode" id="{item.editlink.id}" type="button" class="link hide ${item.polarity}" onclick="this.form.edit('{item.editlink.href}"/>
            ##END MODIFICATION
          </div>
        </td>

        <td colspan="1" rowspan="1" class="translate-translation translate-translation-${item.polarity}" ">
          <div id="{item.trans.itemid}" class="translate-translation ">
            <span py:if="item.editable and reviewmode" class="translation-text">
              <b py:content="item.trans.current_title">Current Translation:</b><br />
              <span py:for="form in item.trans.forms" py:strip="True">
                <span class="translation-text-headers" py:if="form.title" py:content="form.title">Plural Form ${form.n}</span>

```

```

    <br py:if="form.title" />
    <span py:replace="XML(form.diff)">Text with diffs</span><br />
</span>
<span py:for="sugg in item.trans.suggestions" py:strip="True">
    <b py:content="sugg.title">Suggestion by testuser:</b><br />
    <span py:for="form in sugg.forms" py:strip="True">
        <span class="translation-text-headers" py:if="form.title" py:content="form.title">Plural Form ${form.n}</span>
        <br py:if="form.title" />
        <span py:replace="XML(form.diff)">Text with diffs</span><br />
        <input type="hidden" name="${form.suggid}" value="${form.value}" />
        <br />
    </span>
    <br />
    <span py:if="sugg.canreview" py:strip="True">
        <input type="submit" name="accept${sugg.suggid}" value="${accept_button}" />
        <input type="submit" name="reject${sugg.suggid}" value="${reject_button}" />
    </span>
    <input py:if="sugg.skip" type="submit" name="skip${sugg.skip.item}" value="${sugg.skip.text}" />
</span>
<input py:if="item.trans.skip" type="submit" name="skip${item.trans.skip.item}" value="${item.trans.skip.text}" />
</span>
##MODIFIED Achilles
<span py:if="not reviewmode" py:strip="True">
##END MODIFICATION
    <span py:if="not item.trans.isplural" py:strip="True">
        <!--! TODO: spell checking -->
        ##MODIFIED Achilles
        <textarea class="${item.visibility} ${item.polarity}" name="${item.trans.itemid}" rows="${item.trans.rows}" cols="${item.trans.cols}" py:content=
        ##END MODIFICATION
            translatable text
        </textarea>
    </span>
    <span py:if="item.trans.isplural" py:strip="True">
        <span py:for="form in item.trans.forms" py:strip="True">
            <span class="translation-text-headers" py:content="form.title">Plural Form ${form.n}</span><br />
            <!--! TODO: spell checking -->
            ##MODIFIED Achilles
            <textarea class="${item.visibility} ${item.polarity}" name="${form.name}" rows="${item.trans.rows}" cols="${item.trans.cols}" py:content="XML
            ##END MODIFICATION
                translatable text for plural form
            </textarea>
        </span>
    </span>
</span>
<span py:strip="True" py:for="buttons in [item.trans.buttons]">
    ##MODIFIED Achilles
    <div>
        <input class="${item.visibility}" py:if="'skip' in buttons.desired" type="submit" name="skip${buttons.item}" accesskey="k" value="${buttons.s
        <input class="${item.visibility}" type="button" py:if="'copy' in buttons.desired" accesskey="c"
            onclick="document.forms.translate.trans${buttons.item}.value = document.getElementById('orig-pure${buttons.item}.0').value ; document.forms
            value="${buttons.copy_text}" />
        <input class="${item.visibility}" py:if="'suggest' in buttons.desired" type="submit" name="submitsuggest${buttons.item}" accesskey="e" value=
        <input class="${item.visibility}" py:if="'translate' in buttons.desired" type="submit" name="submit${buttons.item}" accesskey="s" value="${bu
    </div>
    <div>
        <input type="text" class="${item.visibility} ${item.polarity}" readonly="true" py:if="'specialchars' in buttons" value="${buttons.specialchar
    </div>
    <div py:if="'resize' in buttons.desired">
        <input type="button" class="link ${item.visibility} ${item.polarity}" onclick="this.form.expandTextarea()" accesskey="=" value="${buttons.gro
        <input type="button" class="link ${item.visibility} ${item.polarity}" onclick="this.form.contractTextarea()" accesskey="-" value="${buttons.s
        <input type="button" class="link ${item.visibility} ${item.polarity}" onclick="this.form.broadenTextarea()" accesskey="+" value="${buttons.br
        <input type="button" class="link ${item.visibility} ${item.polarity}" onclick="this.form.narrowTextarea()" accesskey="_" value="${buttons.nar
        <input type="button" class="link ${item.visibility} ${item.polarity}" onclick="this.form.resetTextarea(${buttons.rows}, ${buttons.cols})" val
    </div>
    ##END MODIFICATION
</span>
</span>
##MODIFIED Achilles
<script type="text/javascript" py:if="item.editable and not reviewmode">
    <!-- //We should comment this for old browsers, but then kid doesn't evaluate the expression -->
    getElement("translate${item.itemid}")["trans${item.itemid}"].focus();
    <!-- //-->
</script>
</div>

```

