The role of Computer-Assisted Translation in the field of software localization

Alberto Fernández Costales
University of Oviedo

This article analyses the effectiveness of computer-assisted translation (CAT) in the field of software localization. In order to measure and assess the advantages of using translation tools, a program is localized using Passolo, a specialized software localization application. The study is intended to calibrate how CAT can improve translators' performance in a localization project and also to appraise the main features of the selected application, focusing on functionality, usability and reliability. The article outlines some of the challenges and difficulties of software localization, aiming to test how the process of adapting a product to a particular locale can be optimized by the use of computer assisted translation. The case study focuses on the main challenges of the process from the point of view of translation. To conclude, the study formulates a number of conclusions and evaluates the performance of the selected application.

1. Introduction

The development of new technologies in the latter half of the 20th century and the explosion of the Internet as a promotional, informative and communicative tool have modified how companies traditionally engaged in international trade. On the other hand, the political and economic changes occurring in recent decades have shaped a new global panorama involving new players and social agents.

Globalization has allowed small and medium-sized companies to increase their market shares by competing at an international level with the support of the Internet and new software applications.

In this scenario, translation is not only the basic tool for intercultural communication and a vehicle for understanding among nations but has turned into an essential element for the economy of any company seeking an international presence beyond the borders of its home country (Corte, 2002).

Companies adapting their products to a particular market will increase their sales figures as they improve their brand images. But a wrong Web site translation or a failure to localize software applications can lead to commercial failure even if the quality of the product is widely recognized.

As explained in this paper, localization involves not only textual adaptation but also modifying the non-verbal, semiotic and cultural elements of a product in order to make it suitable for the target audience. In order to achieve this goal, translators and localizers can rely on the support provided by computer-assisted translation (CAT)[1].

Traditionally, research in translation technology has been linked to machine translation. The lack of conclusive results in this area in the last decades of the 20th century opened new lines of research the target of which was not achieving machine translation but developing new tools to assist human translators. The application of CAT has been widely studied by translation scholars (Austermühl, 2001; Nogueira, 2002; Biau & Pym, 2006; Somers, 2003; Melby, 2006), and research in this field has contributed to improving and enhancing translation technology. One of the main advances in this field has been the setting of a standard format, known as Translation Memory eXchange, or TMX, which allows the exchange of translation memories among different applications (Abaitua, 2001). This implies that more than one single tool can be used to carry out a project, fostering competition among translation software providers and giving translators more flexibility and freedom of choice.

The main hypothesis of this paper is that the use of CAT is profitable for translators since it provides more efficiency and consistency in software localization. The more technical terminology is used and the more repetition occurs in a software application, the more suitable the use of CAT will be.

The basics of localization will be overviewed in section 2, focusing on the main methodological alternatives available in the case of software. The case study is detailed in section 3, where the key elements of the project will be explained and the way in which translation tools can effectively optimize localization from the translator's point of view will be analysed. In addition, the tool's performance will be evaluated, with comments on its strengths and weaknesses, and possible alternatives to Passolo will be suggested. Finally, conclusions are presented in section 4, where the initial hypothesis will be discussed.

2. Localization

Localization is the process of adapting a product to the local market where it is going to be sold, so that it seems that it has been originally designed for this particular audience. For a product to be effectively localized, users should not be aware that it has been designed in an other part of the world, with a different language and with an other cultural background. That is to say, the final consumer should not detect that a particular product has been created with other cultural parameters (Corte, 2002). Bert Esselink provides the following definition:

> Generally speaking, localization is the translation and adaptation of a software or Web product, which includes the software application itself and all related product documentation. The term "localization" is

> derived from the word “locale” which traditionally means a small area or vicinity. (Esselink, 2000, p. 1)

The localization industry started to flourish in the 1980s, together with the development of new and more powerful translation tools, and it was consolidated in the 1990s with the creation of big multinational giants such as SDL (Esselink, 2003). Today, localization is a blooming sector, with a growing number of professionals and researchers working in the field. According to the Localization Industry Standard Association (LISA), the growth of the industry accounts for $30 billion per year (LISA, 2007, p. iii).

