No. 1, Volume 1
July 1997

Translation Journal

Facets of Software Localization

A Translator's View

by Per N. Dohler

Per N. Dohler has been an independent translator between English and German since 1984. He specializes in information technology, software localization, and the Internet, as well as dentistry, medicine, medical technology, and pharmacology. He lives and works in Barendorf in Northern Germany, where he and his wife Thea run their small service company Triacom serving customers in the U.S., Germany, and a number of other countries.

Prompted by a mixed ethnic background and early interest in language and languages at large, Per studied linguistics, with a minor in American Studies and History, at UC Berkeley and at the University of Hamburg/Germany, where he received his M.A. degree in 1984. Additional university-level and practical computer science and dental/medical training helped him get started as a technical and dental/medical translator.

Per was one of the first German translators to take his activities to Cyberspace, which he considers to be his most important step away from the isolation that so often engulfs freelances. He is an ardent fan of the translators' "cyber-cafeteria" on CompuServe, FLEFO. Per is a corresponding member of the ATA, whose publications he occasionally contributes to, and a member of the North German translators' association, ADÜ Nord, and of the Swedish Association of Professional Translators, SFÖ.

Per Dohler can be reached at

Contents

Introduction

That information technology has revolutionized the translator's working environment is a fact so obvious that it no longer even bears mentioning. For the vast majority of translators and their clients, computers have long replaced typewriters and reams of paper. Modems and e-mail have replaced manila envelopes, mass storage devices have replaced drawers full of folders, CD-ROMs supplement dictionaries and encyclopedias, and more recently the Internet and its resources more and more often save us a trip to the library, while various online communities have brought many translators out of their isolation.

But information technology not only offers us tools. It is itself a field in which more and more translation work is actually performed. This is certainly true of other fields like marketing materials, packaging materials, advertising copy, and manuals. But in the case of information technology products, it is frequently the products themselves that need to be translated. Whenever a program or process displays a word or a phrase on the screen, this means potential work for a translator or many translators.

In an article on a recent market study by Ovum Reports, *The Expanding World of Globalization* (Language International **9.2** 1997), Liza Loughman reports that the total worldwide revenues for globalization-related services will increase from USD 2.8

per@triacom.com

Home

- **From the Editor/Webmaster**
  What Is the Translation Journal
  by Gabe Bokor
- **Translator Profiles**
  Saga of a Scientific Translator
  by Cathy Flick, Ph.D.
- **Feature Article**
  Facets of Software Localization
  by Per N. Dohler
- **Science & Technology**
  A Translator's Guide to Organic Chemical Nomenclature
  by Chester E. Claff, Jr., Ph.D.
- **Caught in the Web**
  Web Surfing for Fun and Profit
  by Cathy Flick, Ph.D.
  Translators' On-Line Resources
  by Gabe Bokor
- **Translators' Tools**
  Electronic File Transfer and Conversion
  by Gabe Bokor
- **Translators' Events**
- **Letters to the Editor**
- **Call for Papers**

billion in 1997 to USD 6.2 billion in 2000. Ovum forecasts that Japanese will account for 31% of all localization service revenues in 2000, up from 18% in 1994, with German falling from 18% in 1994 to 12% in 2000 — which in absolute figures still represents an increase from USD 352 million in 1995 to USD 639 million in 2000 and continues to put German in second place. Relatively smaller slices, but a larger pie. A *very* large pie.

International users

International users of computer software have come to expect their software to "talk" to them in their own language. This is not only a matter of convenience or of national pride, but a matter of productivity. Users who understand a product fully will be more skilled in handling it and avoid mistakes. So they will prefer applications in their language and adapted to their cultural environment.

The international market has become very important for software manufacturers. Many U.S. manufacturers derive a large percentage of their revenues and profits from international sales. Competition is fierce, and those companies who are best at anticipating the needs and preferences of their international users will benefit most.

With the advent of the Web, it has become vital for many manufacturers and distributors to maintain a world-wide presence. For this presence to be effective, it is not enough to simply be there. It is equally important to make sure the message gets across, and once again, that means: translation.

Localization and internationalization

Internationalization , also referred to as *globalization*, is the process of designing or redesigning a product so that it can be localized with minimal changes.