```

        </td>
        <input type="hidden" name="searchtext" value="{searchtext}" />
        <input type="hidden" name="pofilename" value="{pofilename}" />
    </form>
    ##END MODIFICATION
</tr>
</span>
</table>

<div class="item-statistics" py:if="pagelinks">
    <span py:strip="True" py:for="pagelink in pagelinks">
        <a href="{pagelink.href}">{pagelink.text}</a>{pagelink.sep}
    </span>
</div>
</div>
</div>

<div id="links" over:include="pootlepage#links">
    <form id="searchform">
        <input type="hidden" name="pofilename" value="{pofilename}" />
    </form>

    <div py:if="defined('assigns')" class="sidetitle" py:content="assigns.title">Assign Strings</div>
    <form py:if="defined('assigns')" action="?index=1" name="assignform">
        <select name="assignto" title="{assigns.user_title}">
            <option py:for="user in assigns.users" value="{user}" py:content="user">Username</option>
        </select>
        <input name="action" value="translate" title="{assigns.action_title}" />
        <input name="doassign" type="submit" value="{assigns.submit_text}" />
    </form>

    <div class="side" py:if="defined('checking_text')" py:content="checking_text">
        checking has-suggestion, etc
    </div>
</div>

##MODIFIED Achilles
<!--I *WAS* going to put this inside "links" but *OF COURSE* IE has a problem with it.-->
<div id="fixedhelp">
    <div id="additionalinfo">
        <div class="sidetitle" py:content="additionalinfo.title">Additional Information</div>
        <div id="informations" class="side">
            <div py:for="info in additionalinfo.information" py:content="info" />
        </div>
    </div>

    <div id="termbase">
        <div class="sidetitle" py:content="termsearch.title">Search termbase</div>
        <form id="searchtermform" class="side" action="translate.html">
            <input name="searchterm" value="" title="{termsearch.usage}" />
        </form>
        <div id="termbaseanswer" class="side">
            <div py:for="term in termsearch.answer">{term}</div>
        </div>
    </div>
</div>
##END MODIFICATION

</body>
</html>

```

ANEXO J – tranlate.js

```

function init() {
  roundClass("div", "sidetitle", { corners: "tl tr" });

  pageforms = document.getElementsByName("translate");
  for (var i=0; i<pageforms.length; i++) {
    form = pageforms[i];
    form.toggleFormVisibility = toggleFormVisibility;
    form.toggleTMVisibility = toggleTMVisibility;
    form.getTextareas = getTextareas;
    form.getEditButton = getEditButton;
    form.expandTextarea = expandTextarea;
    form.contractTextarea = contractTextarea;
    form.broadenTextarea = broadenTextarea;
    form.narrowTextarea = narrowTextarea;
    form.resetTextarea = resetTextarea;
    form.edit = edit;
    form.loadTMMatches = loadTMMatches;

    if (i < (pageforms.length-1)) {
      var nextid = parseInt(form.id.replace("translate", ""));
      form.next = getElement("translate" + (nextid+1));
      for (var j=0; j<form.elements.length; j++) {
        if (form.elements[j].type == "submit") {
          connect(form.elements[j], "onclick", submitTrans);
        }
      }
    }
  }

  var formTextareas = form.getTextareas();
  if (formTextareas) {
    tableRow = form.parentNode;
    tableRow.editbutton = form.getEditButton();
    if (tableRow.editbutton) {
      connect(tableRow, "onmouseover", mouseGoesOver);
      connect(tableRow, "onmouseout", mouseGoesOut);
    }

    for (var j=0; j<formTextareas.length; j++) {
      if (hasElementClass(formTextareas[j], "hide")) {
        formTextareas[j].readOnly = true;
      }
    }
  }
}

var searchtermform = getElement("searchtermform");
if (searchtermform) {
  connect(searchtermform, "onsubmit", submitTerm);
}
}

function mouseGoesOver() {
  textareas = this.editbutton.form.getTextareas();
  for (var i=0; i<textareas.length; i++) {
    if (hasElementClass(textareas[i], "show")) {
      return false;
    }
  }
}

```

