Sandrini, Peter (2008): Localization and Translation. In: MuTra Journal, Vol 2 2008. LSP Translation Scenarios. Selected Contributions to the EU Marie Curie Conference Vienna 2007. Edited by Heidrun Gerzymisch-Arbogast, Gerhard Budin, Gertrud Hofer. Saarbrücken: ATRC 167-191.

Peter Sandrini (Innsbruck)

Localization and Translation

Abstract – After a short introduction into the main concepts of localization the paper discusses the two main areas of software and website localization. It first focuses on the object of the localization process and its pecularities and then describes the translation procedure as well as the specific tools to use.

1 Introduction

Localization is a relatively new field of activity for language experts. It is closely linked to digital media and computer products. It is a field of activity where technology is deeply involved and which would be inconceivable without thorough technical preparation.

Before we can explore the methods and procedures of localization, we need to define the most important terms: globalization and internationalization, localization and locale which have been abbreviated by the acronym GILT (globalization, internationalization, localization and translation).

Globalization has two meanings: in general, it refers to the globalizing scope of the economy and of business activity. In the context of localization, it refers to the business activities related to marketing a product or service in multiple regional markets.[1]

Internationalization describes the "process of enabling a product at a technical level for localization" (Lommel/Ray 2007: 17) so that it can be easily adapted for a specific market after the engineering phase.[2]

1 The acronym G11n (11 characters between G and n) is used for globalization
2 The acronym I18n (18 characters between I and n) is used for internationalization

Locale is a set of parameters used to identify the user's language, country and other preferences. It is roughly the combination of a language and a geo graphical region with all the cultural implications involved. For example, a country, a region, or a city. A locale is not a culture since a culture is more comprehensive, much more rooted and much less volatile than a locale. Take the EU for example, you could adapt a software product for the EU market to be used in English with all the necessary legal and cultural adaptations. A locale is not a language, though language is a vital part of a locale. However, there could be locales which use the same language as the original but nonetheless require an adaptation of the product to be successful, e.g. French in France and French in Canada. Localizing a product means adapting the linguistic and cultural specifics of content (text, images, voice sequences, etc) to a given geographical or demographic locale. It includes adapting content to the local conventions for such features as date and time formats, currencies, numbers, language, colour coding, cultural choices, writing systems. For data representation a locale sets sorting algorithms and also upper- or lower casing characters. . Technically, locales are represented by the two-letter code for the representation of languages (ISO 639-1) combined with the standardized country codes (ISO 3166-1), such as *de-BE* which refers to German spoken in Belgium, or *en-CA* meaning English in Canada.

Localization: Generally defined, we can define localization[3] as „the provision of services and technologies for the management of multilinguality across the global information flow“ (Reinhard Schäler 1999 cited in Localization Focus September 2002: 21). This comprises the industry, the providers of tools as well as the localization work itself. More practically, localization is the „process of modifying a product for a specific locale“ (Yunker 2002: 17). The aim of localization should be that people from a specified locale can use the product without any difficulty in their own language. A given product could be everything you can sell to an international consumer group, but in practice the term localization is usually used for software application programs and websites.

The following introduction into the basic concepts of localization will be twofold: the first part addresses the issues that arise when localizing a software product while the second part is dedicated to website localization.

3 The acronym L10n (10 characters between L and n) is used for localization

2 Software Localization

If we speak of software we mean all non-material components of an electronic system, which can be software such as operating systems, compilers, hardware drivers, utilities and testing tools on one hand and application software such as databases, image manipulation, office application suites, desktop publishing, games, etc. on the other. It should be possible for users in different locales using different languages to make use of localized software systems without difficulties: this is the overall aim of localization efforts.

Technically speaking, software is basically programming code assembled in an executable file which tells the hardware processor what to do. This program code is dependent on the type of processorbeing used, and its instructions must work the same way before and after the localization process. Inside the program code there are elements which provide interaction interfaces with the user: menus, dialog boxes, text strings. These must be translated. The code could also contain locale specific elements which must be localized as well, such as date and time formats, currency units, paper formats, hot keys, etc. Buried in the code we could for example find other text as direct instructions such as *dir, list, copy,* etc. which cannot be regarded as English text to be translated. Such direct instructions are part of the programming language and should not be changed in any way.

Apart from the program code there will be an online help system, in most cases a hypertext type document similar to the windows help system in compressed HTML format .chm, or plain HTML in Unix environments. An online help resource will be linked to the program code, but tutorials and manuals are independent documents. Both need to be localized.

Usually a software product will come with a printed manual, quick reference guide, registration cards, promotional material, etc. Today, increasingly more software producers rely on electronic guides either online or on a CD-Rom, limiting the printed material to an installation and quick reference guide. These documents in paper format also have to be translated.

In the following chapter we will concentrate on electronic resources rather than printed documents, since the translation of the latter does not differ much from other technical documents.

The main problem in software localization is separating the elements which should be localized from the programming code which should not be altered in any way, or otherwise the software will not function properly.

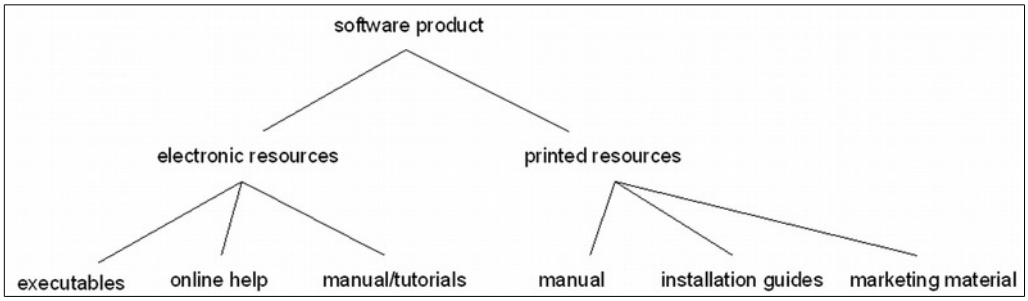

Fig. 1 Objects of Software localization

Executable program files will come in different formats depending on the OS platform, e.g. Windows executables will be .exe, .com, .dll, .bat or .drv files, whereas Unix programs will be .bin files. Generally speaking, we can identify three steps in the localization of binary files or executable program files:

1. preparing the programming code
2. translating elements
3. testing the code

The first task in a software localization project is to analyse the product in order to understand how and with what tools the software was produced. The built environment and the source files – i.e. the code in the original programming language before it was compiled – should be made accessible to the localization professional by the customer in the so-called localization kit. A localization kit contains all necessary files as well as the documentation for the localization project: source files, built environment, guidelines, available glossaries and translation notes.

