El software

PID_00250671

Manuel Mata Pastor

Universitat Oberta de Catalunya

Índice

Introducción

En este módulo, se estudian los aspectos fundamentales del elemento central de un producto informático, su software, y se analizan su proceso de localización y las estrategias y herramientas más habituales.

Además, se persigue que el estudiante se familiarice con algunas nociones básicas de diseño y programación de aplicaciones. Así, estará en mejor disposición de entender las peculiaridades y estructura de un programa informático y de ofrecer un servicio integral que, amén de la traducción propiamente dicha, incluya todos los ajustes necesarios para localizar íntegramente una aplicación informática de mediana complejidad.

Objetivos

Tras la lectura de este módulo y la realización de los ejercicios y actividades que lo acompañan, el estudiante debería haber alcanzado los siguientes objetivos:

1. Profundizar en las principales diferencias entre un proyecto convencional de traducción y uno de localización.

2. Familiarizarse con los conceptos básicos de diseño y programación de aplicaciones informáticas necesarios para entender las peculiaridades y estructura de un software.

3. Entender las implicaciones técnicas, comerciales, lingüísticas y culturales que llevan aparejadas la localización y la internacionalización de un producto informático.

4. Identificar los componentes del software de un producto informático y los materiales que integran un proyecto de localización, así como los formatos de archivo más habituales.

5. Aplicar los principales criterios y mecanismos para reconocer y diferenciar los elementos localizables y los no localizables del software de un producto informático.

6. Entender y aplicar las estrategias y prácticas habituales, y evaluar y utilizar distintas herramientas para localizar el software de un producto informático.

7. Comprender el procedimiento de localización de la interfaz y los demás componentes que integran el software de un producto informático, y saber cómo se localizan en un entorno de programación, con un editor de recursos, con una herramienta integral de localización y con otras herramientas genéricas o especializadas.

1. Algunas nociones básicas

Antes de abordar la problemática que plantea la localización del software de un producto informático y las estrategias y herramientas más comunes para llevarla a cabo de manera integral, a continuación se resumen, a título introductorio, algunas nociones fundamentales sobre programación.

Ved también

Es aconsejable asimismo repasar los conceptos estudiados en el módulo «Conceptos básicos» de esta asignatura y, en particular, su apartado 3, «Características definitorias de la localización».

El concepto de *software* suele definirse como «conjunto de instrucciones y reglas que permiten llevar a cabo determinadas tareas en una computadora (hardware)».

El software es, por tanto, el **componente lógico o intangible** de un equipo informático, que está integrado por una serie de dispositivos internos (procesador, memoria, tarjetas, etc.) y de periféricos (monitor, teclado, ratón, etc.) que, sin la ayuda del software, no son más que aparatos inertes. No obstante, esta definición admitiría incontables matizaciones según a qué tipo de «instrucciones» o «reglas» nos refiramos y en qué clase de equipo o dispositivo se ejecuten estas. Ya se trate de un procesador de textos, de la interfaz de un teléfono móvil o de un sofisticado sistema de gestión del control aéreo, la función del software es actuar como intermediario entre una persona y una máquina, o entre máquinas.

Así, *software* podría considerarse **hiperónimo** de términos como *aplicación*, *comando*, *dispositivo lógico*, *instrucción*, *macro*, *orden*, *programa*, *rutina*, *script* o *utilidad* entre otros, los cuales restringen y matizan el significado de *software* según su cometido y entidad. A grandes rasgos, cabría clasificar el software en **tres grandes familias**:

Otros calificativos del software

Atendiendo a **otros factores**, como las condiciones de creación, modificación, uso o distribución del software, se emplean otros calificativos como «de código fuente abierto» (open source), «comercial» o «privativo» (proprietary), «compartido» (shareware), «gratuito» (freeware), «libre» (free software), «con limitaciones funcionales» (crippleware), «con publicidad» (adware), «de prueba» (demo), etc.

- Los **sistemas operativos** gestionan las operaciones básicas de un ordenador, controlan los dispositivos y periféricos que integran un equipo informático y actúan de intermediario entre el microprocesador de una computadora y las aplicaciones que se instalen en ella, de modo que se hacen cargo de la coordinación y ejecución de las actividades más comunes. Por lo general, se diseñan de manera específica para un equipo informático de determinadas características. Entre los actuales más conocidos están MS-DOS, Mac OS, Linux, Unix o Windows para ordenadores portátiles y de sobremesa; o Android, iOS o Windows Phone para dispositivos móviles, como teléfonos inteligentes o tabletas.

- Los **lenguajes de programación** son sistemas formales de codificación que sirven para escribir programas utilizando distintas técnicas y estrategias. Cuanto más se acerque el lenguaje en cuestión al código que entiende un

procesador (lenguaje máquina) o al que utilizamos los humanos (lenguaje natural) para comunicarse con él, los lenguajes de programación son, respectivamente, de **bajo, medio o alto nivel**. Para traducir un programa escrito en un lenguaje de bajo nivel a código binario –o sea, ceros y unos–, se utilizan **ensambladores**; mientras que, en el caso de los lenguajes de alto nivel, se emplean **compiladores**. Existen, además, **lenguajes de programación interpretables**, con los cuales es posible escribir programas que no hace falta convertir a lenguaje máquina –o sea, compilar– porque se interpretan y sus instrucciones se procesan sobre la marcha mientras se ejecutan, como sucede, por ejemplo, con los lenguajes de etiquetado (*mark-up languages*) o los de *script*.

- Los **programas de aplicación (o aplicaciones)** se diseñan para la resolución de un problema concreto o para la ejecución de una serie de tareas específicas, como sucede con los procesadores de textos, las hojas de cálculo, los programas de presentación o los gestores de bases de datos. En esta tercera categoría cabría englobar también otros programas de menor entidad, como los **accesorios** (un editor de textos o de imágenes básico con las prestaciones mínimas) o las **utilidades** (caso de los antivirus o los programas de compresión y descompresión de archivos con una finalidad muy específica).

De las tres categorías anteriores, lo habitual es que **se localicen tanto sistemas operativos como aplicaciones**, puesto que los lenguajes de programación no se localizan, al estar creados a partir de constructos inalterables, casi siempre basados en el inglés, que se usan en esa lengua.

A riesgo de simplificar, cabría dividir **el proceso de programación de una aplicación** en varias etapas: las fases previas de planificación para identificar, recopilar y analizar necesidades y requisitos; y las fases de programación propiamente dicha, que abarcan el diseño, desarrollo y codificación (incluidas la depuración y puesta a prueba), e implantación y posterior mantenimiento.

Desde el punto de vista del localizador, el último eslabón de la cadena, la **compilación**, es probablemente la tarea que en mayor medida condiciona la elección tanto de la estrategia idónea como de las herramientas más adecuadas

Entorno de programación

Debe quedar claro que una cosa es el lenguaje de programación en sí (BASIC, C y sus derivados, COBOL, Pascal, Perl, Python, etc.), en el cual se redactan las instrucciones de un programa –«tirando» o «picando código», como se dice coloquialmente–, y otra es la herramienta, o el entorno de programación (*programming environment*), que permite crear y compilar programas. Tanto los entornos como la documentación que los acompaña (en formato electrónico o impreso) se pueden y suelen localizar, como si de cualquier otro producto informático se tratara.

para localizar un producto informático. Desde su perspectiva, lo recomendable es que la **internacionalización** de la aplicación esté presente en todas las fases previas de planificación y desarrollo de una aplicación, de manera que no solo se tenga muy en cuenta que esta se localizará con posterioridad, sino que se diseñe y se programe de tal modo que su ulterior localización resulte lo más fluida, eficiente y barata que sea posible.

Localización de software frente a la de contenido web

Hay que recordar que, como se explica en el módulo «Conceptos básicos», lo habitual es que los programas se construyan a partir lenguajes de programación compilables, mientras que los sitios web suelen utilizar lenguajes de etiquetado y de *script*, que se interpretan sobre la marcha y no es preciso compilar. Esta diferencia entre localización de software y de contenido web condiciona sobremanera tanto las estrategias y herramientas que se usen como la complejidad técnica de un proyecto y las competencias y herramientas necesarias para acometerlo íntegramente.

2. El proceso y sus fases

Una vez que se han elegido la estrategia y la(s) herramienta(s) más adecuadas, según los criterios explicados en los correspondientes apartados de este módulo, el proceso de localización del software de un producto informático se divide en las siguientes fases:

- **Planificación** del proyecto **y preparación** de los materiales (en particular, el *lockit*).
- **Extracción** de los recursos de su contenedor original.
- **Traducción** de los textos y ajuste de los elementos localizables.
- **Reinserción** de los recursos en su contenedor original.
- Reajuste de las dimensiones y posición de los recursos (***resizing***).
- Comprobación (**testeo**) y depuración (***debugging***).

Las fases iniciales de planificación y preparación atañen globalmente al proyecto de localización, y al software como parte de él.

Ved también

Las fases iniciales de planificación y preparación se explican en el módulo «La gestión de proyectos».

Las fases de **extracción y reinserción** son optativas, según la estrategia de localización elegida. Como se explica más adelante en este módulo, se puede decidir extraer los recursos de su contenedor original para traducirlos fuera de él (y reinsertarlos luego), u optar por localizar directamente los archivos originales en su contenedor original, tanto si son archivos compilados en formato binario como si se trata de los archivos fuente del programa.