Localization is the process of adapting a product, in our context a software program, to a specific locale, i.e., to its language, standards and cultural norms as well as to the needs and expectations of a specific target market. A properly localized product also meets all the legal requirements in force in the user's region.

Users can interact with a successfully localized product in their own language and in a setting that feels natural to them. This means, for instance, that all messages are in their own language, that they can input names, addresses, dates, and other data in the same way they would write them down on paper, that they can freely use their standard keyboard characters wherever an entry can be made, and that any error messages are comprehensible to them rather than representing American geekspeak. Localization comprises the program itself and any online or printed documentation.

Typically, localization is undertaken from English into other languages, as most software continues to be developed in the U.S., and even some of the software developed elsewhere is originally developed in English.

Localization is performed by taking the source code for a product developed for one country and modifying the source code and product to satisfy the needs of other countries. Often teams of developers in different countries are needed to adapt products. If the original product is not built with a view toward being localized, this can be a very expensive and time-consuming process. There is the direct cost of multiple development teams modifying the source code of the original product. This process also produces multiple code bases, which makes future development and maintenance more complex.

Plenty of books on localization and internationalization have been written that are addressed to software engineers, web site designers, and others. Unfortunately, the role of the translator in the localization process has, to my knowledge, not been given any specific attention.

Introductions to localization and internationalization

International Consulting has an instructive overview over localization and internationalization, including many technical and business aspects.

ILE in Boulder, Colorado, has compiled an interesting document in Adobe Acrobat format that lists a lot of interesting facts about everything that should be taken care of before we translators see the text to be translated. “This informative guide to product internationalization explores the ways you can develop your software, on-line help systems, documentation and audio that will make localization as straight-forward as possible. This overview can serve ... as a primer for anyone involved with localization.” (Note: This document is protected and cannot be printed, only viewed.)

How localization is usually organized

Translation companies are often chosen to provide localization. They eliminate the need for companies to maintain expensive in-house personnel. In addition, they offer a broader range of services, technical expertise, and flexibility than freelance translators and also eliminate the need for the company to manage a large pool of freelancers.

Translators who work on localization projects are often part of a large and distributed team. Its complex, multi-layer organization poses new challenges to those independent translators who usually work on neatly delimited projects which they have sole responsibility for. Working on localization projects is more akin to working in the translation department of a larger company, only that the translator does not enjoy many of the benefits of on-the-job training and rapid information flow which characterizes an in-house setting. In actual reality, translators are expected to have complete command of the tools required for software localization, know the market and a lot of products,

know their own position in the process and workflow, and understand the constraints involved in ever shorter production cycles — all on their own.

It is for this same reason that most translators working in localization do so through intermediary translation bureaus, which are often specialized localization or "language engineering" companies. The majority of software manufacturers now outsource their localization activities and maintain only a skeleton team for interfacing with their supplier. These companies, which can be responsible for as many as 10 to 20 languages or even more, will pass the documents on to subsidiary partners in the target countries or directly to freelance translators, either in the U.S. or overseas. These will typically be some in-house staff, but most of the actual translation work will probably be done by freelance translators. It is not even uncommon for corporations to outsource the localization of their documentation to multiple translation companies.

Project scheduling

Ideally — marketing managers say — the localized versions will be completed at the same time the original version goes to market. Ideally — translators say — localization begins when the software and documentation are finalized. It is immediately apparent that these are irreconcilable demands. In actual practice, therefore, it is attempted to follow a stringent project schedule where localization is only one step behind development.

Unfortunately, development does not follow a linear schedule; features are developed concurrently with the documentation, and the latter is written, rewritten, and re-rewritten all the time. What translators get to translate is not infrequently a not-quite-final version of the ultimate product. It would seem that programs should be developed from specifications, and documentation should be developed from, and document, the completed program. We have to know how we translate "Press Enter to continue" in the software itself before we can tell the presumptive user what the program says when we describe the various steps in the online help.

Furthermore, translation follows a cycle of comprehension. As translators are virtually never given more than the briefest of summaries of the product specifications (if that!), when it is time to translate the software, we must do a lot of guessing as to what function actually does what and consequently what to call it. Often enough, though, the purpose of a function, dialog box, or command will become apparent to the translator only when he or she finally gets to the help file that explains it. In this case the translator may have to go back and change the term that was used in the first version of the software translation; and it may not even be the same person doing the software and the help, which complicates matters. Too often the software is already

“frozen”, that is, ready for production with no additional changes possible by the time enlightenment comes.

