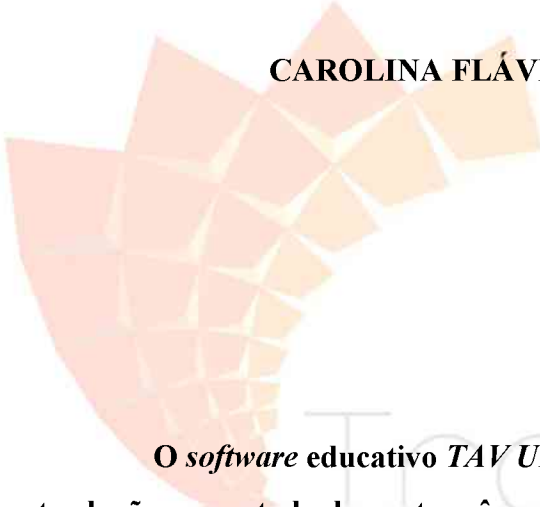


UNIVERSIDADE FEDERAL DE UBERLÂNDIA
INSTITUTO DE LETRAS E LINGUÍSTICA
CURSO DE TRADUÇÃO

CAROLINA FLÁVIA DE HENRIQUE



O software educativo TAV UFU – Editor de Comentários:
tradução comentada do português para o inglês em projeto de localização

UBERLÂNDIA/MG

2018

CAROLINA FLÁVIA DE HENRIQUE

O software educativo TAV UFU – Editor de Comentários:
tradução comentada do português para o inglês em projeto de localização

Monografia apresentada ao Curso de Tradução do Instituto de Letras e Linguística da Universidade Federal de Uberlândia, como requisito parcial para a obtenção do Grau de Bacharel em Tradução.

Orientadora: Prof^ª Dr^ª Marileide Dias Esqueda

UBERLÂNDIA/MG

2018

CAROLINA FLÁVIA DE HENRIQUE

O software educativo TAV UFU – Editor de Comentários:
tradução comentada do português para o inglês em projeto de localização

Monografia apresentada ao Curso de Tradução do Instituto de Letras e Linguística da Universidade Federal de Uberlândia como requisito parcial para a obtenção do título de Bacharel em Tradução.

Banca de Avaliação:

Prof. Dra. Marileide Dias Esqueda – UFU
Orientador

Prof. Dr. Pedro Malard Monteiro – UFU
Membro

Prof. Me. Arthur Melo de Sá – UFU
Membro

Uberlândia (MG), 26 de Junho de 2018

Agradecimentos

Aos meus pais, em especial minha mãe, pelo apoio e incentivo incondicionais em todas as minhas escolhas.

Às minhas amigas Brunna, Luísa, Jahynne, Camila e Bárbara, por fazerem todo esse processo mais leve e menos doloroso.

À minha orientadora, Profa. Dra. Marileide Dias Esqueda, pela orientação, disponibilidade, dedicação e atenção para com os detalhes que foram essenciais durante este trabalho. Deixo aqui minha mais sincera e profunda admiração.

A todos os professores do Curso de Bacharelado em Tradução da Universidade Federal de Uberlândia, por todos os ensinamentos.

Aos meus cantores preferidos, Shakira e Corey Taylor, por embalarem as longas horas de escrita desta monografia.

A todos, o meu muito obrigada!!!

RESUMO

Com a popularização dos computadores pessoais no início dos anos de 1980, grandes empresas do ramo da tecnologia de computação viram-se tentadas a expandir seus negócios para além de seus países de origem, fazendo surgir os processos de tradução e localização de produtos e serviços digitais. Dunne (2015) define que a localização é um hiperônimo que se refere aos processos através dos quais conteúdos e produtos digitais desenvolvidos em um determinado *locale* são adaptados para uso e comercialização em outros *locales*, ou regiões com línguas e culturas diferentes daquelas onde o produto foi originalmente produzido. Assim, este trabalho, além de buscar estudar esses processos, não apenas em nível teórico, mas também em nível prático, tem como proposta traduzir e localizar um *software* educativo do português para o inglês, explorando o uso de ferramentas de tradução que se originam no interior dos avanços tecnológicos ligados à localização. Utilizando a ferramenta específica para localização encontrada no *Visual Basic Community 2017*, será feita a localização do *software* educativo *TAV UFU – Editor de Comentários 1.0*, e comentários serão tecidos a partir dessas traduções, seguindo o modelo de Tradução Comentada estabelecidos por Zavaglia (et al) (2015). Além disso, este trabalho se pauta nos preceitos da tradução comentada para tecer as fases e estratégias adotadas para a tradução e localização de um *software*.

Palavras-chave: Tradução e Localização. Tradução Comentada. *Software* Educativo.

ABSTRACT

With the popularization of personal computers in the early 1980s, large companies in the field of computer technology have been tempted to expand their business beyond their home countries, resulting in the translation and localization of digital products and services . Dunne (2015) defines that localization is a hyperonym that refers to the processes through which contents and digital products developed in a given locale are adapted for use and commercialization in other locales, or regions with languages and cultures different from those where the product was originally produced. Thus, this work seeks to study these processes, not only at a theoretical level, but also at a practical level, aiming at translating and localizing an educational software from Portuguese into English, exploring the use of translation tools that have been emerging within the technological advances related to localization. With the help of the localization tool which is available in Visual Basic Community 2017, the Localization of the educational software TAV UFU – Comment Editor will be made, and comments will be written about these translations, following the models of Zavaglia (et al) (2015) In addition, this work is based on the precepts of the annotated translation to weave the phases and strategies adopted for the translation and localization of a software.

Keywords: Translation and Localization. Annotated Translation. Educational software.

LISTA DE ILUSTRAÇÕES

Figura 1: Processo <i>GILT</i>	11
Figura 2: Ícone de abertura do <i>software</i>	15
Figura 3: Captura da tela principal do <i>software</i> TAV UFU Editor de Comentários 1.0	16
Figura 4: Menu “Arquivo”	17
Figura 5: Menu “Inserir comentário”	17
Figura 6: Menu “Arquivo”	18
Figura 7: Comentário no arquivo de legenda e vídeo por meio do TAV UFU Editor de Comentários.....	18
Figura 8: Tela inicial do <i>Microsoft Visual Basic Studio 2017</i>	23
Figura 9: Tela inicial do <i>Microsoft Visual Basic Studio 2017</i>	24
Figura 10: Tela inicial do <i>Microsoft Visual Basic Studio 2017</i>	24
Figura 11: Ferramenta de localização <i>multilanguage</i> do <i>software Visual Studio Community</i> <i>2017</i>	25
Figura 12: Tela para adicionar um novo idioma ao projeto de localização, clicando-se no ícone +, escolhendo idioma	25
Figura 13: Seleção do idioma	26
Figura 14: Seleção do idioma	27
Figura 15: Seleção da tradução automática	31
Figura 16: <i>TAV UFU Comment Editor</i>	33
Figura 17: <i>TAV UFU Comment Editor menu localized into English</i>	34
Figura 18: Aba “Sobre”	36
Figura 19: Aba <i>About</i>	37
Figura 20: <i>Github</i>	38
Figura 21: <i>Github</i>	39

LISTA DE QUADROS

Quadro 1: <i>Software</i> em português e inglês.....	30
Quadro 2: Texto da aba “Sobre”	36

SUMÁRIO

INTRODUÇÃO	10
CAPÍTULO 1 - A CRIAÇÃO E APLICAÇÃO DO <i>SOFTWARE</i> EDUCATIVO <i>TAV UFU EDITOR DE COMENTÁRIOS 1.0</i>.....	13
CAPÍTULO 2 - PROCEDIMENTOS METODOLÓGICOS QUE EMBASARAM A TRADUÇÃO E LOCALIZAÇÃO DO <i>SOFTWARE</i> EDUCATIVO <i>TAV UFU EDITOR DE COMENTÁRIOS 1.0</i>	19
CAPÍTULO 3 - A TRADUÇÃO E LOCALIZAÇÃO DO <i>SOFTWARE TAV UFU EDITOR DE COMENTÁRIOS 1.0</i>: TECNOLOGIA, TRADUÇÃO E PUBLIACAÇÃO	21
3.1 – AS TECNOLOGIAS ENVOLVIDAS NA TRADUÇÃO DO <i>TAV UFU</i>	22
3.2 – A DESCRIÇÃO DOS DESAFIOS TRADUTÓRIOS	27
3.3 – PUBLICAÇÃO E COMPARTILHAMENTO EM REDE DO EDITOR DE COMENTÁRIOS <i>TAV UFU 1.0/ COMMENT EDITOR TAV UFU 1.0</i>	37
PALAVRAS FINAIS.....	40
REFERÊNCIAS	42
APÊNDICE A - LOCALIZATION, KEIRAN DUNNE (2015).....	43

INTRODUÇÃO

As origens da localização remontam ao surgimento do computador pessoal no final dos anos de 1970 e início de 1980. Os computadores pessoais começaram a se popularizar entre usuários que não possuíam habilidades de programação, e, como resultado, muitas empresas de computação dos EUA se propuseram a atender o mercado de forma mais abrangente. Uma vez que empresas como *Sun Microsystems*, *Oracle* ou *Microsoft* conseguiram popularizar seus produtos nos EUA, elas se voltaram aos mercados internacionais; os alvos iniciais eram o Japão e os chamados países FIGS (França, Itália, Alemanha e Espanha, do inglês *France*, *Italy*, *Germany* e *Spain*). Razões econômicas são, portanto, facilmente identificáveis como o principal impulsionador para o surgimento e a evolução da localização (JIMÉNEZ-CRESPO, 2013; DUNNE, 2015).

O termo localização, no entanto, não existe isoladamente, mas faz parte de um amplo e complexo processo inter-relacionado conhecido como GILT, ou *Globalization, Internationalization, Localization and Translation*, em português, Globalização, Internacionalização, Localização e Tradução.

A globalização, representada pelo acrônimo G11n (onze caracteres entre G e n), diz respeito à organização geral do processo. Trata-se de um processo cíclico que ocorre não apenas antes, em nível de projeção econômica dos setores, mas também durante o processo e seu suporte subsequente.

A internacionalização ocorre durante os estágios de desenvolvimento de todo o produto digital e pode ser descrita como a etapa que visa garantir que o produto não esteja atrelado a uma única cultura, e que seja independente de qualquer língua, evitando um *future redesign* (nova programação do *software*). A internacionalização consiste basicamente em separar a funcionalidade de um produto de qualquer língua em específico, de modo que o suporte ao idioma possa ser integrado ao produto de forma simples.

A localização é o processo que contempla a preparação, manipulação, engenharia e qualidade de *softwares* utilitários, de entretenimento e de *websites*, sendo *locale* (local), a combinação de uma região sociocultural com uma língua. Refere-se ao processo por meio do qual os textos contidos nesses produtos, bem como suas especificações técnico-tecnológicas são modificadas para serem usadas em diferentes regiões. Tomando o Brasil como exemplo, pode-se adaptar um *software* para ser utilizado em português do Brasil, demandando várias modificações legais e adaptações culturais. Um *locale* não é uma língua, apesar de a língua ser um dos componentes vitais da localização. Tecnicamente, os *locales* são representados por

um código de duas letras combinado com o código do país, como no caso de pt-BR, ou pt-PT, para designar o português do Brasil e o português de Portugal, respectivamente.

A tradução, como parte do processo GILT, é geralmente terceirizada e considerada pela indústria uma etapa separada da localização. A integração do texto que o tradutor traduz pode ser feita pelos engenheiros, programadores ou diretamente por tradutores-localizadores que são assistidos por tecnologias desenvolvidas para esse fim. A Figura a seguir ilustra a interdependência do processo GILT.

Figura 1: Processo GILT



Fonte: adaptado e traduzido a partir de JIMÉNEZ-CRESPO (2013) pela autora e orientadora

Inicialmente, foram contratados profissionais da linguagem para traduzir as linhas de texto, os chamados linguistas (DUNNE, 2015). As primeiras tentativas mostraram-se impraticáveis, já que o texto era extraído, traduzido e depois reintroduzido no produto, sendo que as traduções com segmentos muito longos não se encaixavam no espaço a eles destinado, além de exigir conhecimento sobre linguagem de programação dos profissionais envolvidos. Dunne (2015) ressalta ainda que os problemas relacionados à localização de *software* envolviam correção de erros e aumento nos custos.

Assim, no começo os anos de 1990, diante do desafio de controlar a complexidade e o custo dos processos de localização, os desenvolvedores começaram a perceber que as etapas do processo poderiam ser executados com antecedência para tornar a localização mais fácil e, com isso, passaram a separar as linhas de texto traduzíveis do código-fonte. Nas palavras de

Dunne (2015), a internacionalização não só eliminou a necessidade de manter um conjunto de código-fonte para cada *locale*, mas também esclareceu os respectivos papéis dos programadores, engenheiros e não mais linguistas, mas os já nomeados tradutores.

A separação dos textos traduzíveis do código-fonte do *software* para fins de localização facilitou a tarefa dos tradutores que, na etapa da tradução, passaram a manipular as linhas texto do menu, das caixas de diálogos e de outras partes do *software* por meio de ferramentas tecnológicas visuais de localização, ou as chamadas *WYSIWYG* (do inglês, *What you see is what you get*; em português, o que você vê é o que você tem), que lhes permitiam visualizar os recursos do *software*, simulando sua futura interface com o usuário, ao mesmo tempo em que está sendo traduzido.

Tradicionalmente, as ferramentas disponíveis para a localização de *software* são específicas e podem ser acopladas a sistemas de memória de tradução ou a sistemas de tradução automática.

Tendo como base as atuais tendências e possibilidades de manipulação do conteúdo linguístico de *softwares*, que cada vez mais apontam para a necessidade e vantagem de implementação de processos de localização, haja vista que o usuário prefere consumir produtos em sua própria língua (DUNNE, 2015), este trabalho tem como proposta explorar o tema em nível teórico e traduzir e localizar, do português para o inglês, o *software* educativo *TAV UFU Editor de Comentários 1.0*, cuja função será explicitada adiante.

Assim, esta monografia está dividida da seguinte forma: além desta breve introdução, o capítulo 1 contextualiza a criação e aplicação do *software* educativo *TAV UFU Editor de Comentários 1.0*; o capítulo 2.0 delinea os procedimentos metodológicos que embasaram sua tradução e localização, sendo que o capítulo 3.0 apresenta os resultados de tais processos e seus desafios tradutórios e tecnológicos envolvidos na forma de tradução comentada. O capítulo intitulado “Palavras Finais” resume os achados deste trabalho e vislumbra possíveis encaminhamentos de pesquisas dessa natureza. Não menos importante, o Apêndice A desta monografia apresenta a tradução do texto *Localization* de Keiran Dunne (2015), realizada pela proponente desta monografia, que embora não seja comentada nos moldes da metodologia de “tradução comentada”, serviu como exercício tradutório do inglês para o português e também como embasamento teórico para que se pudesse melhor compreender os conceitos, as definições e incrementar a metalinguagem que permeia essa subárea dos Estudos da Tradução, *i.e.*, a Tradução e Localização.

Capítulo 1 - A criação e aplicação do *software* educativo TAV UFU EDITOR DE COMENTÁRIOS 1.0

Da mesma maneira que um tradutor literário necessita conhecer a escola literária à qual pertence o autor que irá traduzir, um tradutor-localizador de *software* necessita conhecer a que serve um determinado *software*. Assim, primeiramente, busca-se contextualizar a criação do *software* educativo que será aqui traduzido e localizado.

Pode-se dar início a essa contextualização afirmando que o ensino de tradução, de modo geral, tem se consolidado nos contextos universitários brasileiros e do exterior, e várias modalidades de tradução foram incorporadas aos currículos dos cursos de formação de tradutores, dentre elas a tradução audiovisual. Ensinada em programas de graduação ou pós-graduação, a depender do perfil da instituição e dos estudantes, o ensino da tradução audiovisual sempre esteve permeado pelas tecnologias, exigindo que professores e alunos tenham contato direto e contínuo com as novas tecnologias, especialmente para que possam adquirir as habilidades necessárias para o uso de programas computacionais, manipulação de arquivos de áudio e vídeo em diferentes formatos, sem contar as buscas em bancos de dados terminológicos e na internet de modo geral, tornando possível perceber que a tradução para produção de legendas e dublagens não significa transcrever, intra ou interlingualmente, o discurso oral de uma língua para outra em duas linhas, no rodapé de vídeos.

Para Diaz Cintas (2015), antigamente, na década de 1970, os profissionais de legendagem apenas necessitam de ferramentas tais como: um computador, um *player* externo de vídeo compatível ao formato VHS, os materiais a serem traduzidos, um monitor de televisão para exibir os materiais audiovisuais a serem legendados.

Para o autor, tal situação mudou bastante e, atualmente, com um computador pessoal, uma cópia digital, um *software* especialmente voltado à legendagem, um suplemento *spell checker* (ou corretor ortográfico), os tradutores-legendadores podem simular as legendas de forma bastante eficaz, restando pouco trabalho aos técnicos dos estúdios. Para Diaz Cintas (2015):

A capacidade e a funcionalidade da maioria dos programas de legendagem profissional foram melhoradas em um ritmo incrivelmente rápido nas últimas décadas, com alguns dos principais fabricantes comerciais sendo EZTitles (www.eztitles.com), FAB (www.fab-online.com), Miranda Softel (www.miranda.com/softel), Spot (www.spotsoftware.nl) e Screen Systems (www.screensystems.TV), os últimos desenvolvedores do programa WinCAPS.¹ (DIAZ CINTAS, 2015, p. 634, tradução da autora).

¹ *The capability and functionality of most professional subtitling programs have been improved at an incredibly fast pace in recent decades, with some of the leading commercial manufacturers being EZTitles (www.eztitles.com), FAB (www.fab-online.com), Miranda Softel (www.miranda.com/softel), Spot*

O autor ainda explica que o fato de tais programas profissionais serem tradicionalmente caros e complexos, muitos tradutores e também universidades dedicadas à formação de tradutores em legendagem e dublagem têm recorrido aos programas livres, sendo os mais conhecidos e citados, por Diaz Cintas (2015), o *Subtitling Workshop* (<http://subworkshop.sourceforge.net>), *DivXL and Media Subtitler* (www.divxland.org/en/media-subtitler), *Aegisub* (www.aegisub.org) e *Subtitle Edit* (www.nikse.dk/SubtitleEdit), sendo este último utilizado especialmente na disciplina Tradução de Filmes no âmbito do Curso de Tradução da Universidade Federal de Uberlândia, pois aumenta o processo de marcação e segmentação (em inglês, *timing and spotting*) das legendas em sincronia com a imagem, detectando as mudanças na trilha sonora ou nas inserções de unidades auriculares diversas.

Diaz Cintas (2015) ainda cita as tecnologias voltadas à legendagem utilizadas por fãs, as chamadas *cloud subtitling platforms* (plataformas para legendagem nas nuvens, citando como exemplo a *ZOOsubs* (www.zoosubs.com), desenvolvida por *Zoo Digital*, e a *iMediaTrans* (www.imediatrans.com), desenvolvida por *i-Yuno Media Group*.

Com relação a programas para produção de legendas e dublagens desenvolvidos para gerar memórias de tradução, o autor cita a iniciativa de uma empresa de Taiwan, a *Webtrans Digital* (www.webtrans.com.tw), que desenvolveu a ferramenta *Wados*, aumentando a produtividades dos tradutores que trabalham com essa modalidade, com suporte para tradução automática. Nessa direção, vale a pena também ressaltar as tecnologias de reconhecimento automático de voz da *Google*, que geram legendas em vídeos do *YouTube*, fazendo uso dos avanços da tradução automática.

Com relação às tecnologias 3-D, Diaz Cintas (2015) cita ainda haver um avanço tecnológico com o desenvolvimento de ferramentas, como no caso da *Poliscript 3DITOR*, da empresa britânica *Screen Subtitling*, que reposiciona automaticamente as legendas para que essas não destruam a ilusão 3D e não causem náuseas e dores de cabeça no espectador.

Percebe-se, que todo esse arsenal tecnológico traz avanços à tradução audiovisual, mas, por outro lado, apresenta uma certa falta de preocupação com a profissionalização na atividade, colocando pouca ênfase no trabalho do tradutor e requerendo deste mais conhecimento tecnológico do que atenção à atividade de produção e criação de qualidade das legendas em si. Além disso, a indústria cinematográfica e as empresas de *software* de um

(www.spotsoftware.nl), and *Screen Systems* (www.screensystems.tv), the latter developers of the program *WinCAPS*.

modo geral também não se preocupam com a formação de tradutores-legendadores, restando às universidades o uso passivo das ferramentas ou uma busca incessante pelas novas funcionalidades que são constantemente a elas agregadas.

Pensando em formas de suprir tais lacunas, pelo menos parcialmente, o *software* educativo *TAV UFU 1.0*, desenvolvido exclusivamente para a sala de aula, objetiva servir como ferramenta para o professor de tradução audiovisual. Criado por iniciativa² da Profa. Dra. Marileide Dias Esqueda, professora de tradução audiovisual do Curso de Bacharelado em Tradução e do Programa de Pós-Graduação em Estudos Linguísticos da Universidade Federal de Uberlândia, o programa pode servir para que o professor de tradução audiovisual insira comentários nos vídeos e legendas em que os alunos estão trabalhando, com o propósito de sugerir alterações e correções envolvendo a segmentação, a marcação, bem como o componente linguístico em si.