```

    }
    showElement(this.editbutton);
}

function mouseGoesOut() {
    textareas = this.editbutton.form.getTextareas();
    for (var i=0; i<textareas.length; i++) {
        if (hasElementClass(textareas[i], "show")) {
            return false;
        }
    }
    hideElement(this.editbutton);
}

function toggleFormVisibility() {
    for (var i=0; i<this.elements.length; i++) {
        if ((this.elements[i].id.indexOf("edit") != -1) && hasElementClass(this.elements[i], "hide")) {
            continue;
        } else {
            toggleElementVisibility(this.elements[i]);
        }
    }
}

function toggleTMVisibility() {
    var idnum = this.id.replace("translate", "");
    var tmid = "tmtitle" + idnum;
    toggleElementVisibility(getElement(tmid));
    tmid = "tmorig" + idnum;
    toggleElementVisibility(getElement(tmid));
    tmid = "tmtrans" + idnum;
    toggleElementVisibility(getElement(tmid));
}

function loadTMMatches(text) {
    response = DIV();
    response.innerHTML = text;
    var idnum = this.id.replace("translate", "");
    if (!getElement("tmtitle" + idnum)) return;
    tmid = "tmorig" + idnum;
    getElement(tmid).innerHTML = getElementById("orig", response).innerHTML;
    tmid = "tmtrans" + idnum;
    getElement(tmid).innerHTML = getElementById("trans", response).innerHTML;
    // Bonus treat: load additional information!
    getElement("informations").innerHTML = getElementById("info", response).innerHTML;
}

function getElementById(id, parent) {
    for (var i=0; i<parent.childNodes.length; i++) {
        if (parent.childNodes[i].id == id) {
            return parent.childNodes[i];
        }
    }
    return null
}

function toggleElementVisibility(elem) {
    if (hasElementClass(elem, "hide")) {
        if (elem.nodeName == "TEXTAREA")
            elem.readOnly = false;
        showElement(elem);
    } else if (hasElementClass(elem, "show")) {
        if (elem.nodeName == "TEXTAREA")
            elem.readOnly = true;
        hideElement(elem);
    }
}

function showElement(elem) {
    removeElementClass(elem, "hide");
    addElementClass(elem, "show");
}

```

```

function hideElement(elem) {
    removeElementClass(elem, "show");
    addElementClass(elem, "hide");
}

function getTextareas() {
    textareas = new Array();
    for (var i=0; i<this.elements.length; i++) {
        if (this.elements[i].nodeName == "TEXTAREA") {
            textareas.push(this.elements[i]);
        }
    }
    return textareas;
}

function getEditButton() {
    for (var i=0; i<this.elements.length; i++) {
        if (this.elements[i].id.indexOf("edit") != -1) {
            return this.elements[i];
        }
    }
    return null;
}

function expandTextarea() {
    var textareas = this.getTextareas();
    if (textareas == null) return true;
    for (var i=0; i<textareas.length; i++) {
        if (textareas[i].rows >= 3)
            textareas[i].rows += 3;
        else
            textareas[i].rows += 1;
    }
    return false;
}

function contractTextarea() {
    var textareas = this.getTextareas();
    if (textareas == null) return true;
    for (var i=0; i<textareas.length; i++) {
        if (textareas[i].rows > 3)
            textareas[i].rows -= 3;
        else if (textareas[i].rows > 1)
            textareas[i].rows -= 1;
    }
    return false;
}

function broadenTextarea() {
    var textareas = this.getTextareas();
    if (textareas == null) return true;
    for (var i=0; i<textareas.length; i++) {
        if (textareas[i].cols >= 40)
            textareas[i].cols += 10;
        else
            textareas[i].cols += 5;
    }
    return false;
}