One of the attributes that characterize successful and sought-after software translators is precisely the ability to guess correctly about what a given software string or dialog box or function actually does, so as to avoid having to loop back wherever possible. It is here that experience plays an important role. All software projects should include at least some “old hands.” Software translations done only by inexperienced translators usually give away the fact that the translators did not have a clue. Their experienced colleagues may not have had a clue either, but if they have a feel for their work, often no one will notice.

It may irk those of us for whom quality exclusively means the linguistic quality of the final product, but: there are other aspects to quality, and they do tend to come to the fore in software localization. To the client, sometimes a linguistically mediocre translation (craftsperson translation) delivered on time is much more valuable than a perfect one (artist translation) that is three days late. Sometimes errors and inelegancies can be caught further down the line, in the various editing steps that (should) follow translation, but a file that is delayed three days may hold up an entire project and cost the client thousands of dollars in lost time-to-market.

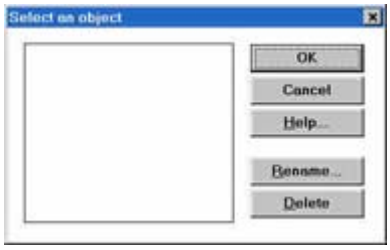
Components of a software localization project

When localizing software, we are dealing with a number of different file types. The following subsections will present a few selected aspects of frequently encountered file types for the various Windows platforms. I apologize to all prospective Mac localizers, but I have no idea how these things work on a Mac. I am therefore a good example of someone who should not undertake localization for the Macintosh platform. If you do not know either and cannot find out, neither should you...

Resource files

In localization, the visible part of the software, the user interface, is usually simply called *software* per se. In properly developed software, the texts the user sees are included in separate files, the so-called *resource files*. They contain everything the user is likely to see and that is not created when the program actually runs *(at runtime)*: menus, dialog boxes, error messages, bitmaps, cursor shapes, and so on. Help files are resources, too, but they are usually treated differently.

The following is an example of a very simple Windows dialog box and its representation in a resource file (in a Windows environment this is called an *.RC file*, where .RC stands for resource compiler).




```
IDD_SELECT DIALOG DISCARDABLE 0, 0, 167, 106
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION |
WS_SYSMENU
CAPTION "Select an object"
FONT 8, "MS Sans Serif"
BEGIN
DEFPUSHBUTTON "OK",IDOK,108,8,50,14
PUSHBUTTON "Cancel",IDCANCEL,108,24,50,14
LISTBOX IDC_TOOLBAR_NAMES,8,8,92,88,LBS_SORT |
LBS_NOINTEGRALHEIGHT | WS_VSCROLL | WS_TABSTOP
PUSHBUTTON "&Help...",IDHELP,108,40,50,14
PUSHBUTTON "&Rename...",IDD_RENAME,108,64,50,14
PUSHBUTTON "&Delete",IDD_DELETE,108,80,50,14
END
```

You will recognize the various elements. The texts marked in red here (in actual files they are not marked in color) are the texts to translate. You must not translate anything else, including the “MS Sans Serif” which deceptively enough is also enclosed in quotes but is the name of the font used in the dialog box. The “&” precedes a letter (shortcut) that is underlined in the actual box. The “...” must be preserved as it tells the user that another dialog box follows when that button is clicked. — Everything else is information that tells the compiler what to put where in the box. Chances are that the boxes need to be resized because the translation comes out longer than the original. This is not something that is usually included in the translation part of localization, although some translators possess the requisite tools to do so and offer this as a value-added service.