A título de ilustração, a Figura 2 mostra o ícone para abertura do *software*, que utiliza o logotipo do Curso de Tradução da UFU para sua identificação.

Figura 2: Ícone de abertura do software



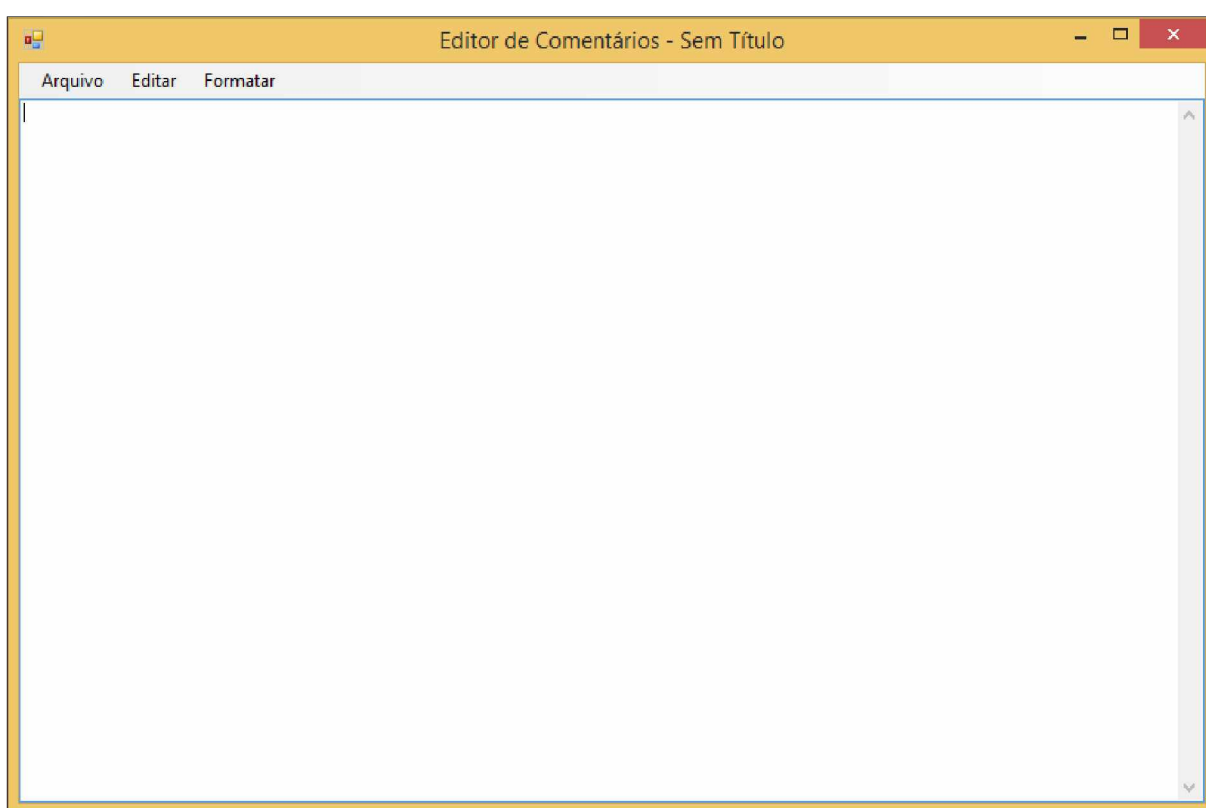
Fonte: *TAV UFU Editor de Comentários 1.0*

O *software* foi criado e desenvolvido, entre os anos de 2014 e 2015, no *Visual Basic Studio (VB)*, que possui versão gratuita para desenvolvimento de *softwares* (<https://www.visualstudio.com>). O *TAV UFU*, que é autoexecutável e *open source*, pode se acoplar a quaisquer outras aplicações de reprodução multimídia, tais como *VLC*, *Media*

² Após aprovação de seus projetos na modalidade PBG – Programa de Bolsa de Graduação, a professora contou com o auxílio de alunos-bolsistas da FACOM UFU (Faculdade de Computação da Universidade Federal de Uberlândia) para o desenvolvimento, teste e aplicação do *software* educativo.

Player, *KMPlayer* ou outros. De usabilidade simples, amigável e com pouca densidade informacional (Figura 3), o editor de comentários utiliza o arquivo de legenda gerado por outros *softwares* tais como *Subtitle Workshop*, *Subtitle Edit* ou outros e o edita na interface do *TAV UFU Editor de Comentários 1.0*, que auxilia o professor de tradução audiovisual, como o próprio nome já deixa antever, a inserir comentários, em posições por ele escolhidas no arquivo de legendas gerada pelo aluno, possibilitando as sugestões de correção, edição e aprimoramento das legendas mediante os critérios de avaliação adotados pelo professor.

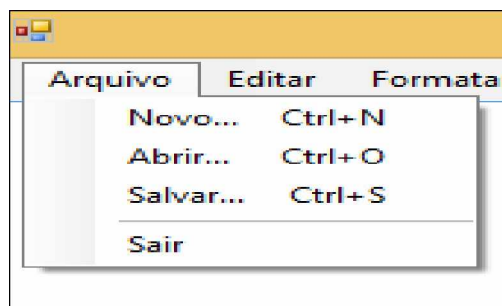
Figura 3: Captura da tela principal do *software TAV UFU Editor de Comentários 1.0*



Fonte: *TAV UFU Editor de Comentários 1.0*

Conforme mostra a Figura 4 no item “Novo...”, pode-se criar um novo arquivo de comentários, também com padrão de extensão “.srt”, o mesmo de edição de legengas, para que possa ser posteriormente utilizado nos *softwares* de reprodução de vídeo ou nos *softwares* de edição de legendas. O item “Abrir...” permite abrir um arquivo de comentário já existente (neste caso os arquivos em .srt enviados pelos alunos) e os itens “Salvar...” e “Sair” desempenham as funções designadas por seus títulos.

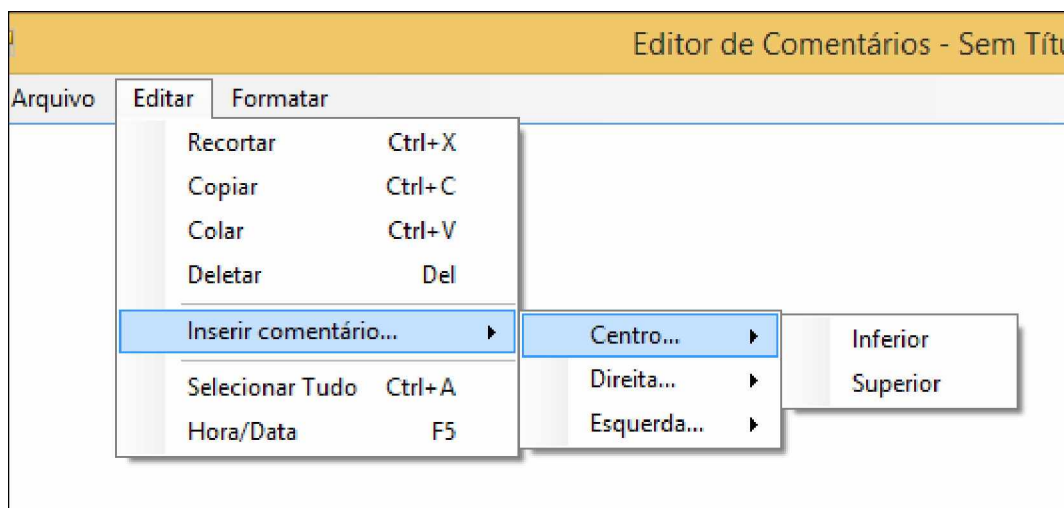
Figura 4: Menu “Arquivo”



Fonte: TAV UFU Editor de Comentários 1.0

O item “Inserir comentário...”, como mostra a Figura número 5, permite que o professor insira um texto breve ou comentário à legenda produzida pelos alunos. Esse comentário aparecerá em uma área específica do vídeo de acordo com a posição que o docente escolher (centro; direita ou esquerda; superior ou inferior).

Figura 5: Menu “Inserir comentário”



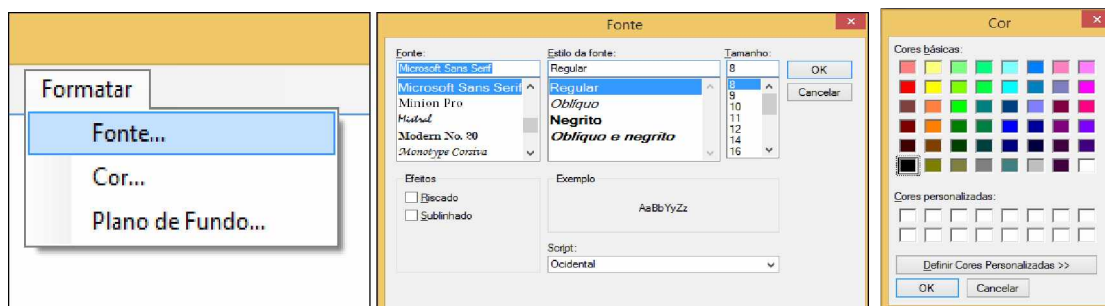
Fonte: TAV UFU Editor de Comentários 1.0

Ao escolher a posição do comentário, uma janela será aberta para que seja inserido o comentário, antecedido sempre de uma tag, (por exemplo: $\{a3\}$).

O professor também pode, para facilitar a leitura do comentário, escolher em “Formatar”, a cor da fonte, tamanho, estilo e efeitos dos comentários, principalmente para que possam ser distintos do padrão e cor da fonte das legendas para o cinema e televisão, que normalmente é *Helvetica*, *Arial* e *Times New Roman*, de tamanho 12, predominantemente em branco e em menos ocorrência em amarelo (DIAZ CINTAS, 2003). O item “Plano de fundo”

do comentário também pode ser alterado, facilitando a visualização e acessibilidade para alunos com certa ou acentuada dificuldade visual.

Figura 6: Menu “Arquivo”



Fonte: TAV UFU Editor de Comentários 1.0

Na Figura a seguir, pode-se averiguar como os comentários do professor podem ser acrescentados às legendas produzidas, em conjunto com o vídeo. Esta figura ilustra a funcionalidade do *TAV UFU*. Utilizado na disciplina de Tradução de Filmes, ministrada pela professora Marileide, o *TAV UFU* recebe o arquivo *.srt* que os alunos enviam para a professora, as abrem no *software* educativo, realiza os comentários acerca da legenda, salva o arquivo contendo os comentários feitos por ela também em *.srt* e envia para os alunos, que podem abrir as correções em qualquer *software* de legendagem que consegue identificar o *.srt*.

Figura 7: Comentário no arquivo de legenda e vídeo por meio do *TAV UFU Editor de Comentários*



Fonte: Cena do Filme *The Bling Ring* no software de reprodução de vídeo VLC.

Uma vez que o *software* educativo *TAV UFU Editor de Comentários 1.0* foi desenvolvido no *software Visual Basic Studio 2015*³, optou-se, assim, que sua tradução e localização fossem realizadas também nesse mesmo *software*, porém em sua versão *Visual Studio Community 2017* e cujos detalhes serão explicitados no próximo capítulo.

Capítulo 2 - Procedimentos metodológicos que embasaram a tradução e localização do *software* educativo *TAV UFU EDITOR DE COMENTÁRIOS 1.0*

Tendo como base a necessidade cada vez maior de internacionalização de *softwares*, que consiste na preparação do produto para receber a tradução para outras línguas e locais, assim democratizando os avanços tecnológicos (DUNNE, 2015), indagou-se quais poderiam ser os desafios presentes no processo de tradução e localização, na direção linguística português-inglês, do *software* educativo *TAV UFU Editor de Comentários 1.0*.

A metodologia utilizada nesta monografia pode ser descrita a partir de um modelo aplicado processual, que está interessado na sequência de escolhas e tomada de decisão na tarefa de tradução e localização.

Trata-se de uma pesquisa empírica, de viés experimental, que ao produzir a tradução de um determinado produto, isola características particulares deste para fins de interpretação de seus problemas e desafios. Não se trata de levantar dados estatísticos, mas sim de investigar aspectos mais técnicos, tecnológicos e linguísticos do *software* ora apresentado, destacando as complexidades linguísticas no par e direção linguística português-inglês e suas demandas de localização.

Embora não seja o propósito discutir os preceitos teóricos da tradução comentada, este trabalho alia-se, de outro lado, aos tipos de comentários sobre textos traduzidos que normalmente são elaborados em obras editoriais (ZAVAGLIA et al., 2015), adequando-os aos comentários que podem ser feitos em relação à tradução e localização de *softwares*.

Nesse prisma, a tradução comentada que será aqui produzida mostrará e discutirá, primeiramente, as tecnologias envolvidas na tradução e localização do *software* educativo *TAV UFU Editor de Comentários 1.0*, que possibilitam a manipulação do conteúdo linguístico do *software*, para que, em seguida, se possa produzir os comentários de caráter propriamente linguístico. A proposta é relatar o percurso tradutório de uma estudante de Tradução, suas dúvidas, escolhas iniciais e finais e seu embasamento teórico. Para Williams e Chesterman

³ *Microsoft Visual Studio* é um ambiente de desenvolvimento integrado da *Microsoft* para desenvolvimento de *software* especialmente dedicado ao *.NET Framework* e às linguagens *Visual Basic*, *C*, *C++*, *C#* e *J#*.

(2002), esse tipo de trabalho aumenta a autoconsciência da qualidade de tradução realizada por tradutores, fazendo com que a tradução não exerça função acessória, mas revele-se em uma atividade que, independentemente de sua modalidade ou tipo, sempre demandará a tomada de decisão e a produção criativa e original do tradutor.

A tradução comentada será feita a partir dos desafios linguísticos que a localização de um *software* apresenta e será organizada com base nos seguintes procedimentos metodológicos: 1. Apresentação das tecnologias envolvidas em projetos tradutórios de natureza digital e interativa; 2. descrição e desafios tradutórios do componente linguístico do *software*, em confluência com as teorizações versadas nesse tipo de tradução; 3. descrição da publicação do *software* e seu compartilhamento em rede.

A localização do *TAV UFU – Editor de Comentários 1.0* foi feita no *software Visual Studio Community 2017*, mais especificamente na aba *Multilanguage*, ferramenta essa utilizada exclusivamente para fins de localização. As linhas de texto do *software* educativo estão disponíveis dentro uma pasta cujo nome é “*kit loc*”, e a partir do momento em que o *software* é aberto no *Visual Studio Community*, essas linhas de textos são separadas da linguagem de binária, ou seja a linguagem própria de desenvolvimento de *softwares*, fazendo com que o responsável pela tradução não corra o risco de alterar algum código de programação. E quando as linhas de texto (natural) foram retiradas do código fonte, o processo de tradução se inicia, e uma vez terminado, o tradutor volta a salvar o trabalho na mesma pasta, e o programa funcionará com o sistema de dois idiomas, que é o caso do *software* educativo *TAV UFU – Editor de Comentários*.

Assim, com vistas a descrever seus possíveis desafios tradutórios, o objetivo geral deste trabalho é traduzir e localizar, do português para o inglês, o *software* educativo *TAV UFU Editor de Comentários*, já que também não se encontra na literatura internacional *softwares* com esse propósito didático-pedagógico, *i.e.*, direcionado ao professor de tradução audiovisual. Por ser um *software* inédito, também torna-se inédita sua tradução e localização para a língua inglesa, tornando-se um exercício tradutório autêntico.

Para cumprir tal objetivo geral, são traçados os seguintes objetivos específicos:

1. Traduzir do português para o inglês o *software TAV UFU Editor de Comentários 1.0*, avaliando os desafios tradutórios nele contidos, através do *software Visual Studio Community 2017*, específico para manipulação de outros *softwares* e suas interfaces com o usuário;
2. Coletar e categorizar o componente linguístico presente no *software*;

3. Discutir os aspectos linguísticos presentes na aplicação e relacioná-los ao arcabouço teórico sobre tradução e localização de *software*;

Vale ressaltar que utilizar parcialmente os preceitos da Tradução Comentada neste trabalho é mostrar que essa abordagem pode ir além da área na qual ela geralmente está associada, a Literatura. No caso aqui apresentado, pareceu ser possível associar uma linguagem mais curta e objetiva com a análise de seu processo tradutório, pois todas as traduções e quaisquer traduções, por mais simples que sejam, apresentam desafios passíveis de comentários. Para Zavaglia (et al) (2015), os comentários podem ser feitos de quatro maneiras:

1. Forma de discussão sobre tarefa tradutória;
2. Análise do texto fonte e do contexto em que foi escrito;
3. Justificativas sobre os problemas encontrados;
4. Soluções propostas na tradução.

Nesta pesquisa, utilizou-se os dois últimos tópicos, pois foram encontrados desafios tradutórios durante o percurso. E justificativas foram feitas para a defesa da escolha tradutória que se foi feita.

Capítulo 3 - A tradução e localização do *software* TAV UFU EDITOR DE COMENTÁRIOS 1.0: tecnologia, tradução e publicação

De acordo com Prudêncio (2004), com a globalização da economia e o advento da Internet, tornou-se muito mais fácil atingir os mercados externos. Quem usa o computador diariamente já notou que a maioria dos programas usados hoje em dia não foi criada no Brasil, embora muitos deles estejam em português. A globalização e a consequente interação de mercados fizeram com que a internacionalização e a localização de *software* e de *websites* passassem a ser uma preocupação constante para as empresas que agem ou pretendem agir globalmente. Para exportar um *software*, assim como para exportar qualquer outro produto, é necessário atender a certos requisitos de qualidade. Alguns países inclusive exigem por lei que o *software* vindo de países estrangeiros seja traduzido para o seu idioma. Para tal, é necessário que se pense, desde o princípio, no desenvolvimento de programas que possam ser distribuídos e comercializados em outros idiomas.

Para o autor, o tradutor-localizador deve ser o responsável pela tradução de conteúdo eletrônico (*websites*, textos de ajuda *on-line* de *software*, a interface de *software* – botões, menus, janelas), da documentação técnica de *software* e material extra (material de *marketing*, embalagens de produtos, folhetos de referência rápida ou passos de gerenciamento de um *software*, documentação de registro da aplicação, manual do usuário etc. É de sua responsabilidade também definir (em cooperação com o desenvolvedor do *software*), traduzir e gerenciar a terminologia envolvida nas atividades de tradução. O foco do trabalho do tradutor-localizador é a busca pela qualidade, eficiência e precisão nas tarefas, coerência terminológica, redução de custos totais, com reutilização de traduções.

Prudêncio (2004) ainda explica que o fato de todas as *strings* de texto (sequências de caracteres que formam orações ou frases) da interface visível ficarem em arquivos separados do código-fonte do programa, torna todas as ações mais fáceis, tanto para os programadores quanto para a equipe de tradução que realizará a localização do *software*. Os programadores não precisam se preocupar com esses textos e tampouco com suas traduções, apenas precisam ter em mente que cada identificador representa um texto que será substituído automaticamente por outra função.

Neste capítulo, serão apresentados os resultados da tradução do *software* educativo *Editor de Comentário TAV UFU 1.0*. Conforme explicitado antes, este capítulo terá como subdivisão: (1) a apresentação das tecnologias envolvidas nesse projeto tradutório; (2) a descrição e desafios tradutórios do componente linguístico do *software*, em confluência com as teorizações na área de Estudos da Tradução desse tipo de tradução; e, por último, (3) a descrição da publicação do *software* e seu compartilhamento em rede.