The source files are usually programming code files – e.g. instructions in the programming language C++ - and resource files (with the extension .rc), which include user interface elements such as menus and dialog boxes, which constitute the main elements to be translated. It is a good programming standard to have separate resource files which is more or less the case nowadays. Some resource files must be compiled with the programming code however, in order to obtain a binary executable program file. More and more software programmers extrapolate the resource file straight away into a binary resource file, a .dll or an .exe; thus you can localize only this file without the need to recompile the entire program.

If there is no resource file whatsoever, translatable items are mixed up with the programming instructions, and it is impossible to evaluate whether a word

like 'copy' would be an English word in a dialog box, a menu item to be translated or a programming instruction in a programming language that should not be tampered with.

As a result, a localizer would have to check whether there are translation resources for re-use in this project, i.e. whether the files had already been translated for another version of the product and the translations saved in a translation memory, or whether a terminology database exists for this software or a previously localized similar product, etc.

The decision as to which tools should be used in the project is based on the following criteria:

- file format support;
- safety of program code: the tool should help you to avoid inadvertently overwriting chunks of machine code;
- client requirements;
- leveraging or re-use of translations.

These criteria follow a hierarchy where the file format is by far the most important factor. The main types of file formats in software localization are the following:

- .exe, .dll: standard binary files can only be localized with a localization tool due to the necessity to separate localizable elements from programming instructions. However, binary files compiled in a non-standard way, i.e. programming instructions without an original resource file and compiled into one application – result in non-standard .exe, .dll files: in this case the localizer must resort to the source code, translate all elements within the native development environment and recompile the whole application;
- .rc: text-based resource files can be localized using a localization tool or within the native development environment, e.g. an object-oriented programming language like Visual C++ or Delphi. Not all localization tools, however, support .rc files.
- Online help: given the repetitive and highly structured nature of online help texts the appropriate tool would be a translation memory system. Most translation memory systems can handle the HTML file format of the standard Windows help system(.chm) very well.
- Printed documents: provided all the documents are delivered in digital format, a translation memory system should be used to translate them. This would allow for re-use of repetitive text, increase consistency

throughout the document and facilitate the translation of new versions later on.

The right tool for the file format should allow for the efficient translation of text elements, but it also protects the sensitive programming instructions from inadvertently being changed by the user. It would be a risky undertaking to try to translate a binary file from within a general text editor without the strict division between translatable text elements and programming instructions provided by localization tools.

Situations could arise where the customer requires the localizer to use a specific tool either because there are resources available like a translation memory or because the software publisher has developed their own proprietary tool, as is the case. for Sun or Oracle for example..

Traditionally, the tools available are divided into the following groups: translation memory tools, software localization tools, native development environments, machine translation, terminology tools. Meanwhile, many tools have a broad range of features although many of them are designed for a specific task. If we leave aside the programming environments and concentrate on their functionality, we can clearly see that translation tools have become more linked. In the area of software localization tools the following systems are the most notable.

- Passolo (http://www.passolo.com)
- Alchemy Catalyst (http://www.alchemysoftware.ie)
- Language localizator (http://www.localizator.com)
- Multilizer (http://www.multilizer.com)
- RC-WinTrans (http://www.schaudin.com)
- Visual Localize (http://www.visloc.com)

A detailed comparison chart of these localization tools is available online at http://www.localizationworks.com/DRTOM/Conclusions.html.

Correspondingly, we will now explain briefly how to translate a small piece of software using Passolo. Firstly, a new project must be created where you have to specify the source file to translate, the type of binary file, the language of the text elements to extract from the binary file and the target language. The tool will then generate a string list extracted from the binary file. These strings can be marked to exclude elements which should not be translated or changed. Next, you have to create the bilingual string list where each string is listed in a table with the source language on the right and the target language on the left. Clicking on a string will open the edi-

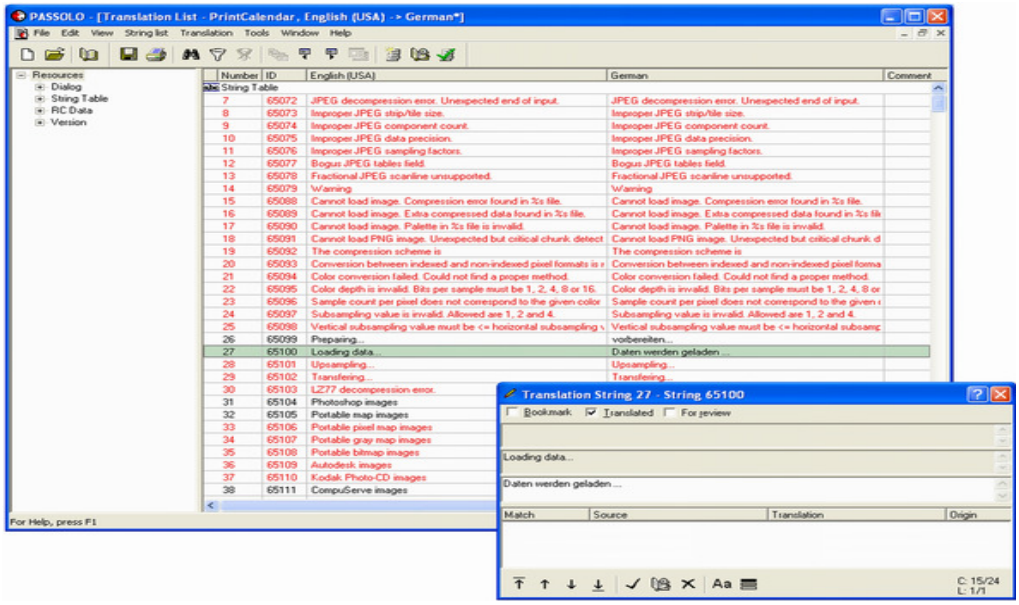

Fig. 2: Localizing with Passolo

tor window where the translation can be typed in. Translations can be automated, at least partially, using a glossary list. After translating all the strings in the table, you will generate the target file, resulting in a new binary file .exe being created and launched.

While this roughly outlines the procedure, there are a few other problems which have to be dealt with, such as the re-sizing of dialog boxes or the assignment of hot keys. During translation, text strings will inevitably change in length and this will affect dialog boxes as well as text messages. Alternatively, when translating from English to German, target text elements will usually be considerably longer and graphical elements must be adapted. A good localization tool will assist the localizer in this respect and provide a WYSIWYG (what you see is what you get) environment for changing graphical elements, which could then be just as simple as re-sizing a window on the Windows desktop.