Localization has been approached from different viewpoints (Dohler, 1997; Esselink, 2000; Pym, 2006) and it is a meeting point for translators, linguists and professionals from the computer industry. The controversial question of the status of localization within Translation Studies has been widely discussed in the literature (Esselink, 2000, p. 2; Mangiron & O’Hagan, 2006; Pym 2006) but is beyond the scope of this paper.

Localizing a product goes beyond the textual adaptation of contents from the source to the target language. In addition, this process deals with the adaptation of multiple non-verbal elements to the target locale: Images and other visual components (icons) that the product may include, colours, cultural references, numbers, date formats, currency, flags, music and legal issues (Yunker, 2002, p.477).

2.1. Software localization

Localization deals with different fields, namely, Web sites, videogames and computer applications. Software is one of the most interesting targets since it allows millions of users to access computers in their own language.

Arguably, the US is the leading country in the computer industry (with the exception of videogames, where Japan rules the roost) and American English is the *lingua franca* for software. Original American programs are localized into the so-called FIGS languages (French, Italian, German and Spanish) (Esselink, 2000, p. 8).

In addition to the textual component (menus, help files, etc.), a computer application includes images and other non-verbal elements that should be localized. Colours, for example, must be adapted, taking into account the values they may express in the target culture: Green is a sacred colour in Arab countries, and white is used for funeral pyres in China (Yunker, 2002, p. 485). Even when these colours are harmless in the US or Europe, they must be adapted when localizing the product to different locales.

Cultural issues such as measurement units, currency, numbers, and date formats must be taken into account in the localization process. If we are working with a spreadsheet created in the United Kingdom, the pound will be the default currency. This element should be modified if the program is to be exported to Greece, Spain, France or any other member state of the euro zone.

Technical questions must also be addressed, as they can affect the correct functionality of an application. Dialogues and menus, for instance, must be edited due to possible enlargement of the text produced by translation. Such enlargements can create functionality problems in the user interface and affect the application's appearance (words exceeding allotted space, unreadable menus, etc.). When translating from English into other languages the text is expanded between 20% and 30% (Esselink, 2000, p. 33; Yunker, 2002, p. 176).

As explained in this paper, assisted translation tools provide a helping hand in many of the localization challenges here mentioned. One of the main advantages of CAT is commented on in the next section.

2.2. Source code or assisted translation?

There are two main alternatives to localize a computer application: Dealing directly with the source code file or localizing the binary files of the program once they have been compiled (that is, working with the translatable text).

The files used to create computer applications are known as "Resource Files", and are easily recognizable through the extension ".rc", which stands for "Resource Compiler" (Dohler, 1997). These files contain all information about the program's architecture, its functions, etc. Once the Resource Files have been compiled, the so-called "Executable Files" (usually with the ".exe" extension) are created[2].

2.2.1. First option: Source code file

If we work with the source code directly, we will have to cope with the previously mentioned Resource Files. This process requires expertise and technical skills since not all translators and localizers can read and master the source code file proficiently. This code is the language used by developers to design software and is based on instructions and commands in a specific programming language (such as C++). These instructions configure and adjust the parameters of the program and how it works.

```
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,108,80,50,1
        END
```

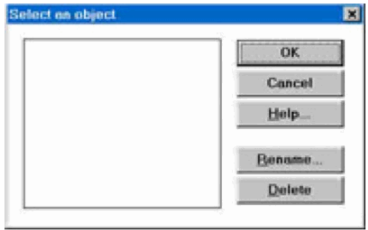

Figure 1: Example of a dialogue box and its source code file

A key problem in localizing the source code files is that working with the code requires surgical precision to accurately modify the translatable information without damaging peripheral text. Specific information (translatable strings) must be spotted very carefully since even a slight mistake can lead to functionality problems (the so-called “bugs”) once the files have been compiled.

This is an extremely complicated process due to the difficulty of separating translatable text (usually between commas) and the code surrounding it. Given the hurdles and technical requirements involved in editing Resource Files, a more reliable alternative should be considered: Using computer-assisted translation to filter translatable strings.