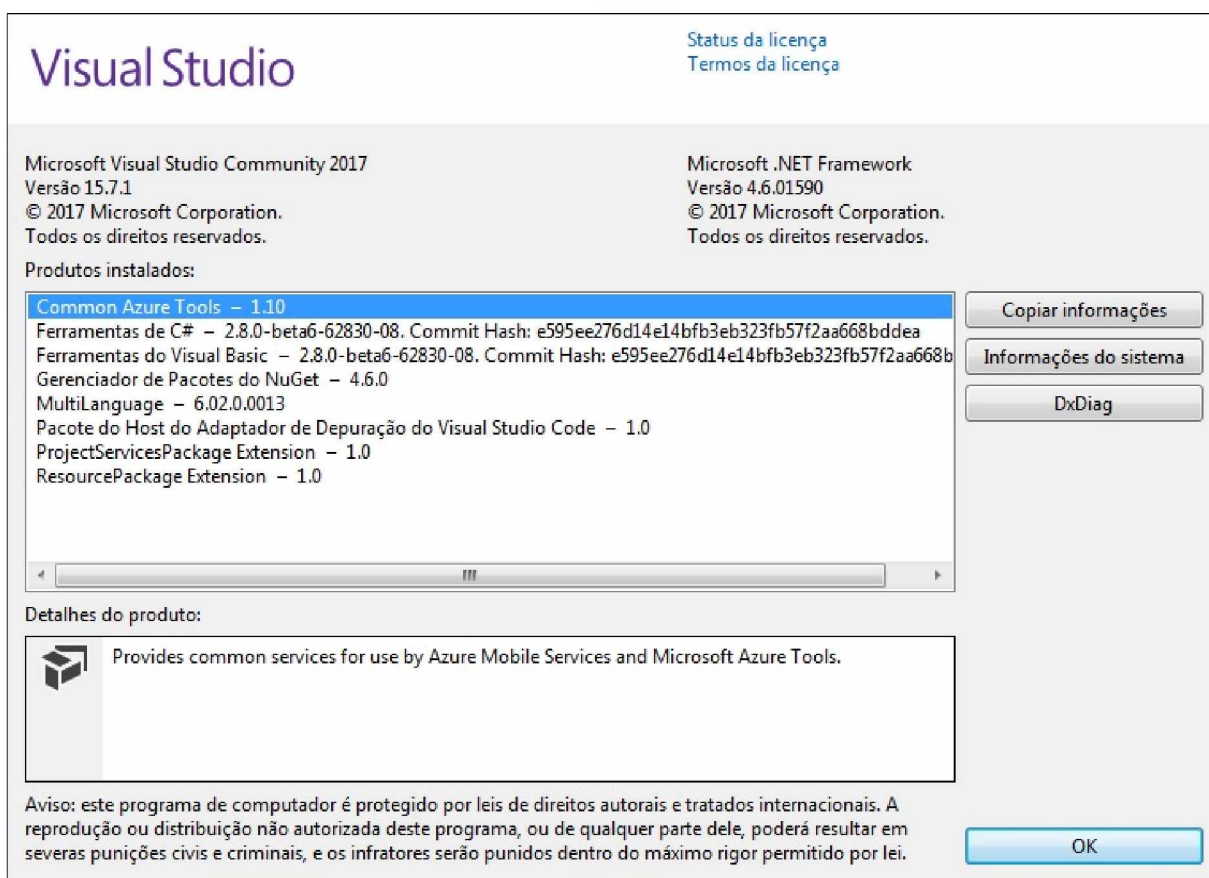
3.1 – As tecnologias envolvidas na tradução do *TAV UFU*

Preferiu-se a ferramenta em que este foi desenvolvido, o *Visual Basic*, evitando-se, assim, conversões de arquivos para sua compilação final. Nesse sentido, a proponente desta monografia pode expandir seus conhecimentos acerca de programação de *software*, utilizando um produto com o qual não teve contato anterior.

O *Visual Basic* é da empresa norte-americana *Microsoft* e tem várias versões, sendo algumas pagas e outras comercializadas especialmente para grandes empresas.

Neste estudo, foi utilizada a versão *Visual Studio Community 2017*, uma versão gratuita.

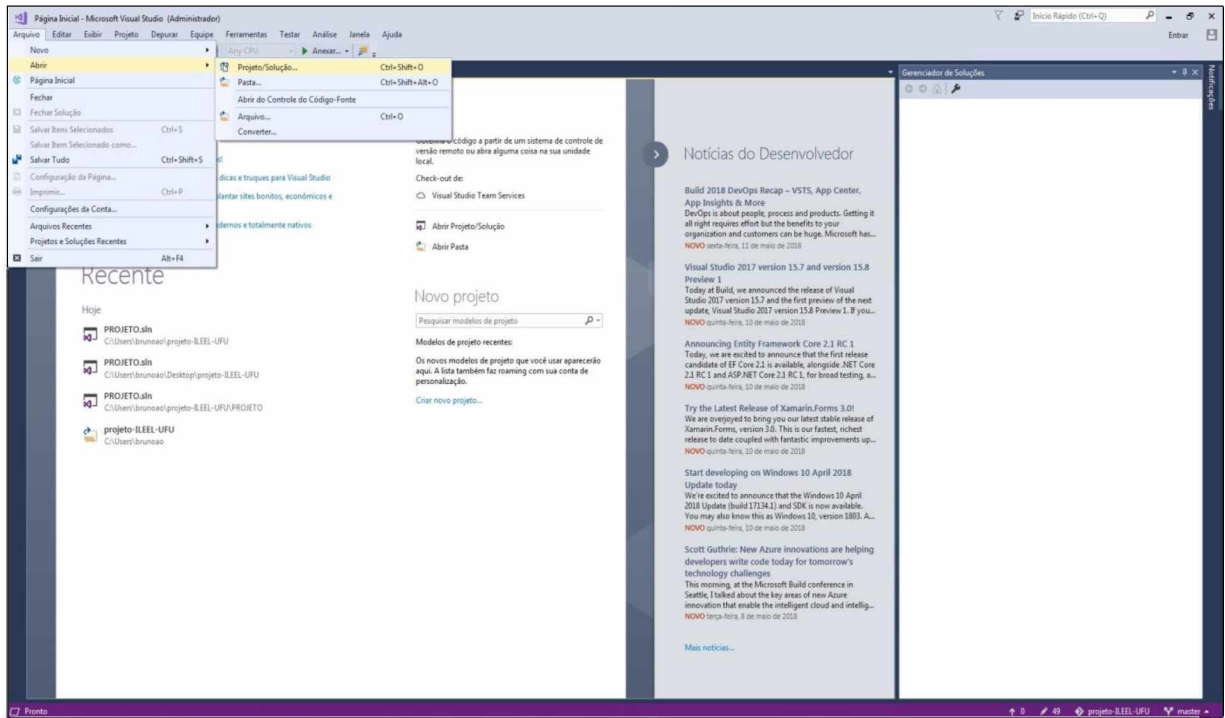
A Figura 8 traz a visão ampla das configurações do *Visual Studio Community 2017*.

Figura 8: Tela inicial do *Microsoft Visual Basic Studio 2017*

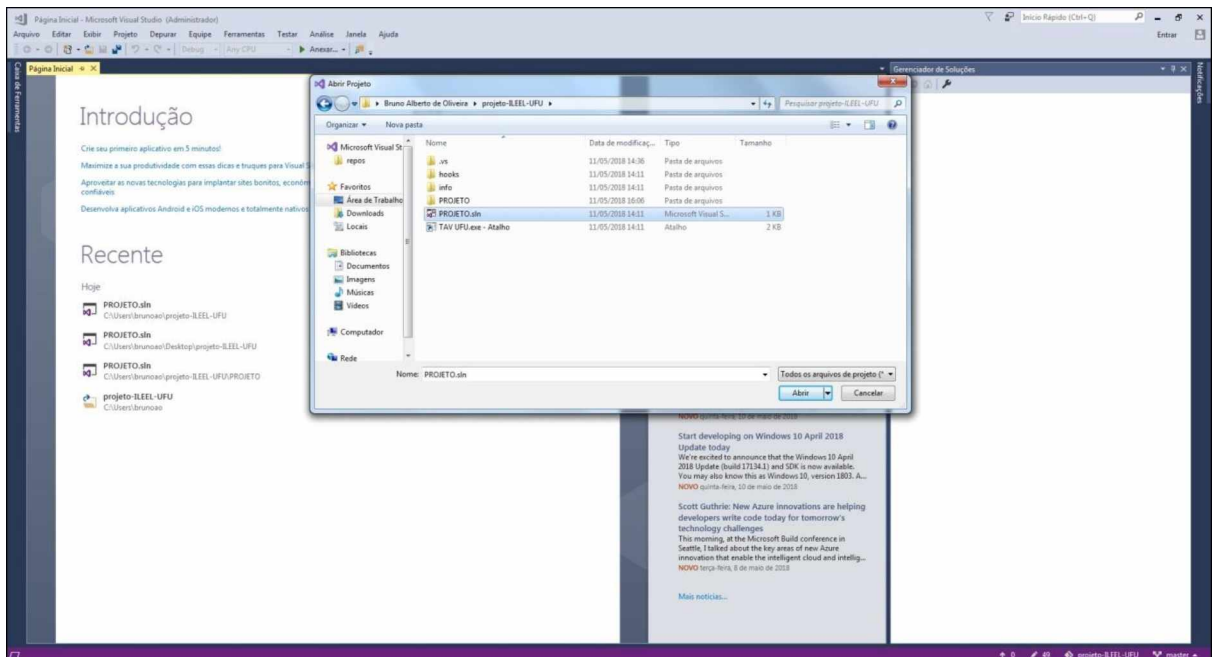
Fonte: *Visual Studio Community 2017*

Tendo sido instalada em um computador *LG, i5, Windows 7* utilizado no do LABTRAD – Laboratório do Curso de Tradução do ILEEL-UFU, este *software* possui uma ferramenta específica de localização e manipulação do componente linguístico do *software* a outras línguas, a ferramenta *Multilanguage*.

Para que o *TAV UFU Editor de Comentários* pudesse ser traduzido e localizado para o inglês, uma pasta contendo todos os arquivos do *software* foi instalada no referido computador. Assim, pode-se proceder à abertura do *TAV UFU* no *Visual Studio* e iniciar seu carregamento de arquivos (Figuras 9 e 10).

Figura 9: Tela inicial do *Microsoft Visual Basic Studio 2017*

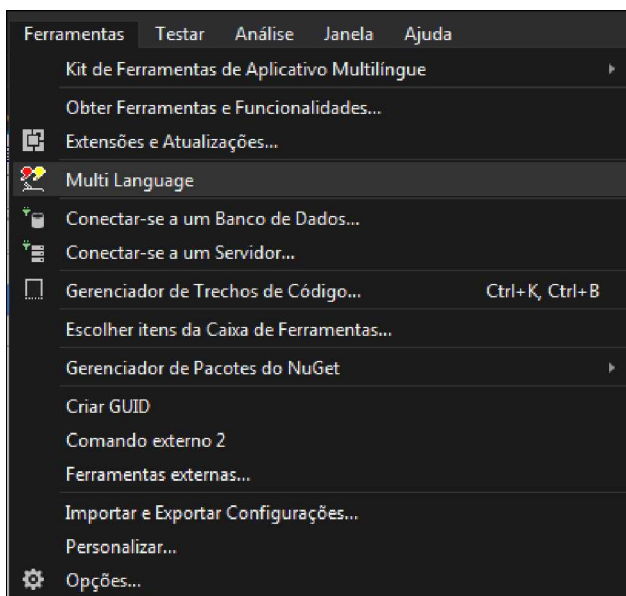
Fonte: *Visual Studio Community 2017*

Figura 10: Tela inicial do *Microsoft Visual Basic Studio 2017*

Fonte: *Visual Studio Community 2017*

Assim, o *TAV UFU* foi aberto e, por meio da ferramenta de localização *Multilanguage*, iniciou-se o processo de tradução e localização.

Figura 11: Ferramenta de localização *multilanguage* do software *Visual Studio Community 2017*

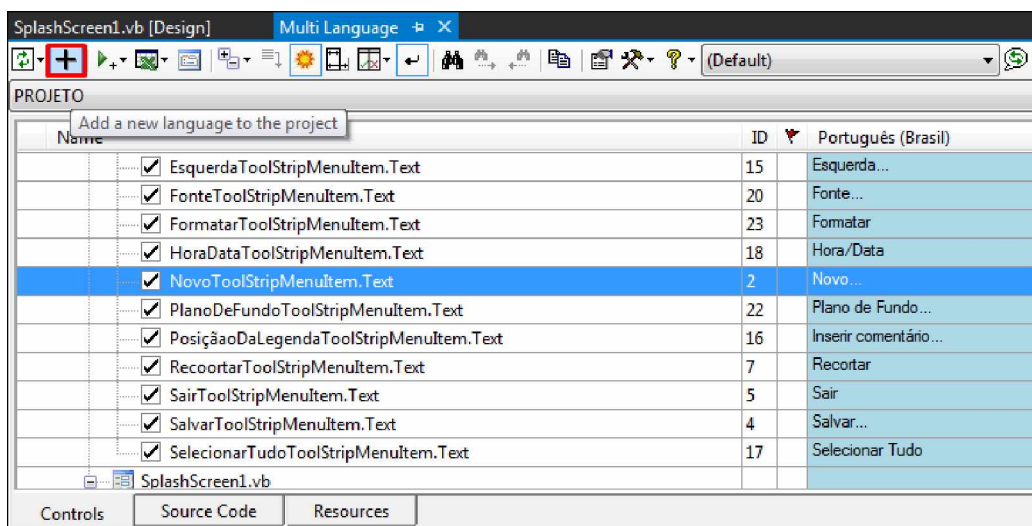


Fonte: *Visual Studio Community 2017*

Sua interface com o usuário (menus, caixas de diálogo etc.) puderam ser abertas, manipuladas e traduzidas para a língua inglesa.

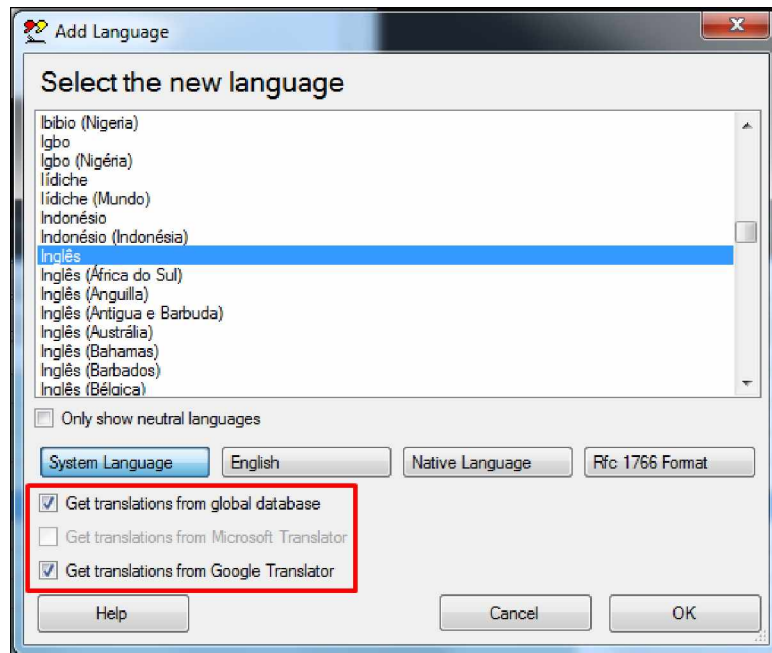
Na Figura 12 pode-se encontrar o processo de localização dentro da ferramenta *Multilanguage*.

Figura 12: Tela para adicionar um novo idioma ao projeto de localização, clicando-se no ícone +, escolhendo idioma



Fonte: *Visual Studio Community 2017*

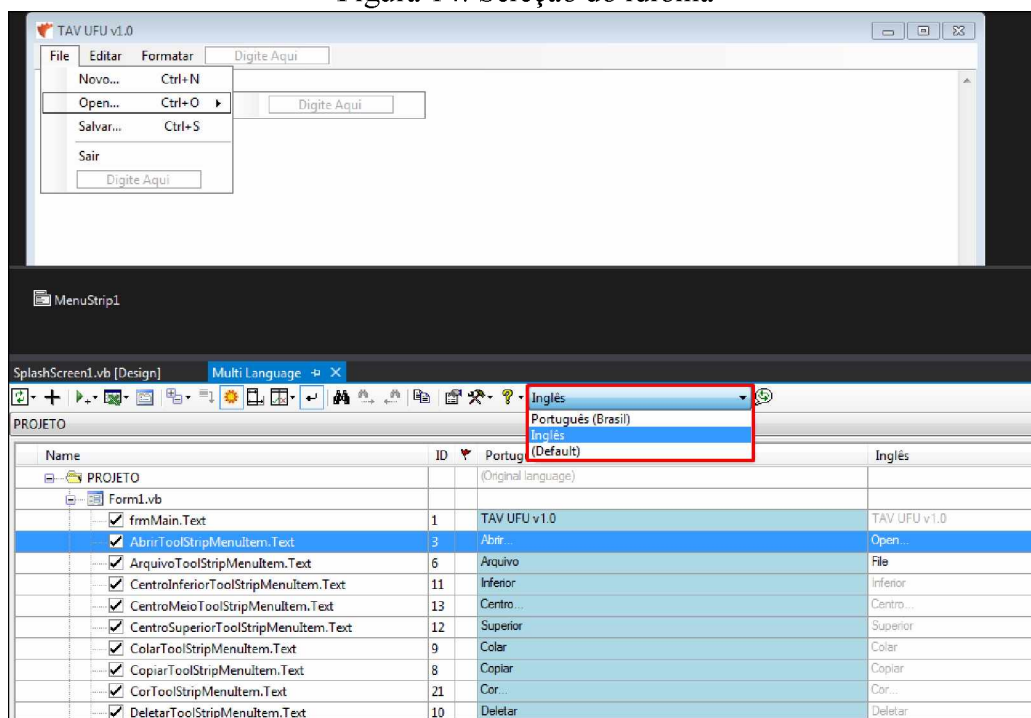
Figura 13: Seleção do idioma



Fonte: *Visual Studio Community 2017*

Na sequência, pode-se visualizar o *software*, no modo WYSIWYG, e proceder à tradução do menu inicial. *What You See is What You Get* (WYSIWYG) é a disposição de tela que é possível encontrar em *softwares* com ferramentas de localização. O tradutor pode ver em tempo real como o seu *software* ficará quando traduzido. A tela pode ser dividida meio a meio ou lado a lado, no caso do *Multilanguage*, a tela foi dividida ao meio, e a parte de cima geralmente contém o *software* que está sendo localizado e na parte de baixo as linhas de texto a serem traduzidas.

Figura 14: Seleção do idioma



Fonte: Visual Studio Community 2017

3.2 – A descrição dos desafios tradutórios

Para Keniston (1997), qualquer um que já tenha trabalhado com a localização de *softwares*, sejam eles sistemas operacionais ou aplicativos, pode atestar que os desafios surgem em nível técnico e estão longe de serem universalmente resolvidos. Se a ‘localização técnica’ é complexa, cara e exigente, o segundo aspecto da localização, que o autor chama de ‘localização linguístico-cultural’, que é a parte onde o tradutor entra em contato com o conteúdo linguístico e cultural de cada localidade, em que ele precisa pensar nas melhores estratégias e léxicos para o conteúdo textual que se encontra no material digital que ele tem em

mãos, e é ainda mais complexa, exigente e difícil, até mesmo para defini-la, sendo pouco discutida na literatura da área.

No que tange a problemas técnicos, não apenas Keniston (1997), mas também Collins (2001) e Sandrini (2005; 2008), dentre outros, explicam que quando a tradução é feita, precisa-se levar em consideração a interface com o usuário, ou seja, procurar não ultrapassar os limites delimitados para o itens linguísticos dos menus do *software* quando traduzidos a outras línguas.

Diante desse desafio, neste trabalho, buscou-se encontrar termos em língua inglesa que tivessem o número aproximado de letras que o português, porém houve ocorrências em que isso não foi possível, como por exemplo na tradução para o inglês do item “Idioma”. Em português, o termo é comumente utilizado na barra de ferramentas, indicando na aba que é possível a mudança do idioma da interface com o usuário. Em inglês, utilizar sua tradução literal, *idiom*, além de não ser o mais usual em barras de ferramentas de programas cuja interface com o usuário está em inglês, a palavra *idiom* diz respeito ao idiomático das línguas, expressões que caracterizam a língua, como é o caso das expressões idiomáticas. Assim, a opção de tradução mais satisfatória poderia ser *Choose language*, porém ultrapassando a questão do espaço reservado para a aba no *software*, portanto a opção final foi *Language*, como será mostrado no quadro 1 a seguir, que contém essa e outras linhas de texto do *software*.

Esse pode ser descrito como um desafio tradutório na tradução e localização de *softwares*, que se revela ao mesmo tempo como um desafio técnico. Nishiro (2013) também relatou que depois de trabalhar vários anos como tradutor, começou a estudar o desenvolvimento de *software* na graduação. O autor relata que aprendeu como criar uma aplicação com suporte para várias línguas, e como formatos, tais como datas ou números diferem de região para região. O autor ainda coloca que desconhecia o fato de que a separação decimal em alemão ser feita com vírgula em vez de ponto final e de como isso interfere na interface com o usuário.

A tradução e localização do componente linguístico diz respeito às hipóteses, aos valores e às perspectivas da cultura para a qual se traduz. Para Keniston (1997), um programa localizado culturalmente deve ser distinto daquele escrito por membros da cultura-fonte.

Segundo Collins (2001), a localização de *software* é baseada no mercado de cada país, bem como na implementação de testes de usabilidade no país-alvo. Vários aspectos culturais de um país são muitas vezes imutáveis, assim como a língua; outros aspectos são

mais sutis e suscetíveis a mudanças culturais, como as cores, o *design* dos produtos, que sofrem as demandas dos governos ou empresas.

Para Cyr e Lew (2003), a indústria da localização é motivada pelo *marketing* e a globalização do comércio. Se um cliente está escolhendo entre duas impressoras com preços e funcionalidades similares vendidas no Líbano, por exemplo, qual ele ou ela escolherá? Apesar da maioria das pessoas que procuram um computador para comprar no Líbano saberem ler em inglês, a localização afeta a preferência do comprador. Estudos de *marketing* mostram que provavelmente os consumidores vão comprar a impressora que vem com um manual em árabe. O retorno do investimento feito (em inglês, *ROI – Return On Investment*) em serviços de localização é imediato e pode ser lucrativo se os processos de tradução forem adotados de forma correta.