Hot keys are marked letters in menus which can be used in combination with the ALT key to access the menu command, e.g. ALT F for the menu File and ALT O for the menu command Open file used in many Windows applications. These keys obviously change with localization and become e.g. ALT D for Datei (File) in German and ALT f for Öffnen (Open). All hot keys must be unique within the menu where they are used to ensure that the functionality of

the user interface is guaranteed. Hot keys must also be chosen so as not to collide with key combinations from the operating system.

Today, many localization tools integrate some form of translation memory component. Passolo and Catalyst in their current versions integrate a translation memory engine from SDL/Trados and can, therefore count on one of the most sophisticated tools in this regard with fuzzy matching and extended functionality.

After the translation process the software package should be run and tested. Such a quality check on the localized application implies some knowledge of software quality assurance principles and requires a testing plan which should take into account with focus on at least the following three aspects:

- *linguistic test*: This involves questions such as: Have all the text strings been translated? Are special characters displayed correctly? Is the text in dialog boxes and error messages truncated? Is text wrap and hyphenation ok? Are all menu items and titles used consistently? Is assignment of hot keys correct? Some localization tools offer routines for this kind of testing such as finding duplicate hot keys automatically.
- *user interface test*: This kind of testing refers to the aesthetic questions of the user interface in the localized version of the application.
- *functional test*: The functionality of the localized version will be tested against the functionality of the source language product to assure that there were no bugs introduced during the localization process. Also, the localized application must work properly in the target language context, which means that full interoperability with the corresponding localized operating system and with other localized products should be tested. And finally, all deliverables and installing procedures should be tested also..

3 Website Localization

Localization is the process of adapting a product to a new locale. We have seen how a software product is localized, Website localization, however, is something quite different. It is not only relatively new, as the World Wide Web we now take for granted only took off in the early 1990s and Website localization emerged a few years later. Website localization is a “specialised service that has emerged in recent years (since 1999). It is basically a packaging of translation services with technical services that ensure the proper functioning of the translated sites” (van der Meer 2002: 10). Translation and technical services consti-

tute the two main aspects of Website localization which can be defined as the process of modifying an existing Website to make it accessible, usable and culturally suitable to a target audience. Website localization is also much more open and less specialised than software localization.

The main differences between software localization and Website localization are:

- Update frequency: Websites are subject to constant change and localization is therefore a program-based task, whereas software localization is a project-based task where each new version of a software product will become a new localization project.
- Translation expertise: the Web is as multifaceted as the real world and websites contain many different types of text which require different translation strategies: marketing texts, product descriptions, legal information, manuals, listings, etc. In many cases, website translation requires much more particular subject-related knowledge than software localization.
- The relationship between the localizer and the customer in software localization is a project- oriented relationship where a defined task is carried out and the relationship is ended.. For a website there is the need for an ongoing relationship with the localizer as a result of the ever changing content of the website. The customer decides between either fully outsourcing where the localizer does the entire job or employing a translation service where only the planning and coordination remain with the customer.

Now that we have defined website localization, let us move on to examine the object of localization and what kind of content has to be localized.

A Website can be “a marketing channel, a software product, a brochure, a shopping mall or all of the above“ (Yunker 2003: 4), indeed a complex matter as a source text for translation. To make things easier we will speak of the *content* of a Website which is composed of *digital assets*. These digital assets constitute different aspects of website content:

1. common content: texts, images, links which constitute the main structure of a website.
2. multimedia assets: audio- and video streaming, flash animations.
3. application-bound assets: files and documents which are accessible only with special software applications (e.g. MS-Word documents or Adobe PDF), in this case the Web acts as a distribution service without displaying the content.

4. transactional assets: information about transactions (e.g. shopping baskets, sessions) in e-commerce.
5. community assets: dynamic content of discussion forums and chat rooms created by Website visitors.

All these digital assets possibly contain localizable information with the primary localizable component obviously being the texts. In relation to the need for change in the content and the durability of the content, the information can be divided into:

- static information (e.g. records of historical events, biographies, documentation of hardware and software, economic figures, manuals, laws and bills and legal documents, text passages from books or periodicals, press memos, etc.);
- dynamic information (e.g. stock exchange rates, warehouse inventory, the content of an email account, sports results, prices at online auctions, date and time, etc.);
- semi-dynamic information (e.g. people at a company, bibliographies, biographies of living people, price lists, etc.).

All information on the Web is subject to a *content life cycle* which describes the usability of the information from the time of creation to publication and finally archiving (internal or public). Dynamic information has a very short life cycle, while static information can have quite a long life cycle.

Not all assets are equally important for localization. Dynamic information would be rather difficult to localize efficiently for a website. Decisions have to be taken on what kind of information should be localized. We will come back to the criteria on which such decisions should be based.

All digital assets on a website need a structure to be presented to the visitor, and many websites use some form of integrated software to offer services or interactivity to the visitor, usually Java or Javascript code or Perl scripts embedded in the website. In this regard, website localization has much in common with software localization as these smaller programs need to be localized as well.

Information and digital assets will be made accessible in the form of files. Let us have a look at the most popular file formats on the Web:

HTM/HTML (Hypertext Markup Language), is localizable/translatable;
XML (Extensible Markup Language), is localizable/translatable;
CSS (Cascading Style Sheet), no content to localize;
XSL (Extensible Style Sheet Language), is localizable/translatable;
JS (Javascript) is localizable/translatable;

ASP (Active Server Pages), is localizable/translatable;
PHP (Hypertext Preprocessor) is localizable/translatable;
JSP (Java Server Pages), is localizable/translatable;
GIF (Graphics Interchange Format), no (only when text is embedded);
JPG (Joint Photographic Experts Group), no (only when text is embedded);
PSD (Photoshop Document), is localizable/translatable when one of the layer contains some text.

Website localization is not about translating single documents. The object of the localization process is an entire website composed of many pages, just as the text is the object of translation and not single sentences. For website localization, stretching this metaphor, the source text would be the complete website.

Before a website can be localized, the localizer should advise the publisher about the pros and cons of localization. The following items have to be considered by the publisher of the website:

- language choice: An international company would choose languages on the basis of their involvement in a certain market, the market potential measured in GNP growth rates, the number of active Internet users, costs, available staff for a particular language, etc.
- Return on Investment (ROI): will the necessary costs for localization be covered by the resulting advantages? Does the new market need the products/services and are they affordable there? How can potential customers pay and how can goods/services be delivered? Is there customer support for this market and language? Are there legal problems with the products in this market?