La fase de ***resizing*** consiste en reajustar la posición y las dimensiones de los controles (botones de comando, casillas de verificación, listas desplegables, etc.) de un recurso (un cuadro de diálogo, por ejemplo) para dar cabida a los textos traducidos cuando no existía espacio suficiente en la versión original. Estos reajustes pueden hacerse **de forma individual, al mismo tiempo que se traduce** (redimensionando y reubicando cada control cuando se localiza su contenido textual), o bien **de manera global al final del proceso de traducción** (reajustando todos los recursos después de haberlos traducido). Los tres tipos de herramientas analizados en este módulo (entornos de programación, editores de recursos y herramientas integrales de localización) disponen de editores visuales que permiten llevar a cabo estos ajustes en un entorno gráfico o WYSIWYG; algunas permiten, incluso, hacerlo de manera automatizada. Por lo demás, el procedimiento es sencillo e intuitivo.

Ved también

La fase final de testeo y *debugging* se explica en el módulo «La gestión de proyectos» de esta asignatura. La fase de traducción propiamente dicha, tanto de la interfaz como de los demás elementos localizables de un programa, se explica en los subapartados que siguen.

2.1. Elementos característicos del software y pautas para su localización

A continuación, se describen e ilustran los elementos más característicos de cualquier programa informático y las principales dificultades que se plantean en la **fase de traducción**, y se proponen algunas pautas para resolverlas, con independencia de la herramienta y estrategia elegidas.

2.1.1. Los separadores

Como se explica en el módulo «Conceptos básicos», la principal dificultad que entraña localizar el software de un producto informático tiene su origen en la convivencia entre instrucciones escritas en algún sistema de codificación (en este caso, un lenguaje de programación) y texto en lenguaje natural susceptible de ser traducido. Los dos, código y texto, suelen ir separados por algún tipo de delimitador que, al diferenciarlos, hace posible, en un extremo, que el programador pueda intercalar entre las órdenes del programa textos destinados a los usuarios de la aplicación (y que, en su mayoría, se mostrarán a través de la interfaz del programa) y, en el otro extremo, que la máquina no confunda instrucciones y textos, y así pueda ejecutarlas correctamente.

> En la práctica, una de las principales dificultades a las que se enfrenta el localizador –y que solo consigue superar con mucha práctica– es diferenciar lo que se traduce y lo que no, sobre todo, cuando no existe un delimitador para separar código y texto, o cuando se usa uno que puede incluir tanto código como texto.

En tales casos, las **cadenas que plantean dudas** (por contener texto en inglés que, a primera vista, podría parecer traducible) suelen:

- Llevar todas sus letras en mayúsculas (`WRITE`) o minúsculas (`print #main, "flush"`).
- Estar formadas por varias palabras juntas (`StartButtonIcon`).
- Contener varias palabras o abreviaciones separadas por algún signo (`SYS $OUTPUT`), a menudo, un punto (`Err.Number`) o un guión bajo (`SWI "OS_Exit"`).
- Componerse de combinaciones alfanuméricas (`print #main, "place 50 50"`) o de palabras con un prefijo o sufijo numérico (`ID="Label1"`).
- Estar formadas por seudoabreviaciones (`Ctxtdelrecord`), a menudo con las iniciales de cada palabra en mayúsculas (`RecToRename`).

- Incluir el nombre de algún archivo o ruta (`F:\WINDOWS\system32\stdole2.tlb`), tipo de letra (`Name = "MS UI Gothic"`) o cualquier otro recurso del sistema.

- Emplear un segundo delimitador, diferente del utilizado para identificar el texto traducible (`"Congratulations, "+name+"."`).

- Presentar alguna combinación de las convenciones anteriores (`Compile: "FASM HELLO.ASM HELLO.COM"`).

El código informático no debe modificarse en ninguna de las situaciones anteriores, porque hacerlo puede provocar problemas funcionales en el programa o, incluso, impedir su compilación.

A pesar de que la inmensa mayoría de los programas se traducen desde el inglés hacia otros idiomas, localizar un programa cuyos textos originales estén otra lengua que no sea el inglés puede suponer paradójicamente una enorme ventaja, puesto que la pauta para identificar los elementos susceptibles de ser traducidos se reduciría a diferenciar lo que está en esa otra lengua y lo que no.

OpenPeerTV

OpenPeerTV es una aplicación para sintonizar canales de televisión a través del ordenador, y puede descargarse gratuitamente de www.peertv.eu/content-sid-2-none-0-openpeertv.html.

Por ejemplo, en las siguientes líneas extractadas del código fuente de la aplicación francesa OpenPeerTV, lo que habría que traducir es lo que está en francés; lo que está en inglés es código informático que no debe modificarse.

```
%GenreHash=();
$GenreHash{'ALL'}='Tous';
$GenreHash{'ANI'}='Anime';
$GenreHash{'CIN'}='Cinema';
$GenreHash{'DOC'}='Docus';
$GenreHash{'MIX'}='Generaliste';
$menu = new Win32::GUI::Menu(
 "&Fichier" => "File",
 " > &Ouvrir" => "openurl",
 " > -" => 0,
 " > &Quitter" => "quit",
 "Liste" => "Favs",
 " > &Favoris" => "viewfav",
```

El delimitador más habitual de los textos susceptibles de ser traducidos son las comillas rectas, (sencillas, `''`, o dobles, `""`), aunque también se emplean otros signos dobles, como paréntesis, `( )`, corchetes, `[ ]`, o llaves, `{ }`. Sea cual fuere el delimitador, no solo no debe eliminarse nunca sino que tampoco puede incluirse como carácter dentro de una cadena de texto, puesto que no se interpretaría como carácter textual sino como delimitador. Existen distintos mecanismos para evitar el conflicto de delimitadores (*delimiter collision*), como, por ejemplo, emplear dos delimitadores diferentes (`'I said "Hello, word!"'`), reduplicar el carácter en cuestión, (`"I said ""Hello,`

word!"""), codificarlo de algún modo ("I said \x22Hello, word!\x22"), o anteponerle un carácter de escape (*escape character*), como la barra invertida ("I said \"Hello, word!\""), que anule su función.

Aunque los programas de traducción asistida y de localización conocen las pautas anteriores y disponen de filtros para separar elementos textuales traducibles y código no traducible, **es necesario conocer las principales convenciones**, para comprender y resolver los errores interpretativos que pudiera cometer la herramienta elegida para localizar.

Hello, world!

Como ilustración de los mecanismos y delimitadores que emplean algunos lenguajes de programación para separar código y texto, valgan las numerosas muestras del célebre programa *Hello, world!*, disponibles, por ejemplo, en http://en.wikibooks.org/wiki/List_of_hello_world_programs o en http://en.wikibooks.org/wiki/Computer_programming/Hello_world.

2.1.2. Los comentarios

Programadores de software y diseñadores de sitios web suelen –aunque quizá no tan a menudo como sería deseable– intercalar en el código informático observaciones (*remarks*) que comentan, explican, ilustran, firman y, en definitiva, **documentan el código**. Esta práctica tan recomendable tiene, entre otras, la finalidad de facilitarle a un tercero –otro programador o un localizador, por ejemplo– la comprensión del código, para entender así qué cometido cumple cada fragmento en cada momento. Aunque los comentarios no suelen traducirse (salvo por petición expresa del cliente), el localizador se beneficia de manera indirecta de ellos, puesto que le facilitan enormemente la contextualización de las cadenas de texto que debe localizar.

Cada lenguaje tiene un carácter (comúnmente, la comilla sencilla, ', la almohadilla, #, la pleca, |, o algún signo duplicado, como el guión, --, o la barra inclinada, //), o bien un prefijo (por ejemplo, REM en el lenguaje BASIC), que se anteponen a los comentarios para identificarlos y excluirlos del flujo de código (y texto) del programa.