2.2.2. Second option: translatable text

This alternative implies that we will not face the source code file, but will instead deal with the binary files once they have been created. Thanks to CAT, localizers are provided with a tool that supplies the same functions without requiring any editing of the source code. Translation tools use parsers and filters to detect translatable strings, thus allowing localizers to focus exclusively on software adaptation.

When extracting the translatable strings from the source files, translation tools perform a segmentation process in which the text is broken down into translation units. Segmentation rules can be adjusted according to the file format (for example, localizing a Web site in HTML may require different settings than a program in C++).

Furthermore, CAT supports localizers with translation memory systems, which allow previous translations to be re-used and recycled (leveraging). As translation units are stored by the application, localizers will be provided with suggestions for new translations according to the matching percentage (in Passolo, 100% coincidences provide exact matches and 75% fuzzy matches).

Moreover, translation quality can be improved by increasing coherence in the target text thanks to certain features such as “Translate Replicates”, which are included in many applications. This is a useful tool for localizing software or Web sites, where there is a high level of textual repetition (and an important number of technical terms).

The next case study focuses on some of the most important features that use of CAT can add to the process of localization.

3. Case study

The aim of this case study is to test how CAT can support translators in localizing a software application and to provide evaluation of Passolo, a well-known localization tool. To do this, the user interface of a program from the field of logistics management was adapted from American English into Castilian Spanish. We will focus on the linguistic and technical aspects of the process. As this paper is written under the scope of Translation Studies, issues regarding marketing or project management are not addressed.

3.1. Localization tool

The localization software chosen for the case study is Passolo 6.0 Team Edition[3]. Selection of this particular tool was done according to several parameters. Firstly, we wanted to evaluate a Windows-based standalone application (Trados, Transit and Wordfast used Microsoft Word macros at the time this paper was written), with full-localization features (such as user interface resizing functionality, image and bitmap edition tools, automated localization tests, etc.). The best-known multi-faced localization tool meeting these requirements (together with Passolo) is Alchemy Catalyst, which offers a user-friendly interface and additional interesting features. However, using Catalyst was not possible since version 6 did not support 16-bit binaries (the format of the files to be localized). This circumstance made Passolo the most suitable candidate for this particular project.

The tool is evaluated mainly by its functionality, since the hypothesis in question asks how CAT can improve translators' performance in a localization process. Other aspects, such as usability and reliability, will also be commented upon.

Passolo can be used to localize 16-bit binary files (.exe, .dll) as well as software developed with 32-bit applications (Visual C++, Borland Delphi, and Borland C++ Builder). It also supports ASCII and Unicode (allowing localization to Asian languages). Additional features include Trados and Transit interface, terminology management, and a powerful tool for statistical analysis. The program has a rather short learning process (regarding the basic and most common localization functions) and the provided documentation is quite complete. Negative remarks regarding some of the mentioned features will be reported in the task description of the case study.

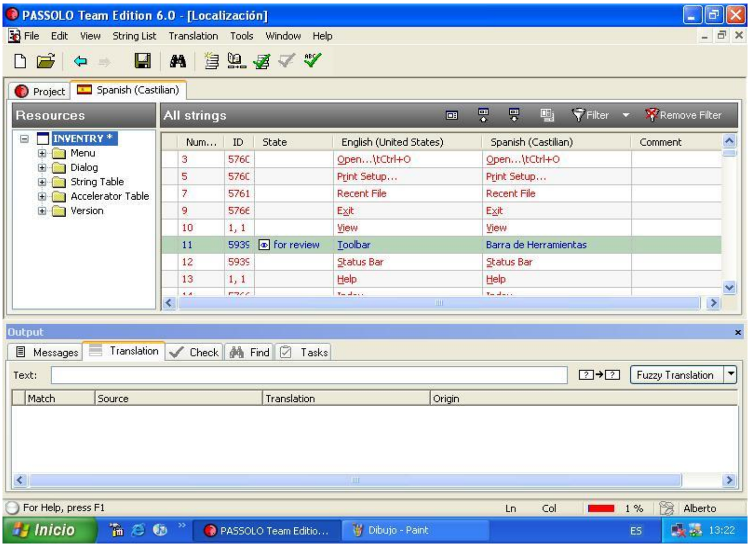

Figure 2: Passolo 6.0 user interface

3.2. Target application

The software to be localized is HOM (version 3.0), a widely known application in the field of transport and logistics management. This tool was developed at the Leonard N. Stern School of Business at New York University by Professors Michael Moses and Sridhar Seshadri, along with software consultant Michael Yakir[4]. The program was designed specifically for the Competitive Advantages course in the business school's Operation Management program and was released in November 1998.