Only by answering these questions can the localization of a website be a success. If the publisher has uncertainties or doubts about these facts, every cent spent on localization will be too much. One of the tasks of the localizer is not only to advise the publisher on the advantages of localization but also on the implications of dealing with people from another culture and language.

Different approaches can be taken with localization, but it is never an a*ll or nothing* decision. There are various levels of localization possible depending on the volume of text and the type of Web content to be translated (global or regional content) or produced directly in the target language (local content). Regarding the websites of international companies, Rose Lockwood

(2000) has identified three main strategies for the management of multilingual and multicultural content:

1. The monarchist approach with central control over the content where content is translated but seldom adapted. The result is a website which is not sensitive to local markets.
6. The anarchist approach with multiple local sites without coordination, each using a different design. In this case there would be high costs and no corporate strategy.
7. The federalist or subsidiary approach would be a compromise between the first two as it integrates global regional and local content (GRL). Global content will be produced centrally, translated and used internationally; regional content is also translated and used in a regional context while local content will be produced locally in the local language without the need for translation.

Each company decides what kind of approach matches its requirements and possibilities. Another study (Schewe 2001: 204) establishes a close link between the marketing policy of a company and the choice of languages for its Web presence. The six main website language design strategies identified cover a broad range, from a monolingual site written in the home language reflecting a domestic marketing strategy to the multilingual website with English or the home language and several local independent websites in the respective local language reflecting a global player strategy.

Once an appropriate Web publishing strategy is in place, the first thing to do is to analyse the content in order to make it locale independent. This process which is called *internationalization* should adapt any locale-specific information such as dates, times, numbers, currencies, etc. It is about changing and adapting a website to simplify localization.

Texts have to be also adapted in order to be suitable for new cultures or to be at least culturally neutral. In comparison to software localization, website localization involves a slight shift of priorities. Software localization concentrates on functionality – the application must work in the target language – and language quality concerns are less critical to a certain extent. For websites, however, language quality is crucial as it is the medium to convey a specific content to the target audience.

Careful consideration should be given to which texts can be translated into which languages and which texts should be omitted or at least shortened. localization has its cost and a text should therefore be as concise as possible while still clear and comprehensible.

There are a few things to consider in source texts in order to simplify localization: use consistent terminology throughout the website, eliminate political, religious or specific references to individual countries, do not use slang or jargon, delete references to humour. All these elements are deeply rooted in culture and cannot be transferred easily across languages. Special attention should be given to visual effects such as graphics, symbols, the use of colours, etc., which also strongly depend on culture. Their reception differs significantly across languages and cultures. Culturally sensitive elements in texts also concern formatting conventions such as the representation of currency units, dates, time zones, time units, measurement units, numbers (e.g. the difference of meaning of *1,000* between US-English and German), addresses (zip-codes), international phone numbers, etc.

Another important preparatory step is to ensure that the multilingual website is capable of processing and displaying all the required languages and scripts. This is a technical challenge for the Web developers involved since there are several possibilities of displaying writing systems and languages on the Web. Computers understand only numeric codes and every input must be encoded using ordered numeric codes which are stored in bytes within the memory of the computer. There are different character types for the various languages and consequently the encoding must be specified for each web page.

The oldest encoding standard is the ASCII-code based on a 7-bit byte representation which has space for 128 characters. Thus, it was able to support only English and other languages without special characters while most other Western languages with their special characters could not be represented in ASCII. So, a new encoding was developed, the ISO-8859 series of character sets: ISO 8859-1 (Latin 1) for Western European languages, ISO-8859-2 (Latin 2) for Eastern European languages, ISO-8859-3 (Latin 3) for Southern European languages, ISO-8859-4 (Latin 4) for Northern European languages, ISO-8859-5 (Cyrillic) for Russian, Bulgarian and Ukrainian, and so on through ISO-8859-16.

At the same time, some encoding standards for languages with more than 256 characters or symbols were developed using a double-byte encoding such as Big5 for Chinese, Shift-Jis for Japanese. Still, these encoding standards are not able to represent two or more different languages on the same page since another code page has to be used for every language. This led to the development of a single standard which tries to combine all existing character representations into one encoding scheme: Unicode. Unicode is a 16bit encoding

system that currently has space for more than 96,000 characters or symbols meaning that in one encoding standard we have a unique numeric value for each conceivable character or symbol: each Latin character, each Chinese symbol or for that matter each Korean symbol or just about every other character possible. Almost all modern browsers support Unicode encoding (UTF-8).

An international multilingual website requires some planning. A directory structure has to be set up which should reserve one directory for each language where all the files for the localized versions are saved. A very simple example could look like this:

The most important thing to remember concerning the layout of an international website is the link between the different localized versions. This is one of the changes in a website which become necessary with localization. There has to be a gateway which lets the user choose the right language version or the version of the website for his or her specific country or region. The criteria by which the website is subdivided into localized versions reflects the locales chosen and may be languages, regions or countries. These three types of gateways can also be combined hierarchically where the user chooses first the region, then the country and the language, e.g. region EU, country Belgium, language French. Examples for combined gateways can be found at http://www.olympus.com, http://www.ikea.com, http://www.canon.com.

Certain factors have to be considered for the global gateway:

- the global gateway must be located in a prominent place on the website so that users can easily make their choices;
- it must support all the necessary character types;
- all choices must be available in the proper language, a country choice should read e.g. *Deutschland* not *Germany*;
- graphical and visual elements (maps, globe) are highly versatile since they are neutral;

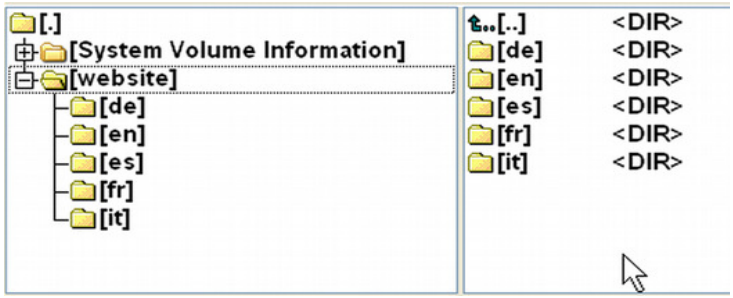

Fig. 3 Simple directory structure

- avoid flags, this is a bad habit born out of the early days of the Web. A national flag seldom reflects a language in its entirety: what kind of flag would you use for Spanish? Argentina, Spain, Mexico, Guatemala or even the US; what flag for German, English, Portuguese or French?
- user preferences regarding language may be saved in cookies on the local computer so that the user has to choose only once.