Por ejemplo, en las siguientes líneas de código del programa CompleteWordCount (cuya localización se propone en el ejercicio práctico P06), escrito en el lenguaje Visual Basic for Applications (VBA), se incluyen algunos comentarios en varias líneas que van encabezadas por una comilla sencilla ('). Así, muchas herramientas los reconocen como tales y los colorean, para diferenciarlos del resto del código.

```
Option Explicit

Public Function IsBounded(vArray As Variant) As Boolean
'Thanks to Astrid Zeelenberg for this (http://www.mvps.org/word/FAQs/MacrosVBA/AvailablePrinters.htm)
    'If the variant passed to this function is an array, the function will return True;
    'otherwise it will return False
    On Error Resume Next
    IsBounded = IsNumeric(UBound(vArray))

End Function
```

En este otro ejemplo, extraído del ejercicio de testeo *LocTest* (propuesto en los ejercicios prácticos P04 y P05), su autor incluye algunos comentarios (coloreados en verde), tanto dentro de un *script* escrito en el lenguaje JavaScript (anteponiéndoles dos barras inclinadas seguidas, //),

```
<SCRIPT LANGUAGE="JavaScript">
//  LocTest
//
// Autor: Rainer Schl-tterer
//  Datum: 17/10/00

[...]

function set_var1()
{
    if (var1 == "1")
    {
        alert("You've already spotted that one.\n\nThere are "+bugs+" more bugs.");
    }

    else
    {
        var1 = "1";
        var10++;
        bugs--;
    //  displayimage('images/dialogklein-de.jpg');
        alert("Correct "+name+". 'Benutzername' is not properly aligned.\n\nThere are "+bugs+" more
        bugs.");
    }   check_for_all();
}
```

como en el código HTML que lo alberga (en el que los comentarios van precedidos de <!-- y seguidos de --> para delimitarlos).

```
<map name="dialogklein-de">
<area shape="rect" coords="21,56,54,71" target="empty" onClick='set_var1()'><!--not properly
aligned-->
<area shape="rect" coords="32,85,85,100" target="empty" onClick='set_var2()'><!--not translated
'password'-->

[...]

</map>
</p>
<!--
<a href=http://www.theverybestofstuff.de/index.htm>localization</a>
<a href=http://www.theverybestofstuff.de/contents/contents.html>localization</a>

[...]

-->
</BODY>
</HTML>
```

2.1.3. Los códigos especiales

Para localizar software, conviene conocer –y, sobre todo, ser capaz de reconocer– determinados códigos especiales que aparecen intercalados en las cadenas de texto que se deben traducir. En general, estos códigos pueden desplazarse, pero no eliminarse. A continuación, se describen algunos de los más comunes:

1) Los caracteres de control (*control characters*) son códigos de un juego de caracteres que no se representan como un carácter concreto y que se utilizan, entre otras cosas, para controlar la disposición de los textos de la interfaz de un programa, espaciándolos horizontal y verticalmente. Los más comunes son `\t` (*tab*), `\n` (*new line*) y `\r` (*return*), para introducir, respectivamente, un tabulador, un salto de línea o un retorno de carro. En ocasiones, un carácter de control aparece representado por una barra invertida, `\`, seguida del código que este tenga asignado en el juego de caracteres en cuestión (por ejemplo, en el juego de caracteres ASCII: `\011` para el tabulador, `\012` para el salto de línea y `\015` para el retorno de carro).

Así, el carácter de control `\t` se emplea a menudo para separar una opción de menú de la combinación de teclas que tenga asignada (por ejemplo, la cadena `"&Open\tCtrl+O"` se visualizará como `Open Ctrl+O`). Los caracteres

de control \n y \r suelen utilizarse para dividir en líneas un mensaje de texto o para espaciarlas intercalando alguna en blanco (como en `"This is not a bug.\n\nPlease retry."`).

2) El signo *et* (&) –denominado *ampersand* en inglés– precede a la letra de un comando u opción que se convertirá en su carácter o tecla «de acceso rápido», el cual aparecerá subrayado en la interfaz de un programa (`&Abrir -> Abrir`) y, combinado con la tecla ALT, provocará su ejecución. Dado que los caracteres de activación varían de un idioma a otro, al traducir puede ser preciso desplazar el signo `&` para anteponérselo al carácter que le corresponda en la versión localizada (`&Help -> Ay&uda`).

A la hora de **elegir el carácter de activación** (*hot key* o *hit letter*, en inglés) de un comando u opción en la versión localizada de un programa, hay que tener en cuenta que los comandos y opciones más comunes de un entorno o sistema operativo ya tienen asignados un carácter por convención (como la *a* en la opción de menú `Archivo` de cualquier aplicación de Windows); cuando no es así, deben seguirse algunas pautas fundamentales:

- Dentro del mismo contexto funcional (un menú, un cuadro de diálogo, etc.), **no puede haber dos o más opciones que tengan asignado el mismo carácter de activación** (muchas herramientas disponen de funciones para controlar automáticamente la indebida duplicación de caracteres de activación).

- El carácter de activación ha de ser preferiblemente la **letra inicial mayúscula** de la opción, por resultar ésta más visible y legible.

- Es **desaconsejable** elegir como carácter de activación alguna **letra propia de un idioma** (como la *ñ* en castellano) o, en general, **caracteres acentuados** o con alguna marca diacrítica (como la diéresis), o bien aquellos para cuya reproducción sea necesario pulsar más de una tecla (conque, en la práctica, conviene ceñirse a las letras del abecedario del inglés).

- Suele ser más fácil recordar las **letras menos comunes** (si, por ejemplo, no pudiera elegirse la *e* en la opción `Exportar`, nemotécnicamente podría ser aconsejable usar la *x*).

- Es preferible elegir **letras de base ancha** (como la *m*) en lugar de otras de base estrecha (como la *f*, la *i*, o la *l*) o de **caracteres que tengan terminaciones** por debajo de su base (como la *g*, la *j* o la *p*), con el objetivo de que el carácter subrayado resulte más visible y legible.

- En algunos menús o cuadros con muchos comandos u opciones, estos se numeran para poder utilizar el **número como carácter de activación**,

como sucede cuando se enumeran al final del menú `Archivo` los últimos documentos abiertos.

3) Las **variables (*variables*)** son elementos que se definen en las instrucciones de un programa de manera que sean sustituidos por un valor cuando se cumplan determinadas condiciones durante la ejecución de la aplicación.

Así, en la cadena `"This is not a bug "+name+". \n\nThere are "+bugs +" more` **`bugs.`**`"`, las variables `+name+` y `+bugs+` son contenedores (*placeholders*) que albergarán sus respectivos valores (definidos en las instrucciones del programa) cuando el mensaje se muestre en la pantalla; aquí, el nombre del usuario y un número.

Es habitual que las variables de un programa se definan mediante una palabra precedida y/o seguida de un signo (el + en el mensaje anterior, o los símbolos `$` o `#` en otros lenguajes), o prefijada con el símbolo de tanto por ciento (`%`) seguido de una letra que indica el tipo de valor por el que será reemplazada la variable. Así, se encuentran intercaladas en las cadenas de texto variables como `%d` (*decimal integer*), `%p` (*page number*), `%s` (*string*) o `%u` (*unicode character*), que respectivamente serán sustituidas por un número entero decimal, un número de página, una cadena de texto o un carácter Unicode, como en el mensaje `"The file %s has been processed. %d records read."`.

A menudo, las variables dificultan el proceso de traducción y obligan al traductor a reestructurar el orden oracional, o le plantean problemas de concordancia de género y número, como los que entrañaría traducir una cadena como `"%d %s%s copied."`, que pudiera arrojar resultados como `"0 files copied"`, `"1 folder copied"`, etc.

> Al enfrentarse a códigos como los anteriores o a otros similares cuando se localiza un programa, amén de proceder con cautela y minuciosidad, no ha de olvidarse que un lenguaje de programación es un sistema formal de codificación sujeto a normas muy estrictas, en el que un carácter, un signo o un simple espacio cumplirán con toda probabilidad un cometido. Es aconsejable, por tanto, no manipular ni, mucho menos, eliminar aquello cuyo significado se desconozca, por más que aparente ser una errata o algo sin sentido o que está de más.

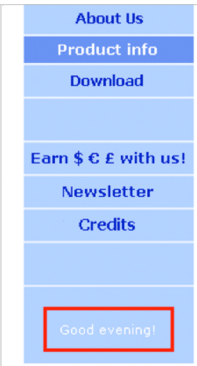
2.1.4. La concatenación

La **concatenaciónde secuencias de texto para construir mensajes** durante la ejecución de un programa constituye una práctica de programación muy común por su eficiencia y por la economía de recursos que supone. Sin embargo, paradójicamente se convierte en una de las peores pesadillas del loca-

lizador. Consiste en incluir de manera separada en el código varias piezas textuales con las que es posible generar diferentes mensajes cuando se cumplan determinadas condiciones.

Por ejemplo, las siguientes líneas de código JavaScript consiguen que, cuando alguien visite la página web en la que van incluidas como *script* (www.catcount.com), se visualice un saludo (***Good morning***, ***Good afternoon***, ***Good evening*** o ***Good night***) que se ajusta dinámicamente según la hora que marque el reloj del ordenador del usuario. El saludo se construye a partir de las cadenas `"Good"`, por un lado, y `"morning"`, `"afternoon"`, `"evening"` o `"night"`, por otro, en cuatro posibles combinaciones.

```
datetoday = new Date();
timenow=datetoday.getTime();
datetoday.setTime(timenow);
thehour = datetoday.getHours();
if (thehour > 18) display = "evening";
else if (thehour >12) display = "afternoon"
else if (thehour <4) display = "night";
else display = "morning";
var greeting = ("Good " + display + "!");
document.write(greeting);
```



Es frecuente que las cadenas concatenadas de un programa también incluyan variables, que pueden sustituirse por valores textuales, numéricos o una combinación de los dos, como en este otro ejemplo, tomado de la página principal del mismo sitio web.

```
var dayNames = new
Array("Sunday","Monday","Tuesday","Wednesday","Thursday",
"Friday","Saturday");
var monthNames = new
Array("January","February","March","April","May","June","July",
"August","September","October","November","December");
var dt = new Date();
var y = dt.getYear();
// Y2K
if (y < 1000) y +=1900;
document.write(dayNames[dt.getDay()] + ", " +
monthNames[dt.getMonth()] + " " + dt.getDate() + ", " + y);
```

Thursday, May 1, 2008

El férreo orden oracional del inglés (SVC) y su configuración morfológica y gramatical, así como las convenciones formales de puntuación de miles y decimales, fechas, etc., no siempre coinciden con los propios de las lenguas a las que se localiza un producto, y esto pone en serias dificultades al localizador cuando tiene que traducir cadenas concatenadas. Aunque, en teoría, cabría la posibilidad de retocar las instrucciones del programa para facilitar la localización de los textos, incluyendo uno por uno en el código todos los posibles mensajes concatenados, **lo más práctico suele ser optar por soluciones lingüísticas** –a veces, imaginativas– que preserven la integridad del código.