HOM provides several tools designed to analyse a company's operations and services, and it helps to work out suitable solutions for specific problems regarding logistics and supply chain management. HOM comprises seven different modules: Quality Control, Inventory Management, Process Management, Forecasting Techniques, Project Management, Integral Planning and Queue Theory.

The program's seven modules share the same appearance, and the user interface is practically the same: The central part of the screen displays a table where data are introduced, and in the upper part is the classical toolbox with the typical menus (File, Edit, View, etc.) and 14 different icons. As seen in Figure 3, ten of these icons are common to any Windows-based application: New Document (a white sheet), Open File (a yellow folder), Save (a disc), Print (a printer), Preview (a magnifier and a sheet of paper), Cut (a pair of scissors), Copy (two pages overlapping), Paste (a clipboard), Help (a pointer with a question mark), and About (represented by a question mark). The other four icons of the toolbox have been specifically designed for HOM: Parameters (represented by a table), Run (a runner), Last Results (a graph) and Log (a writing hand).

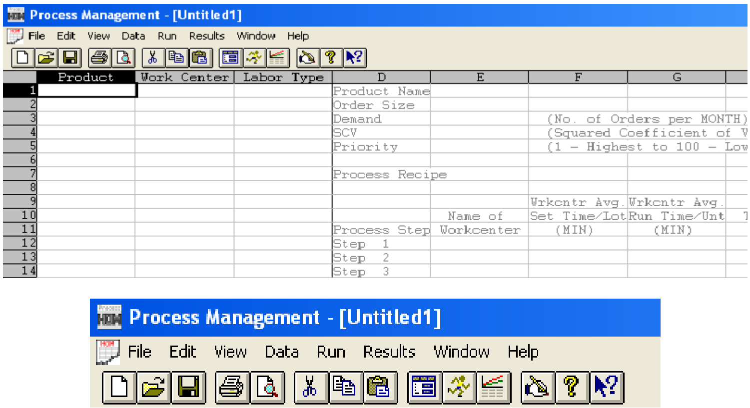

Figure 3: Screenshots of the Process Management Module and the toolbox of HOM 3.0.

3.3. Localizing the application

In order to achieve complete localization of the product, the seven modules of HOM were translated into Spanish and all non-verbal elements of the program were modified using Passolo: In addition to the translation of the text, the user interface was adjusted to the target language and some cultural elements (i.e. icons) were adapted.

Finally, when the localization stage was concluded, several quality and functionality tests were conducted using tools provided by Passolo (as we wanted to check whether the whole localization process could be completed with one single application). Quality tests included text proofreading, spell checking and terminology coherence. Passolo has simple tools (similar to those included in any text editor) to carry out the first two tests. As for coherence, the option Concordance Search allows the translator to check how a single term has been translated in all entries. Some possibilities of the program were not fully explored since we did not rely on the support of Trados or Transit.

On the other hand, functionality tests focused on user interface (size of windows, menus and boxes) and verification of shortcuts and other commands. In this sense, Passolo offers competitive advantage over its competitors: With a single option, all translations stored in the project can be checked. The command Check All performs a complete verification of the whole project (or a selection) and detects possible errors regarding size of menus and duplicate access keys (Figure 4). The application spots the exact line where a mistake was produced, making it is easy for translators to correct errors.

In this respect, CAT provides an important support, as functionality tests can be time-consuming for translators and localizers who do not rely on translation tools.

