>>> revista tradumàtica
Traducció i Tecnologies de la Informació i la Comunicació
Número 1 : La Localització : Octubre 2002 : http://www.fti.uab.es/tradumatica/revista

Localization Engineering: The Dream Job?

By Bert Esselink

Resum
This article provides an introduction to the fundamental concepts of localisation engineering and the tasks performed by the localisation engineer. The author also comments on the profile and training of the localisation engineer and the main changes occurring in the nature of the products being localised.

Paraules clau
Localisation, localisation engineering, project management.

Introduction

A few years back I wrote an article for an ALPNET magazine called "So what do our localization engineers do?" The article introduced the concepts of localization engineering, the activities performed by localization engineers, and some of the tools they used. The information was very much welcomed, especially by those who had wondered for a long time about this *species* that was playing such an important role in localization but whose job was never really understood.

In this contribution to TRADUMÀTICA, I will take a closer look at the job of localization engineer and discuss some of the changes that have happened in software localization and that have had an impact on this profession.

What Is Localization Engineering?

So what is localization engineering? It is not translation but still very related to language. It is not engineering but still much related to *building* products. Localization engineering basically is taking the development environment of a product, taking it apart, and putting it together again after all the text has been translated. Actually, localization engineering is probably the main difference between *localization* and *translation*. It's all that "technical stuff" that needs to happen when a software user interface is translated, or an online help file, or even an HTML file.

When the localization industry started taking off, in the early 1980s, localization engineering as an independent job was not known. In those days usually the original developers or programmers of the source language product collaborated directly with the translators to build localized versions of the product. This worked relatively well, even though there was an obvious gap between the language background of translators and the technical background of developers. Communication wasn't always running smoothly and both parties didn't always understand or appreciate each others priorities or challenges.

So a bridge had to be built between all this "technical stuff" and the translation work. This came as a fairly natural evolution as many software publishers started outsourcing more than only the translation work to "localization vendors" that clearly recognized many opportunities. Developers had to focus on making the release dates of the –normally– English products, and had no time and/or interest to deal with all the complexities of multilingual versions of the product. Basically, the ideal model for software publishers was to send out the build environment of their product and receive a fully localized and ready to ship version of the product a few weeks or months later.

When the localization industry matured, software publishers realized the competitive advantages of releasing all language versions of their products simultaneously, which called for even more technical work (use of translation technology, processing updates, testing beta versions) in the localization and translation process.

So the answer to the question "what is localization engineering" is a very broad one. In general, localization engineering consists of all the work that translators cannot do, as well as all the work that the original product developers cannot do. Below we will take a closer look at what this work exactly entails and what the profile is of the people who do this work.

Localization Engineering Tasks

Traditionally (read: since the early 1980s) the tasks of localization engineers primarily consisted of preparing projects, building and compiling localized products, and supporting translators. The "products" are typically Windows or Macintosh software applications and online help systems. Let's now take a closer look at what localization engineers do.

Project Preparation

A "localization kit" normally consists of hundreds of files, of which some are translatable and many are not. It has build environments for software applications and/or the accompanying online help files. To build a software application, you need all the resource files and code files, which are then *compiled* into a binary file or executable which can be run on a computer. For example, a Windows application developed using Visual C++ consists of hundreds of files with programming code, resources, and resource files. Examples of resources are the bitmaps used in toolbars, such as the printer icon which executes the Print command. In most cases, these resources do not have to be changed for any localized versions of the product. The translatable information, such as the menu text, dialog box options, and error messages, is stored in *resource files*, which in the Visual C++ environment normally have the .RC extension. A software product which has been well-internationalized stores all translatable text in one or more of these resource files, which makes the localization job relatively simple. However, in many cases, files containing translatable text are found all throughout the build environment.