En cualquier caso, como ya se ha mencionado repetidamente, lo ideal es que se internacionalice el programa en sus etapas iniciales de diseño, evitando en lo posible el empleo de cadenas concatenadas. También constituye una prác-

tica recomendable documentar el código mediante la inclusión de comentarios que adviertan de la presencia de cadenas concatenadas y expliquen su funcionamiento.

2.1.5. El espacio disponible

Además de la delimitación y diferenciación de código y texto, y de la presencia de determinados códigos especiales, entre todos los escollos que ha de superar el localizador, destaca la **falta de espacio disponible** para acomodar los textos traducidos, que a menudo son más largos que los originales.

La escasez de espacio pone a prueba constantemente la habilidad y el ingenio del localizador, y le obliga a recurrir a soluciones lingüísticas (abreviación, calco, omisión, reformulación, sinonimia, síntesis, etc.), técnicas (reajuste de las dimensiones o ubicación de un determinado elemento) o a una combinación de las dos. En líneas generales, lo recomendable es optar por soluciones lingüísticas antes que técnicas, no solo por el menor riesgo que suponen para la integridad del código del programa, sino también porque reajustar las dimensiones de, por ejemplo, un botón puede suponer con frecuencia hacer lo propio con los demás del mismo cuadro de diálogo o incluso de todos los cuadros de diálogo similares de toda una aplicación.

En este sentido, la correcta internacionalización de las aplicaciones que pretendan localizarse resulta crucial, puesto que una misma cadena de texto puede tener **una longitud muy diferente en distintas lenguas**. En el caso de las lenguas románicas, el aumento suele rondar no menos de entre una tercera parte y la mitad de la longitud de la cadena original en inglés, pero en otros idiomas puede llegar a ser mucho mayor. No en vano, algunas recomendaciones para la internacionalización de aplicaciones aconsejan prever un aumento de espacio adicional para albergar los textos localizados de no menos del doble de la longitud de los originales.

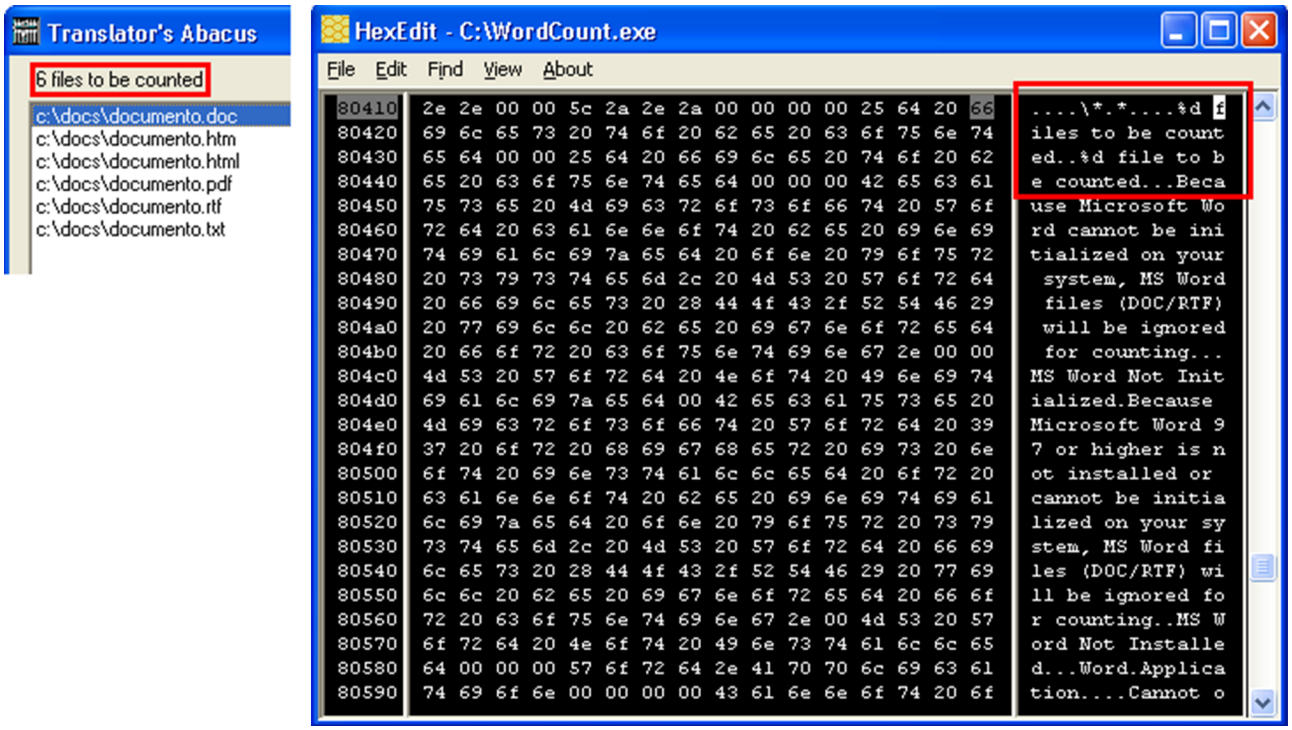
Ha de tenerse en cuenta, además, que muchas veces no se trata de frases convencionales sino de breves mensajes o palabras sueltas, en los que **el aumento del espacio necesario puede ser proporcionalmente considerable**. Piénsese, por ejemplo, en cadenas tan comunes en cualquier interfaz de usuario como el botón de comando `OK` (con dos caracteres en inglés y más del doble en muchas lenguas) o la opción de menú `Tools` (`Herramientas` duplica su longitud en español, con doce caracteres frente a los cinco del inglés).

En los albores de la localización, cuando aún era frecuente localizar programas escritos en lenguajes de bajo nivel, las limitaciones de espacio podían llegar a ser muy restrictivas, y el mero hecho de que una cadena traducida tuviera solo un carácter más que su correspondiente original o de que un archivo ejecutable

pesara un byte más podía provocar un funcionamiento anómalo o, incluso, impedir la compilación del programa. Aún hoy, no es infrecuente que, por falta de previsión, **determinadas cadenas de texto acaben incrustadas** (*hard-coded*) en las entrañas del código, y sea preciso utilizar, por ejemplo, un editor hexadecimal para modificarlas carácter por carácter.

Tal es el caso de algunas cadenas de texto del programa Translator's Abacus, cuya localización se propone en el ejercicio práctico P07; por ejemplo, las cadenas de texto `"file to be counted"` y `"files to be counted"` (que aparecen en la interfaz del programa) o de algunos de los epígrafes que encabezan los apartados del informe en formato HTML resultante del recuento de palabras.

A la izquierda, cadena incrustada (*hard-coded*) en el archivo WordCount.EXE. A la derecha, modificación de la cadena incrustada en un editor hexadecimal (HexEdit).



En una situación así, sería preciso abrir el archivo en cuestión en un editor hexadecimal para poder modificar las cadenas de texto, tomando la precaución de que la longitud del texto traducido sea exactamente la misma que la del original.

Editores hexadecimales

Algunas de las herramientas estudiadas en este módulo disponen de un editor en formato hexadecimal. También existen infinidad de editores hexadecimales gratuitos, como HexEdit.

Aunque los modernos entornos de programación permiten que algunos de los espacios destinados a alojar el texto visible en la interfaz de un programa se reajusten automáticamente (como, por ejemplo, sucede con los menús o los cuadros que enmarcan los avisos y mensajes de error), como pauta general debe procurarse **que la longitud de los textos traducidos no sea mucho mayor que la de sus correspondencias originales**. No conseguirlo pondrá al traductor en apuros para producir una traducción igualmente certera pero más

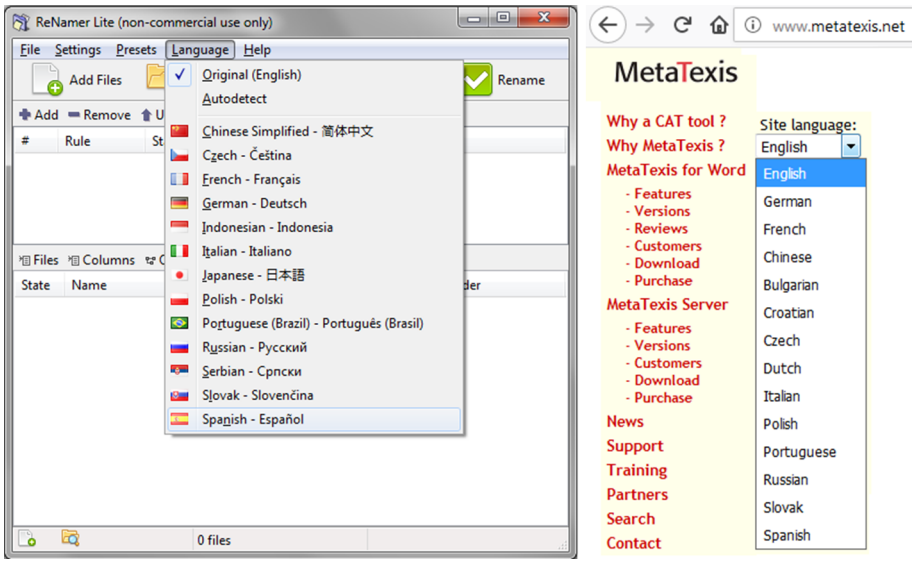
corta, y llevará aparejado en muchos casos el coste adicional de la corrección del texto traducido o del reajuste manual de las dimensiones y la ubicación de los elementos gráficos.