To construct a global gateway you will need an HTML Editor, a software package which will allow you to edit the HTML code directly. The global gateway can be constructed just like any other website component using either a WYSIWYG-editor or a text-based HTML-editor. A text-based editor is the tool of choice when HTML-files should be compared or tested for errors, good freeware examples are the following editors:

NVU http://www.nvu.com;
Notetab light http://www.notetab.com;
Phase 5 http://www.qhaut.de/forums/index.php?dlcategory=1;
PSPad editor http://www.pspad.com/.

Websites are accessible through the net and can be easily downloaded. Such offline versions are not well suited for translation purposes because of possible problems that result from direct downloading. It is good practice, therefore, to ask for a localization kit from the customer. A localization kit is the complete set of files necessary for the localization project saved on a disk, CD-Rom or received via email and consists of:

1. the entire original operational website as it would appear on the Web, as well as
2. all the files scheduled for localization;
3. resources for re-use such as glossaries or translation memories;
4. guidelines and style sheets.

The original website serves as a reference for the localizer regarding the functionality of the website. Furthermore, the customer should specify which parts of the original website are intended to be localized and identify all the respective files. And finally, all language resources should be made available to the localizer, such as existing glossaries and translation memories.

A common standard file format has been developed by a consortium of localization tools providers to overcome difficulties in sharing files for localization: XLIFF or the XML based localization Interchange File Format. XLIFF supports the lossless interchange of localizable data and its related information

and ensures the compatibility of tools because it is tool-neutral. A few tools which already support XLIFF are:

- Enlaso's Rainbow which is essentially a file converter between HTML, XML and RTF files and the XLIFF file format http://www.translate.com/technology/tools;
- Heartsome's XLIFF Translation Editor, a translation memory system which uses XLIFF as the format of choice http://www.heartsome.net.

Apart from file conversions, website localization needs tools that can:

1. separate content from form: meaning basically, separating HTML or XML code from the text;
2. re-use already translated text (translation memories);
3. detect changes and updates in already translated websites (translation memories);
4. edit Web documents: HTML/XML-editors.

Additional requirements may include terminology management and project management features.

The most popular file format on the Web is still the HTML-Format (Hypertext Mark-up Language), a relatively simple file format which is used to mark up text elements either to assign them a layout or to give them a certain meaning. The mark up element <H1>*text*</H1>, for instance, stands for a heading of first order, a *text* for a paragraph, a *text* for bold formatting of the text. While HTML does both layout (e.g. bold formatting) and content mark up (e.g. header), a newer mark up language has been developed to separate both aspects more clearly. XML *extended mark up language* uses only content mark up and leaves layout and formatting to external style sheets. XML has been applied to Web documents and the new standard is called XHTML. In the following we will outline briefly how such a tagged file is translated; we cannot, however, talk about other types of files such as script files, written in JavaScript or VBScript, or server-side technologies such as ASP, PERL or PHP.

The mark up language and its version as well as the encoding used in a Web document will be inserted in the header of the Web page and is visible in the browser under the menu item view/source.

In the first line you see the specification DOCTYPE stating that HTML version 4.01 is used in this document, transitional refers to the pre-XML/

```
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
    <meta name="Description" content="The Open Source Home Page">
    <meta name="Keywords" content="open source, open-source, free
software">
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1"><title>Open Source Initiative OSI -
Welcome</title></head>

    <body bgcolor="#ffffff">...
```

Fig. 4 HTML-Header

XHTML status of HTML version 4.01. In line number 5 you see the specification <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"> meaning that this document uses the encoding ISO-8859-1 (Latin 1).

Apart from these specifications, the header of a Web page also contains some elements which must be translated: e.g. the description of the content in the <meta name="Description" content=...> element, or the keywords used by search engines in the <meta name="Keywords" content=...> element, and most importantly, as it is directly visible to the visitor, the title as displayed in the title bar of the browser window in the element <title>...</title>, and possibly quite a bit more as specified in the RDF-standard (Resource Description Framework).

A good tool will recognise and protect all the tags in the Web page. It should, however, highlight all the elements which should be translated. In figure 5 you see how Trados® TagEditor handles the HTML mark up: the tags are grey and blocked out leaving the text editable for translation.

Not all tools allow the translation of meta-tags in the header: see in figure 6 how the freeware tool OmegaT hides all the information in the header except the title. Some tools hide all tags from the translator, others protect them but make them visible to the translator. Even when tags are protected, they often need to be moved within the text to accommodate target language requirements. Some tools allow moving of tags, others do not. This ability to move, edit or delete tags, however, is a must if you wish to use a comprehensive tool; or you would have to otherwise adjust the tags in a HTML-editor.

Another way to translate HTML files is to prepare the files before translating them. This must be done when you want to use an MS-Word based translation tool like Wordfast® or Trados® Workbench without the TagEditor