Conforme explica Sandrini (2005),

[...] Vista a partir da perspectiva dos Estudos da Tradução, o processo tradutório envolve a transferência linguística bem como cultural, e a intenção comunicativa ou função do texto-alvo é de importância geral. Logo, a tradução sempre envolve alguma forma de adaptação com respeito aos textos em si ou outros itens relevantes para o documento, como gráficos etc. (SANDRINI, 2005, p. 2, tradução da autora)⁴.

Sandrini (2005) ainda explica que na literatura da área existem indicações para ambas as hipóteses: de um lado existem cursos específicos para localizadores profissionais, associações para localizadores profissionais para apoiar a estruturação da profissão de localizador e lidar com todos os desafios que ela apresenta, como no caso da escolha de itens do menu; de outro lado, os cursos de formação de tradutores são mais lentos para se adequarem aos desafios desse novo campo de pesquisa. De qualquer modo, pode ser útil para ambos os campos aprenderem um com o outro ao invés de tentar reinventar a roda de um lado ou ser reduzido a uma mera substituição de itens linguístico, como tem sido o caso da tradução.

Um *software* é basicamente o código instalado em um arquivo executável que diz ao disco rígido o que fazer. Esse código de programação é dependente do tipo de processador sendo usado, e suas instruções necessitam funcionar da mesma maneira, antes e depois do processo de localização. Dentro do código de programa existem elementos que oferecem interações com o usuário: menus, caixas de diálogos, linhas de textos etc, que são traduzidos.

⁴ “[...] Seen from a translation studies perspective, however, the process of translation involves linguistic as well as a cultural transfer and the communicative intention or function of the target text is of overall importance. Translation, therefore, always involves some form of adaptation with respect to the text itself or other items relevant to the document such as graphics etc”.

O código pode também conter elementos específicos do *locale*, que são localizados, como por exemplo datas e horas, moeda, formato de papel etc. Acoplados ao código, encontram-se outros códigos que dizem respeito a instruções diretas, tais como *dir*, *list*, *copy* etc., que não podem ser considerados como texto a serem traduzidos. Tais instruções diretas são parte de uma linguagem de programação e não devem ser alteradas.

Para facilitar a visualização das linhas de texto contidas no *TAV UFU*, resume-se, no Quadro 1, as linhas do texto original em português do Brasil (pt-PB) e seus respectivos textos traduzidos pela autora desta monografia para o inglês norte-americano (en-US). Em sequência, quando pertinentes, são tecidos os comentários sobre os desafios encontrados.

Quadro 1: Software em português e inglês

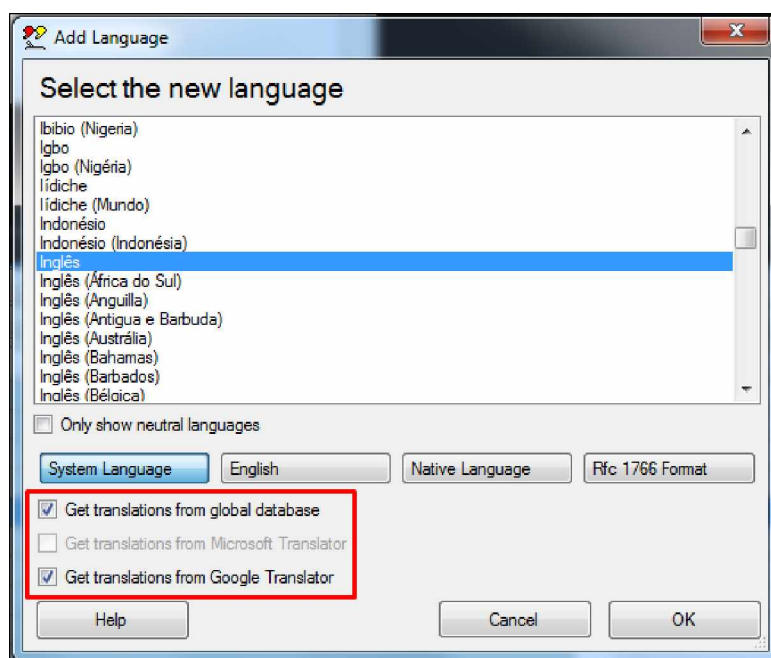
TO: PT-BR	TT: EN-US
Novo...	New...
Abrir...	Open...
Salvar...	Save...
Arquivo...	File...
Sair...	Close...
Recortar	Cut
Copiar	Copy
Colar	Paste
Deletar	Delete
Inferior	Bottom
Superior	Top
Centro	Middle
Direita	Right
Esquerda	Left
Inserir Comentário	Insert Comment
Selecionar tudo	Select All
Hora/Data	Time/Date
Editar	Edit
Fonte	Font
Cor	Color

Plano de Fundo	Background
Formatar	Settings
Idioma	Language
Selecionar Idioma	Choose Language

Fonte: a autora

Ressalta-se que o *Visual Studio* possui conexão direta, tanto com o *Global Database* (Base de Dados Global) da *Microsoft*, quanto do *Google Translate* (Google Tradutor) da empresa Google. Caso essa função de tradução automática esteja habilitada, ao se abrir o projeto para tradução e localização, a ferramenta consulta e exibe a tradução em sistemas de tradução automática de cada linha de texto, cabendo ao tradutor verificar sua adequação.

Figura 15: Seleção da tradução automática



Fonte: *Visual Studio Community 2017*

Neste estudo, foram consultados, além dos sistemas de tradução automática apresentados pela própria ferramenta, e de dicionários monolíngues e bilíngues *on-line*, outros *softwares* em língua inglesa para se compreender e optar por traduções mais satisfatórias e conhecidas de usuários brasileiros, neste caso professores, daí a adequação da tradução para o *locale*, ressaltada pelos autores que estudam o fenômeno da localização. (KENISTON, 1997; COLLINS, 2001; SANDRINI, 2005; 2008, DUNNE, 2015)

De acordo com a classificação de Zavaglia (et al) (2015), aqui, e também no caso da aba “Sobre”, que se encontra na página 35, utilizamos as justificativas sobre os problemas encontrados, pois desafios foram encontrados e logo abaixo eles serão listados e também

justificados, no sentido de escolha lexical e também de contexto, uma vez que é preciso levar em consideração as palavras já utilizadas em *softwares* sobre conteúdo audiovisual.

No caso de “Sair”, embora seja possível admitir um correspondente literal em inglês do tipo *Exit*, optou-se por *Close*, que indica maior relação e correspondência com os menus de *softwares* (tais como aqueles disponíveis em língua inglesa no LABTRAD, tais como *Subtitle Edit* ou *Notepad++*, por exemplo).

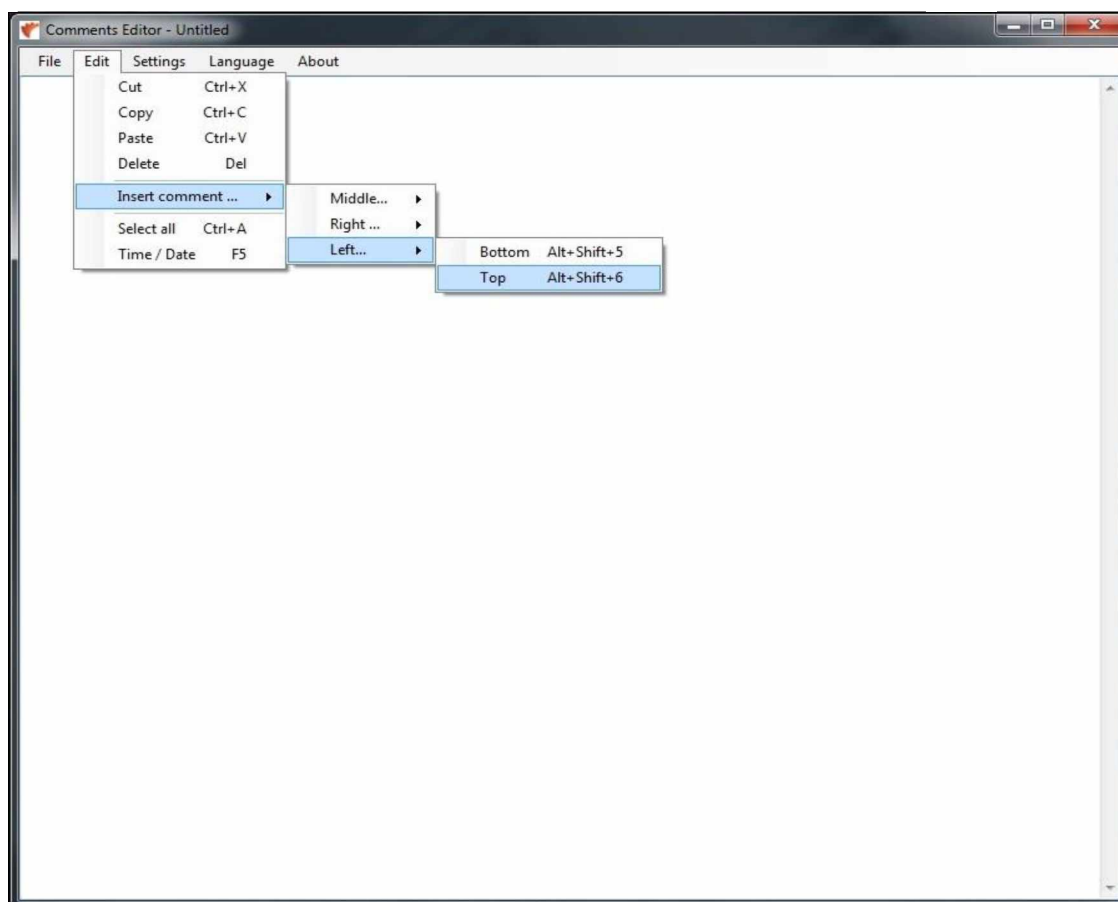
No caso da tradução da linha “Colar”, embora a tradução automática do *Global Database* e *Google Translate* tivessem exibido a tradução *to glue* e *to crib*, respectivamente, sendo essas traduções literais e inadequadas, a opção por *Paste* indica maior relação com os menus de *softwares* consultados.

Na sequência, manteve-se o calco *delete*, que possivelmente pode ser explicado a partir do surgimento do termo nos teclados dos *Personal Computers* de décadas anteriores (DUNNE, 2015).

Para as linhas de texto “Inferior”, “Superior” e “Centro”, embora seja possível admitir um correspondente literal em inglês do tipo *Inferior*, *Superior*, *Center*, optou-se por *Bottom*, *Top* e *Middle*, respectivamente, uma vez que o objetivo do *software* é comentar as legendas e exibir os comentários em diferentes posições na tela do computador.

Embora seja possível admitir um correspondente literal em inglês do tipo *Source* para a linha “Fonte”, principalmente em outros contextos, optou-se por *Font*, que indica maior relação e correspondência com os menus de *softwares*.

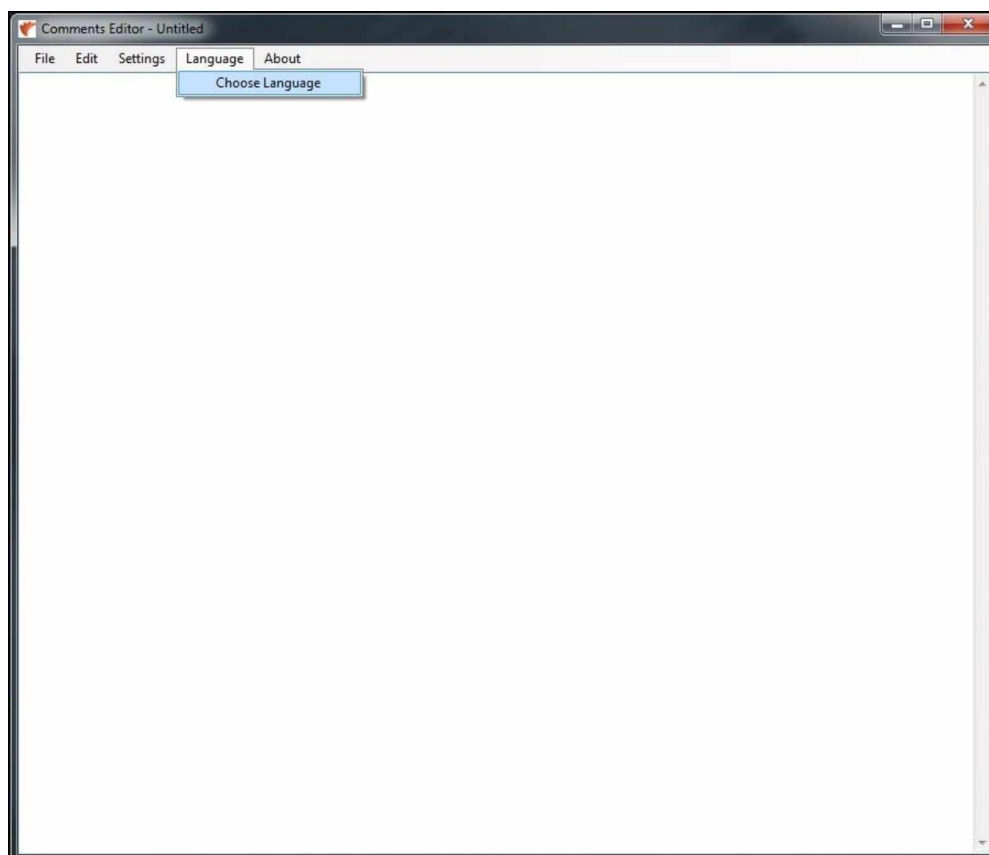
No caso da linha “Formatar”, embora seja possível admitir um correspondente literal em inglês do tipo *Format*, optou-se por *Settings*, que indica maior relação com os menus de *softwares* quando o usuário irá formatar ou configurar as opções, neste caso, de comentários no TAV UFU.

Figura 16: *TAV UFU Comment Editor*

Fonte: *Visual Studio Community 2017*

Com relação à já mencionada linha de texto “Idioma”: optou-se pela tradução *Language* para “idioma”, já que *idiom* em língua inglesa se refere, usualmente, a uma palavra ou grupo de palavras cujos significados não podem ser aferidos a partir de seus constituintes, como no caso do exemplo dado pelo <https://www.thefreedictionary.com/idiom> “*It was raining cats and dogs*”, que demanda uma tradução que evidencia características peculiares de um determinada língua. Como em ambas as linhas de texto a função do comando é que o usuário escolha a língua de interface da aplicação, optou-se por *Language* e *Choose Language*, respectivamente.

Figura 17: TAV UFU Comment Editor menu localized into English



Fonte: *Visual Studio Community 2017*

Bathia, Tomar e Sharma (2013) colocam que a localização de *software* é a tradução e adaptação de um *software* ou *web*, incluindo o *software* em si e toda a sua documentação para outras línguas. A tradução tradicional é uma atividade tipicamente feita depois que o documento-fonte for finalizado. Os projetos de localização de *software*, por outro lado, ocorrem em paralelo com o desenvolvimento do produto-fonte para permitir o carregamento simultâneo para todas as versões linguísticas. Por exemplo, a tradução de linhas de *software* pode orientar o processo, enquanto o produto está ainda na fase beta.

Para os autores:

Comparada à tradução tradicional, gerenciar projetos de localização de *software* ou *website* pode ser muito complexo. Primeiro, os projetos de localização contêm um grande número de componentes, tais como *software*, amostras de arquivos, ajuda online, documentação online e impressa; materiais extras como caixas de produtos e rótulos, e demos multimídia. Na maioria dos casos, a tradução começa antes que o material-fonte esteja em sua versão final, então na maioria dos projetos os arquivos-fonte são atualizados várias vezes durante a tradução. (BATHIA, TOMAR E SHARMA, 2013, p.46, tradução da autora).⁵

⁵ “Compared to traditional translation projects, managing software or web localization projects can be very complex. First of all, localization projects contain a large number of components, such as software, sample files, online help, online and printed documentation, collateral materials such as product boxes and disk labels, and

Segundo Massion (2011), Kanwal e Agarwal (2012), e Avni (2012) para se localizar qualquer *software* para outras línguas é crucial que ele esteja pronto para a tradução. Assim, ele deve ser preparado a partir do processo inicial de internacionalização do produto, que diz respeito ao seu processo de planejamento e organização física. Isso faz com que a localização seja mais fácil, rápida, auxiliando o alcance de maior a qualidade.

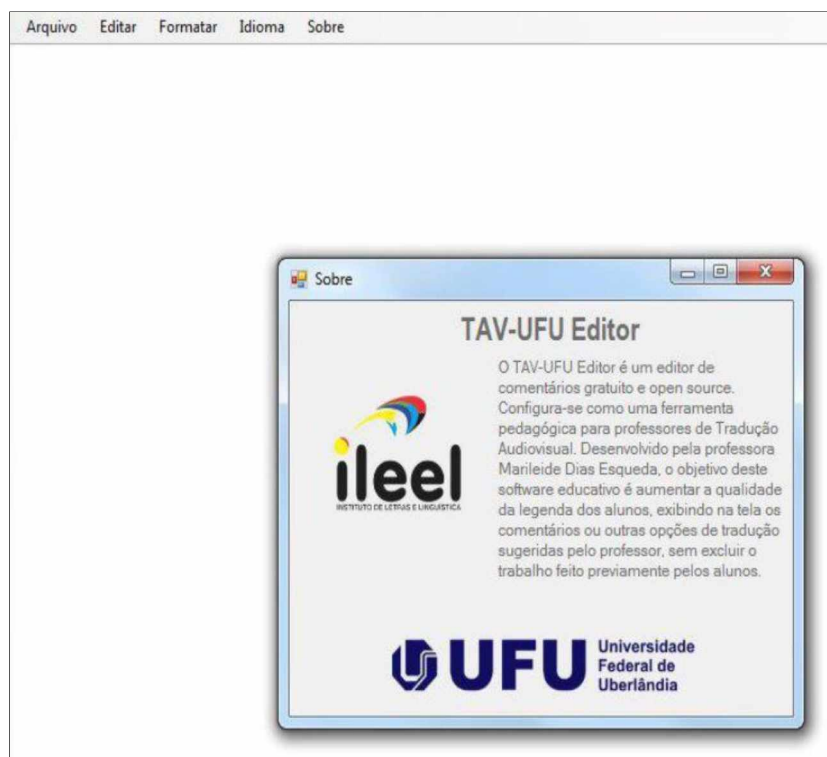
Os autores, de modo geral, afirmam que os desenvolvedores trabalham por meses ou até anos para fazer um *software* perfeito em uma língua, e a maioria dos elementos de um *software* – interface com usuário (IU), ajuda online (AO), documentação etc. – serão impactados pela localização. Se o produto já foi internacionalizado, uma revisão final do *design IU* será implementada, evitando a propagação de erros em cada língua, reduzindo erros, prevenindo atrasos e diminuindo custos, tendo sido esse o caso do *software* aqui descrito.

A linguagem na qual foi desenvolvido o *TAV UFU* já previa, desde sua origem, um processo de localização. Assim, durante o processo de localização para o inglês, foi apenas adicionada uma aba “Sobre” em seu formato inicial, *i.e.*, uma breve explicação sobre o *software*⁶, que nada interferiu em seu *design* geral.

multimedia demos. In most cases translation starts before the source material is final, so in most localization projects the source files are updated several times during translation”.

⁶ Agradecemos a indispensável ajuda no aluno-estagiário, Bruno de Oliveira, da FACOM UFU, Faculdade de Computação da UFU, que, como membro do Programa Bolsa de Graduação 2017-2018, e vinculado aos projetos acadêmico-científicos da Profa. Marileide Esqueda, auxiliou na manipulação de ambos os *softwares Visual Studio* e Editor de Comentários e seu processo de localização.

Figura 18: Aba “Sobre”



Fonte: TAV UFU

Quadro 2: Texto da aba “Sobre”

TO: PT-BR	TT: EN-US
TAV-UFU Editor é um editor de comentários gratuito e <i>open source</i>. Configura-se como uma ferramenta pedagógica para professores de Tradução Audiovisual. Desenvolvido pela professora Marileide Dias Esqueda, o objetivo deste <i>software</i> educativo é aumentar a qualidade da legenda dos alunos, exibindo na tela os comentários ou outras opções de tradução sugeridas pelo professor, sem excluir o trabalho feito previamente pelos alunos.	TAV-UFU Editor is a free and open source comment editor. It is configured as a pedagogical tool for Audiovisual Translation trainers. Developed by Professor Marileide Dias Esqueda, the purpose of this educational software is to increase the quality of the students' subtitles by displaying on the screen comments or other translation options suggested by the trainer, without excluding the translation previously done by the students.

Fonte: a autora

Ressalta-se que o título do *software* não foi alterado, buscando-se manter sua referência local. É neste texto da aba “Sobre” / *About*, que o usuário é levado a saber que

TAV ou “Tradução Audiovisual”, corresponde à *Audiovisual Translation*, e que se trata de uma aplicação destinada aos professores de tradução. No caso da tradução de “professores” para *trainers*, registra-se o fato de que, independentemente dos vários títulos atribuídos à carreira de um professor, a acepção *trainer* pareceu englobar as mais categorias de professores.