2.1.6. Otras dificultades

A veces, la interfaz de una aplicación incluye **listas de elementos ordenados alfabéticamente**. Como es lógico, después de traducir los elementos enumerados, es preciso reordenarlos en la lengua de destino, tal y como sucede en la traducción de un texto convencional. Sin embargo, que vayan intercalados en el código del programa puede dificultar la tarea y casi siempre obliga a retocarlo, para lo cual es necesario contar con conocimientos de programación o recurrir a la ayuda de un especialista. Esta es justamente la situación que se produce, por ejemplo, en un glosario cuyos términos estén ordenados alfabéticamente.

Así, en el siguiente cuadro de diálogo de la aplicación Translator's Abacus (cuya localización se propone en el ejercicio práctico P07), los idiomas parecen estar organizados por grupos. Si hubiesen aparecido ordenados alfabéticamente, habría sido preciso reordenarlos en la versión localizada del cuadro de diálogo, con la complicación técnica que con toda seguridad esto supondría.

Este contratiempo de la reordenación alfabética de una enumeración en la versión localizada también se produce a menudo en las listas desplegables en las que el usuario ha de elegir una opción, tan comunes tanto en los productos informáticos como en los sitios web. Un ejemplo frecuente es el de las listas de idiomas o versiones en que un software o una web están disponibles, como respectivamente se ilustra en las siguientes capturas de la aplicación ReNamer o del sitio web de la herramienta de traducción asistida MetaTexis.



Es frecuente encontrar en el código de un programa **espacios extra**, prefijados o sufijados a una cadena de texto, que en apariencia no cumplen ninguna función y parecen simples erratas. A menudo, se trata de cadenas que se concatenarán para construir mensajes durante la ejecución del programa, y el cometido de estos espacios de relleno (*trailing spaces*) es separar los elementos o sintagmas que los componen. En tales casos, por precaución, es preferible no eliminar ni desplazar los espacios, salvo que se tenga la absoluta certeza de que efectivamente son erratas o están de más.

También es fácil encontrar en las entrañas de cualquier producto informático **instrucciones y textos obsoletos** o que ni siquiera emplea el programa, y que quedaron abandonados en algún momento en la maraña de su código fuente, por distintas razones (simple descuido, un posible uso futuro en versiones posteriores del programa, reutilización de elementos o módulos prefabricados o tomados de otros programas o bibliotecas de recursos, etc.). No es extraño, por tanto, que al localizar un producto el localizador pueda tropezarse con cadenas de texto que parecen no tener sentido o no encajar en el conjunto, o que no se consiguen encontrar en la interfaz de la aplicación cuando esta se ejecuta y que, si se confirma que son obsoletas o no se usan, no sería preciso localizar.

2.2. La interfaz y sus elementos localizables

La interfaz de una aplicación es su elemento más visible al usuario, y constituye el medio que este utiliza para comunicarse con la máquina e interactuar con ella.

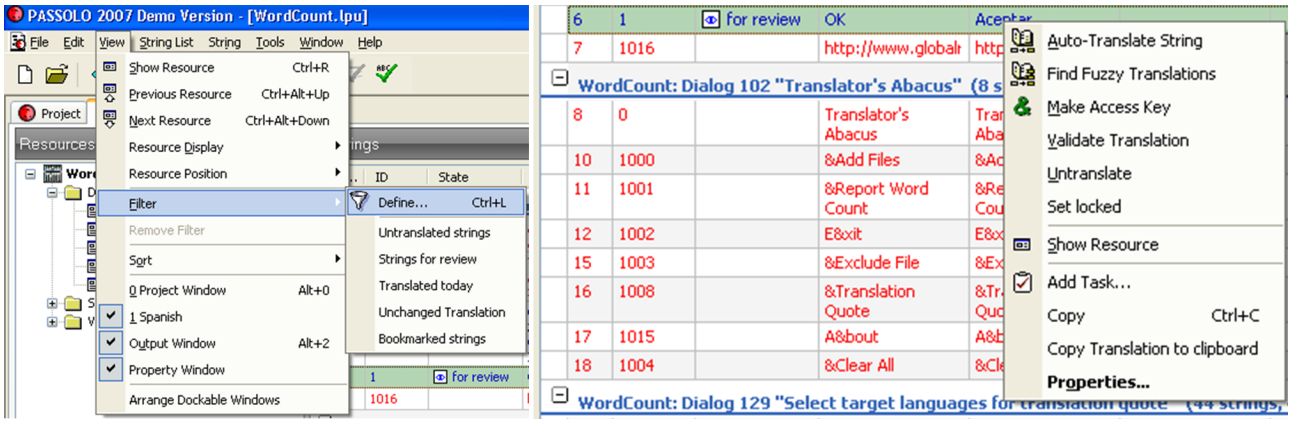
Por sencilla que sea, una interfaz suele incluir los componentes que a continuación se describen y que, en su mayoría, contienen **elementos textuales, funcionales y gráficos** susceptibles de ser localizados. Es frecuente, como sucede en las herramientas integrales de localización, que el **área de trabajo** de una aplicación se subdivida en varios paneles o ventanas.

Passolo

Las imágenes de este subapartado que ilustran los componentes de la interfaz de un producto informático han sido capturadas de la herramienta de localización Passolo (versión *demo*), con el objetivo de que el estudiante se familiarice con ella.

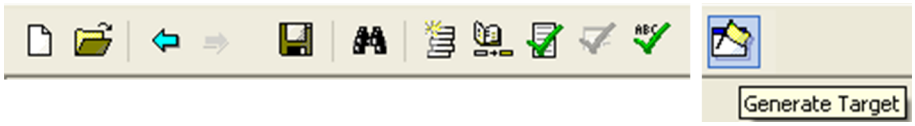
1) Los **menús** (*menus*) se agrupan en el **menú principal** de un programa (que suele disponerse horizontalmente debajo de la barra de título de la ventana de la aplicación), y reúnen en **submenús** (*submenus*) los comandos y funciones más comunes. Los **menús contextuales** (*context menus*) son variantes abreviadas de un menú, incluyen dinámicamente los comandos relacionados con la tarea que se esté haciendo en un momento dado y se activan con el botón derecho del ratón o la tecla equivalente del teclado. Por convención, las opciones de menú se escriben con inicial mayúscula, van seguidas de puntos suspensivos (...) cuando su ejecución no es inmediata, y suelen tener asignado un carácter de activación o tecla de acceso rápido que facilita su selección. A menudo, las opciones más comunes de un menú van acompañadas de la tecla de función (*function key*) o el atajo de teclado o combinación de teclas (*keyboard shortcut*) que tengan asignados y que agilizan su ejecución. Según el contexto operativo en el que se halle un usuario, algunas opciones de menú pueden no estar disponibles, en cuyo caso aparecen difuminadas en un color más claro (por lo general, gris).

A la izquierda, menú principal y submenú. A la derecha, menú contextual.