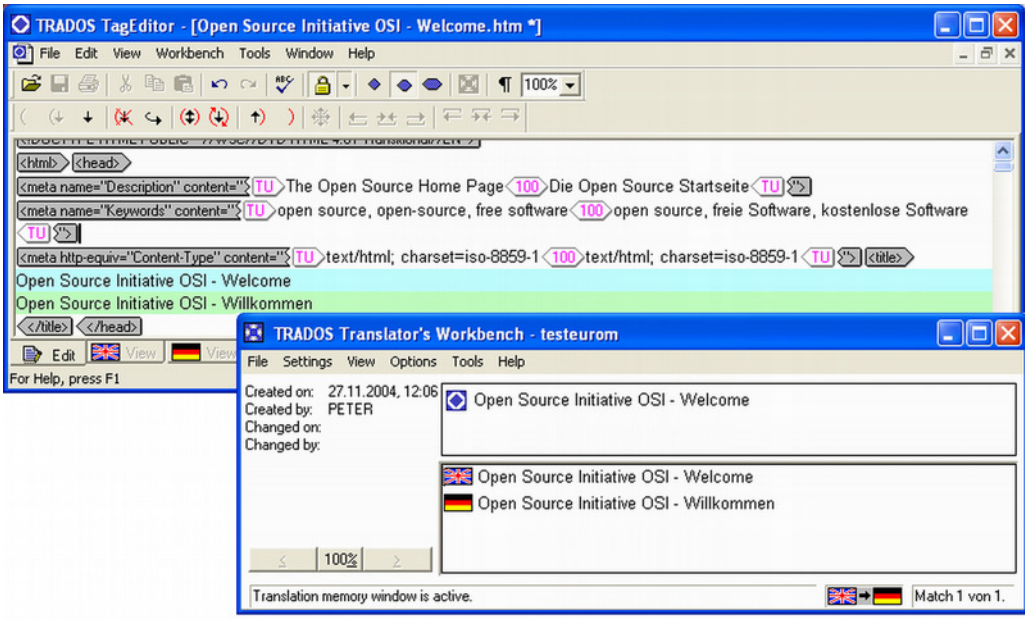

Fig. 5 Trados TagEditor and Trados Workbench

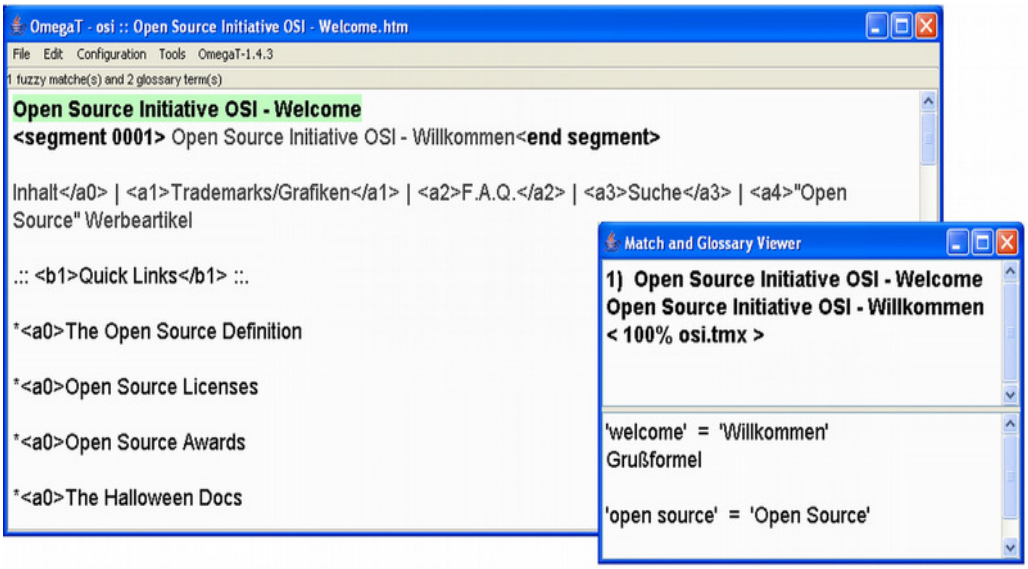
OmegaT - osi :: Open Source Initiative OSI - Welcome.htm
File Edit Configuration Tools OmegaT-1.4.3
1 fuzzy matche(s) and 2 glossary term(s)
Open Source Initiative OSI - Welcome
<segment 0001> Open Source Initiative OSI - Willkommen<end segment>
Inhalt</a0> | <a1>Trademarks/Grafiken</a1> | <a2>F.A.Q.</a2> | <a3>Suche</a3> | <a4>"Open Source" Werbeartikel
.:: <b1>Quick Links</b1> ::.
*<a0>The Open Source Definition
*<a0>Open Source Licenses
*<a0>Open Source Awards
*<a0>The Halloween Docs
Match and Glossary Viewer
1) Open Source Initiative OSI - Welcome
Open Source Initiative OSI - Willkommen
< 100% osi.tmx >
'welcome' = 'Willkommen'
Grußformel
'open source' = 'Open Source'

Fig. 6 OmegaT open source translation memory system

for instance. Since MS-Word can not edit HTML files directly, it loads the HTML code, converts it and recompiles it using a very proprietary HTML output which inevitably leads to a multitude of problems. To avoid this, you

can use a tool (e.g. the freeware Enlaso Rainbow) to convert the HTML file to an RTF file which you can load into Word and translate with Wordfast® or Trados Workbench®.

Separating the HTML tags from the localizable text is the most important feature of a Web localization tool. A translation memory feature makes translations more cost-effective allowing for the re-use of previously translated text. It also saves time on translation projects and increases consistency. Other criteria influencing the choice of a translation memory system could be:

- Segmentation rules: The underlying principle by which the system segments the text. Some tools segment by sentences, some by paragraphs, others by word sequences. The smaller the unit the more matches the system is able to propose, which is potentially attractive, but not really practical if you have to choose the right translation from more than ten matches. The larger the segmentation unit, the fewer results will be produced by the system, but the results will be better as more context is taken into account and a word-for-word or sentence-for-sentence translation procedure will be avoided.
- Matching algorithm: not just perfectly corresponding units should be found and proposed for re-use, but also more or less similar units. Such a search strategy is called *fuzzy matching*.
- Support of standards: this is important if you plan to exchange translation memories. A standard file format for translation memories based on XML has been developed: the Translation Memory Exchange Standard TMX. Most tools support the TMX standard today, either by providing import/export utilities or by using the TMX format directly like OmegaT (http://www.omegat.org). A tool completely dedicated to creating and maintaining translation memory databases in the TMX format is the Heartsome TMX Editor (http://www.heartsome.net).
- Compatibility with your clients requirements: cases could arise where a client wants the localizer to use a proprietary tool or a certain commercial tool because he wants to re-use translation memories he has or he wants to obtain the new translation memory in a proprietary format.

Meanwhile there quite a few translation memory systems for Web localization on the market, here are a few examples:

Déjà Vu: http://www.atril.com;

SDL/Trados: http://www.sdl.com;

STAR Transit: http://www.star-group.net;

Webbudget: http://www.aquino.com;

Heartsome TMX Editor: http://www.heartsome.net;

Foreigndesk: http://sourceforge.net/projects/foreigndesk/;

OmegaT:http://www.omegat.org.

Translation Memory rationalises the translation of repetitive texts, but this is not the only advantage. Even more important for the localization of websites is the possibility to automatically detect changes and updates in websites. Websites are constantly updated and the task of identifying new content can be very time consuming. With a translation memory from the previouly translated version of the website, the system matches the content of the new version against the translated content of the old version. As a result, all the old text will be replaced by the saved translations in the database and only new text will have to be translated manually.