It is the localization engineer's responsibility to locate and identify all these translatable files and to prepare them for translation. Localization engineers should ensure that translators know exactly what they need to do, so they can get started quickly. Software localizers normally translate resource files in a translation memory tool such as TRADOS or a user interface localization tool such as Alchemy Catalyst or Passolo.

```
14 DIALOG FIXED IMPURE 0, 0, 356, 196
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CONTEXTHELP | WS_POPUP |
WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Page Setup"
FONT 8, "MS Sans Serif"
BEGIN
        GROUPBOX "Paper",1073,8,9,224,56,WS_GROUP
        LTEXT "Si&ze:",1089,16,24,36,8
        COMBOBOX 1137,64,23,160,160,CBS_DROPDOWNLIST | CBS_SORT |
        WS_VSCROLL | WS_GROUP | WS_TABSTOP
        LTEXT "&Source:",1090,16,45,36,8
        COMBOBOX 1138,64,42,160,160,CBS_DROPDOWNLIST | CBS_SORT |
        WS_VSCROLL | WS_GROUP | WS_TABSTOP
        GROUPBOX "Orientation",1072,8,69,64,56,WS_GROUP
        CONTROL "P&ortrait",1056,"Button",BS_AUTORADIOBUTTON |
        WS_GROUP |
```

Figure 1 - A resource file: spot the translatables.

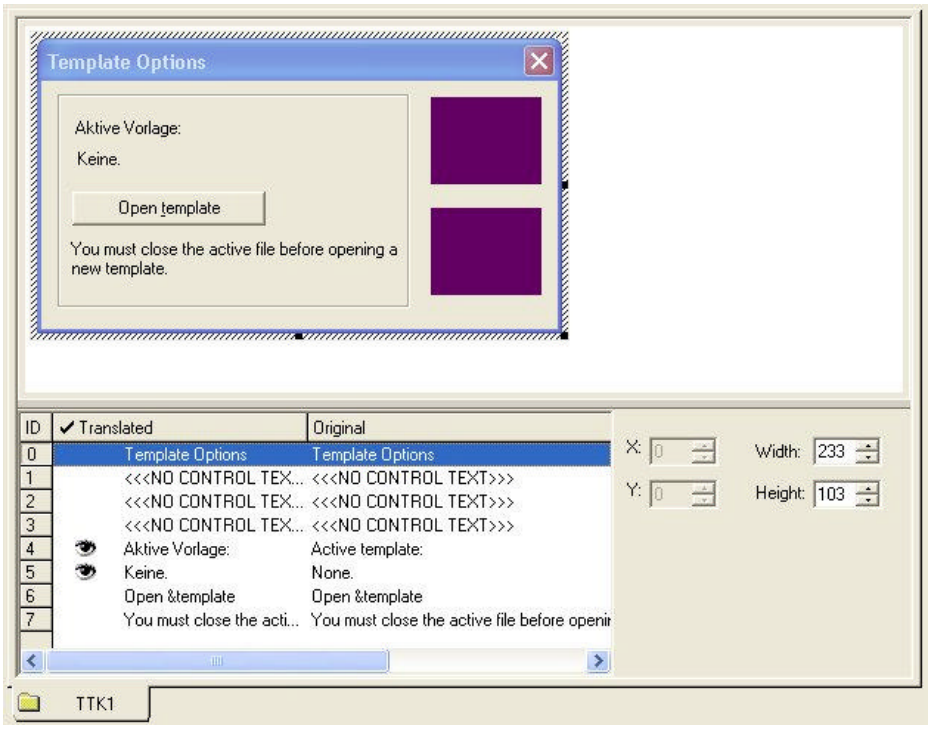

Figure 2 - Translating a dialog box in Alchemy Catalyst.