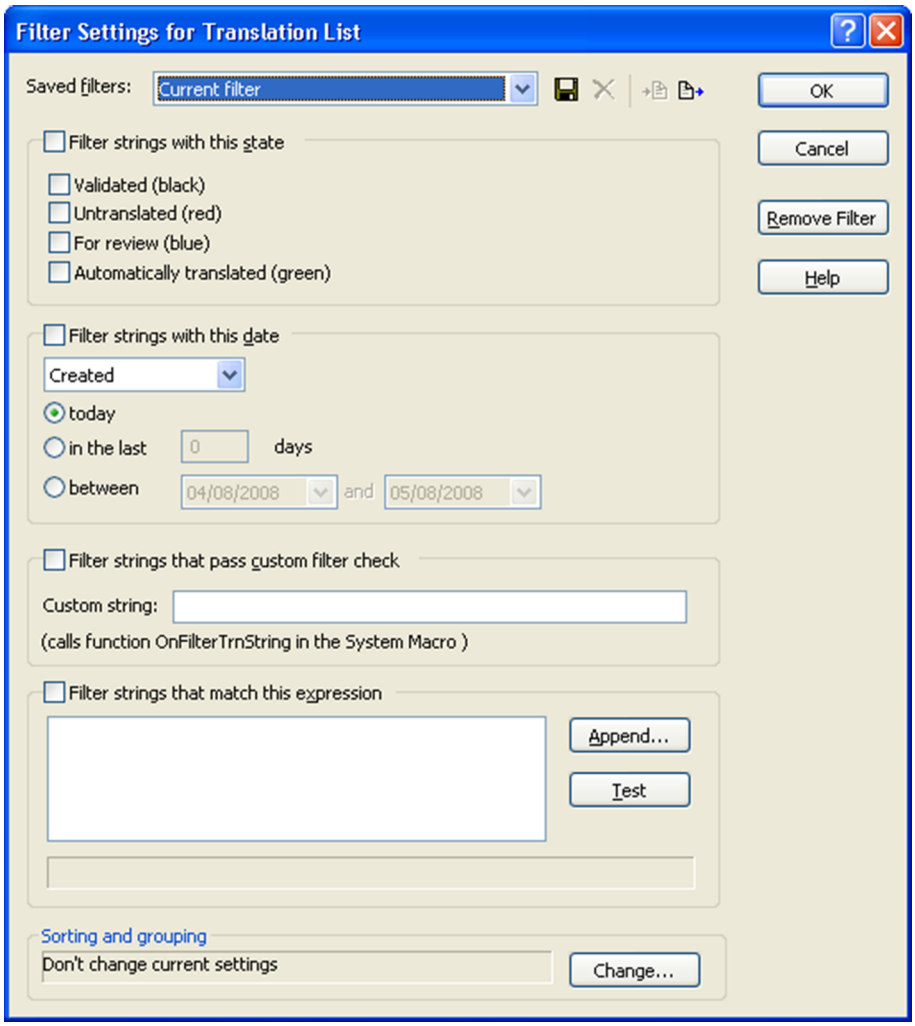
2) Las **barras de herramientas** (*tool bars*) son paneles que reúnen, en forma de iconos (*icons*), los comandos más comunes relacionados con una tarea o serie de tareas. Aunque los iconos de las barras de herramientas suelen ser representaciones gráficas, en ocasiones incluyen no solo texto o caracteres que es preciso localizar, sino mensajes explicativos (*tool tips*) que aparecen, al situar el cursor sobre ellos, en un pequeño recuadro o bocadillo, y que también hay que localizar.

A la izquierda, barra de herramientas. A la derecha, descriptor de icono (*tool tip*).



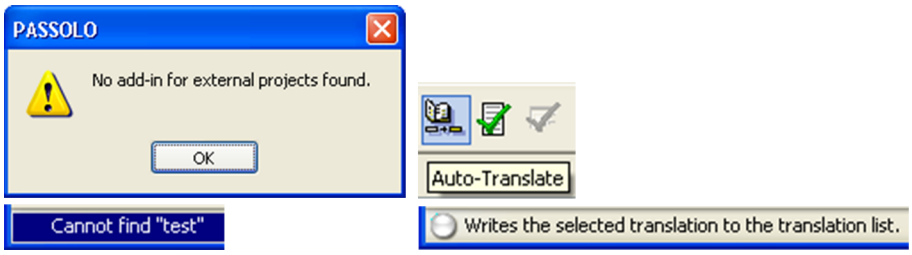
3) Los **cuadros de diálogo** (*dialog boxes*) son formularios electrónicos de forma rectangular que agrupan, en una o varias secciones identificadas mediante pestañas (*tabs*), distintas opciones relacionadas con uno o varios comandos. Como su propio nombre indica, estos cuadros le permiten al usuario «dialogar» con la aplicación e indicarle con detalle qué desea hacer a través de sus controles. Estos pueden tener forma de cuadro de texto (*text box*), casilla de verificación (*check box*), botón de opción (*radio button*), lista desplegable (*drop-down list*) o botón de comando (*command button*), entre otras. Los controles de un cuadro de diálogo también suelen tener asignado un carácter de activación (que se indica mediante el subrayado), se agrupan en marcos y están sujetos a un orden de tabulación, que permite pasar de uno al siguiente o al anterior mediante la tecla tabulador o la combinación mayúscula + tabulador, respectivamente.

Cuadro de diálogo con distintos tipos de controles (botones de acción y opción, casillas de verificación, cuadros de texto, listas desplegables, etc.).



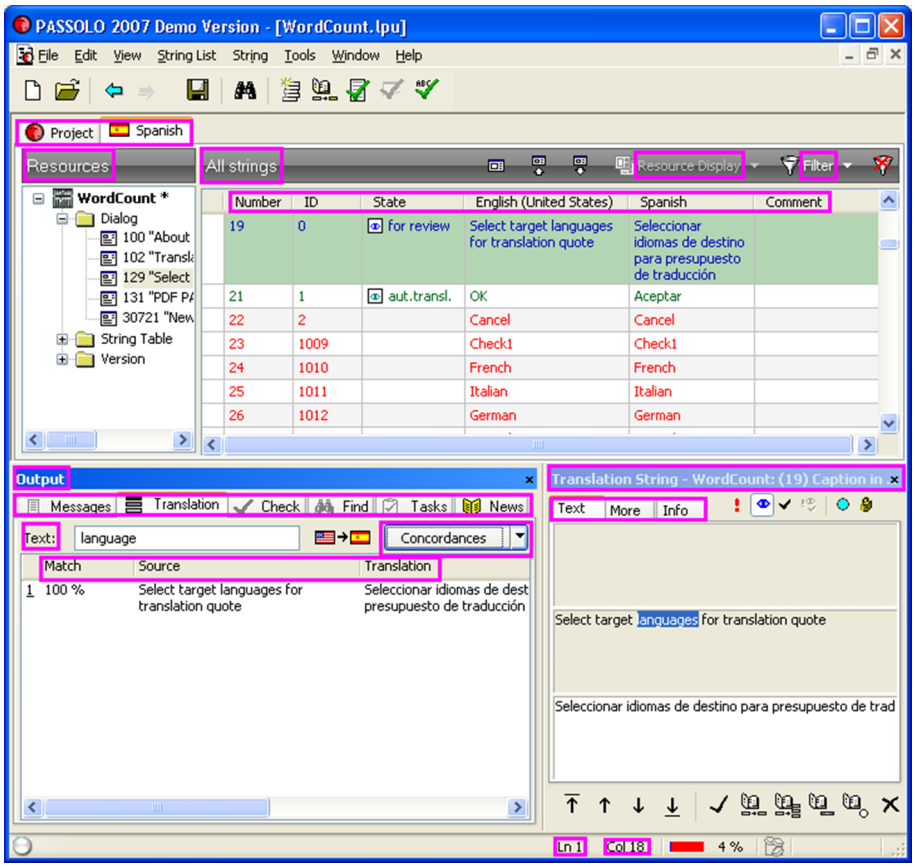
4) Además de distintos **mensajes textuales** que aparecen en forma de advertencias, mensajes de error, avisos de la barra de estado, descriptores de comandos, etc.,

A la izquierda, mensajes de error en recuadro y en la barra de estado. A la derecha, descriptor de comando en recuadro y en la barra de estado.



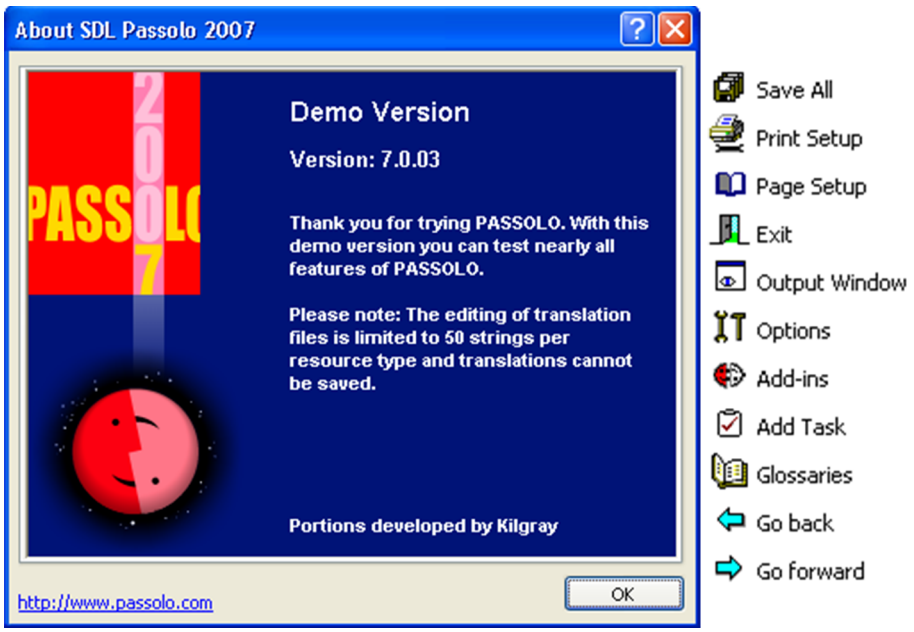
la interfaz de un programa contiene **otros textos localizables** en todos sus recovecos.

Cadenas de texto traducibles repartidas por la interfaz de una aplicación.



5) Los **elementos gráficos** son imágenes que se incluyen en distintos lugares de una aplicación en forma de mapas de bits (*bitmaps*), iconos (*icons*) o cursores (*cursors*), entre otras. Al igual que sucede con los iconos de las barras de herramientas, tanto si incluyen texto como si no, a veces es preciso localizarlos.

A la izquierda, *splash screen* con elementos textuales y gráficos. A la derecha, iconos con y sin texto que podría ser preciso localizar.