All the aforementioned tools are a wise choice for the localization of static Web pages, a process which could be described as repurposing existing information resources for the multilingual Web. It is, however, becoming more and more a task of developing entirely new types of interactive information resources which can be maintained and delivered multilingually. Initially, Content Management Systems CMS have been developed to maintain extensive websites. These tools save all information in a database and produce HTML output on the fly using style sheets the moment a visitor to a site requests a Web page. However, traditional CMS do not support multiple languages. Only recently have a new type of CMS come forward with multilingual support and incorporated terminology and translation memory tools. These applications are called Global Content Management Systems (GMS). The additional benefits for these kind of systems, besides the advantages already mentioned for translation memory systems, are the decentralised management of websites with different access rights for central and local Web developers/content providers, less manual work and faster turn around times, support for the localization workflow and also translation vendor management where data about the costs of translations will be stored centrally and can be accessed immediately for statistical purposes by language, vendor or text type. Examples for GMS are:

Idiom WorldServer: http://www.idiominc.com;

Transware: http://www.transware.com;

SDL GIM Platform: http://www.sdl.com.

Once the website has been adapted and translated, all files should be verified and tested. Inevitably, changes have been made during the localization process which can lead to errors in the target files. Changes in character and language encoding or the translation of text elements may cause problems regarding the layout of the target files or functional problems so that the target file will not display or work properly.

Thus, the first thing to assess after translation is whether the localized site functions in exactly the same way as the source site. All the hyperlinks must work properly, locale-specific formats must have been adapted like currency, dates, addresses, etc. A special problem with interactive websites can be Web forms, where many locale-specific formats will be used, which may cause text length and encoding problems.

Having tested the functionality of the localized site, all the language-related aspects must be checked. Have all the source text elements actually been translated? Are the target text elements of a satisfactory linguistic quality?

And finally, the appearance of the localized website must be checked. Is all the text visible? Has text embedded in graphics been translated? Is the formatting correct?, etc. Different browsers display Web pages differently, so the visual aspects must be tested on a number of popular browsers on different OS platforms (e.g. Internet Explorer, Mozilla, Firefox, Opera, Konqueror, Apple Safari).

The easiest way to test a website is to click through all the pages in turn but this may be a very time consuming and boring task to perform. There are tools which will do many different testing procedures automatically, among them SDL HtmlQA (http://www.sdl.com/htmlqa). One of the above mentioned text based HTML editors will be beneficial when correcting errors and performing basic testing of the code.

After the testing phase, the site is uploaded to the web server. The next task is to make the new website visible to visitors who come from this locale and speak the language of the localized site. This is done through the global gateway integrated into the website by the localizer. This is, however, not enough to attract foreign visitors. The new localized Web page must be promoted on the Web, a task which has to be performed in the foreign language. Apart from registering the site with international indexes and search engines, it is advisable to promote it with country or language-specific indexes and search engines. Announcements in local newsgroups and forums, link exchange agreements with strategic websites, as well as traditional press releases are another possibility. And finally, country-specific domain names can

be registered with the local Internet authority, such as www.yourname.fr for the French version, www.yourname.it for the Italian version, etc.

New opportunities for translators emerge with these services (language-specific promotion of websites, building a global gateway) which require language fluency and cultural knowledge. To take advantage of these new opportunities, new skills and competences must be integrated into translator training curricula.

Granted that a translator has acquired the necessary language competence and a broad base of general knowledge which enables him or her to gain familiarity with a specific field rather quickly, as well as translation skills and a general awareness of cultural issues, basic knowledge about terminology management and translation memories, the following additional requirements have also been identified:

- computer applications: basic knowledge of operating systems and platforms as well as standard software types. Experience with programming languages and compilers for software localization could also be specified;
- HTML- and XML-authoring: a must for all types of localization as the XML standard gains broad acceptability;
- language-specific technical knowledge: the technical infrastructure of the Web with regard to language and character encoding;
- translation tools for localization purposes: neither type of localization is strictly feasible without such tools. Thus, a vital requirement;
- basic knowledge of project management, quality assurance and business models applied to localization. A translation student or for that matter every person emanating from a strictly linguistic or philological context should bear in mind that localization is a business orientated activity which requires at least some insight into business practices and procedures.

A good localizer also has a basic knowledge of Machine Translation, Controlled Languages, writing for an international audience and International Marketing. Trained localizers must be able to provide consultancy services to companies who want to operate internationally. In doing so, they must be able to communicate withmanagers in companies using their language regarding costs and benefits of localization projects.

Localization can be a great opportunity for translators if they have the ambition to go beyond translation, to acquire the necessary skills and to take responsibility for the whole process. If not, or if translators insist on doing only

text translation without taking into account the larger picture, their role will be reduced to mere free-lance contributors for localization professionals.

This article is only a basic overview of localization principles and tools. Much more can be added about localization if we were to delve into greater detail.. What has not been touched upon is, for instance, the workflow of localization projects and project management, the people involved, the differences between in-house localization and outsourcing, writing for the Web and an international audience, or the localization of graphics or images.

As an introduction, this overview should awaken interest in localization and stimulate further reading about this fascinating activity.

4 References

Ballstaedt, Steffen-Peter (2003): "Anforderungen an die Gestaltung elektronischer Kommunikate: Texten und Visualisieren". Fachsprache. Volume 25. 1-2. Vienna: Braumüller. 6-13.

Bruner Rick & Harden Leland & Heyman, Bob (2000): *NetResults.2 Best Practices for Web Marketing.* Indianapolis: New Riders.

Bungarten, Theo (ed) (1999): *Sprache und Kultur in der interkulturellen Marketing kommunikation.* Toestedt: Attikon.

Cheng, Susan (2000): "Globalizing an e-Commerce website". In: Sprung, Robert C. (ed) (2000): *Translating into success. Cutting-edge strategies for going multilingual in a global age*. Amsterdam: John Benjamins. 29-42.

Crystal, David (2001): *Language and the Internet.* Cambridge: CambridgeUniversity Press.

DePalma, Donald (2002): *Business without Borders. A Strategic Guide to Global Marketing.* Wiley and Sons.

Dreikorn, Johannes (2001): "Der Reiz des Besonderen. Tipps zum Schreiben von online-Texten. Technische Kommunikation". Volume 5/01. 23-31.

Dunne, Keiran J. (ed.) (2006): *Perspectives on Localization*. ATA Scholarly Monograph Series XIII. - Amsterdam/Philadelphia: John Benjamins,

Esselink, Bert (2000): *A Practical Guide to Localization.* Amsterdam: John Benjamins.

Esselink, Bert (2001): "Web Design: Going Native". Language International. Volume 2/2001. 16-18.