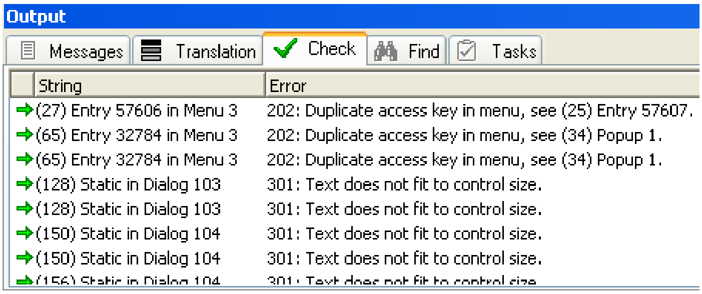

Figure 4: Functionality test in Passolo 6.0.

Arguably, textual coherence is an issue to be addressed in software localization: This is a key element not only for translation quality, but also for the program's functionality,as it is underlined in most localization guides (Lingo, 2000). Bert Esselink suggests the following:

> Create a glossary of terms relating to the product, company or industry, and apply this consistently across all documentation and online help sets. Use consistent phrases and terminology. The importance of simple, concise language is magnified when writing for translation. For example, decide at the outset if you want to use phrases like "click on", "click", "choose", or "select" when describing software commands. (Esselink, 2000, p. 28).

When starting a new project important decisions concerning terminology must be made. For example, if we are going to adapt an application from Spanish into English and we find the term *teléfono móvil* we will have to decide whether to use the British expression ("mobile") or the American one ("cellular", or even "cell"). Whatever the decision, it must be consistent and the same term must be used systematically throughout the translation of the application and all additional materials (manual, licenses, box, starting guides, etc.).

In order to maintain terminological coherence, Passolo provides the possibility of creating a technical glossary linked to the localization project. This can be extremely useful for two main reasons. First, by creating a glossary we contribute to maintaining coherence. Second, the glossary provides the translator with valuable information for new projects, making his or her job faster and more accurate.

In our project, the glossary was created after translating the first module of HOM, and it became a powerful tool for localizing the rest of

HOM since a huge amount of recurrent expressions was to be translated (even in the first module, Queue Theory, 40% of total strings were auto-translated by Passolo due to the high number of repetitions). Furthermore, glossaries can be stored for use in future translations with the same customer or similar projects.

Another interesting option is Translating Replicates. When translating a term for the first time, the program detects if there are more repetitions and asks the translator if he or she wants to export the translation and keep it for the rest of the entrances (see Figure 5).

This is a common option not only in Passolo but in CAT, and it can be extremely useful in large projects, especially if there is a high number of repetitions. Furthermore, it provides terminological coherence to the localized application.

Figure 5: Translating replicates with Passolo

Other features included in Passolo (and in most localization tools such as Catalyst) are Auto Translation and Fuzzy Translation. The first option automatically translates the selected term if it has been previously translated in the project and there is 100% matching. The second option translates the term according to similar or partial expressions that have already been translated.

Some features described in this section support the idea that using CAT can improve the translator's performance in two different ways: By reducing the time needed to complete a project (thanks to leveraging, the re-use of previously translated strings) and contributing to maintaining textual coherence. It should be mentioned that in the case of software localization, functionality tests are a must: Passolo provided specific tools to perform these tests in a reliable and fast way.

In the next sections, some challenges faced in the program localization are commented upon, as are the solutions proposed for each case with the support of Passolo.

3.4. Linguistic aspects

Computer applications present a notable amount of technical terminology; this pool of terms should be translated so as to conform (as much as possible) to the standards of the sector (Windows operating system, in this case). If the string Out of Memory is translated as *Fuera de memoria*, Spanish native speakers would perceive it as an artificial expression. The sentence *No hay suficiente memoria* would be more accurate for users of computer applications. Similarly, the term Tile may not be translatable as *Teja* or *Azulejo* in the context of computers, where the word *Mosaico* refers to the layout and distribution of several windows in a computer screen.

Some of the most repeated terms in HOM can be found in any simple Windows-based application. Table 1 shows the most repeated terms in the Queue module of HOM:

Table 1: Recurrent terms in the Queue module of HOM 3.0.

Term	Repetitions
Help (*Ayuda*)	25
File (*Archivo*)	23
Open (*Abrir*)	21
New (*Nuevo*)	17
Print (*Imprimir*)	13

As mentioned earlier, Passolo provides several tools and options to deal with terms that are repeated in a text.