Figura 19: Aba *About*



Fonte: *TAV UFU Comment Editor*

3.3 – Publicação e compartilhamento em rede do editor de comentários *TAV UFU 1.0/ COMMENT EDITOR TAV UFU 1.0*

Souphavanh e Karoonboonyanan (2005) explicam os diversos tipos de *software*, com ênfase aos *softwares* livres, cabendo aqui portanto uma distinção entre *software* livre e *software open source*.

Os autores afirmam que um *software* livre significa um *software* que é disponibilizado gratuitamente ao usuário, respeitando-se sua liberdade de acesso, execução e distribuição de seus serviços. No caso dos *softwares open source*, característica do *software TAV UFU*, não só os usuários são livres para acessar os programas mas também para estudá-los e alterar seu código-fonte, sendo esse tipo de *software* desenvolvido de forma legível e inteligível por quaisquer programadores. Assim, o *software* foi publicado e disponibilizado no repositório *Github*⁷ para livre acesso.

⁷ Rede de compartilhamento de produtos *open source*, na qual é possível baixar vários *softwares*. O GitHub, em termos gerais, é um repositório na internet que distribui o código fonte de diversos desenvolvedores e aspirantes a desenvolvedores. Ligado à iniciativa *open source*, estima-se que 8,2 milhões de desenvolvedores compartilham seus projetos nesta rede, formando mais de 19 milhões de repositórios. (<https://opensource.com/life/15/2/beginners-guide-github>)

Figura 20: *Github*

The screenshot shows the GitHub interface for the repository 'Marileide / TAV-UFU'. At the top, there's a navigation bar with 'This repository', 'Search', 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below this, the repository name 'Marileide / TAV-UFU' is displayed with 'Watch', 'Star', and 'Fork' buttons. The 'Code' tab is selected, showing a list of files and folders. The files list includes folders like '.vs', 'PROJETO', 'hooks', 'info', and various files like 'Criando localizacao.txt', 'HEAD', 'Manual de Uso v1.0.pdf', 'MultiLanguage.zip', 'PROJETO.sln', 'PROJETO.v11.suo', 'PROJETO.v12.suo', 'Projeto Apresentado.zip', 'README.md', 'Relatório REVISAR.doc', 'TAV UFU.exe - Atalho.lnk', 'Thumbs.db', 'config', and 'description'. Each file entry shows the commit message and the time since the last commit. Below the files list, the 'README.md' file is expanded, showing the project description in Portuguese and English.

Marileide / TAV-UFU

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided. Edit

Add topics

3 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

Marileide Enviando arquivos Latest commit d983ce3 7 minutes ago

File/Folder	Commit Message	Time
.vs	Enviando arquivos	7 minutes ago
PROJETO	Enviando arquivos	7 minutes ago
hooks	Enviando arquivos	7 minutes ago
info	Enviando arquivos	7 minutes ago
Criando localizacao.txt	Enviando arquivos	7 minutes ago
HEAD	Enviando arquivos	7 minutes ago
Manual de Uso v1.0.pdf	Enviando arquivos	7 minutes ago
MultiLanguage.zip	Enviando arquivos	7 minutes ago
PROJETO.sln	Enviando arquivos	7 minutes ago
PROJETO.v11.suo	Enviando arquivos	7 minutes ago
PROJETO.v12.suo	Enviando arquivos	7 minutes ago
Projeto Apresentado.zip	Enviando arquivos	7 minutes ago
README.md	Update README.md	11 minutes ago
Relatório REVISAR.doc	Enviando arquivos	7 minutes ago
TAV UFU.exe - Atalho.lnk	Enviando arquivos	7 minutes ago
Thumbs.db	Enviando arquivos	7 minutes ago
config	Enviando arquivos	7 minutes ago
description	Enviando arquivos	7 minutes ago

README.md

TAVUFU

O TAV-UFU Editor é um editor de comentários gratuito e open source. Configura-se como uma ferramenta pedagógica para professores de Tradução Audiovisual. Desenvolvido pela professora Marileide Dias Esqueda, o objetivo deste software educativo é aumentar a qualidade da legenda dos alunos, exibindo na tela os comentários ou outras opções de tradução sugeridas pelo professor, sem excluir o trabalho feito previamente pelos alunos.

TAV-UFU Editor is a free and open source comment editor. It is configured as a pedagogical tool for Audiovisual Translation trainers. Developed by Professor Marileide Dias Esqueda, the purpose of this educational software is to increase the quality of the students' subtitles by displaying on the screen comments or other translation options suggested by the trainer, without excluding the work previously done by the students.

Projeto desenvolvido inicialmente pelo bolsista Diego Lucas da Silva do curso de Ciências da Computação da UFU, juntamente com a Professora pesquisadora Marileide Dias Esqueda do curso de Bacharelado em Tradução da UFU (2014/2016).

Projeto localizado para a língua inglesa pelos bolsistas Bruno Alberto de Oliveira do curso de Sistemas de informação da UFU e Carolina Flávia de Henrique do curso de Bacharelado em Tradução da UFU (2017/2018).

Fonte: *Github*

Figura 21: *Github*

The screenshot displays the GitHub interface for the repository 'Marileide / TAV-UFU'. At the top, there are navigation links for 'This repository', 'Search', 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below these, the repository name 'Marileide / TAV-UFU' is shown with 'Watch', 'Star', and 'Fork' buttons. The 'Code' tab is selected, showing repository statistics: 3 commits, 1 branch, 0 releases, and 1 contributor. A progress bar indicates the commit history. Below the statistics, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The file list shows various files and folders, including '.vs', 'PROJETO', 'hooks', 'info', 'Criando localizacao.txt', 'HEAD', 'Manual de Uso v1.0.pdf', 'MultiLanguage.zip', 'PROJETO.sln', 'PROJETO.v11.suc', 'PROJETO.v12.suc', 'Projeto Apresentado.zip', 'README.md', 'Relatório REVISAR.doc', 'TAV UFU.exe - Atalho.lnk', 'Thumbs.db', 'config', and 'description'. The 'README.md' file is selected, showing its content. The README text describes the TAV-UFU Editor as a free and open source comment editor, developed by Professor Marileide Dias Esqueda, aimed at increasing the quality of students' subtitles by displaying comments or other translation options suggested by the trainer, without excluding the work previously done by the students. It also mentions the project was developed initially by Diego Lucas da Silva and later localized for English by Bruno Alberto de Oliveira and Carolina Flávia de Henrique.

TAVUFU

O TAV-UFU Editor é um editor de comentários gratuito e open source. Configura-se como uma ferramenta pedagógica para professores de Tradução Audiovisual. Desenvolvido pela professora Marileide Dias Esqueda, o objetivo deste software educativo é aumentar a qualidade da legenda dos alunos, exibindo na tela os comentários ou outras opções de tradução sugeridas pelo professor, sem excluir o trabalho feito previamente pelos alunos.

TAV-UFU Editor is a free and open source comment editor. It is configured as a pedagogical tool for Audiovisual Translation trainers. Developed by Professor Marileide Dias Esqueda, the purpose of this educational software is to increase the quality of the students' subtitles by displaying on the screen comments or other translation options suggested by the trainer, without excluding the work previously done by the students.

Projeto desenvolvido inicialmente pelo bolsista Diego Lucas da Silva do curso de Ciências da Computação da UFU, juntamente com a Professora pesquisadora Marileide Dias Esqueda do curso de Bacharelado em Tradução da UFU (2014/2016).

Projeto localizado para a língua inglesa pelos bolsistas Bruno Alberto de Oliveira do curso de Sistemas de informação da UFU e Carolina Flávia de Henrique do curso de Bacharelado em Tradução da UFU (2017/2018).

Fonte: Github

PALAVRAS FINAIS

A necessidade de localização nos dias de hoje é imperativa, trazendo inúmeros benefícios para as universidades, para o comércio, para a indústria e para o público em geral. No entanto, confirmou-se que, embora necessária e essencial, a tradução e localização de um *software* educativo é uma tarefa complexa. A ideia de tornar possível uma versão em língua inglesa do *software* educativo *TAV UFU Editor de Comentários* revelou a preocupação que todo tradutor-localizador tem que ter em relação à língua, às tecnologias, ao conteúdo do *software* em si e a seu público-alvo, neste caso, professores de tradução audiovisual.

Como dito anteriormente, este trabalho objetivou explorar a temática tradução e localização de um *software* para o inglês. Foram extraídas as linhas de texto do *software* por meio do *software Visual Studio Community* e da ferramenta *Multilanguage*. A implantação e compilação do *TAV UFU* com sua versão em inglês também se deu por meio do *Visual Studio*, tornando possível sua publicação no *Github* para fins de compartilhamento.

Do ponto de vista dos comentários tecidos, procedeu-se não apenas ao relato do manuseio das ferramentas envolvidas no processo de localização, mas também da tradução para o inglês das linhas de texto do *software*.

No intuito de obter uma tradução satisfatória, foram consultados outros *software* com semelhantes funções, para que as linhas de texto não provocassem quaisquer estranhamentos ao usuário da atualidade, que possivelmente já deve ter tido a experiência de utilizar outros *softwares* utilitários.

Ao longo do trabalho, recorreu-se à pesquisa em dicionários, dicionários bilíngues e *on-line*, para assegurar os resultados referentes à localização. Também recorreu-se à tradução automática, viabilizada pela própria ferramenta *Multilanguage*, para tradução das linhas de texto. Pode-se dizer que os tradutores automáticos têm sido usados diariamente por milhões de pessoas de forma gratuita e instantânea. Esse tipo de ajuda pode ser útil para traduzir palavras isoladas ou frases curtas, mas eles não são totalmente eficazes quando o assunto é a adaptação linguístico-cultural para fins de localização de um *software* educativo, como no caso de a ferramenta *Multilanguage*, ao recorrer à tradução automática da linha “inferior”, que se refere à posição do comentário que o professor de tradução audiovisual pode inserir na parte inferior da tela, mostrou a opção de tradução literal dessa linha como *inferior*. No entanto, quando se pensa na parte inferior da tela do computador, a opção *bottom of the screen* parece ser mais recorrente.

Em se tratando da experiência de tradução da proponente desta monografia, relata-se que, para a manipulação do conteúdo linguístico do *TAV UFU* foi feito uso de um conjunto de ferramentas tecnológicas que se mostraram adequadas para a localização, permitindo à aluna-tradutora elaborar o seu trabalho baseado no modo *WYSIWYG* (*What You See Is What You Get*). Com essa funcionalidade, o tradutor passa a ter uma visualização instantânea das linhas originais e traduzíveis, tendo mais condições de adaptar o tamanho das linhas de texto dos menus e das caixas de diálogo.

Vale a pena ainda ressaltar que a tradução do texto de Keiran Dunne (2015) também é parte integrante deste trabalho (Apêndice A), e não apenas contribuiu para o enriquecimento dos aspectos teóricos deste trabalho, mas também para a constante busca do aperfeiçoamento do exercício tradutório.

Para finalizar, este trabalho foi de grande enriquecimento para a proponente desta monografia, que pretende atuar na área de localização.

REFERÊNCIAS

- BHATIA, M.; TOMAR, V.; SHARMA, A. A survey of Software Localization work. **Journal of Global Research in Computer Science**, v. 4, n. 8, 2013.
- COLLINS, R. W. Software localization: Issues and methods. **ECIS 2001 Proceedings**, p. 4, 2001.
- CYR, D.; LEW, R. Emerging challenges in the software localization industry. *Thunderbird International Business Review*, v. 45, n. 3, p. 337-358, 2003.
- DUNNE, K. J. **Localization**. The Routledge encyclopedia of translation technology, p. 550-562, 2015.
- KANWAL, M.; AGARWAL, A. Adobe Flash localization. *Multilingual*, v. 23, n. 7, p. 56, 2012.
- KENISTON, K. **Software localization: Notes on technology and culture**. Program in Science, Technology, and Society, Massachusetts Institute of Technology, 1997.
- LOMMEL, A. The economic opportunity for software localization in Africa, the Middle East and Central Asia. **Multilingual**. v. 28, n. 171, p. 36-39, 2017.
- MASSION, F. Localization of machine software. **Multilingual**, v. 22, n. 3, p. 40, 2011.
- NISHIRO, R. Learning localization in context. **Multilingual**. v. 24, n. 140, p. 36-39, 2013.
- PRUDÊNCIO, A. C. ; VALOIS, D. A. ; DE LUCCA, J. E.. Introdução à internacionalização e à localização de softwares. **Cadernos de Tradução**, v. 2, n. 14, p. 211-242, 2004.
- SANDRINI, P. Website localization and translation. In: **MuTra: Challenges of Multidimensional Translation: Conference Proceedings**. 2005. p. 131-138.
- SOUPHAVANH, A. ; KAROONBOONYANAN, T. **Free/open source software-localization**. United Nations Development Programme-Asia Pacific Development Information Programme, 2005.
- WILLIAMS, J. ; CHESTERMAN, A.. **The map: a beginner's guide to doing research in translation studies**. Routledge, 2014.
- ZAVAGLIA, A. ; RENARD, C. M. C. ; JANCZUR, C. A tradução comentada em contexto acadêmico: reflexões iniciais e exemplos de um gênero textual em construção. **Aletria: Revista de Estudos de Literatura**, v. 25, n. 2, p. 331-352.

APÊNDICE A - LOCALIZATION, Keiran Dunne (2015)

LOCALIZATION

Keiran J. Dunne

Localization

Keiran J. Dunne

DUNNE, Keiran J. Localization. IN: SIN-WAI, Chan. **The Routledge encyclopedia of translation technology**, p. 550-562, 2015.

Localization is an umbrella term that refers to the processes whereby digital content and products developed in one locale are adapted for sale and use in one or more other locales. Although the term ‘localization’ has been in use since the early 1980s, confusion persists as to what exactly it means. To understand localization, it is necessary to consider when, why and how it arose, the ways it has changed over time, and its relationship to translation and internationalization. Thus, this chapter will examine localization and its evolution from the 1980s to present.

The practice of translation remained relatively unchanged from the dawn of writing until the commoditization of the PC and the advent of mass market software ushered in the digital revolution in the 1980s. As increasing numbers of computers appeared in homes and business offices, ‘typical users were no longer professional computer programmers, software engineers or hardware engineers’ (Uren, Howard and Perinotti 1993: ix). U.S.-based software companies quickly realized that by developing products such as spreadsheet programs and word processors that average people could use for work or leisure, they could sell to a vastly larger potential market. Targeting average users instead of computer professionals was not without challenges, however.

Localização é um hiperônimo que se refere ao processo através do qual conteúdos e produtos digitais desenvolvidos em uma determinada localidade são adaptados para uso e comercialização em outras localidades. Apesar de o termo "localização" ser utilizado desde o começo da década de 80, sua definição ainda não está consagrada. Para entender o que é localização, é necessário considerar quando, como e por que surgiu, o que mudou ao longo dos anos, e sua relação com a tradução e internacionalização. Desse modo, este capítulo examinará a localização e sua evolução a partir dos anos de 1980 até o presente.

A prática da tradução manteve-se relativamente inalterada desde o surgimento da escrita até a comoditização dos computadores pessoais, sendo que o advento da comercialização em massa de *softwares* inaugurou a revolução digital nos anos de 1980. Com o aumento do número de computadores em residências e em ambientes de trabalho, “os usuários comuns não eram mais programadores profissionais, engenheiros de *software* e *hardware*” (UREN, HOWARD e PERINOTTI, 1993, p. ix). As empresas de *software* dos Estados Unidos perceberam de imediato que, desenvolvendo produtos tais como programas de planilhas e processadores de texto, pessoas comuns poderiam usá-los em contexto doméstico ou para lazer, e, dessa forma, a comercialização poderia alcançar um mercado ainda maior. Almejar o mercado de usuários comuns, no entanto, ao invés de profissionais, trouxe desafios.

While experienced professionals had become adept in detecting bugs and working around them, the new users expected, indeed demanded, that the software they bought operate exactly as described in the manuals. Benign acceptance of anomalies in the operation of software could no longer be tolerated.

(Uren, Howard and Perinotti 1993: x)

Initial efforts by software publishers to develop this embryonic mass market thus focused on improving software reliability and user-friendliness.

U.S.-based software companies soon broadened the scope of their marketing efforts beyond the domestic market to target international users. Expansion into international markets required that software publishers offer products in languages other than English. 'For a software product to have wide market acceptance in a non-English-speaking environment, it was essential to convert the software so that users saw a product in their own language and firmly based in their own culture' (Uren, Howard and Perinotti 1993: x). Software publishers thought that adapting their products for international markets was merely a matter of 'translating' software. As a result, initial attempts to adapt software for international users were characterized as 'translation on the computer for the computer' (van der Meer 1995). However, it soon became clear to practitioners that this work was 'related to, but different from and more involved than, translation' (Lieu 1997). Indeed, the scope of the undertaking was not confined to translation of text in the user interface but rather encompassed all target market requirements for culturally dependent representation of data, including but not limited to the following:

Enquanto os profissionais mais experientes já estavam aptos a detectar falhas e solucioná-las, os novos usuários esperavam, e até mesmo exigiam, que o *software* que eles adquiriam operasse exatamente como descrito no manual. Já não se tolerava mais o mau funcionamento de um *software*.

(UREN, HOWARD e PERINOTTI, 1993, p. x).

Dessa forma, os desenvolvedores de *software* se concentraram no aprimoramento da confiabilidade e usabilidade para desenvolver esse amplo mercado embrionário.

As empresas de *software* dos Estados Unidos logo ampliaram o escopo de suas estratégias de marketing para além do mercado doméstico, focando, assim, em usuários internacionais. Ao se expandirem para mercados internacionais, fez-se necessário que os produtos fossem disponibilizados em outras línguas, além do inglês. "Para que um *software* pudesse ter grande aceitação de mercado em uma localidade que não falava inglês, era necessário convertê-lo para que este pudesse ser visto por usuários em sua própria língua e com sólida inserção em sua própria cultura." (UREN, HOWARD e PERINOTTI, 1993, p. x). As empresas consideravam que a adaptação de seus produtos para mercados internacionais era só uma questão de "tradução". Consequentemente, as primeiras tentativas de adaptar um *software* para usuários internacionais foram descritas como 'uma tradução no computador para o computador' (VAN DER MEER, 1995). No entanto, logo ficou claro aos profissionais que esse trabalho estava 'relacionado a, mas diferente de e mais complexo que, tradução' (LIEU, 1997). Essa empreitada, portanto, não estava limitada apenas à tradução do texto presente na interface com o usuário, mas sim, envolvia as exigências do mercado-alvo com relação à representação cultural dos dados que incluíam, e que não estavam limitados a:

- character sets, scripts and glyphs for the representation of various writing systems
- encodings to enable the storage, retrieval and manipulation of multilingual data
- text comparison, searching and sorting (collation)
- line and word breaking
- calendars (e.g., Buddhist, Coptic, Gregorian, Hebrew lunar, Hijri, Japanese Emperor Year, Julian, Year of the Republic of China and Tangun Era calendars)
- date formats (MM/DD/YYYY, DD/MM/YYYY, YYYY-MM-DD, etc.; for example, 5/11/2014 would be read as May 11, 2014 in the United States but as November 5, 2014 in Italy)
- time formats (12-hour vs. 24-hour clock; use of AM and PM)
- number formats, digit groupings and decimal separators (period vs. comma)
- paper sizes (A3, A4, legal, letter)
- units of measurement (metric vs. imperial)
- conjuntos de caracteres, *scripts* e glifos para a representação de vários sistemas de escrita;
- codificação que permite o armazenamento, recuperação e manipulação de dados multilíngues;
- comparação de texto, busca e classificação (agrupamento);
- quebra de linhas e palavras;
- calendários, dentre eles Budista, Alexandrino, Gregoriano, Hebraico, Islâmico, Japonês, Juliano, Chinês e Juche;
- formatos de data, que podem ser mês/dia/ano, dia/mês/ano, ano/mês/dia etc. Por exemplo, nos Estados Unidos 5/11/2014 lê-se maio 11, 2014, mas, na Itália, lê-se novembro 5, 2014;
- formatos de hora, que podem ser o sistema horário de 12 horas e o sistema horário de 24 horas;
- formato de números, agrupamentos de dígitos e separadores decimais (ponto ou vírgula);
- tamanhos de papel (A3, A4, Ofício e Carta);
- unidades de medida (métrico e imperial).

In software engineering, these local market requirements are referred to using the hypernym 'locale'. Locales are expressed as language-country pairs. Thus, 'French-Canada is one locale, while French-France is another' (Cadieux and Esselink 2002). The need to account not only for translation but also for 'locale' explains why and how the process of adapting software for international markets came to be known as 'localization' in the early 1980s. The scope and scale of this new activity expanded so rapidly that within less than a decade localization was perceived as an industry unto itself, as reflected by the creation of the Localization Industry Standards Association in 1990 (Lommel 2007: 7).