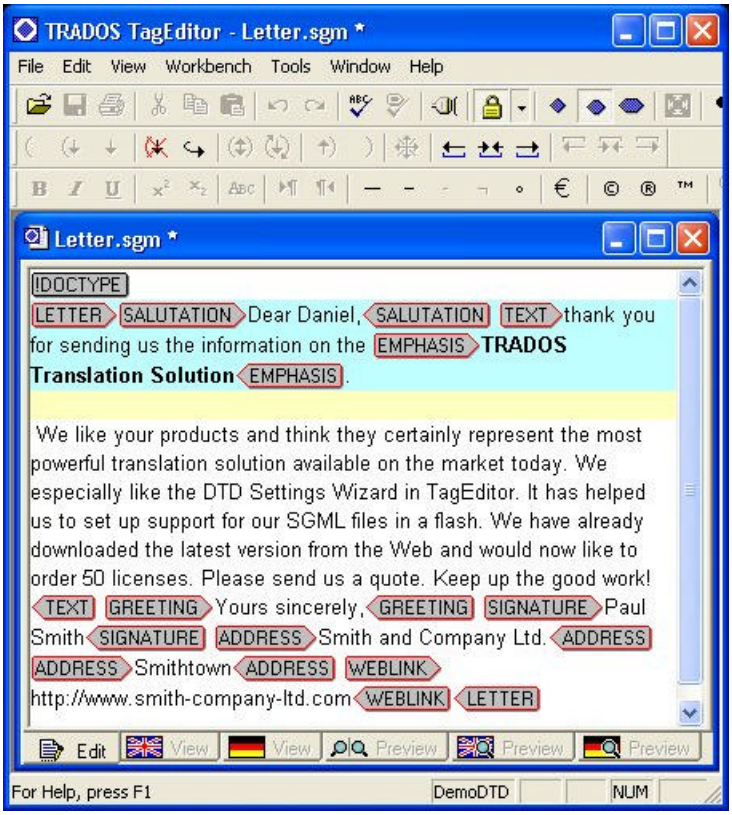

Figure 3 - Translating an SGML file in TRADOS TagEditor.

Another important activity in the project preparation phase is testing the build environment and setting up all the software required to compile the localized product. Finding out compilation errors early on in the process could prevent considerable trouble later on in the process when the localized versions of the product are compiled. This is essential for software applications as well as online help systems.

The data collected by the localization engineer in the preparation phase will be essential for project managers to schedule their products or to define the scope of work and general project approach.

Building & Compiling & Testing
When all the translatable files have been identified and prepared for translation, a "translation kit" is sent out, which contains all the resource files that need to be translated, all the tools required, plus instructions for the translators on how to deal with the various files and file types. Building these "translation kits" is an important task for localization engineers. Considering the fact that most software localization projects involve a large number of target languages, getting the translation kit right from the start would prevent many similar or identical questions from the various translators.

Once the translated files are returned, the localization engineer can start building the localized products. To do this, translated resource files are verified and then reintegrated in (a language-specific copy of) the build environment. Although software and online help compilation is largely an automated process, there is normally quite a bit of bug fixing to be done by the localization engineers. Also various language settings and code page configurations may need to be changed. For Asian languages, other fonts need to be chosen to display the localized products.

The bulk of the work when building a localized software product is getting the user interface to look good in the localized version. Translated strings are normally longer than the English source, which means that dialog boxes need to be resized and fields need to be moved. This not only requires a basic understanding of the language that is being worked on, it also requires a good understanding of user interface guidelines. For each platform (Windows, MacOS, Unix) guidelines exist that describe how user interface elements should be positioned and sized.

The resizing work is normally done using the Resource Editor which comes with most development environments. Resource editors are used by developers to design the user interface, i.e. create the dialog boxes, toolbars, icons, menus, etc. Localization engineers use them to "re-design" the layout for the localized versions.

When applications or help files have been compiled, it is the localization engineer's task to test the localized versions, usually in a number of target languages. The testing work involves a certain level of functional testing (does the localized product still work well?) as well as cosmetic testing (does the localized interface look good and are all the translated strings displayed correctly?).

Supporting Translators
One of the most important and challenging tasks for a localization engineer is to support the translators in their work. I already mentioned the "translation kit" that needs to be created for translators, but localization engineers also need to be available to support translators during their work.