3.5. Technical issues

3.5.1. Shortcuts

Shortcuts allow users to execute an action or command without opening a menu with the mouse. By using a key combination (Alt and another key, in Windows-based applications), we can perform the same action with a few keystrokes. These elements are defined in the source code file and are meant to give users faster access to menus and dialogue boxes.

Shortcuts are usually represented in menus and toolboxes with an underlined letter (Open or *Abrir*) that indicates the combination required to perform that particular function with the keyboard. Computer-assisted translation tools (including Passolo) detect shortcuts in the source text, thereby allowing localizers to translate them into the target language in an effective way. By placing the character & before a specific letter we create a shortcut in the localized version (the command <u>O</u>pen would appear in Passolo as &Open). As mentioned, Passolo is a rather functional tool regarding shortcuts, as it facilitates easy detection of duplicates.

3.5.2. Dialogue boxes

Another element to be tackled concerns the three dots (…) that appear after some translatable terms. The dots indicate that the term they accompany opens a dialogue box, and so the three dots should be kept in the target text for usability reasons. In this sense, Passolo does not allow Placeables translation. Placeables are elements (years, cities, proper nouns, etc.) that are not translated into the target text (they remain the same). When a term is marked as Placeable, it is copied into the target text without translation. This option can be useful when adapting dialogue boxes, as translators must copy (or type) the three dots into the target text. This function is common in translation memories (Trados, Wordfast) and even other localization tools (Catalyst).

3.5.3. Space restrictions

Space limitations are one of the most serious difficulties in localizing a software application. When translating from English into other languages, text can be expanded considerably: The Edit menu becomes *Bearbeiten* in German with a 100% expansion (Esselink, 2000, p. 33). This is a major concern for translators, as they must include long text-strings in limited spaces. However, translation tools enable redesign of the graphic user interface to adapt it to the target text.

This is an easy task with Passolo, since it includes a WYSIWYG (What You See Is What You Get) editor, where menus can be resized using drag-and-drop techniques, as shown in Figure 6.

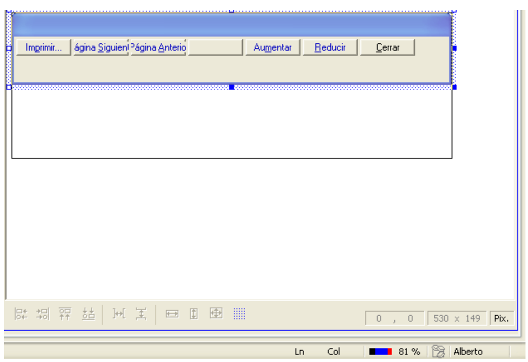

Figure 6: Redesign of a dialogue box using Passolo.

3.5.4. Icons

Despite the reduced number of images and icons in the user interface of logistics software, one specific bitmap was modified in the toolbox of HOM, where the following icon was found:

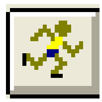

Figure 7: Icon of the Process Management module of HOM 3.0.

Although the expression *correr* (the Spanish equivalent for "run") is used in the field of computers as a synonym for *ejecutar* or *funcionar* ("execute" or "work")—as in the expression *la aplicación* ***corre*** *bajo Windows* ("the application runs in Windows")—a Spanish native speaker who is unfamiliar with this kind of (computer) language would not understand the meaning in this context.

The verb *correr* ("to run") does not refer to anything related to computer applications in Spanish, although the expression is widely used in the sector due to direct translation of the English "run", which is effectively meaningful in that context. However, the translation is not accurate and only minor numbers of people (computer industry professionals, advanced users, etc.) would understand *corer* in this sense.

In this particular case, there is a disruption between the meaning (execute, make something work) and the image representing it, since the icon of the runner will not make Spanish users understand its function. Noelia Corte (2000), in her research on Web site localization, notes that "Negative reactions may also come from the use of colour or icons. The metaphor of a person running to symbolise the running of a program does not work in all languages" (p. 19).