function narrowTextarea() {
    var textareas = this.getTextareas();
    if (textareas == null) return true;
    for (var i=0; i<textareas.length; i++) {
        if (textareas[i].cols > 40)
            textareas[i].cols -= 10;
        else if (textareas[i].cols > 1)
            textareas[i].cols -= 5;
    }
    return false;
}

function resetTextarea(rows, cols) {

```

```

var textareas = this.getTextareas();
if (textareas == null) return true;
for (var i=0; i<textareas.length; i++) {
    textareas[i].rows = rows;
    textareas[i].cols = cols;
}
return false;
}

function contains(arr, elem) {
    for (var i=0; i<arr.length; i++) {
        if (elem == arr[i]) {
            return true;
        }
    }
    return false;
}

function edit(link) {
    var textareas = getElementsByTagAndClassName("textarea", "show");
    lastForms = new Array();
    for (var i=0; i<textareas.length; i++) {
        if (!contains(lastForms, textareas[i].form)) {
            lastForms.push(textareas[i].form);
        }
    }
    var req = getXMLHttpRequest();
    if (req.overrideMimeType) {
        req.overrideMimeType("text/plain");
    }
    req.open("GET", link + "%gettmatches", true);
    d = sendXMLHttpRequest(req);
    d.addCallback(loadEditResponse, lastForms, this);
    d.addErrback(loadResponseError);
}

function loadEditResponse(lastForms, currForm, req) {
    for (var i=0; i<lastForms.length; i++) {
        lastForms[i].toggleFormVisibility();
        lastForms[i].toggleTMVisibility();
    }
    currForm.loadTMMatches(req.responseText);
    currForm.toggleTMVisibility();
    currForm.toggleFormVisibility();
    currForm.getTextarea().focus()
}

function submitTrans(event) {
    event.preventDefault();
    var poststr = "";
    for (var i=0; i<this.form.elements.length; i++) { // Aren't you tired of "fors"? I'm sure *I* am!
        var elem = this.form.elements[i];
        var type = elem.type.toLowerCase();
        if ((type == "submit" && elem != this) || (type == "button") || (type == "text")) {
            continue;
        }
        if (type == "select-multiple") {
            var seleclist = elem.options;
            for (var j=0; j<seleclist.length; j++) {
                if (seleclist[j].selected) {
                    poststr += encodeURIComponent( elems.name + "=" + seleclist[j].value + "&");
                }
            }
        }
        else if (type == "radio" || type == "checkbox") {
            if (elem.checked) {
                poststr += encodeURIComponent( elem.name + "=" + elems.value + "&");
            }
        }
        else {
            poststr += encodeURIComponent( elem.name + "=" + elem.value + "&");
        }
    }
    poststr += encodeURIComponent( "gettmatches" );
}

```



```

var req = getXMLHttpRequest();
if (req.overrideMimeType) {
    req.overrideMimeType("text/plain");
}
req.open("POST", this.form.action, true);
req.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
req.setRequestHeader("Content-length", poststr.length);
req.setRequestHeader("Connection", "close");
d = sendXMLHttpRequest(req, poststr)
d.addCallback(loadSubmitResponse, this.form);
d.addErrback(loadResponseError);
}

function loadSubmitResponse(form, req) {
    form.toggleFormVisibility();
    form.toggleTMVisibility();
    form.next.loadTMMatches(req.responseText);
    form.next.toggleTMVisibility();
    form.next.toggleFormVisibility();
    form.next.getTextareas()[0].focus(); // There's at least *ONE* textarea...
}

function loadResponseError(error) {
    alert(error.req.responseText);
}

function submitTerm(event) {
    event.preventDefault();
    var poststr = "";
    for (var i=0; i<this.elements.length; i++) {
        var elem = this.elements[i];
        poststr += encodeURIComponent( elem.name + "=" + elem.value + "&" );
    }
    poststr += encodeURIComponent( "gettbmatches" );

    var req = getXMLHttpRequest();
    if (req.overrideMimeType) {
        req.overrideMimeType("text/plain");
    }
    req.open("POST", this.action, true);
    req.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    req.setRequestHeader("Content-length", poststr.length);
    req.setRequestHeader("Connection", "close");
    d = sendXMLHttpRequest(req, poststr)
    d.addCallback(loadTermResponse);
    d.addErrback(loadResponseError);
}