Error messages are more straightforward:

```
STRINGTABLE DISCARDABLE
BEGIN
  IDS_WINEXEC_ERROR0 "The following error occurred:\n\nSystem was out
of memory, executable file was corrupt, or relocations were invalid."
  IDS_WINEXEC_ERROR2 "The following error occurred:\n\nFile was not
found."
  IDS_WINEXEC_ERROR3 "The following error occurred:\n\nPath was not
found."
  IDS_WINEXEC_ERROR5 "The following error occurred:\n\nAttempt was
made to dynamically link to a task, or there was a sharing or network-protection
error."
  IDS_WINEXEC_ERROR6 "The following error occurred:\n\nLibrary
required separate data segments for each task."
  IDS_WINEXEC_ERROR8 "The following error occurred:\n\nThere was
insufficient memory to start the application."
  IDS_WINEXEC_ERROR10 "The following error occurred:\n\nWindows
version was incorrect."
END
```

The “\n" stands for a newline character. i.e. this is where a new line will begin in the actual text.

Help files The source files for Windows help files are usually .RTF files. Windows help compilers take this file and convert it to the familiar hypertext formats you see when pressing F1 in a Windows program.

The Hypertext organization (hypertext is text in which you can click on certain hot spots that will take you somewhere else, usually to a subtopic relevant to what you want to know) is represented in .RTF files in the form of single- and double-underlined texts, hidden text, and footnotes.

The **title** must be translated. All the normal text must be translated as well.

The **double-underlined parts** represent hyperlinks. In the .RTF file they are followed by **hidden text** that tells the compiler what page to go to when this link is clicked. The double-underlined part must be translated. The hidden text must remain untouched.

The **single-underlined parts** represent links to a help file glossary. They, too, are followed by **hidden text** that tells the compiler what entry in the glossary to show in a small window when this link is clicked. The single-underlined part must be translated. The hidden text must remain untouched.

The **colors** guide users. They should be retained for the translated text.

The help page starts with several **footnotes** before the title. These footnotes must never be removed. As can be seen at the bottom of the example, there are different types of footnotes.

"#" footnotes are never translated. They represent hyperlink targets. If a different page wants to link to this page, it would do so by placing “Example_Application_Welcome_Menu" in hidden text behind a hyperlink. Any changes here destroy the help file.

"$" footnotes are titles as they appear in the Help content list. These footnotes must be translated. It is recommended that this text be the same as the help page title.

"k" footnotes are entries in the Help index list. They must be translated, but with the same caveats that hold for index entries in documents.

"+" footnotes are internal compiler information and must not be translated.

Some help compilers support additional footnote classes.

If you now remember that graphic elements are treated like letters by the program with which you work on the .RTF file (probably MS Word), i.e. they can be underlined and used as hyperlinks, you know most of what you have to know to confidently start translating help files without destroying them.

Readme files

Readme files usually contain late-breaking news that did not make it into the documentation, additional setup information, or corrections and additions to the manual. Usually these are plain text files. If you use your word processor to work on these, you must remember not to save them in the word processor's format but as plain text, using the correct character set (e.g. *Text only* or *MS-DOS Text*). Otherwise you should try to break lines so the result looks pleasing. Readme files are pretty much the only place where hard returns at the end of a line and formatting text with spaces are ever allowed!

Screen shots and bitmaps

Some of the elements of a user interface may be present in the form of image files in which the information is present as pixels and cannot be edited in the word processor or editor. These bitmaps will be recreated with an image-processing program, and the translator's job is to furnish the translations to be included. In simpler cases the translator may be asked to overwrite graphics files. This should be billed as a value-added service.

Examples of bitmaps are the *splash screens*, the colorful rectangular messages that often greet users when a program is started.

Word processing and DTP files

Documentation (manual) files are often presented to translators in heavily formatted form, with many hallmarks of document automation, with the expectation that translators overwrite them; this of course presupposes that translators know how to use their tools. You will find more details in the section Writing tools and their limits.

Client-prepared file formats

A number of mostly larger localization companies take the resources and files to be translated and produce text files from them. These text files are the files that the translators get to work on.

These companies will send you, along with the text to be translated, instructions about how to use their formats, where you should write your translations, what elements you must not change, and the meaning of various tags and hints you may encounter in the files you are working on.

One advantage of this procedure is that translators can be

provided with one set of instructions for a variety of different source file formats. A second advantage is that you can translate files for operating system environments other than your own. If the target system is a system that few people have installed, this may be the only way to work on the files to be translated.

Incidental files
There are a variety of other file types: for packaging, warranty cards, and other purposes; these are too diverse to be listed here. They do not usually present any specific problems.