This support could be anything from answering questions on the meaning of certain terms used in the software application to assistance with the tools the translators use. For example, when a software resource file contains the translatable text *file*, without any further context, a localization engineer should be able to tell a translator if the word is used as "to file" or "a file", although in many cases the engineer will have to divert the question to the software developer. It may even be possible that the word is an internal program command and should not be translated at all. In cases where translators need help solving these types of issues, localization engineers bridge the gap between translators and developers. Localization engineers can provide developers with feedback on specific issues that translators have come across while localizing the product, so these can be fixed in future releases.

What Is a Localization Engineer?

From what I explained above, it seems that a localization engineer needs to be almost a "superhuman". Who on earth can combine knowledge of programming languages, development environments, user interface design guidelines, localization and translation tools, language characteristics, and translation challenges, all in one brain? Well, it may seem like a mission impossible but still there are thousands of people doing these types of activities on a daily basis. And, believe it or not, many of them once started out as translators or have a background in languages.

Although localization engineering can be very tricky at times, the basic idea is that of "recreation" of something that already exists. So a localization engineer doesn't need to be a programmer, a translator, or a designer. Most importantly, a localization engineer should be someone who is not afraid of technology, who loves to experiment and troubleshoot, and who is extremely flexible. The job hardly ever gets boring, as each customer in the localization industry has different build environments,

different programming methods, and different levels of making a product "localization-friendly". Localization engineers never stop to learn, and should never stop to teach others.

In fact, the job of localization engineer is ideal for those who like to fiddle around with tools, are keen on solving problems, and love to communicate with developers and translators alike.

There is no "official" education for localization engineers. Most localization engineers are self-taught and have learned the necessary skills on-the-job. However, there are some courses in localization engineering, for example the ones at the University of Washington. The Localization Engineering I course covers the following topics: background on localization, localization process models, optimizing localization, platform specific localizations, cross-platform solutions and tools, web-based localization, localizing multiple resource formats, no-compile localization technology, pseudo localization technology. The localization engineering course at the University of Limerick in Ireland raises the technical bar even higher, with topics such as character codes and character sets, data security and integrity, disk imaging tools, etc. Both courses are targeted at the more technically-savvy people, although the University of Washington course is advertised as good for "People with language experience seeking an engaging career in an international environment."

In summary, a localization engineer is an explorer, someone who is always at the front line of new technologies, and someone who plays a key role in helping translators cope with the many challenges of localization.

Past, Present & Future

Above I have presented the "traditional" role of localization engineering in the localization process. Although this model is still very valid today, the past few years the nature of the products that localization engineers are dealing with has changed dramatically.

Several years ago there was some level of standardization with respect to development platforms. For Windows applications, for example, most development was done using Visual C++ or Visual Basic, and online help systems were usually RTF or HTML-based. With the growing importance of HTML as the main interface for software applications, the way localized products are produced has also changed. Where Windows applications could be compiled, executed, and tested on one single machine, new generations of Web-based applications require a more complex setup, both for building and execution.

New development techniques have evolved, such as Java, XML, .NET and database platforms. At the same time, the level of localized software has changed. The majority of the applications localized used to be desktop applications; today, more and more business-to-business or enterprise scale applications are localized. Translatable text is increasingly separated from layout or design, which means that localization engineers and translators are working more and more out-of-context. Another result is that much of the layout (both user interface or online help/documentation layout) is *templatized*, which means that much of the formatting is automatically generated, resulting in less engineering work.

Even the way *content* is published, has changed rapidly. Manuals that were traditionally written using applications such as Microsoft Word or Adobe FrameMaker are now dynamically built using database-driven publishing systems or content management systems. Where translators could get started quickly by just working in Word or importing the document into a translation memory system, now often a localization engineer is needed to produce a "translation kit" from a series of complex SGML or XML files containing the manual text.

In summary, localization engineers will play an increasingly important role in making translatable information "translator-accessible". This, combined with the fact that good localization engineers are difficult to find in the marketplace, would make this career move very interesting for the more technically-savvy translators. So if you're a techie and you want to move into languages, or if you are a translator and you want to move into technology, this may very well be the dream job for you.

Bert Esselink