function loadTermResponse(req) {
    response = DIV();
    response.innerHTML = req.responseText;
    getElement("termbaseanswer").innerHTML = getElementById("termbaseanswer", response).innerHTML;
}

addLoadEvent(init);

```

ANEXO K – content.js

```

// Utility functions
function addEvents(name, ev, fn) {
    links = document.getElementsByName(name);

    for (var i=0; i<links.length; i++) {
        connect(links[i], ev, fn);
    }
}

function init() {
    var manager = new IndexContent();
    addEvents("sidelink", "onclick", manager.anchorClick);
    roundClass("div", "sidetitle", { corners: "tl tr" })

    var url = String(document.location);

    initScript(url);
}

function initScript(url) {
    var manager;
    if (url.indexOf("admin") != -1) {
        manager = new AdminContent();
    }
    else if (url.indexOf("home") != -1) {
        manager = new HomeContent();
    }
    else {
        manager = new IndexContent();
    }
    manager.init();
}

// Content manager objects
IndexContent = function()
{
    var self = {};
    self.content = "content"
    self.url = "";

    self.init = function()
    {
        addEvents("contentlink", "onclick", self.anchorClick);
    };

    self.anchorClick = function(event)
    {
        event.preventDefault();
        self.url = String(event.target());

        var req = getXMLHttpRequest();
        if (req.overrideMimeType) {
            req.overrideMimeType("text/plain");
        }
        req.open("GET", event.target() + "?contentonly", true);
        d = sendXMLHttpRequest(req).addCallbacks(self.loadPage, self.loadError);
    };
};

```

```

self.loadPage = function(req)
{
    getElement(self.content).innerHTML = req.responseText;
    initScript(self.url);
};

self.loadError = function(error)
{
    getElement(self.content).innerHTML = error.req.responseText;
};

return self;
}

AdminContent = function()
{
    var self = new IndexContent();
    self.responseid = "";

    self.init = function()
    {
        addEvents("contentlink", "onclick", self.anchorClick);
        addEvents("general", "onsubmit", self.submitClick);
    };

    self.submitClick = function(event)
    {
        event.preventDefault();

        var form = event.src();

        self.responseid = String(form.id)
        self.responseid += "response"
        self.clearResponse(self.responseid);

        var felems = form.elements;
        self.url = String(form.action);

        var poststr = "";
        for (var i=0; i<felems.length; i++) {
            var type = felems[i].type.toLowerCase();
            if (type == "select-multiple") {
                var seleclist = felems[i].options;
                for (var j=0; j<seleclist.length; j++) {
                    if (seleclist[j].selected) {
                        poststr += encodeURIComponent( felems[i].name + "=" + seleclist[j].value + "&");
                    }
                }
            } else if (type == "radio" || type == "checkbox") {
                if (felems[i].checked) {
                    poststr += encodeURIComponent( felems[i].name + "=" + felems[i].value + "&");
                }
            } else {
                poststr += encodeURIComponent( felems[i].name + "=" + felems[i].value + "&");
            }
        }
        poststr += encodeURIComponent( "contentonly" );

        var req = getXMLHttpRequest();
        if (req.overrideMimeType) {
            req.overrideMimeType("text/plain");
        }
        req.open("POST", form.action, true);
        req.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        req.setRequestHeader("Content-length", poststr.length);
        req.setRequestHeader("Connection", "close");
        d = sendXMLHttpRequest(req, poststr).addCallbacks(self.loadResponse, self.loadResponseError);
    };

    self.loadResponse = function(req)
    {
        getElement(self.responseid).innerHTML = req.responseText;
    }
}

```

```
    roundElement(self.responseid, { corners: "tl tr bl br" });
};

self.loadResponseError = function(error)
{
    getElement(self.responseid).innerHTML = error.req.responseText;
    roundElement(self.responseid, { corners: "tl tr bl br" });
};

self.clearResponse = function(responseid)
{
    getElement(responseid).innerHTML = "";
};

return self;
}

HomeContent = function()
{
    var self = new AdminContent();

    self.init = function()
    {
        addEvents("contentlink", "onclick", self.anchorClick);

        if (getElement("personal") != undefined)
            addEvents("personal", "onsubmit", self.submitClick);
        if (getElement("interface") != undefined)
            addEvents("interface", "onsubmit", self.submitClick);
        if (getElement("useroptions") != undefined)
            addEvents("useroptions", "onsubmit", self.submitClick);
    };

    return self;
}

addLoadEvent(init);
```

ANEXO L - translatepagetb.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns:py="http://purl.org/kid/ns#" py:strip="True">
  <span id="termbaseanswer">
    <div py:for="match in tmatches" py:content="match" />
  </span>
</html>
```