Basic references

Bilingual dictionaries
In many other fields, one of the most important and most frequently consulted references are bilingual dictionaries, often in paper form, sometimes on CD-ROM. Bilingual dictionaries always have their limitations, but working in a field that does not have these references makes you appreciate them more. Bilingual dictionaries in this field are often partly obsolete as soon as they appear. Not only do new words come up and old words fall into disuse, but the way existing words are used will also change rapidly. Remember what a Winchester drive is?

Web sites and computer magazines
The vast majority of original published material, digital or otherwise, is written in English today. This is why localization is so often a one-way street from English to other languages. At the same time, we are often faced with the necessity of coining the words we need ourselves, or making an executive decision between several alternatives, none of which has yet become established.

Carefully written and edited computer magazines in the target language are a very useful source of computer terminology. If you don’t live in the country of your target language, you should still be able to read them on the Web. Supposedly, computer magazines are a little less ephemeral than web sites, but a web site may be more on top of late-breaking news and is also easier to search.

However, the approach of jumping on the bandwagon of preliminary journalese “translations” as they are usually found in the first excited account of a new program, process, or device and of sanctioning them by copying is, in my opinion, a serious mistake. It is precisely in the computer field that we can do something for our readers by choosing our words conscientiously and prudently.

Those of us who are confronted with English and with computers and computer-related texts in our own languages every day might tend to think that our languages are just inundated with English loan words. On the whole, however, this impression is misleading. Of course there are differences in

degree between the various European languages (I cannot speak about any others) but it is probably true everywhere that modern sports, some technical and scientific fields, computers, and computer-related topics, and certain aspects of cultural — particularly those that address a more youthful audience — tend to be full of "cool" loan words. But if you read newspaper and magazine articles that do not happen to concentrate on these topics (and a good number of those that do), chances are that the proclaimed "inundation" of our language with Anglicisms is fortunately still more or less limited to what is reasonable.

So with these caveats, computer magazines and carefully edited web sites (too many are, unfortunately, hastily thrown together by people with not much respect for their readers) in the target language are probably the best sources of computer terminology.

General monolingual computer-related glossaries

Fortunately, it is not usually a problem finding out what a technical term or abbreviation means. There are too many glossaries to list here individually, but here are some good starting points for finding them:

The University of Wasa, Finland, has an impressive collection of computer glossary links for Chinese, Croatian, Dutch, English, Finnish, French, German, Greek, Hungarian, Italian, Norwegian, Portuguese, Spanish, Swedish, Russian, Vietnamese plus some multilingual ones: Term-Online, Online Special Language Glossaries — Computing, Internet.

The Translator's Home Companion also lists a number of interesting starting points for terminology searches.

A by now famous source for finding out the meanings of obscure computer-related terms is the Jargon File, the electronic Version of *The New Hacker's Dictionary*.

A well-known and well-kept listing is BABEL: Glossary of Computer Abbreviations and Acronyms.

Most of the time when we translate software and related documentation, we are not concerned with the fine points of computer terminology, as the software is a tool that is supposed to help the user get some specific work done. So the translator's requisite expertise must be in the field in which the software is supposed to do its job. All the computer expertise in the world still leaves the translator clueless when a help file contains pages upon pages of dense accounting jargon. Translators must not fall into the trap of accepting any software-related assignment just because they know a lot about computers. The results can be disastrous.

Your computer

Your most important tool is also your most important reference: your computer itself. I am going on the assumption that you are working on the same platform that the software product is for — which is the recommended procedure. In that case it makes eminent sense to use the localized version of your operating system in the target language. One of the most important aspects of a good user interface is that it has to be consistent. So it is

always a good idea to have everything that your translation is supposed to be consistent *with* at your fingertips.

Industry-standard glossaries

For the same reason, user interface consistency, it is important that all references to the operating system and other industry standard programs are correct and use the same terminology. If a user, for instance, is directed to click on an icon in the Control Panel of the operating system employed, but the name of that icon is translated by a term that is different from what the actual icon is called, the instruction becomes useless, and the user is led astray.

For some manufacturers there are glossaries that contain every menu item, every message, every dialog box entry that appears in their products.

Microsoft has made the glossaries for its products (operating systems and applications) available. They can be downloaded with any Internet browser.