Na engenharia de software, essas exigências do mercado local são denominadas pelo hiperônimo localidade. Localidades são designadas como pares de língua-país. Desse modo, 'Francês-Canadá é uma localidade, enquanto Francês-França é outra' (CADIEUX, ESSELINK, 2002). A necessidade de levar em consideração não só a tradução, mas também a localidade, explica como e por que o processo de adaptação de *software* para mercados internacionais começou a ser conhecido como "localização" no início da década de 1980. A magnitude dessa nova atividade expandiu-se tão rápido que, em menos de uma década, a localização se destacou como uma indústria própria, tal como demonstrado pela criação da *Localization Industry Standards Association*, em 1990 (LOMMEL, 2007, p. 7).

The costs of adapting software products for

Os custos de adaptação de *softwares* para

other locales seemed like a small price to pay given the sizable international markets and potential revenues to which localized products could enable access. Software publishers approached localization in different ways: some performed the work using in-house teams and some outsourced the work to specialized service providers, whereas others assigned responsibility for localization to in-country subsidiaries or distributors (Esselink 2003b: 4). Despite the ostensible differences between these approaches, they all shared one fundamental characteristic: in each case localization was performed apart from, and subsequent to, the development of the original, domestic-market products. 'This separation of development and localization proved troublesome in many respects,' observes Esselink (2003b: 4).

outras localidades pareciam reduzidos se comparados aos grandes mercados internacionais e os possíveis lucros que os produtos localizados poderiam alcançar. As empresas de *software* abordaram a localização de diversas formas: algumas optaram por utilizar equipes internas, algumas terceirizavam o processo para provedores especializados e, em alguns casos, a responsabilidade da localização foi atribuída para subsidiárias ou distribuidores no país. (ESSELINK, 2003b, p. 4). Apesar das diferenças entre tais abordagens, todas elas partilhavam a seguinte característica fundamental: em todos os casos, a localização era realizada à parte, e subsequente ao desenvolvimento do produto original. "Essa separação de desenvolvimento e localização mostrou-se problemática em diversos aspectos", conforme observa Esselink (2003b, p. 4).

First, the software provided to localization teams often could not be localized because it lacked certain fundamental capabilities, such as the ability to display target-language scripts and writing systems. In such cases, the localization teams had to send the software back to the development teams for implementation of the necessary capabilities, such as support for the display of Asian languages or of right-to-left scripts for languages such as Arabic and Hebrew. Second, translatable text was typically embedded in the software source code. Identifying and locating translatable text was very difficult for localization teams that had not participated in the development of the software (see Figure 33.1). Finally, and perhaps most critically, localization required that changes be made directly to the source code of the software. And how working directly in source code caused problems, it is important to note that software source code is the raw material from which the running copy of a program is created. In other words, source code must be compiled, or built, into a machine-readable (binary) executable file, which in turn must be tested (and debugged, if any bugs are found) before the software can

Primeiramente, o *software* fornecido às equipes de localização muitas vezes não poderia ser localizado, devido à falta de determinados recursos fundamentais, tais como a capacidade de exibição de scripts da língua-alvo e os sistemas de escrita. Nesses casos, as equipes de localização devolviam o *software* aos seus desenvolvedores para que fossem implementados os recursos necessários, como suporte para a exibição de idiomas asiáticos ou scripts da direita para a esquerda para idiomas como o árabe e o hebraico. Em segundo lugar, os componentes linguísticos a serem traduzidos normalmente encontravam-se incorporados ao código-fonte do *software*. Identificar e localizá-los era um trabalho penoso para as equipes, que não haviam participado do desenvolvimento do *software*. (ver Figura 33.1). Por fim, e talvez o ponto mais crítico a ser observado é que a localização exigia mudanças diretas no código-fonte do *software*. Como trabalhar diretamente no código-fonte causava problemas, passou a ser importante saber que este é a matéria-prima a partir da qual a cópia em execução do programa é criada. Em outras palavras, o código-fonte deve ser compilado ou

be released for use (see Figure 33.2).

construído em um arquivo executável legível por máquina (binário), que por sua vez deve ser testado (e corrigido, caso algum erro seja encontrado) antes de o *software* ser liberado para uso (ver Figura 33.2).

```
IDD_PEN_WIDTHS_DIALOG 0, 0, 203, 65
STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION |
WS_SYSMENU
CAPTION "Pen widths"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON "OK", IDOK, 148, 7, 50, 14
    PUSHBUTTON "Cancel", IDCANCEL, 148, 24, 50, 14
    PUSHBUTTON "Default", IDC_DEFAULT_PEN_WIDTHS, 148, 41, 50, 14
    LTEXT "Thin Pen width:", IDC_STATIC, 10, 12, 70, 10
    LTEXT "Thick Pen width:", IDC_STATIC, 10, 33, 70, 10
    EDITTEXT IDC_THIN_PEN_WIDTH, 86, 12, 40, 13, ES_AUTOHSCROLL
    EDITTEXT IDC_THICK_PEN_WIDTH, 86, 33, 40, 13, ES_AUTOHSCROLL
END
```

Figure 33.1 - Source-code representation of the dialog box shown in Figure 33.5(a)

Figura 33.1 – Representação do código-fonte da caixa de diálogo exibida na Figura 33.5 (a)

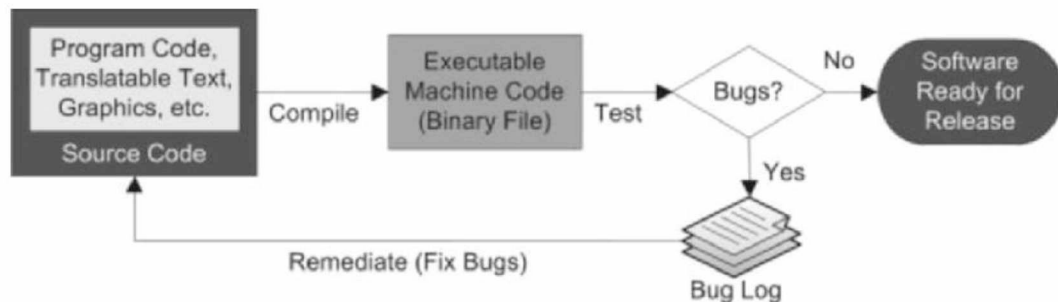


Figure 33.2 - Because source code must be compiled and tested anew whenever it is modified, localizing directly in source code is labor-intensive

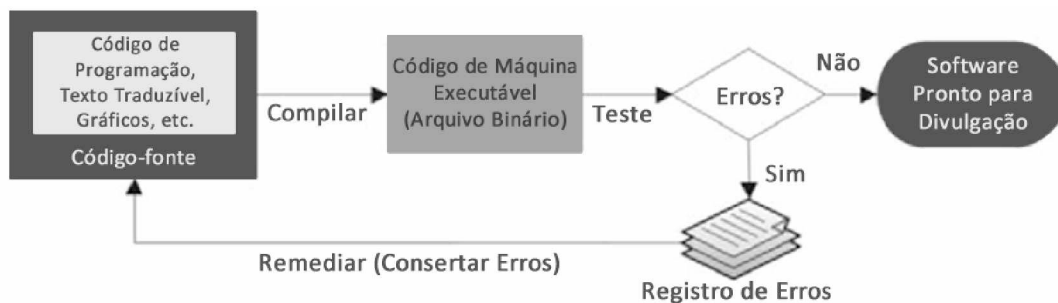


Figura 33.2 – Devido ao fato de que o código-fonte precisa ser compilado e testado sempre que for modificado, localizar diretamente no código é um trabalho bastante árduo.

Identifying translatable text can be difficult for non-programmers. In the example shown in Figure 33.1, items in quotation marks are translatable except for the name of the default font (MS Sans Serif). Each group of four digits separated by commas represents layout coordinates.

Para aqueles que não são programadores, identificar os textos a serem traduzidos não é um trabalho fácil. No exemplo mostrado na Figura 33.1, os itens entre aspas são traduzíveis, com exceção do nome da fonte padrão (MS Sans Serif). Cada grupo de quatro dígitos separados por vírgulas representa a formatação.

Working directly in source code had profound ramifications for localization. Indeed, the adaptation of software products for other locales did not merely entail a few changes to compiled, tested, and debugged versions of programs that had already been released to the domestic market. Instead, localization of a given program required that a separate set of source code be maintained and that a different executable be compiled, tested and debugged for each target locale. Consequently, creating N localized versions of a program required that the publisher maintain $N + 1$ sets of source code: one for each target locale plus one for the domestic market. In addition, each set of source code had to be localized, compiled, tested, debugged, updated and managed separately (Luong, Lok, Taylor and Driscoll 1995: 3). For instance, a U.S.-based publisher that wanted to market a product in three international locales, such as German-Germany, French-France, and Japanese-Japan, was required to manage four different versions of source code in parallel, one set for the domestic English-United States locale plus one set for each of the three international locales (see Figure 33.3). The process of compiling and testing localized software products as distinct from source-locale versions is called localization engineering (Esselink 2002).

Creating, maintaining and supporting multiple localized versions in parallel proved to be time-consuming and expensive. Testing and debugging software was inherently labor-intensive and costly even without adding localization as a variable. Seminal work on software engineering economics by Boehm had demonstrated that 'uncorrected errors become exponentially more costly with each phase in which they are unresolved' (1981: 8). The exponential cost increase of error

Trabalhar diretamente no código-fonte ocasionou consequências sérias para a localização. Como resultado, a adaptação de *softwares* para outras localidades não implicou apenas meras mudanças nas versões já compiladas, testadas e corrigidas de programas que já haviam sido comercializados no mercado interno. Em vez disso, a localização de um dado programa exigia que um conjunto separado de código-fonte fosse mantido e que um executável diferente fosse compilado, testado e corrigido para cada localidade. Consequentemente, criar várias versões localizadas de um programa exigia que o desenvolvedor mantivesse uma versão adicional do conjunto de código-fonte: um para cada localidade e mais um para o mercado doméstico. Além disso, cada conjunto de código-fonte tinha que ser localizado, compilado, testado, corrigido, atualizado e gerenciado separadamente. (LUONG, LOK, TAYLOR e DRISCOLL, 1995: 3). Por exemplo, um desenvolvedor dos Estados Unidos que quisesse comercializar seu produto em três localidades internacionais, tais como alemão-Alemanha, francês-França e japonês-Japão, precisaria gerenciar paralelamente quatro versões diferentes do código-fonte, um conjunto para a localidade doméstica inglês-Estados Unidos, mais um conjunto para cada uma das três localidades exemplificadas (ver Figura 33.3). O processo de compilar e testar um produto localizado de forma distinta das versões de origem é chamado de engenharia de localização (ESSELINK, 2002).

Criar, gerenciar e prover manutenção de várias versões localizadas em paralelo mostrou ser um processo caro e demorado. Testar e corrigir *softwares* já era um trabalho intensivo e dispendioso sem a localização. Um trabalho inaugural sobre os custos da engenharia de *software* de Boehm mostrou que "erros que não são corrigidos tornam-se muito mais caros a cada fase em que eles não são solucionados." (1981, p. 8). Os valores relacionados à correção de erros

correction was exacerbated by the post hoc approach to localization in which the adaptation of software for other locales – and thus the discovery of localization bugs – did not begin until the development of the domestic-market versions had been completed. This cost multiplier problem was compounded by the management of a distinct set of source codes for each target locale, since a bug discovered in one set of code might need to be fixed in all other sets. Indeed, localization engineering has traditionally involved ‘quite a bit of bug fixing,’ as Esselink observes (2002: 4). Not surprisingly, complexity soon established itself as a hallmark of localization (Esselink 2000b). Ultimately, most of the problems posed by early localization efforts stemmed from a single root cause: the failure to effectively plan for the reuse of software source code across multiple locales.²

aumentaram ainda mais devido à abordagem de localização *post hoc*, na qual a adaptação de um *software* para outras localidades – e consequentemente a detecção de erros – começava, apenas, após desenvolvidas as versões do mercado interno ser concluído. Esse problema no aumento dos custos estava relacionado ao gerenciamento de um código-fonte distinto para cada localidade-alvo, já que o problema descoberto em um dos conjuntos deveria ser corrigido em todos os outros. Esselink (2002, p. 4) aponta que a engenharia de localização envolvia tradicionalmente “correção de erros”. Era de se esperar, portanto, que a localização logo se revelasse também complexa (Esselink, 2000b). Dito isso, a maior parte dos problemas impostos pelos primeiros esforços da localização surgia de uma única causa: o planejamento ineficaz do código-fonte do *software* para as múltiplas localidades.

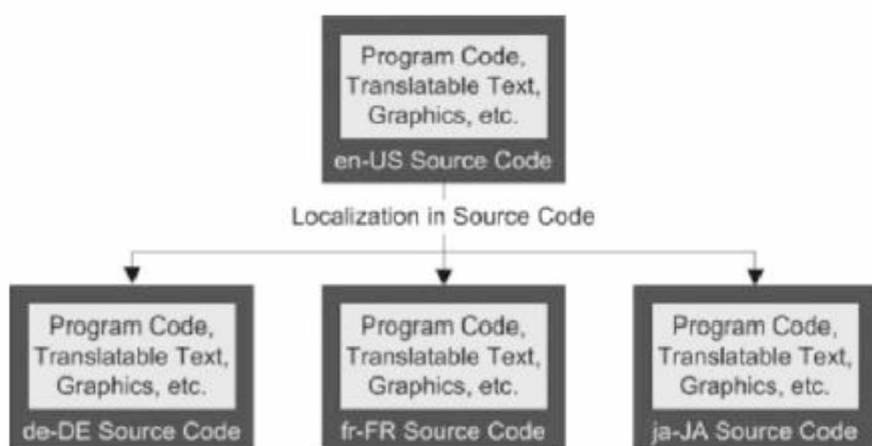


Figure 33.3 - When localization is performed in the source code, it is necessary to maintain a separate set of code for each target locale plus one set for the domestic market



Figura 33.3 – Quando a localização é realizada no código-fonte, é necessário manter um conjunto do código separado para cada localidade-alvo, mais um conjunto para o mercado doméstico.

Most software and hardware companies that made forays into international markets quickly concluded that localization and translation were not an integral part of their business. As Esselink (2000a: 5) observes, '[t]he increasing size and complexity of localization projects soon forced companies to an outsourcing model. Most software publishers simply did not have the time, knowledge or resources to manage multilingual translation or localization projects.' As a result, most companies decided that it would be more efficient to outsource the adaptation of software products for international markets to external language service providers as project work. In addition to the adaptation of the software application, a typical software localization project might also involve the translation and/or adaptation of various other components such as sample files, demos and tutorials, Help systems, printed and online user documentation, as well as marketing collateral (see Figure 33.4). The fact that these components were authored in a variety of digital formats, some of which needed to be compiled and tested prior to release, meant that localization involved a number of new forms of work in addition to traditional translation, including software and online Help engineering and testing, conversion of documentation to different formats, translation memory creation and management, as well as project management (Esselink 2000b; Esselink 2003a: 69; Dunne and Dunne 2001). Localization thus required that translators possess strong instrumental and technical skills in addition to traditional translation and domain expertise. 'Throughout the 1990s, the localization industry tried to turn translators into semi-engineers', recalls Esselink (2003b:7).

A maioria de empresas de *software* e *hardware* que fizeram investimentos em mercados internacionais concluíram que localização e tradução não eram parte integral de seus negócios. Esselink (2000^a, p.5) ressalta que "o aumento e a complexidade de projetos de localização logo forçaram as empresas a um modelo terceirizado. A maioria dos desenvolvedores não tinha tempo, conhecimento ou recursos para gerenciar traduções multilíngues ou projetos de localização. Como resultado, tais empresas decidiram que seria mais eficiente terceirizar a adaptação de seus produtos para mercados internacionais a fornecedores de serviços linguísticos externos, como parte do projeto de trabalho. Além da adaptação do *software*, um projeto comum de localização também envolvia tradução e/ou adaptação de vários outros componentes, como arquivos de amostra, demos e tutoriais, sistemas de ajuda, documentos de usuário *on-line* e impressos, além de garantia de marketing (ver Figura 33.4). O fato de que esses componentes foram criados em uma variedade de formatos digitais, alguns dos quais precisavam ser compilados e testados antes da divulgação, significava que a localização envolvia novas formas de trabalho além da tradução convencional. Nisso, estavam inclusos materiais de ajuda *on-line*, conversão de documentos para diferentes formatos, criação de memória de tradução e seu gerenciamento, e também o gerenciamento de projetos. (ESSELINK, 2000b; ESSELINK 2003^a, p. 69; DUNNE e DUNNE, 2001). A localização exigia, portanto, que tradutores possuísem fortes habilidades instrumentais e técnicas, além do conhecimento e domínio de tradução. "Ao longo da década de 1990, a indústria da localização buscou tornar tradutores em semi-engenheiros", relembra Esselink (2003b, p.7).



Figure 33.4 - The scope of a traditional software localization project may encompass a number of components in addition to the software application itself

Source: Adapted from Esselink (2000a:10)



Figura 33.4 – O escopo de um projeto de localização tradicional de *software* pode abranger vários componentes além do aplicativo de *software* em si. Fonte: Adaptado de Esselink (2000a, p. 10)

Outsourcing shifted the challenges of managing complex multilingual localization projects to external service providers but did not address the fundamental problem of the duplication of effort required to manage multiple sets of source code in parallel. Faced with the challenge of controlling the complexity and cost of the localization process, software publishers in the late 1980s and early 1990s began to realize that ‘certain steps could be performed in advance to make localization easier: separating translatable text strings from the executable code, for example. This was referred to as internationalization or localization-enablement’ (Cadieux and Esselink 2002). Internationalization is an engineering process that precedes localization and entails the

A terceirização transferiu as dificuldades dos gerenciamentos complexos multilíngues a fornecedores externos, mas não abordou o problema fundamental de duplicação de esforços exigidos para gerenciamento paralelo de múltiplos conjuntos de código-fonte. No fim dos anos de 1980 e começo os anos de 1990, diante do desafio de controlar a complexidade e o custo do processo de localização, desenvolvedores começaram a perceber que “dadas etapas poderiam ser executadas com antecedência para tornar a localização mais fácil: separar as linhas de texto traduzíveis do código executável, por exemplo. Isso se refere à internacionalização ou à possibilidade de localização” (CADIEUX e ESSELINK, 2002). A internacionalização é um processo de

separation of '[a]ll the culturally and linguistically sensitive software components ... from the core of the application' (Hall 1999: 298). In practice, the scope of internationalization is typically confined to the linguistic and culturally dependent contents of the user interface that may require adaptation, which are collectively designated using the hypernym 'resources.' When a piece of software is properly internationalized, '[t]here is no programming code in the [resources] nor is there any [translatable] text in the program code' (Uren, Howard and Perinotti 1993: 60). Resources in a typical desktop software application may include the following:

- Accelerators: keyboard shortcuts that enable direct execution of commands. Accelerators are typically associated with a Function key or with a combination of the Ctrl key plus a specific keyboard letter. For example, pressing the F1 Function key in a typical Windows application launches the Help, while pressing Ctrl+C executes the Copy command.
- Dialog boxes: secondary windows that allow the user to perform a command and/or that ask the user to supply additional information (see Figure 33.5(a)). Common examples include the 'Save As' and 'Print' dialog boxes. Dialog box resources also contain the coordinates that govern the layout and display of the user interface (see Figure 33.1).
- Icons: images that symbolize and provide clickable shortcuts to programs, files and devices (see Figure 33.5(b)).
- Menus: lists of options or commands that display at the top of the main program window (see Figure 33.5(c)). Secondary menus, called 'context' or 'popup' menus, display when the user clicks the right-hand mouse button.
- String tables: 'string' is short for 'string of characters,' and designates any text that is stored and manipulated as a group. Strings

engenharia que precede a localização e implica a separação de "todos os componentes de *software* sensíveis cultural e linguisticamente (...) do núcleo da aplicação" (HALL, 1999, p. 298). Na prática, o escopo da internacionalização está tipicamente restrito aos conteúdos linguísticos e culturalmente dependentes da interface com o usuário, que podem exigir adaptação, designada coletivamente por meio do hiperônimo "recursos". Quando parte de um *software* é internacionalizada corretamente, "não há nem código de programação nos recursos, nem textos traduzíveis no código do programa" (UREN, HOWARD e PERINOTTI, 1993, p. 60). Os recursos em um aplicativo de *software* de *desktop* típico podem conter:

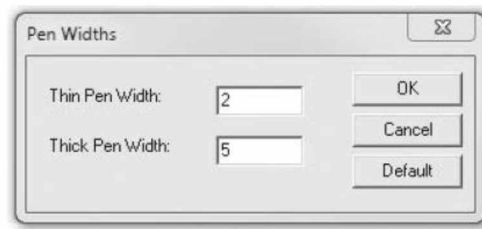
- aceleradores: atalhos de teclado que habilitam execução direta de comando. Aceleradores são tipicamente associados a uma tecla de função ou uma combinação da tecla Ctrl mais uma letra específica do teclado. Por exemplo, pressionar a tecla de função F1 em um aplicativo típico do Windows abre a Ajuda, ao passo que pressionar Ctrl+C executa o comando copiar;
- caixas de diálogo: janelas secundárias que permitem que o usuário execute um comando e/ou pedem ao usuário que forneça informações adicionais (ver Figura 33.5(a)). Exemplos comuns de caixas de diálogo incluem "Salvar Como" e "Imprimir". Os recursos da caixa de diálogo também contêm as coordenadas que configuram o *layout* e a exibição da interface com o usuário (ver Figura 33.1);
- ícones: imagens que simbolizam e fornecem atalhos clicáveis para programas, arquivos e dispositivos (ver Figura 33.5 (b)).
- menus: lista de opções ou comandos que aparecem no topo da janela principal do programa (ver Figura 33.5 (c)). Menus secundários, chamados de "menu contextual" ou "menu popup", que aparecem quando o usuário clica com o botão direito do mouse;
- *strings* (linhas de texto): *string* é a

include button captions, dialog box titles, error messages, menu items, tool tips, and so on. Menu and dialog box strings can often be visually represented in a WYSIWYG (what you see is what you get) editor during localization, whereas string tables typically cannot (see Figure 33.5(d)).