ANEXO M – translatepagetm.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns:py="http://purl.org/kid/ns#" py:strip="True">
  <span id="orig">
    <div py:for="match in tmmatches.orig" title="{match.similarity}">{match.match}</div>
  </span>
  <span id="trans">
    <div py:for="match in tmmatches.trans" title="{match.similarity}">{match.match}</div>
  </span>
  <span id="info">
    <div py:for="info in additionalinfo">{info}</div>
  </span>
</html>
```

Referências

- ALBUQUERQUE, A. *Localização, Globalização, Internacionalização e Tradução: Algumas considerações*. 2005. Disponível em: <http://www.iscap.ipp.pt/cml/pgtac/ficheiros/teoria/Sistemas_TA.ppt>. Acesso em: 04 maio 2005.
- ALFARO, C. *Descobrimdo, Compreendendo e Analisando a Tradução Automática*. Rio de Janeiro, RJ: [s.n.], Jan. 1998. Disponível em: <<http://www.tecgraf.puc-rio.br/~carolina/monografia/>>. Acesso em: 27 nov. 2004.
- COMMONS, C. About us. In: *Creative Commons Website*. [s.n.], 2006. Disponível em: <<http://creativecommons.org/about/history>>. Acesso em: 30 jun. 2006.
- CRACIUNESCU, O.; GERDING-SALAS, C.; STRINGER-O'KEEFFE, S. Machine translation and computer-assisted translation: a new way of translating? *Translation Journal*, Accurapid Translation Services, Inc., New York, NY, v. 8, n. 3, Jul. 2004. Disponível em: <<http://accurapid.com/journal/29computers.htm>>. Acesso em: 27 nov. 2004.
- ESSELINK, B. *A Practical Guide to Localization*. Amsterdam, Netherlands: John Benjamins Publishing Co., 2000. 490 p.
- FOUNDATION, F. S. *Gettext Manual*. [S.l.], 1998. Disponível em: <<http://www.gnu.org/software/gettext/manual/gettext.html>>. Acesso em: 30 jun. 2006.
- FOUNDATION, F. S. Introdução ao projeto gnu. In: *Filosofia do Projeto GNU*. [s.n.], 1999. Disponível em: <<http://www.gnu.org/gnu/gnu-history.pt.html>>. Acesso em: 29 jun. 2006.
- FOUNDATION, F. S. O que é o software livre? In: *Filosofia do Projeto GNU*. [s.n.], 2000. Disponível em: <<http://www.gnu.org/philosophy/free-sw.pt.html>>. Acesso em: 30 jun. 2006.
- FURTADO, O. J. V. *Apostila de Linguagens Formais e Compiladores*. [S.l.], 2006. Disponível em: <<http://www.inf.ufsc.br/~olinto/apostila-lfc.doc>>. Acesso em: 03 jul. 2006.
- JUNIOR, H. C. *Ferramentas para Apoio à Geração de Código Internacionalizável*. Florianópolis, SC: [s.n.], 2004. Disponível em: <http://www.projetos.inf.ufsc.br/arquivos_projetos/projeto.83/TCC_Helion.pdf.1>. Acesso em: 04 nov. 2004.

- LOMMEL, A.; FRY, D. *Manual de Introdução à Localização*. Florianópolis, Fevereiro 2005. Disponível em: <http://www.lisa.org/products/primers/primer2_poBR.pdf>. Acesso em: 27 maio 2005.
- LUCCA, J. E. D. et al. *Relatório técnico interno Softex - Estratégia nacional de componentes de software*. [S.l.], 2005.
- OGBUJI, U. Building dictionaries with sax. In: *Python & XML*. [s.n.], 2004. Disponível em: <<http://www.xml.com/pub/a/2004/01/14/py-xml.html>>. Acesso em: 03 jul. 2006.
- OSCAR GROUP. *TBX specification draft 1j*. [S.l.], Maio 2005. Disponível em: <<http://www.lisa.org/utills/getfile.html?id=8551478>>. Acesso em: 18 agos. 2006.
- STALLMAN, R. Porque o software não deveria ter donos. In: *Filosofia do Projeto GNU*. [s.n.], 1994. Disponível em: <<http://www.gnu.org/philosophy/why-free.pt.html>>. Acesso em: 28 jun. 2006.
- STALLMAN, R. The gnu project. In: *Open Sources*. [s.n.], 1998. Disponível em: <<http://www.gnu.org/gnu/thegnuproject.html>>. Acesso em: 29 jun. 2006.