In the case of the HOM icon, the signifier (the image of the runner) is associated with the term "to run". When localizing the application into Spanish, the most suitable option is to adapt the icon to the target locale.

For this case study, the tool included in Passolo was used to redesign the icon. New versions were created so as to present better associations between the image and its function for the target audience: An icon with a finger pressing a button would match with the term *Ejecutar* ("execute" or "perform"), which is the way "run" is translated into Spanish.

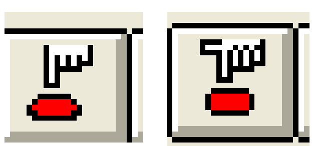

Figure 8: New versions of the Run icon

The adaptation of bitmaps or images is not an easy task, and Passolo's editor is not particularly functional as compared to similar tools in other localization software. Any other simple application (like Paintbrush, included in Windows) provides more functionality and contributes to more accurate results (apart from professional tools such as Photoshop).

3.6. Research output and assessment

The seven modules of HOM accounted for 2,789 segments, that is, 10,228 words (60,670 characters) that were translated from English into Spanish. 46% of the translatable strings were repeated in several modules and consequently auto-translated by Passolo. Due to the high rate of repetitions, we can conclude that the use of CAT clearly improved the translator's performance on this project. In addition, textual consistency was enhanced through some of the tools included in Passolo (concordance searches, glossaries, etc.). Besides, functionality tests were carried out using the same application.

In order to check the real user experience of the localized version of HOM, a copy of the program was delivered to students of the Master in Transport & Logistics Management of the University of Oviedo, as this application is used in several courses. The criticisms of the Spanish version were positive and users reported[5] having a better performance when using the application in their own language. Besides, adapted icons (Figure 8) seem to have improved usability in the target locale.

Passolo, the tool selected for the case study, can be positively evaluated according to several criteria. Regarding functionality, the application provides a number of features that clearly support localization and translation tasks: Translating replicates, fuzzy matches and auto-translation, shortcut edition, etc. are quite well implemented. Functionality tests are easily conducted with one single process. However, some interesting options (such as translation of placeables) are missing.

Reliability is one of the strengths of Passolo: Source text segmentation works properly and files can be successfully exported to other applications. No problems were detected regarding glossary management, and Trados interface works properly (although this was not used for the case study). The program supports a good number of file formats (although some problems have been detected with the HTML parser).

Regarding usability, Passolo is weaker than its main competitor, Catalyst, which has a much more user-friendly interface. Passolo's main layout is correct, but it is less intuitive and usable than those of other CAT tools. Furthermore, the image and bitmap editor is rather poor and some options (such as the possibility to modify background colours) are not available.

As a standalone Windows-based application, Catalyst seems to be the main alternative to Passolo in the field of software localization, offering a more usable interface but less file format compatibility. In the particular case of Web site localization, other alternatives may include small applications such as Catscradle or OmegaT (an open source option).

4. Conclusions

The use of CAT in software localization provides important benefits for translators and localizers. Besides improving text consistency and terminological coherence, assisted translation tools help to save time by recycling previously translated strings (leveraging). In addition, software can be completely localized using only one application, as it can be observed in our case study.

The target of translation tools is to improve translator's performance when completing a given project: Therefore, CAT is not a threat to professionals, since the quality of the final output will be strictly linked to the skills and competence of human translators. Learning curves for using CAT tend to be quite reasonable and multi-faceted applications (such as Passolo) can be handled in a short period of time (although extra time may be required to master it).

Obviously, relevant differences exist among translation tools in regards to not only functionality and usability but also other important issues (such as price, license conditions, etc.). The selection of a particular tool must be done in accordance with the specific requirements and necessities of translators. However, these applications clearly offer an advantage in order to achieve a truly localized product.

Bibliography

Abaitua, J. (2001). Memorias de draducción en TMX compartidas por Internet. *Tradumática, 0*. Retrieved August 25, 2009, from http://www.fti.uab.es/tradumatica/revista/articles/jabaitua/art.htm

Austermühl, F. (2001). *Electronic tools for translators*. Manchester: St. Jerome.