Hoy día, casi todas las aplicaciones de cierta entidad, tanto ofimáticas como especializadas, permiten un enorme grado de personalización, de modo que el usuario puede configurarlas, según sus gustos y necesidades, y modificar, suprimir o añadir cualquiera de los elementos anteriores.

Con independencia de la herramienta que se utilice para localizar la interfaz de un producto informático, y de si se opta por procesar los archivos compilados en formato binario (por lo común, EXE o DLL) o archivos de recursos (en formato RC), al abrirlos, la interfaz se divide en varios grupos o categorías de recursos, que se corresponden *grosso modo* con los tipos de elementos que la componen y que se acaban de describir. Las principales categorías de recursos en las que internamente se subdivide la interfaz de una aplicación se resumen y describen en el siguiente cuadro, por orden alfabético.

Categoría de recursos	Descripción	¿Localizable?
Accelerator	Teclas aceleradoras y de función, y atajos de teclado.	Por lo general, no.
Bitmap	Imágenes que aparecen en distintos lugares de la aplicación (por ejemplo, la *splash screen* o pantalla de presentación que se muestra al ejecutar el programa o al seleccionar la opción Ayuda > Acerca de...).	A veces (contengan o no texto).
Cursor* o *Cursor Group	Cursores y punteros.	A veces (contengan o no texto).
Dialog	Cuadros de diálogo.	Sí.
Icon* o *Icon Group	Iconos de las barras de herramientas.	A veces (contengan o no texto).
Menu	Menús, submenús y menús contextuales.	Sí.

Categoría de recursos	Descripción	¿Localizable?
String Table	Textos y mensajes variopintos que aparecen en distintos lugares del programa (mensajes informativos y de error, avisos, descriptores de comandos, etc.).	Sí.
Version* o *Version Info	Información sobre el producto (denominación, *copyright*, versión, etc.).	Solo en parte.

Las categorías `Dialog`, `Menu` y `String Table` son las que contienen el grueso de los elementos textuales localizables de la interfaz de una aplicación. Los principales problemas que plantea su localización y algunas pautas para resolverlos se describen e ilustran en subapartados anteriores.

Los elementos gráficos (mapas de bits, cursores e iconos) deberán localizarse atendiendo no solo a que incluyan o no texto, sino también al hecho de que pueda ser necesario o recomendable sustituir o adaptar las imágenes que contienen. Bien es sabido que –al igual que sucede en los sitios web– determinados iconos o imágenes pueden no ser adecuados en ciertas comunidades, culturas o religiones, por tener connotaciones negativas, resultar ininteligibles o incluso ser de todo punto inadmisibles o estar prohibidos. Cuando se pretende localizar un producto hasta sus últimas consecuencias, tales imágenes pueden adaptarse, sustituirse por otras o incluso llegar a eliminarse del producto.

Ved también

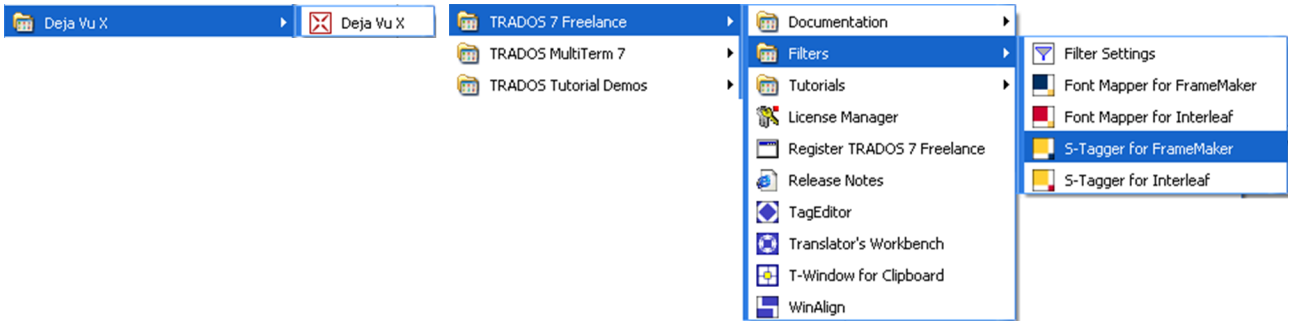
Los aspectos técnicos del tratamiento de imágenes en un proyecto de localización de un producto informático se abordan e ilustran con profusión en el módulo «Las imágenes y el contenido gráfico» y el ejercicio práctico P13.

2.3. Otros componentes localizables

Además de su interfaz, el componente principal y más visible de un programa, la mayoría de las aplicaciones de cierta entidad incluyen **otros módulos accesorios** como subprogramas, filtros, tutoriales, ejemplos, plantillas, complementos, etc. Su cometido suele ser el de explicar, complementar o ampliar las funciones de la aplicación en la que se integran. Su instalación y uso son a veces optativos, de modo que el usuario decide si quiere utilizarlos al instalar el producto.

Algunas aplicaciones concentran sus elementos, funciones y complementos en torno a una única interfaz que los centraliza –de manera más o menos perceptible para el usuario–, como sucede, por ejemplo, con Atril Déjà Vu; otras, por el contrario, son un mosaico de módulos, filtros y accesorios con distintas funciones, como ha ocurrido con las versiones de Trados anteriores a Studio, por ejemplo.

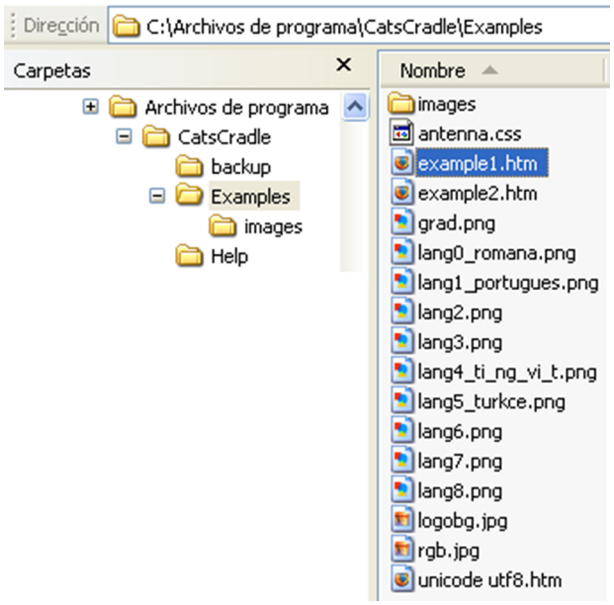
A la izquierda, Déjà Vu versión X. A la derecha, Trados y MultiTerm versión 7.



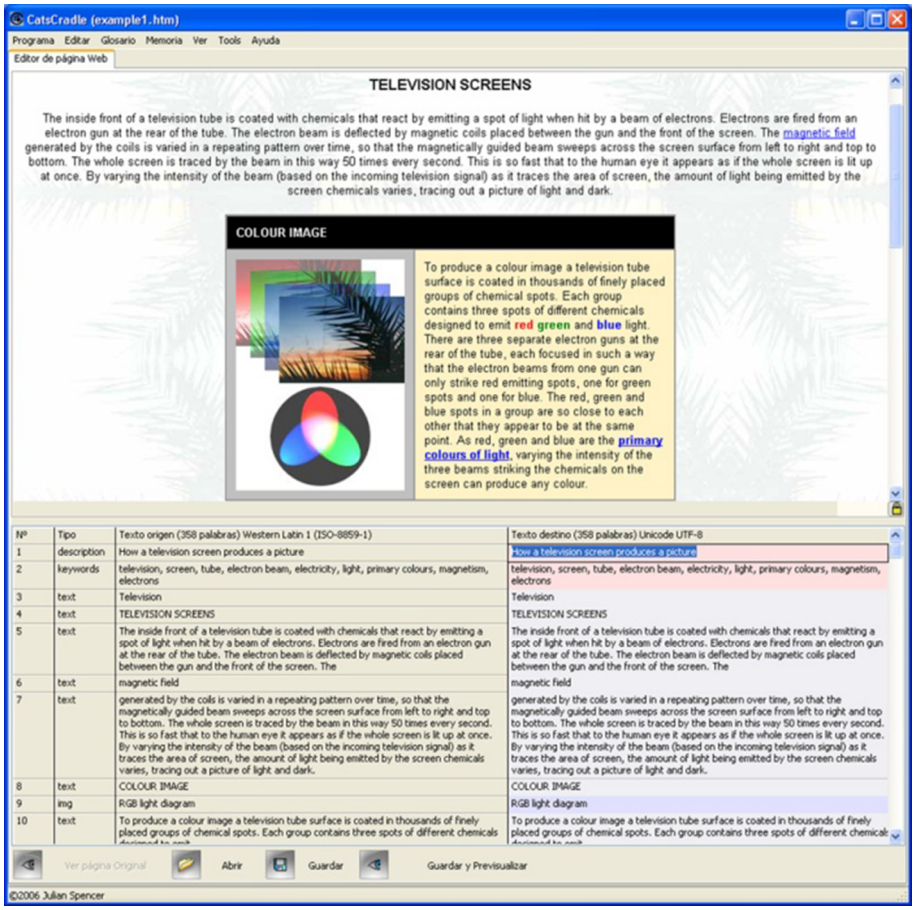
El formato de estos **componentes secundarios** es muy variable y, como es lógico, guarda una relación directa con el programa principal, por lo que la elección de la estrategia y las herramientas más adecuadas para localizarlos depende de la naturaleza de cada módulo o componente.