Novell also offers its glossaries.

There is no similar resource for Unix, and it is also unlikely that there will ever be a general one, since there are too many different flavors of Unix, a fact that is compounded by localization.

To the best of my knowledge, and judging from translators' repeated desperate pleas, Apple has not made its glossaries available.

Client glossaries

Client glossaries, if they exist, are usually straight term lists, often culled from earlier projects. Contextual information is rarely given.

Like so many things in a translator's life, the client glossary situation is often best described by the expression "feast or famine". Sometimes you get nothing at all. But some clients will try to give you a printed glossary, sometimes hundreds of pages long. Working with these is hopeless. You cannot find what you need in a reasonable amount of time. Insist that the glossary be made available in electronic format. But even electronic glossaries are sometimes too large to be useful or fractured into too many files.

The most useful client glossaries come in electronic format, as tables that can be read, searched, sorted and otherwise operated on with a standard word processor or spreadsheet program. There should be columns for source and target terms, domains, syntactic information where needed, and — very importantly — where that term occurs in the project. Tabular output of terminology management systems would seem to be ideally suited for this purpose.

If clients have a problem maintaining proper glossaries, why not convince them that they can save a lot of time in the long run, and offer to do it for them? Although terminology management tools are frequently discussed among translators, most of us do not yet use any organized form of terminology management.

What better way of learning how to use them, or not to use them, than in the context of a billable job?

Client style guides
Most localization companies issue their own style guides for the various languages they handle. Often these appear to be written for non-native speakers, as they have a tendency to belabor points that a native speaker knows without having been told.

However, a well-designed style guide will help guide the translator in making stylistic decisions, such as whether to put commands in infinitive or imperative form, how to phrase chapter headings so they are stylistically congruent, and other things. Of course a style guide cannot substitute for your translator's common sense and writing skills, but it may be a great help in tying the different parts of a project together and smooth out the stylistic differences between different translators.

Project glossaries
The minimum you will need to know to be able to translate help files or manuals is what the software texts in the actual product are. Normally they should be available to you in tabular format; if you don't have one, ask for one, maybe the project manager forgot or did not know.

Sometimes even this minimum does not exist. In that case you can (under Windows) at least put all target terms into one file to search by concatenating them. (Hush — this is best done in DOS: Open a DOS window, go to where the .RC files are, type `COPY *.RC ALL.DOC`, and open ALL.DOC in your word processor. Voilà.)

But seriously, how are you going to translate a manual in which every other sentence literally quotes program messages which you don't know just how they are going to appear in the target language? Surely your customer did not think that you will arrive at the exact same wording as your colleague who translated (or will translate) the user interface because there is always only one correct translation? On second thought, maybe...

Writing tools and their limits

Many translators consider themselves wordsmiths and definitely not layout tinkerers. In translating help files and manuals these days, however, an understanding of word processors that goes beyond mere survival level is usually a must. The PC is not a typewriter! (There is even a book by this name, by Robin Williams.) There should be, however, limits to what should be expected of you "for free."

Word processing
You will most probably be given files with already formatted texts which you will be expected to overwrite, preserving the original format to a tee. Normally there simply is no time for

someone to copy your foreign-language text-only translation into these highly complex files, especially since this would require target-language expertise. Make sure you understand styles, hidden text, fields, tables, graphics, and other objects that will crop up in your texts. If you are open to this way of working, you will find that it has its rewards. You will find that seeing near-final format all the time and having the old and the new text in the same file and thus right in front of you is a relief to the eyes. You can use search and replace (if your target language is amenable to that, there are a few tricks to doing this well.)

I do not believe that overwriting a localization help or manual file in a word processor goes beyond what can be expected from a translator knowing his tools today. Nor do I, frankly, believe that this is worth a surcharge, but should be figured into your general rate for localization.

You might, however, try to negotiate a special (fixed or time-based) price for creating, generating, and checking a foreign-language index. It is usually not enough to simply translate the English entries (they are often hidden inside the text and — caveat vendor! — usually not counted by counting routines). You can check the translation of the entries for conformity with general usage in the target language, understandability, and consistency by generating the index and going back and forth. This should not be left to the engineers who don't speak the target language, so you should offer to do this. It is not that difficult to learn how to generate index entries and indexes in most word processors.