- Toolbars: raster graphics that contain toolbar button images, typically in bitmap (*.bmp) or Portable Network Graphics (*.png) format (see Figure 33.5(e)).

abreviação de "*string* de caracteres" e designa qualquer texto armazenado e manipulado como um grupo. As *strings* incluem as legendas de botões, títulos das caixas de diálogo, mensagens de erro, itens do menu, dicas de ferramenta, e assim por diante. O menu e as *strings* de caixa de diálogo podem ser representados visualmente em um editor que contém o recurso / a funcionalidade WYSIWYG (*what you see is what you get*; em português "o que você vê é o que você tem) durante a localização, ao passo que as *strings* normalmente não (ver Figura 33.5 (d));

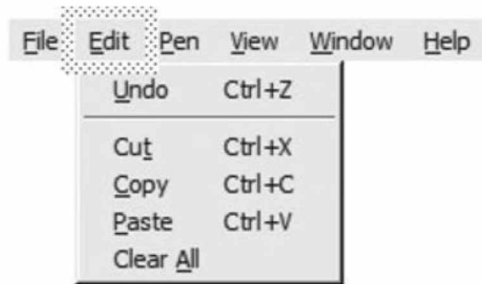
- barra de ferramentas: gráficos *raster* que contém imagens de botão da barra de ferramentas, normalmente em formato bitmap (*.bmp) ou em formato PNG - Portable Network Graphics (*.png), (ver Figura 33.5 (e)).



(a)



(b)



(c)

Value	Caption
57632	Erase the selection\nErase
57633	Clears the drawing
57634	Copy the selection and put it on the Clipboard\nCopy
57635	Cut the selection and put it on the Clipboard\nCut
57636	Find the specified text\nFind
57637	Insert Clipboard contents\nPaste
57640	Repeat the last action\nRepeat

(d)



(e)

Figure 33.5 - Typical resources in a software application include (a) one or more dialog boxes; (b) a program icon and a document icon (left and right images, respectively); (c) one or more

menus; (d) a string table; and (e) one or more toolbars.³ See also Figures 33.8 and 33.9

Figura 33.5 – Recursos típicos encontrados em um aplicativo de software incluem: (a) uma ou mais caixas de diálogo; (b) um ícone do programa e um ícone de documento (imagens a esquerda e a direita, respectivamente); (c) um ou mais *menus*; (d) uma *string*; e (e) uma ou

mais barras de ferramentas. 3 Ver também Figuras 33.8 e 33.9.

The creation of a standardized way to represent culturally dependent user interface material and store it independently from the functional program code greatly facilitated the localization process. No longer was it necessary to modify the source code or to compile, test, and debug each target version of a program separately (Luong et al. 1995: 3). Instead, the development team could simply extract the resources from the binary executable file, provide them to a localization team that would translate the text and perform all other necessary modifications and then return the target resources to the developers, who would integrate them into copies of the binary executable file to create the necessary target version or versions (see Figure 33.6). This extraction-adaptation-integration process is one of the defining characteristics of localization.

A criação de uma forma padronizada para representar o material culturalmente dependente de interface com o usuário e armazená-lo de forma independente do código do programa funcional facilitou muito o processo de localização. Não era mais necessário modificar o código-fonte ou compilar, testar e corrigir os erros de cada versão do programa separadamente. (LUONG et al., 1995, p. 3) Em vez disso, a equipe de desenvolvimento poderia simplesmente extrair os recursos do arquivo executável binário, fornecê-los a uma equipe de localização que traduziria o texto e faria qualquer outra modificação necessária e, em seguida, retornaria os recursos-alvo para os desenvolvedores, que os integrariam em cópias do arquivo executável binário para criar a versão (ou versões)-alvo necessária(s) (ver Figura 33.6). Esse processo de extração-adaptação-integração é uma das características que define a localização.

By enabling the use of a single set of source code to support multiple target locales, internationalization diminished the effort and cost associated with localization and increased the speed and accuracy with which it can be accomplished (Schmitz 2007: 51). It soon occurred to software developers that they could not only embed resources in a program's executable file and bind them directly to the application code, but also externalize the resources, store them in a dedicated file called a satellite assembly and dynamically link this external resource file to the application. To localize an application developed using this internationalization strategy, one simply creates a localized version of the resource file(s) for each target locale. For example, a publisher that created a program for the German-Germany locale and wanted to market localized versions in the United States and the People's Republic of China would create a localized version of the resource assembly for each locale (see Figure 33.7).

Ao permitir o uso de um único conjunto de código-fonte para suportar várias localidades-alvo, a internacionalização diminuiu o esforço e o custo associados à localização, e aumentou a velocidade e a precisão com a qual pôde ser realizada (SCHMITZ, 2007, p. 51). Logo ficou claro aos desenvolvedores que, além de incorporar recursos em um arquivo executável de um programa e vinculá-los diretamente ao código do aplicativo, eles também podiam externalizar os recursos, armazená-los em arquivos dedicados chamados biblioteca-satélite (*satellite assembly*) e conectar dinamicamente esse arquivo de recurso externo ao aplicativo. Para localizar um aplicativo desenvolvido usando essa estratégia de internacionalização, é necessário que se crie uma versão localizada do(s) arquivo(s) de recurso para cada localidade-alvo. Por exemplo, um editor que criou um programa para a localidade alemão-Alemanha e quisesse comercializar versões para os Estados Unidos e para a República Popular da China, deveria criar

uma versão localizada da montagem de recursos para cada localidade. (ver Figura 33.7)

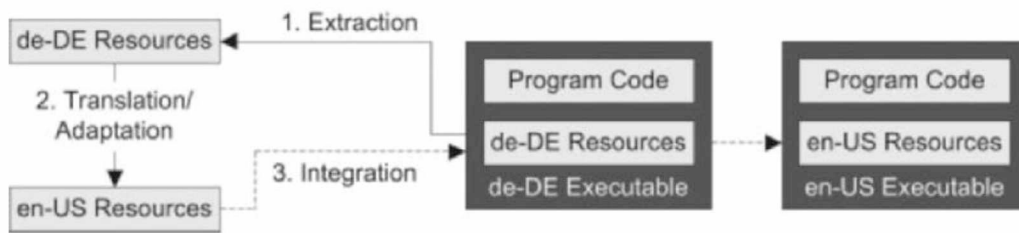


Figure 33.6 - Internationalization enables the logical separation of the culturally dependent contents of the user interface from the functional core of the program, transforming localization into a simpler process of resource replacement.

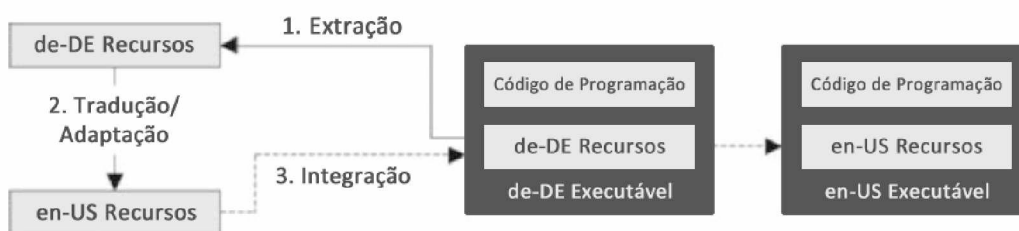


Figura 33.6 – A internacionalização permite a separação lógica de conteúdos culturalmente dependentes da interface com o usuário do núcleo funcional do programa, transformando a localização em um processo simplificado de substituição de recursos.

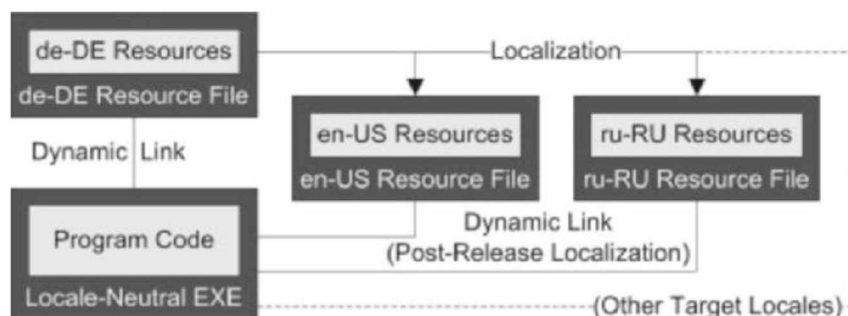
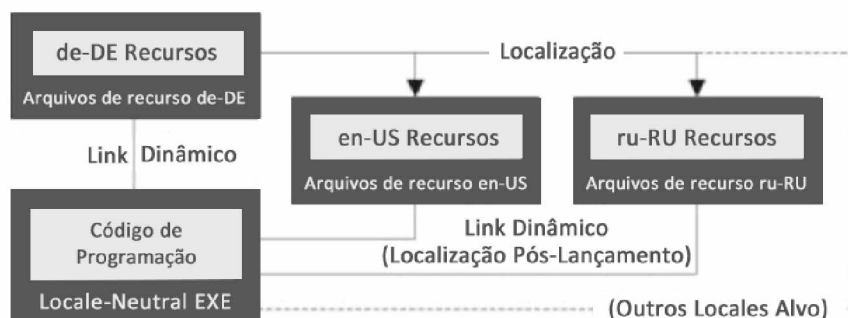


Figure 33.7 - Externalizing resources, storing them in dedicated files and linking them dynamically to a locale-neutral program core is the logical culmination of software internationalization strategies



The advent of software internationalization coincided with, and was facilitated by, the broad shift from procedural and structured programming to object-oriented programming in the 1980s and the 1990s. Procedural and structured programming languages, which predominated during the 1960s and 1970s, were adequate for small, relatively simple standalone applications. However, as applications expanded in terms of size, complexity, and the degree of interaction with other systems, procedural and structured languages began to show their limitations (Clark 2013: 2-5). The larger a program became the harder it was to maintain, and it was difficult to modify one aspect of existing functionality without negatively impacting the system as a whole. Programmers needed a comprehensive understanding of how a program worked and could not focus their efforts on discrete functions. In addition, the absence of standardized notational ways to represent and encode functions hindered the portability and re-usability of a software source code, with the result that programs were typically built from scratch.

Object-oriented programming effectively resolved these problems. In object-oriented programming, data and functions that use those data are grouped and encapsulated in logical structures named objects. One object's encapsulated data and functions can be used, or invoked, by other functions or programs. Communication between objects in a program is carried out through messages. An object's interface is defined by the messages that it can send and receive. In object-oriented programming, sending a message to an object is also called setting a property of that object. Objects are defined by classes, which determine their code, data and the messages they can send and receive (i.e., their properties). Individual objects inherit all of the properties and functions of the class to

O surgimento da internacionalização de *software* coincidiu com e foi facilitado pela ampla mudança de programação procedimental e estruturada para programação orientada a objetos nos anos 1980 e 1990. As linguagens de programação procedurais e estruturadas, que predominavam durante o período de 1960 e 1970, eram adequadas para aplicativos autônomos relativamente simples e pequenos. No entanto, conforme os aplicativos expandiam-se em tamanho, complexidade e níveis de interação com outros sistemas, as linguagens procedurais e estruturadas começaram a mostrar suas limitações. (CLARK, 2013, p. 2-5). Quanto maior era o programa, mais difícil era para mantê-lo, e era difícil para modificar um aspecto da funcionalidade existente sem impactar negativamente o sistema como um todo. Os programadores precisavam de um entendimento abrangente de como um programa funcionava e não poderiam concentrar seus esforços em funções isoladas. Além disso, a ausência de formas de notação padronizadas para representar e codificar funções dificultou a portabilidade e reuso do código-fonte do *software*, com o resultado de que programas eram, normalmente, construídos do zero.

A programação orientada a objetos resolveria esses problemas de forma efetiva. Na programação orientada a objetos, dados e funções que usam aqueles dados são agrupados e encapsulados em estruturas lógicas denominadas objetos. Os dados e funções encapsulados de objetos podem ser usados ou acessados por outras funções ou programas. A comunicação entre os objetos em um programa é realizada por meio de mensagens. A interface deste é definida pelas mensagens que se pode enviar e receber. Na programação orientada a objetos, o envio de uma mensagem para um objeto também é chamado de configuração de uma propriedade desse objeto. Objetos são definidos por classes, as quais determinam seu código, dados e as

which they belong. Inheritance enables the creation of ‘child’ objects and subclasses that inherit all of the properties and functions of the original class of the ‘parent’ object. Inheritance facilitates software maintenance, updates and debugging because a change made to one instance of an object is applied to all instances of objects in that class. Objects can also be reused in and across programs, which simplifies the development of new programs.

mensagens que podem enviar e receber (ou seja, suas propriedades). Objetos específicos herdam todas as propriedades da classe a que pertencem. A herança permite a criação de objetos e subclasses "filho", que herdam todas as propriedades e funções da classe original do objeto "pai". A herança facilita a manutenção, atualizações e correção de erros do *software*, porque uma mudança feita em uma instância de um objeto é aplicada em todas as instâncias de objetos naquela classe. Objetos também podem ser reutilizados dentro e através de programas, o que simplifica o desenvolvimento de novos programas.

Object-oriented programs are not written, but rather drawn in integrated WYSIWYG (‘what you see is what you get’) development environments using a variety of objects including menus, dialog boxes, forms, and user controls such as command buttons, check boxes, and text labels, among others. From the standpoint of object-oriented programming, internationalization standardizes the representation, definition and storage of the inventory of user interface objects as classes of resources. It follows that localization is properly understood as the modification of the properties of objects. For example, translating a command button caption entails the modification of the Caption property of the Button object (see Figure 33.8).

Os programas orientados a objeto não são escritos, mas sim desenhados em ambientes de desenvolvimento WYSIWYG integrados (*what you see is what you get*, em português “o que você vê é o que você tem”) usando a variedade de objetos incluindo menus, caixas de diálogo e controle de usuários, tais como botões de comando, caixas de checagem, linhas de texto, entre outros. A partir da programação orientada a objetos, a internacionalização padroniza a representação, definição e armazenamento do inventário dos objetos de interface do usuário como classes de recurso. Consequentemente, a localização é propriamente entendida como a modificação das propriedades dos objetos. Por exemplo, traduzir a legenda de um botão de comando requer modificação da propriedade Legenda do objeto Botão (ver Figura 33.8).

Internationalization not only eliminated the need to maintain a separate set of source code for each supported locale, but also clarified the respective roles of programmers, engineers and translators.

A internacionalização não só eliminou a necessidade de manter um conjunto de códigos-fonte para cada localidade, mas também esclareceu os respectivos papéis dos programadores, engenheiros e tradutores.

[Internationalization] allows programmers and engineers to focus on code and translators to focus on translation. It means the software with all its complex logic does not have to be touched just because you want to add another language; all you have to do is translate some files. (Uren, Howard and

A internacionalização permite que os programadores e engenheiros foquem no código e os tradutores na tradução. Isso significa que o *software* com toda a sua lógica complexa não tem que ser alterado só porque alguém quer acrescentar um novo idioma. Tudo o que se deve fazer é traduzir

Perinotti 1993:63)

Indeed, most locale-dependent aspects of data storage, retrieval, manipulation and presentation can be managed today through internationalization capabilities built into host operating systems and/or by using development frameworks and runtime environments that offer robust support for internationalization, such as Java (Deitsch and Czarnecki 2001) or Microsoft's .NET Framework (Smith-Ferrier 2007). If a program has been properly internationalized and all translatable text has been externalized from the source code, the localizer works only on resource files and cannot access or modify the program's functional code. Consequently, the nuts-and-bolts work of software localization now primarily involves the translation of strings in menus, dialog boxes and string tables. Dialog boxes and user controls such as buttons and labels may also require resizing to account for translation-related string expansion or shrinkage, depending on the source and target languages involved. Visual localization tools facilitate these tasks by enabling localizers to view resources in context and by enabling them to use translation memory and terminology databases as they work (see Figure 33.9).

alguns arquivos (UREN, HOWARD, PERINOTTI, 1993, p. 63).

De fato, a maioria dos aspectos da localidade dependentes de dados de armazenamento, recuperação, manipulação e apresentação podem ser, atualmente, gerenciados por meio de capacidades de internacionalização construídas no sistema operacional hospedeiro e/ou usando estrutura de desenvolvimento e ambientes que oferecem apoio sólido para a internacionalização, como o Java (DEITSCH e CZARNEKI, 2001) ou a estrutura do Microsoft.NET (SMITH-FERRIER, 2007). Se um programa foi internacionalizado de maneira correta e todos os textos traduzíveis foram retirados do código-fonte, o localizador trabalha somente com os arquivos de recurso e não pode acessar ou modificar o código funcional do programa. Consequentemente, os componentes básicos da localização de *software* agora envolvem a tradução das linhas do menu, caixas de diálogos e tabelas de sequência. Caixas de diálogo e controles de usuário tais como botões e legendas podem demandar redimensionamento para contabilizar a expansão ou encolhimento das linhas de tradução, dependendo das línguas de origem e destino envolvidas. As ferramentas visuais de localização facilitam essas tarefas, ao permitir que localizadores vejam os recursos em contexto e permitindo, também, que usem memória de tradução e bancos de dados terminológicos enquanto trabalham (ver Figura 33.9).

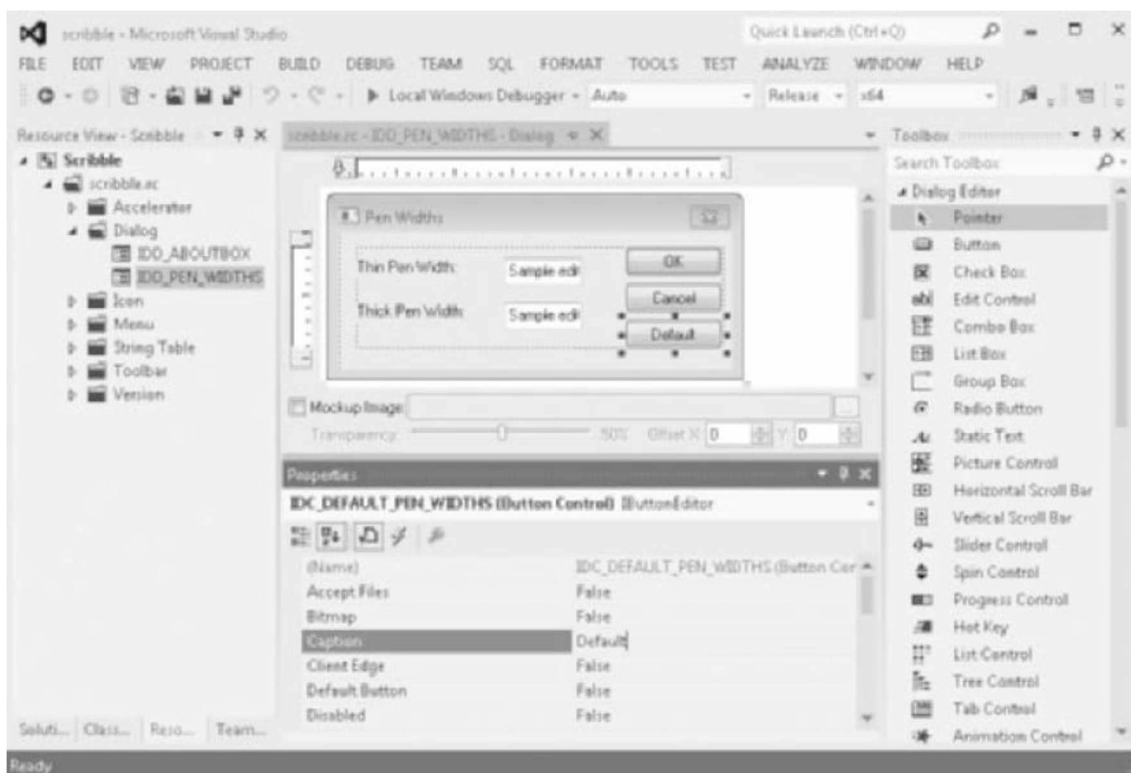


Figure 33.8 - An object-oriented program interface is drawn using classes of user control objects (righthand pane); composite interface objects are defined and stored as resources (left-hand pane). Localization of object-oriented software is properly understood as the modification of the properties of objects, such as the command button caption 'Default' (upper and lower middle panes). This image depicts the creation of the sample application Scribble (see Figures 33.1 and 33.5) in Microsoft® Visual Studio® 2012

Source: - Used with permission from Microsoft

Figure 33.8 – Uma interface de programa orientado a objeto é desenhada usando classes de objetos de controle de usuário (painel direito); objetos de interface composta são definidos e armazenados como recursos (painel esquerdo). A localização de um software orientado a objeto é entendida corretamente como uma modificação das propriedades de objetos, como a legenda do botão de comando “Padrão” (painéis superior e inferior do meio). Essa figura ilustra a criação do aplicativo de amostra Scribble (ver Figuras 33.1 e 33.5) no Microsoft® Visual Studio® 2012.