- WIKIPEDIA. Direito autoral. In: *Wikipedia, a enciclopédia livre*. [s.n.], 2006. Disponível em: <http://pt.wikipedia.org/wiki/Direito_autoral>. Acesso em: 01 jul. 2006.
- WIKIPEDIA. Extensible markup language. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <<http://en.wikipedia.org/wiki/XML>>. Acesso em: 03 jul. 2006.
- WIKIPEDIA. Fuzzy string searching. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <http://en.wikipedia.org/wiki/Fuzzy_string_searching>. Acesso em: 18 agos. 2006.
- WIKIPEDIA. Html. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <<http://en.wikipedia.org/wiki/HTML>>. Acesso em: 03 jul. 2006.
- WIKIPEDIA. Internet. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <<http://en.wikipedia.org/wiki/Internet>>. Acesso em: 02 jul. 2006.
- WIKIPEDIA. Javascript. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <<http://en.wikipedia.org/wiki/JavaScript>>. Acesso em: 02 jul. 2006.
- WIKIPEDIA. Levenshtein distance. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <http://en.wikipedia.org/wiki/Levenshtein_distance>. Acesso em: 18 agos. 2006.
- WIKIPEDIA. List of google services and tools. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <http://en.wikipedia.org/wiki/List_of_Google_services_and_tools>. Acesso em: 03 jul. 2006.
- WIKIPEDIA. N-gram. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <<http://en.wikipedia.org/wiki/N-gram>>. Acesso em: 18 agos. 2006.
- WIKIPEDIA. Rich internet application. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <http://en.wikipedia.org/wiki/Rich_Internet_Application>. Acesso em: 02 jul. 2006.

- WIKIPEDIA. Sistema operativo. In: *Wikipedia, a enciclopédia livre*. [s.n.], 2006. Disponível em: <http://pt.wikipedia.org/wiki/Sistema_operativo>. Acesso em: 01 jul. 2006.
- WIKIPEDIA. Software livre. In: *Wikipedia, a enciclopédia livre*. [s.n.], 2006. Disponível em: <http://pt.wikipedia.org/wiki/Software_livre>. Acesso em: 01 jul. 2006.
- WIKIPEDIA. Standard generalized markup language. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <<http://en.wikipedia.org/wiki/SGML>>. Acesso em: 03 jul. 2006.
- WIKIPEDIA. Translation memory. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <http://en.wikipedia.org/wiki/Translation_memory>. Acesso em: 01 jul. 2006.
- WIKIPEDIA. Tree (data structure). In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <http://en.wikipedia.org/wiki/Tree_%28data_structure%29>. Acesso em: 03 jul. 2006.
- WIKIPEDIA. Vendor lock-in. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <http://en.wikipedia.org/wiki/Vendor_lock-in>. Acesso em: 01 jul. 2006.
- WIKIPEDIA. Web 2.0. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <http://en.wikipedia.org/wiki/Web_2.0>. Acesso em: 01 jul. 2006.
- WIKIPEDIA. Wikipedia: Faq geral. In: *Wikipedia, a enciclopédia livre*. [s.n.], 2006. Disponível em: <http://pt.wikipedia.org/wiki/Wikipedia:FAQ_Geral>. Acesso em: 01 jul. 2006.
- WIKIPEDIA. World wide web. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <http://en.wikipedia.org/wiki/World_Wide_Web>. Acesso em: 02 jul. 2006.
- WIKIPEDIA. Xmlhttprequest. In: *Wikipedia, the free encyclopedia*. [s.n.], 2006. Disponível em: <<http://en.wikipedia.org/wiki/XMLHttpRequest>>. Acesso em: 25 jul. 2006.
- WIKTIONARY. *Markup*. Junho 2006. Disponível em: <<http://en.wiktory.org/wiki/markup>>. Acesso em: 01 jul. 2006.