DTP and engineering tools

These are *not* the translator's responsibility.

Creating a layout, whether in a word processor or in a DTP program such as Framemaker or QuarkXPress, requires expert skills. If you are an expert, go ahead and offer this value-added service. But do charge extra for it.

Overwriting files in a DTP program is something that we are asked to do more and more frequently. What our clients often forget is that DTP programs are not made for writing, they are made for layouting. (Re)creating text in a DTP program can slow you down considerably, and these programs are also expensive and take time to learn. If you want to invest and offer this value-added service, do it by all means, but do remember that the slower speed will reduce your earnings per hour unless you consider this in your price.

Testing files is engineering work. I would go so far as to suggest that checking existing document automation features (index, table of contents) that you overwrote in your regular word processor is something you should be able and willing to do. (But remember that creating a proper index — as opposed to simply translating an English one, which usually yields disastrous results — is a lot of work and should usually not be billed at a word rate.) When you overwrite an HTML file, I think asking you to check the result in a browser to see whether you inadvertently destroyed any HTML tags is not asking too much.

But running help compilers, resizing dialog boxes, following links across files, or testing the runtime behavior of files is not translation. If you offer this, again, this is a special value-added service to be billed for accordingly.

Computer-assisted translation (CAT)

In software localization, translation companies sometimes try to use engineering methods to cut costs, ensure consistency, and assist the translator, in short: integrate aspects of computer-assisted translation (CAT).

The most common strategies are the leveraging of target language terms and phrases from existing translations and the elimination of duplicates. How the result is presented to translators depends on the tools used. In the best of cases, the translator is given something like a table with the source texts in one column, any material that might help understand strings and keep them consistent in a second column, and the translation is then entered in a third column.

Problems with home-grown CAT

What we are faced with is, however, too often not all that helpful. The tools used are often created by software engineers who don't seem to know or care enough about the mechanics of language. I recently received a file from a TC that they had been given by a well-known software manufacturer whose name you would certainly be familiar with. This file incorporated every conceivable sin in this department. It was a short file and should not have taken longer than an hour to do. But the way the file was prepared made it untranslatable.

For instance: The original sentences often consisted of fixed and variable parts, as in the following example.

*No file containing ,"*xxx*, 'will be deleted.*
*The objects named ,"*xxx*, 'will be deleted*

The process had resulted in a file that contained the following in alphabetical order

'will be deleted
No file containing "
The object named "

It is easy enough to see that this is a hopeless endeavor in many languages. In German, for instance, *will be deleted* comes out as *wird nicht gelöscht* and *werden gelöscht*, respectively — these chunks of text are not at all the same, and in fact one is negative and one is positive! Imagine this happening with fifty or so phrases where you can't see which first (prefix) strings go with what last (suffix) strings. Imagine further — this was the case here — German words being stuck into the German text almost arbitrarily from a list. In some cases the suggestions were the opposite of what the original text was trying to say.

What was meant as a time and money saver ended up delayed

and costing twice as much as if it had been done correctly in the first place...

In cases like this one, it is important and in the best interest of our clients to firmly reject these source files. There is often no way to salvage them, and if there is, often enough it takes so much time and guesswork that we lose money on these files. The only professional solution is to work with the client and to explain why this “preparation” is counterproductive. Many of the problems can be eliminated if translators know how to work in the original source files, where (hopefully) strings and texts are given completely and in context. To make your point convincingly, you should know the underlying formats and be able to work in them.

CAT (including the professional tools) suffers from one intrinsic shortcoming: it tends to assume that what is the same in language A must be the same in language B, and what was the same in one situation must be the same in another situation. Not so, as you know. These tools, then depend a lot on whether the project management knows its stuff.

Professional CAT

I am always a bit afraid that techies use those tools to “translate” a lot of text, throw everything the machine was unable to find in a file, and send this residue to the translator, who depending on the circumstances, may be stuck with a chunk of text for which he does not know the precedent, or text where he can’t go back and make necessary changes to precedent (old text may be bad or contain errors or otherwise need improvement).

The second problem is that no CAT system can output better text than is input. All archiving systems — and that is what CAT essentially is, it archives and classifies preexisting translations — suffer from the weakness that mediocrity and mistakes are perpetuated.