Fonte: Usado com permissão da Microsoft

Internationalization and the use of satellite resource files allow software publishers to proactively address potential locale-related problems during the software development process, well in advance of localization. This state of affairs begs the question of how – and perhaps even if – localization differs from translation today. The fact that the translation of strings comprises the bulk of the work in current practice suggests that the term

A internacionalização e o uso de arquivos de recursos-satélite permitem que os desenvolvedores abordem de maneira proativa os problemas relacionados à localidade durante o processo de desenvolvimento do *software*, bem antes da localização. Nesse ponto, é necessário saber como – e talvez até se – a localização difere-se da tradução atualmente. O fato de que a tradução das linhas de textos compreende a

'localization' has come full circle and once again essentially means 'translation on the computer, for the computer.' The blurring of the boundaries between translation and localization can also be seen as evidence of a convergence of these processes as authoring and publishing undergo an evolution similar to that of software localization (Esselink 2003b).

Authoring and publishing were generally separate processes and professions until the 1980s and 1990s, when the advent of digital authoring tools such as word processors turned authors into desktop publishers who were able not only to create digital content, but also to control the manner of its presentation (Rockley, Kostur and Manning 2003: 165). However, the desktop- and document-based approach to authoring and publishing hindered content reuse in much the same way as early localization efforts that required modifications to source code.

maior parte do trabalho na prática atual sugere que o termo "localização" tenha permanecido da mesma forma, significando "tradução no computador, para computador". A confusão dos limites entre tradução e localização também pode ser vista como evidência de uma convergência desses processos à medida que a autoria e publicação sofrem uma evolução semelhante à da localização de *software* (Esselink, 2003b).

A autoria e a publicação foram processos e profissões separadas até os anos 1980 e 1990, quando o advento das ferramentas de autoria, como o processador de textos, transformou autores em desenvolvedores digitais que eram não só capazes de criar conteúdos digitais, mas também de controlar a maneira de sua apresentação (ROCKLEY, KOSTUR e MANNING, 2003, p. 165). No entanto, a abordagem baseada na área de trabalho – e documentos – para autoria e publicação impediu o reuso de conteúdo, da mesma maneira que os esforços iniciais da localização, que exigiam modificação no código-fonte.

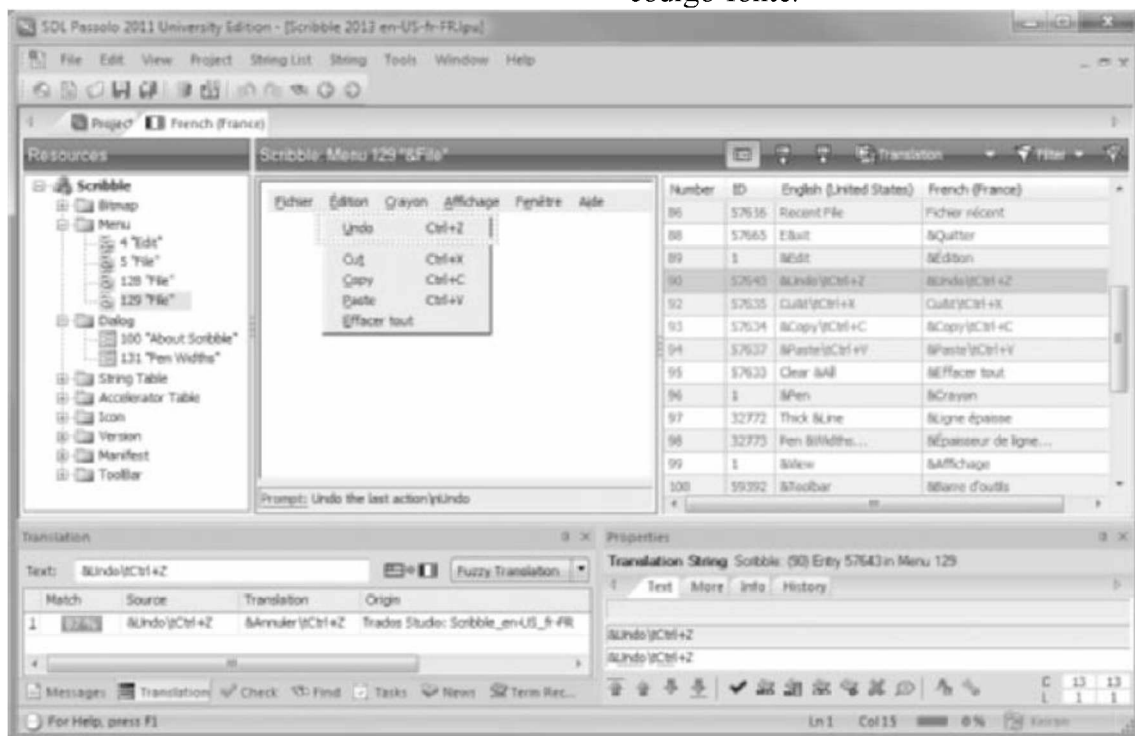


Figure 33.9 - Localization of a sample application named Scribble using a visual localization tool. The left-hand pane displays the resource tree, the middle pane displays the selected resource in WYSIWYG mode, and the right-hand pane displays the corresponding

source and target strings in tabular format

Figura 33.9 – A localização de um aplicativo de amostra chamado Scribble usando uma ferramenta de localização visual. O painel esquerdo exibe a árvore de recurso, o painel do meio exibe o recurso selecionado em modo WYSIWYG, e o painel direito exibe a cadeia de origem e as *strings* de destino em formato tabular.

Repurposing content stored in documents created using word processors and other traditional desktop publishing software is generally a labor-intensive, manual process. A common rule of thumb is that technical communicators who use word processors and document-based authoring tools spend as much as half of their time formatting documents (e.g., Bartlett 1998). 'To reuse the content, authors must apply formatting that is appropriate for each output. Stripping and reapplying formatting is tricky and usually not 100% effective. Format conversions always require correction by hand or complicated scripting' (Rockley et al. 2003: 165). Further complicating content reuse efforts was the need to manage multiple versions of document source files in parallel. Practitioners soon discovered that file management and version control become exponentially more complex as the number of parallel versions of source files and the number of target languages increase. These problems were exacerbated by the Web and its widespread adoption as an enterprise communication platform. Desktop-based authoring and publishing could not keep up with the speed of change on the Web, nor could it meet demand for content in an increasingly wide range of formats for use on an expanding array of devices, from PCs and laptops to PDAs, tablets and smart phones.

The challenges of document-based content reuse were very similar to those associated with early software localization efforts. Thus, it is perhaps unsurprising that the strategies adopted to facilitate content reuse are very

Adaptação de conteúdo armazenado em documentos criados utilizando processadores de textos e outros *softwares* tradicionais é geralmente um processo manual e intenso. Uma regra geral é que comunicadores técnicos que usam o processador de textos e ferramentas autorais de documentação gastam a metade do tempo com a formatação de documentos. (BARLETT, 1998). "Para reutilizar o conteúdo, os autores devem aplicar a formatação apropriada para cada produção. Tirar e reaplicar a formatação é complicado e não é totalmente efetivo. A conversão de formatos sempre requer correção manual ou scripts complicados" (ROCKLEY et al. 2003, p. 165). Outros esforços para reutilizar conteúdos complicados eram necessários para gerenciar várias versões de documentos de arquivos-fonte em paralelo. Os profissionais logo descobriram que o gerenciamento de arquivo e o controle de versão tornaram-se exponencialmente mais complexos à medida que o número de versões paralelas dos arquivos-fonte e o número de línguas de chegada aumentavam. Esses problemas eram agravados pela Web e sua ampla adoção como uma plataforma de comunicação empresarial. A autoria e publicação baseadas em área de trabalho não conseguiram acompanhar a velocidade das mudanças na Web, nem poderiam atender à demanda de conteúdo em uma gama, cada vez maior, de formatos para uso em uma variedade de dispositivos, desde PCs e notebooks a *palmtops*, *tablets* e *smartphones*.

Os desafios de utilizar conteúdos baseados em documentos foram muito semelhantes aos esforços iniciais da localização de *software*. Assim, talvez não seja surpreendente que as estratégias adotadas

similar to strategies developed to facilitate software localization. Just as internalization simplified localization by logically separating the culturally and linguistically dependent aspects of the user interface from the functional core of a program, content reuse strategies are predicated on the separation of content from presentation. This approach is called single sourcing: 'Single sourcing implies that there is a single source for content; content is written once, stored in a single source location, and reused many times' (Rockley et al. 2003: 15).

The implementation of single sourcing typically involves XML-based authoring strategies (Savourel 2001: 7; Rockley et al. 2003: 159-171). XML (eXtensible Markup Language) is a meta-markup language that provides a universal, application-independent mechanism for representing text in a structured format. XML was created in response to the challenges of content reuse associated with large-scale digital publishing. As stated in a December 1997 World Wide Web Consortium Press release announcing the publication of version 1.0 of XML as a proposed recommendation, 'XML is primarily intended to meet the requirements of large-scale Web content providers for industry-specific markup, vendor-neutral data exchange, media-independent publishing, one-on-one marketing, workflow management in collaborative authoring environments, and the processing of Web documents by intelligent clients' (W3C 1997). Whereas HTML is a presentational markup language that specifies how content should be displayed, XML is a semantic markup language that specifies what content means. Because XML provides a structured, semantic representation of content, and does not focus on the presentational aspects of document, authoring of content in XML is often referred to as structured authoring. Formatting of XML in documents is specified by style directives stored in separate files and

para facilitar o reuso de conteúdos sejam semelhantes às estratégias desenvolvidas para facilitar a localização de *software*. Da mesma forma que a internacionalização simplifica a localização separando de forma lógica os aspectos dependentes culturais e linguísticos da interface do usuário do núcleo funcional de um programa, as estratégias de reuso de conteúdo são baseadas na separação do conteúdo da apresentação. Essa abordagem é chamada de fonte única: "Fonte única implica que há uma única fonte de conteúdo, o conteúdo é escrito uma vez, armazenado em um local de fonte única e reutilizado várias vezes" (ROCKLEY et al., 2003, p. 15).

A implementação de suplementos envolve estratégias autorais de XML (SAVOUREL, 2001, p. 7; ROCKLEY et al., 2003, p. 159-171). A linguagem XML (eXtensible Markup Language) é a linguagem de metamarcação que fornece um mecanismo universal, de aplicação independente para representar textos em um formato estruturado. A XML foi criada em resposta aos desafios de reuso de conteúdos associados a publicações digitais de larga escala. Conforme estabelecido em um comunicado de imprensa do World Wide Web de dezembro de 1997, anunciando a publicação da versão 1.0 do XML como uma recomendação proposta, "a XML destina-se principalmente a atender aos requisitos dos provedores de conteúdo da Web em grande escala para marcação específica do setor, troca de dados neutra do fornecedor, publicação independente de mídia, marketing individualizado, gerenciamento de fluxo de trabalho em ambientes de autoria colaborativa e processamento de documentos da Web por clientes inteligentes (W3C, 1997). Enquanto a HTML é uma linguagem de marcação de apresentação que especifica como o conteúdo deve ser exibido, a XML é uma linguagem de marcação semântica que especifica o que o conteúdo significa. Porque a XML provê uma representação semântica e estruturada do conteúdo, e não

applied dynamically in response to user demand. In this way, the same content can be processed and output in any format for which the organization has a defined set of style rules, such as webpages (HTML), PDF documents, Word documents, as well as Eclipse Help, HTML Help, Java Help and WebHelp, to cite but a few examples. Today, the single sourcing of technical and procedural documentation is often implemented using the XML based DITA (Darwin Information Typing Architecture) standard (Bellamy, Cary and Schlotfeldt 2012).

prioriza a apresentação dos aspectos atuacionais do documento, a autoria do conteúdo em XML é geralmente chamada de autoria estruturada. A formatação da XML em documentos é especificada por diretivas de estilo armazenadas em arquivos separados e aplicados de forma dinâmica em resposta à demanda do usuário. Dessa maneira, o mesmo conteúdo pode ser processado e produzido em qualquer formato para o qual a organização possui um conjunto definido de regras, tais como as páginas de web (HTML), documentos em PDF, documentos do WORD, bem como em Ajuda Eclipse, ajuda HTML, ajuda Java e WebHelp, para citar apenas alguns exemplos. Hoje, a única fonte de documentação técnica e processual é frequentemente implementada usando a DITA baseado em XML (*Darwin Information Typing Architecture*, em português Arquitetura Darwin de Informação de Dígito) (BELLAMY, CARY e SCHLOTFELDT, 2012).

The implementation of single sourcing also often involves the use of content management systems (Rockley et al. 2003: 178-191), which are centralized, server-based repositories 'designed to manage "information chunks" (generically known as "content"), usually no longer than a couple of paragraphs' (Biau Gil and Pym 2006: 11). Information chunks, such as DITA topics, are dynamically assembled into documents in response to user requests, typically via a web interface. As is the case with XML-based authoring, content stored in a content management system (CMS) can generally be output in various formats.

A implementação de um suplemento, muitas vezes, também envolve o uso de sistema de gerenciamento de conteúdo (ROCKLEY et al., 2003, p. 178-191), que são depósitos centralizados "designados a gerenciar 'partes de informação' (genericamente conhecido como 'conteúdo'), não sendo maiores que dois parágrafos" (BIAU GIL e PYM, 2006, p. 11). Pedacos de informação, como os tópicos DITA, são montados de maneira dinâmica em documentos em resposta aos pedidos dos usuários, tipicamente via interface na WEB. Já no caso da autoria XML, o conteúdo armazenado em um sistema de gerenciamento de conteúdo (*Content Management System* - CMS) pode, geralmente, ser produzido em vários formatos.

Single sourcing, structured authoring, and 'chunking' can be thought of as applications of the concepts of object orientation and internationalization in the fields of authoring and publishing. Once written, a given information object can be reused

Suplementos, autoria estruturada e "fragmentação" podem ser entendidas como aplicações dos conceitos de orientação de objetos e internacionalização no campo de autoria e publicação. Uma vez escrita, uma informação de objeto pode ser reutilizada

systematically; once translated, the target language versions of that same information object can also be reused systematically. Because content is separated from form, it can be processed using as many different style directives as needed to publish it in the desired output formats (e.g., print, help, web, and mobile devices) without having to modify the content. Just as object orientation and internationalization facilitated the modularization and reuse of software, single sourcing, structured authoring and chunking facilitate the modularization and reuse of content.

Translation of XML content and of information chunks is not 'localization' as the process has been traditionally understood, because it does not entail modification of the properties of objects in a software user interface. Nevertheless, 'content translation projects are now often considered as localization projects simply because of the complex environments in which the content is authored, managed, stored and published,' as Esselink has pointed out (2003b: 7). Complexity was once a defining characteristic of software localization projects, but now characterizes large-scale translation projects as well.

At a more fundamental level, the complexity of software localization and content translation is due largely to the fact that translators and localizers do not work on linear text but rather on decontextualized text strings or chunks. Working on text without context not only complicates the translation decision-making process, but arguably calls into question the very possibility of understanding the text as a whole and the pragmatic act of communication of which it is an ostensible artifact. 'In understanding text, a reader must not only be able to integrate information within sentences but also make connections across sentences to form a coherent discourse representation,' as

sistematicamente; uma vez traduzida, a versão da língua-alvo daquela mesma informação também pode ser reutilizada de forma sistêmica. Por ser separado da forma, o conteúdo pode ser processado utilizando-se quantas diretivas de estilo diferentes forem necessárias para que ele seja publicado, nos formatos de produção desejados (por exemplo: impressão, ajuda, web, e dispositivos móveis) sem ter de modificar o conteúdo. Assim como a orientação de objetos e a internacionalização facilitaram a modularização e reuso do *software*, os suplementos e fragmentações facilitam a modularização e o reuso do conteúdo.

A tradução do conteúdo XML e de informações fragmentadas não é "localização", segundo se compreende tradicionalmente o processo, porque não envolve a modificação das propriedades dos objetos na interface com o usuário de um *software*. Entretanto, "projetos de tradução de conteúdo são agora considerados como projetos de localização por causa dos ambientes complexos em que o conteúdo é desenvolvido, gerenciado, armazenado e publicado", destaca Esselink (2003b, p. 7). Os projetos de localização de software foram uma vez definidos como complexos, mas agora também caracteriza projetos de tradução de larga escala.

Em um nível mais básico, a complexidade da localização de *software* e de tradução de um conteúdo dá-se amplamente pelo fato de que os tradutores e localizadores não trabalham com textos lineares, mas sim com partes e linhas de textos descontextualizadas. Trabalhar com textos sem contexto não só complica o processo de decisão tradutória, mas traz à tona a questão da possibilidade de entender o texto como um todo e o ato pragmático de comunicação, que é um artefato ostensivo. "Ao entender um texto, o leitor não deve somente ser capaz de integrar informação dentro de frases, mas também fazer conexões ao longo de frases para formar uma representação

Rayner and Sereno observe (1994: 73). However, it is not always possible for translators to make connections across sentences while working on software strings. 'Due to their non-linear structure and lack of narrative thread, software programmers cannot be "read" in the same way as [traditional documents]' (Dunne 2009: 197). This also holds true for XML content and information chunks. In single sourcing projects, the 'document' does not exist until it is created dynamically in response to a user request (typically from an end-user). On a surface level, the translation of strings and information chunks may seem technologically simpler than traditional localization because translators do not have to compile or test target files. However, the translation of strings and information chunks is cognitively more complex because reading and comprehending text without context and 'texts without ends' (Biau Gil and Pym 2006: 11) requires translators to construct a situation model of a text that does not yet exist. In other words, the industry is no longer trying to turn translators into semi-engineers, but translators still need to understand the architecture of the components from which software localization project deliverables are created, such as software resource files, and Help topics, tables of contents and indexes. As Esselink observes, 'it looks likely that while translators will be able and expected to increasingly focus on their linguistic tasks ... the bar of technical complexity will be raised considerably as well' (2003b: 7).

coerente de um discurso", observa Rayner e Sereno (1994, p. 73). Entretanto, não é sempre possível para o tradutor fazer conexões ao longo de frases enquanto trabalha com linhas de textos de *software*. "Devido à sua estrutura não linear e à falta de um fluxo narrativo, *softwares* não podem ser 'lidos' da mesma forma que [documentos tradicionais]" (DUNNE, 2009, p. 197). Isso também serve para conteúdo XML e parte de informação. Em projetos de suplementação, o "documento" não existe até que ele seja criado de maneira dinâmica em resposta ao pedido do usuário (tipicamente do usuário final). De forma geral, a tradução de linhas e de parte de informação pode parecer tecnologicamente mais fácil que a localização convencional, pois os tradutores não precisam compilar ou testar arquivos alvo. Entretanto, a tradução dos formatos mencionados anteriormente é cognitivamente mais complexa, pois ler e compreender textos sem o contexto e "textos sem fim" (BIAU GIL e PYM, 2006, p. 11) requer que o tradutor construa uma situação-modelo de texto que não existe ainda. Em outras palavras, a indústria não está mais tentando transformar os tradutores em semi-engenheiros, mas os tradutores ainda precisam entender a arquitetura dos componentes em que os resultados dos projetos de localização de *software* são criados, como os arquivos fonte do *software*, e tópicos de ajuda, tabelas de conteúdos e índices. Como Esselink observa, "parece provável que enquanto os tradutores forem capazes de concentrar-se em suas funções linguísticas [...] a barreira da complexidade técnica também aumentará de maneira considerável" (2003b, p. 7).

Notes

1 For more information on target-market requirements, see Giammarresi 2011, especially 39–40.

2 For a case study that illustrates some of the problems that can occur in the absence of an organized approach to internationalization, see Margulies 2000.

Notas:

1 Para mais informações sobre requisitos de mercado exterior, ver Giammarresi (2011), especialmente as páginas 39–40.

2 Para um caso de estudo que ilustra alguns dos problemas que podem ocorrer na ausência de uma abordagem estruturada de internacionalização, ver Margulies (2000).

3 These resources are derived from a sample application called Scribble developed by the author using Visual Studio 2010 C++ sample files. MSDN Archive, Visual C++ MFC Samples for Visual Studio 2010, <http://archive.msdn.microsoft.com/vcsamplesmfc> (accessed Sep. 8, 2012).

3 Esses recursos são derivados de uma aplicação de amostras de arquivos chamada Scribble, desenvolvido pelo autor usando o Visual Studio 2010 C++. As amostras MSDN Archive, Visual C++ para Visual Studio 2010, <http://archive.msdn.microsoft.com/vcsamplesmfc> (acesso em: 8/09/12).