Biau, G., Ramón, J., & Pym, A. (2006). Technology and translation (a pedagogical overview). In Anthony Pym, Alexander , Perekrestenko, and Bram Starink (Eds.) Translation technology and its teaching (with much mention of localization). International Cultural Studies Group, Universitat Rovira y Virgili. Retrieved August 25, 2009, from http://www.tinet.cat/~apym/on-line/translation/BiauPym_TechnologyAndTranslation.pdf

Corte, N. (2000). *Web site localisation and internationalisation: A case study*. MsC thesis, City University London, London.. Retrieved August 25, 2009 from http://www.localisation.ie/resources/Awards/Theses/Theses.htm

Corte, N. (2002). Localización e internacionalización de sitios web. *Tradumática 1*. Retrieved August 25, 2009, from http://www.fti.uab.es/tradumatica/revista/articles/ncorte/art.htm

Dohler, P. N. (1997). Facets of software localization: A translator's view. *Translation Journal 1* (1). Retrieved August 25, 2009, from http://accurapid.com/journal/softloc.htm

Esselink, B. (2000). *A practical guide to localization*. Amsterdam/Philadelphia, PA: John Benjamins Publishing.

Esselink, B. (2003). The evolution of localization. Retrieved August 25, 2009, from Multilingual Computing, Inc. Web site: http://www.multilingual.com/articleDetail.php?id=646

Herrmann, A. & Florian, S. (2006). *Passolo 6.0*. [computer software]. Bonn: Pass Engineering.

Lingo Systems (2000). *The guide to translation and localization: Preparing products for the global marketplace*. Portland, OR: IEEE Computer Society.

LISA (2007). *The globalization industry primer: An introduction to preparing your business and products for success in international markets*. Retrieved August 25, 2009, from http://www.lisa.org

Mangiron, C. & O'Hagan, M. (2006). Game localisation: Unleashing imagination with 'restricted' translation. *The Journal of Specialised Translation 6*. Retrieved August 25, 2009, from http://www.jostrans.org/issue06/art_ohagan.php

Melby, A. (2006). MT+TM+QA: The future is ours. *Tradumática 4*. Retrieved August 25, 2009, from http://www.fti.uab.es/tradumatica/revista/num4/articles/04/04art.htm.

Moses, M., Sridhar, S. & Mikhail, Y. (1998). *HOM 3.0*. [computer software]. New York, NY: Stern School of Business, New York University. Retrieved August 25, 2009, from http://pages.stern.nyu.edu/ ~sseshadr/hom/

Nogueira, D. (2002). Translation tools today: A personal view. *Translation Journal 6*(1). Retrieved August 25, 2009, from http://accurapid.com/journal/19tm.htm

Pym, A. (2006). Localization, training and the threat of fragmentation. Retrieved August 25, 2009, from http://www.tinet.org/~apym/on-line/translation/translation.html

Somers, H. (Ed.). (2003). *Computers and translation: A translator's guide*. Philadelphia, PA: John Benjamins.

Yunker, J. (2002). *Beyond borders: Web globalization strategies*. Indianapolis; IN: New Riders Publishers.

[1] The term CAT can be associated with a wide variety of tools and applications, but in this article it is used mainly to refer to the general concept of memory tools. Although Passolo, the selected application for the case study, is included in the category of localization tools, providing a series of additional features (such as bitmap editor), translation memories are a core function of these applications (Austermühl, 2001, p. 146).

[2] In other operating systems, such as Linux or Mac OS, this process is performed in a different way. However, this paper focuses on the Windows platform because it is the standard in the field of software localization. New initiatives and research lines on localization of Mac, Linux and open source software are still needed and would contribute to enlarging the range of possibilities for translators.

[3] The case study was carried out before the release of SDL Passolo 2007, so no references are given to the new versions. However, it is notable that the latest release (Passolo 2009) shares the core features mentioned in the paper to support translators. Additional add-ons and characteristics (such as integration with Trados and Multiterm and streamlined and user-friendly interface) require further evaluation.

[4] In order to localize the software, permission was requested from the authors of HOM.

[5] 35 students were sent a questionnaire to appraise the localized version of HOM after one month using the program.