Freigang, Karl-Heinz. (1996): "Zum Stellenwert von Lokalisierungsprojekten in der Übersetzerausbildung". In: Fleischmann, E. et al. (eds): *Translationsdidaktik. Grundfragen der Übersetzungswissenschaft*. Tübingen: Gunter Narr. 122-132.

Fulford, Heather (2000):"Monolingual or multilingual websites? An exploratory study of UK SMEs". In: Schmitz, Klaus-Dirk (ed): *Sprachtechnologie für eine dynamische Wirtschaft im Medienzeitalter*. Proceedings of the XXVI. Annual

conference on Sprache und Wirtschaft. 23.-25. November 2000. Wien: TermNet. 39- 50.

Gondouin, Daniel (2007): "Localization de sites web: constraintes et enjeux". In: Lavault-Olléon, Elisabeth (ed): *Traductrion spécialisée: pratiques, théories, formations*. Bern: Peter Lang, 179-188.

Grebenstein, Kay & Schumann, Christian-Andreas & Tittmann, Claudia; Tsering Gonpo & Weber, Jana & Hennig, Jörg & Tjarks-Sobhani, Marita (eds) (2002): *Lokalisierung von Technischer Kommunikation*. Lübeck: Schmidt-Römhild.

Harris, John & McCormack, Ryan (2000): Sapient Report: *Translation is not enough. Considerations for global Internet development.* Sapient Report.

Langewis, Chris (1998): Avoiding "Weblock". Language International. Volume 10.4. 22-24.

Lockwood, Rose (2000): Brand, Have & Travel, Will. Language International. Volume 12/2, 14-16.

Lockwood, Rose (2000): Have Brand & Will Travel. Language International. #12 volume 2. 14-16.

Lommel, Arle; Ray, Rebecca (2007): *The Globalization Industry Primer. An introduction to preparing your business and products for success in international markets*. The Localization Industry Standards Association.

Loughman, Liza (2000): "Style Online. Language International". Volume 2/2000. 33-34.

McDonough, Julie (2006): "Hiding Difference: On the Localization of Websites". *The Translator*, vol 12, 85-103.

Multilingual Computing (2002): *Multilingual Content Management*. The annual Guide from Multilingual Computing.

Nielsen, Jakob & Del Galdo, Elisa M. (1996): *International User Interfaces*. John Wiley & Sons.

Oldenburg, Jan (2000): "Translating the Web: Into the Future". Translation Journal. Volume 4. 1/2000. http://www.accurapid.com/journal/index.html .

Pearrow, Mark (2000): *Website Usability Handbook*. Charles River Media.

Pym, Anthony (2004): *The Moving Text. Localization, translation and distribution*. Amsterdam: John Benjamins.

Reineke, Detlev, Schmitz, Klaus-Dirk (eds.) (2005): *Einführung in die Softwarelokalisierung.* Tübingen: Gunter Narr Verlag,

Sánchez-Mesa Martínez, Domingo (2001): "Hypertext and Cyberspace: New Challenges to Translation Studies". In: Gambier, Yves; Gottlieb, Henrik. (eds) (2001*): (Multi)media translation: Concepts, Practices and Research.* Amsterdam: John Benjamins. 35-43.

Savourel, Yves (2001): *XML. Internationalization and Localization*. Indianapolis: Sama Publishing.

Schewe, Theo (2001): "Multilingual Communication in the Global Network Economy". In: Eschenbach, Jutta & Schewe, Theo (eds) (2001): *Über Grenzen gehen - Kommunikation zwischen Kulturen und Unternehmen*. Halden/ Norwegen: Hogskolen i Ostfold. 195-209.

Schlutter, Stefanie. (1996): *Sprachliche Gestaltung und internationalizierung von*

Benutzeroberflächen. Erläutert an Beispielen aus dem Englischen, Deutschen und Italienischen. Saarbrücken: Saarbrückner Studien zur Sprachdaten verarbeitung.

Schweibenz, Werner; Thissen, Frank (2003): *Qualität im Web. Benutzerfreundliche Webseiten durch Usability Evaluation*. Berlin, Heidelberg: Springer.

Singh, Nitish; Pereira, Arun (2005): The Culturally Customized Web Site. Customizing Web Sites for the Global Marketplace. - Burlington MA, USA: Elsevier, Butterworth Heinemann,

Van der Geest, Thea M. (1995): *Website Design is Communication Design.* Amsterdam: John Benjamins.

van der Meer, Jaap (2002): "Impact of translation Web services". In: Localization Research Center (LRC) (ed) (2002): *Localization Focus*. #1, volume 2. Limerick: 9-11.

Van der Meer, Jaap (2002): "Impact of translation Web services". In: Localization Research Center (LRC) (ed): *Localization Focus.* Volume 1. 2. Limerick: 9-11.

Wallace, Patricia (1999): *The Psychology of the Internet*. Cambridge: Cambridge University Press.

Weiss, Andreas; Wieden, Wilfried (2000): "Die Herstellung mehrsprachiger Informations- und Wissensressourcen in Unternehmen". In: Schmitz, Klaus-Dirk (ed): *Sprachtechnologie für eine dynamische Wirtschaft im Medienzeitalter*. Proceedings of the XXVI. Annual conference on Sprache und Wirtschaft. 23.-25. November 2000. Wien: TermNet. 25-38.

Wolle, Jürg (2003): „Globale IT-Infastruktur für die interkulturelle Kommunikation". In: Bleich, Susanne & Jia, Wenjian & Schneider, Franz (eds) (2003): *Kommunikation in der globalen Wirtschaft*. Frankfurt/Main: Peter Lang.

Wright, Sue Ellen (2004): "Localization Competence for Translation and Project Management". In: Fleischmann, Eberhard; Schmitt, Peter A.; Wotjak, Gerd (eds) (2004): *Translationskompetenz.* Tübingen: Stauffenburg. 581-595.

Yunker, John (2002): *Beyond Borders. Web Globalization Strategies.* Indiananpolis: New Riders Publishing.

Yunker, John (2002): *Beyond Borders. Web Globalization Strategies*. Indinanpolis: New Riders Publishing.

Yunker, John (2003): "Building a Global Web Site". In: *Multilingual Computing: The Global Web Guid*e. #55 volume 14. 4-9.

Zervaki, Thei (2002): *Globalize, Localize, Translate.* 1StBooks.

Zimmermann, Hansjörg (2001): *Web-Usability - Pflicht oder Kür. technische Kommunikation*. Volume 5/01. 17-22.

Zschau; Traub; Zahradka (2002): *Web Content Management. Websites professionell planen und betreiben.* Bonn: Galileo Business.