The danger is that control over what is translated how passes from the translator to the client or translation company in charge of managing projects, that is, people whose expertise is not necessarily linguistic in nature.

I strongly feel that we as translators should be in control of our tools; any actual savings may well be passed on in the form of attractive prices, but we must be sure that the existence of CAT tools is not simply used as an excuse for putting pressure on our rates. The tools and the time needed to learn how to use them are a substantial investment that has to be recovered, and everything that you or your clients will expect to get out of a CAT system will first have to be put in.

CAT is here to stay, and most of us will have to deal with it in an open and receptive manner to continue to operate profitably. But the introduction of CAT will be a bone of contention between clients, translation companies, and independent translators in the next few years as its potential and limits are not yet fully understood by all those involved in the process. It would seem only fair for everyone to share in the benefits. Certainly CAT can save translators from monotonous repetition (software

translation can be incredibly boring at times. *Press Return to continue!*) but it should be the translator who ultimately decides if what looks like repetition is in fact repetition. Certainly CAT can help make the terminology in software localization projects, especially distributed projects, more consistent, but every now and then someone has to cut through the wallowing dust and start from scratch.

Some of these products are more and some are less friendly to the translator who is actually supposed to feed them. This is not the place to discuss the pros and cons of the various CAT systems. I will restrict myself to presenting a few URLs for some of the more popular ones.

- Translator's Workbench, Multiterm (Trados)
- Translation Manager (IBM)
- OpenTag (ILE)
- Atril (Déjà Vu)
- XL8 (GlobalWare)
- Star Transit (Star AG)

What to do when things go GNORW

A good translator's ingenuity will often be able to overcome the worst effects of poor preparation. But there are situations — unfortunately too many of them — in which we as translators simply have to stand up and state that what is expected is simply not do-able. We must remember that if we do not do this, then the result may be disastrous for the manufacturer, who is, after all, spending a large amount of money on having his product translated and whose business plans may fail if overseas sales fall behind expectations for any reason.

What can be particularly frustrating is to discover an error (and there is hardly a project where you won't) only to find that you can't convey it back up the development chain where it would have to be fixed (and passed back down to all the other languages and translators).

The golden rule here, as elsewhere, is: *Communicate!* Just about anything can be fixed if everyone who should know about it does know about it early enough. Of course no stressed-out project manager will be able to answer a lot of questions about things that are for you, the translator, to know. But if we want to be experts in communication, we need to communicate, and we should do so in a most professional manner: briefly, to the point, and addressing the designated contact or other person in charge. But we have to make it very clear to our contacts that the problems we discover (and who looks more closely at texts, formatting or even the program itself than the translator?) are not simply going to go away by waiting.

Conclusion

Software translation does not have to be the low-paid drudgery where inexperienced translators fumble in the dark. It is a field that has many rewards, not the least of which is that you will come to understand your own computer, its operating system and application programs much better, enabling you to make better use of your tools. With regard to the conditions under which you work, a certain degree of perseverance is often needed to make translation managers, software engineers, and others upstream aware of problems, errors, and the translator's specific needs. In localization we are part of a team, maybe more so than in other fields, and have to show team spirit. But we must make sure that our voice is heard and that our personal and professional needs are recognized by others. If we demonstrate the requisite subject expertise, we will earn the respect of our team partners, our customers, and our customers' customers.

Finally, we should never forget our responsibility to the users of the software we are localizing.

Further reading

Apple Computer, Inc.: *Guide to Macintosh Software Localization.* 1992. ISBN 0-201-60856-1.

Hoft, Nancy L.: *International Technical Communication: How to Export Information About High Technology*. 1995. ISBN 0-471-03743-5.

O'Donnell, Sandra: A Guide to Internationalization. 1994. ISBN 0-13-722190-8.

Uren, Emmanuel, Howard, Robert, and Perinotti, Tiziana: *Software Internationalization and Localization.* 1993. ISBN 0-442-01498-8.

Williams, Robin: *The PC is not a typewriter*. 1995. ISBN 0-938151-49-5.

Send your comments to the Webmaster

URL: http://accurapid.com/journal/softloc.htm

Updated 08/20/97