En ocasiones, se trata de **documentos de muestra** que pueden localizarse –cuando proceda hacerlo– con la aplicación a la que pertenecen. Así, el programa de traducción de páginas web CatsCradle incluye varios archivos HTML de muestra que podrían localizarse con la propia aplicación.

Archivos de muestra incluidos en CatsCradle.

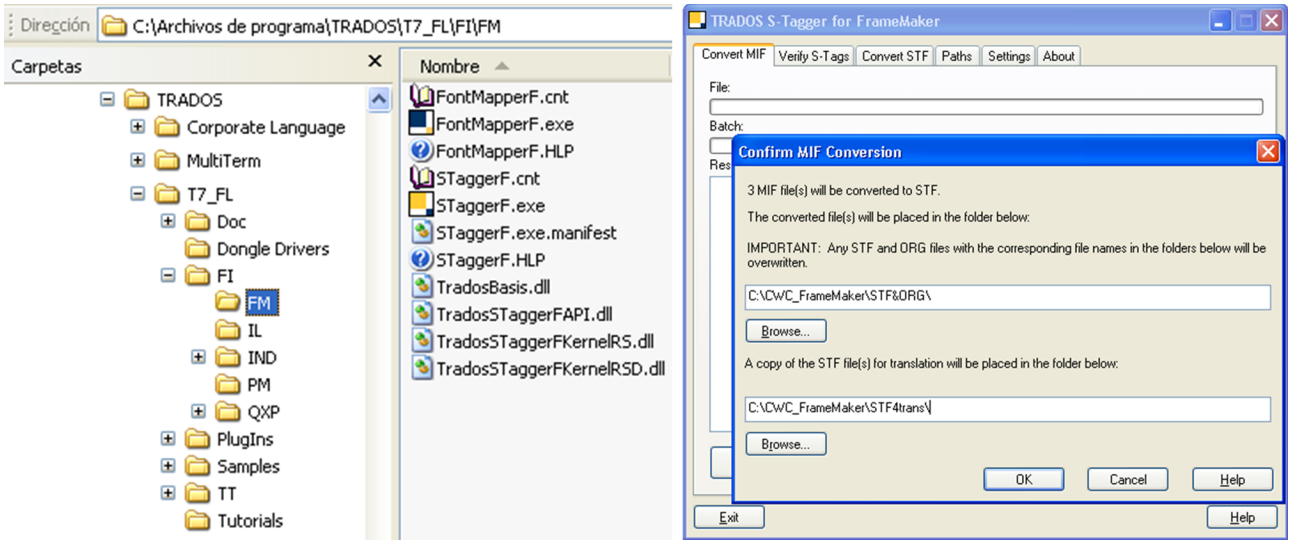


Localización de archivos de muestra de CatsCradle con la propia aplicación.



Otras veces, se trata de **microaplicaciones autónomas** que se integran en la aplicación principal e interactúan con ella –de manera similar a como lo hace el sistema de ayuda de un programa–, y para localizarlas es preciso recurrir a varias estrategias y herramientas, según las pautas propuestas en este y otros módulos y ejercicios. Así, el módulo S-Tagger for FrameMaker, del que viene provisto Trados, constituye una pequeña aplicación en sí y, por tanto, su localización pasaría por el empleo de diferentes herramientas. En concreto, la aplicación consta de varios archivos compilados en formatos EXE y DLL, y de un sistema de ayuda en formato WinHelp (HLP).

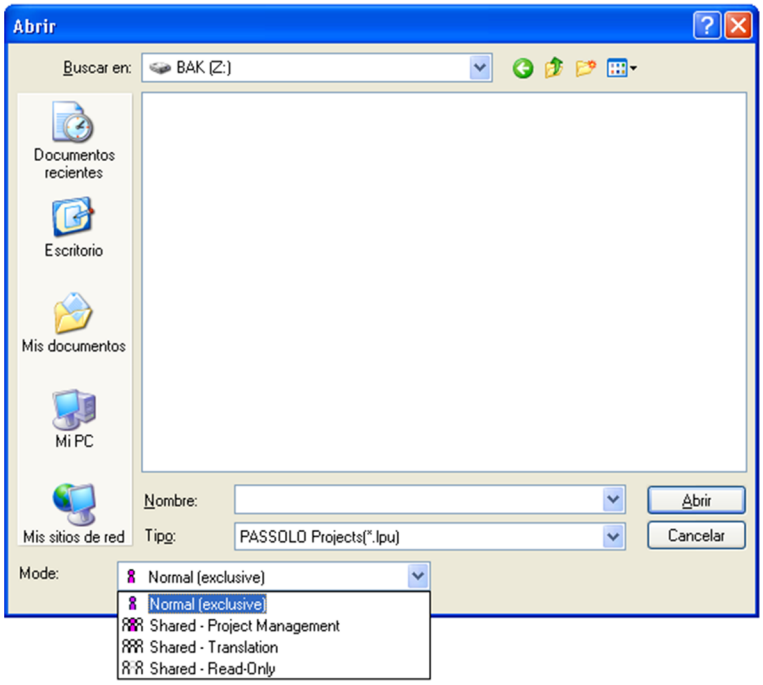
A la izquierda, archivos de la microaplicación S-Tagger for FrameMaker. A la derecha, interfaz autónoma de S-Tagger for FrameMaker.



Una práctica frecuente en la programación de aplicaciones consiste en **reutilizar recursos del sistema operativo o de otros programas**, por las ventajas que de esto se deriva, tanto al diseñar la aplicación como al ejecutarla. Esta es precisamente una de las funciones de los **archivos DLL** (*Dynamic Link Library*), que son bibliotecas de recursos que pueden compartir varias aplicaciones y que contienen elementos de uso común.

Piénsese, por ejemplo, en los cuadros de diálogo más habituales en cualquier aplicación, para abrir un documento, guardarlo, imprimirlo, etc. Cuando un programa se sirve de recursos externos, puede darse la circunstancia de que en su interfaz aparezcan **elementos en dos idiomas**; el del programa en sí y el del sistema operativo en el que se está ejecutando o el del recurso compartido que está reutilizando. Esto es justamente lo que sucede en el siguiente cuadro de diálogo del programa Passolo, en inglés, cuando se ejecuta en un sistema operativo en castellano.

Cuadro de diálogo de la aplicación Passolo en inglés ejecutándose en Windows en castellano.



Como pauta general, lo recomendable es utilizar durante el proceso de localización un sistema operativo que esté en el mismo idioma al que se esté localizando, de manera que se reproduzcan, en lo posible, **las circunstancias reales** en las que el usuario final ejecutará y utilizará el producto localizado.

Por análogas razones, al llevar a cabo el testeo de una aplicación informática o de un sitio web, lo ideal es que su comprobación funcional se haga, como mínimo, ejecutándolos en **los sistemas operativos y navegadores más comunes**, tal y como hacen los grandes fabricantes de software. De lo contrario, se corre el riesgo de no detectar posibles problemas funcionales del producto en algún entorno concreto.

3. Las estrategias habituales

La estrategia elegida para localizar un producto informático depende de infinidad de factores que atañen a **la naturaleza del producto** (su arquitectura informática, envergadura y complejidad, el volumen y distribución del texto, etc.), a **la relación con el cliente** (que puede o no facilitar los archivos fuente sin compilar, e incluso herramientas o utilidades desarrolladas *ad hoc*), o a **los recursos materiales y humanos disponibles** (y a la capacitación de estos), entre otras variables.

A la postre, la estrategia que se adopte condicionará tanto **el nivel de competencia técnica** exigible al traductor/localizador como **el número y la complejidad de las herramientas** necesarias para llevar a buen puerto el proyecto de localización, empresa necesariamente colectiva en la inmensa mayoría de los casos.

Lo habitual es que se opte por una de las dos estrategias más implantadas en el sector, que se explican e ilustran a continuación, o por una combinación de las dos.

3.1. Estrategia 1: extracción de los recursos localizables

La primera estrategia consiste en **externalizar los textos localizables**; bien *ex ante*, internacionalizando el producto desde su fase de diseño, bien *ex post*, durante el proceso de localización. Para ello, es preciso:

- **Ubicar los elementos localizables en archivos diferentes** de los que contienen el código, o **extraerlos** separándolos del código informático para preservar su integridad.

- **Traducirlos** con un simple editor o procesador de textos, una hoja de cálculo o un sistema de memorias de traducción (procesándolos en algún formato ofimático o de intercambio).

- **Reinsertarlos** en su ubicación original, una vez traducidos (a menudo, de manera automatizada).

- **Retocar las dimensiones y ubicación** (*resizing*) de los elementos del programa que lo precisen o hayan sufrido alguna alteración tras ser localizados, y **detectar y enmendar los desajustes visuales y funcionales** (testeo y *debugging*) producidos durante la traducción.

Resizing

Como ya se ha explicado, el proceso de *resizing* también puede hacerse de manera simultánea al de traducción, de modo que se vayan retocando las dimensiones y la ubicación de los controles de los recursos que lo requieran a medida que se va traduciendo su contenido textual.

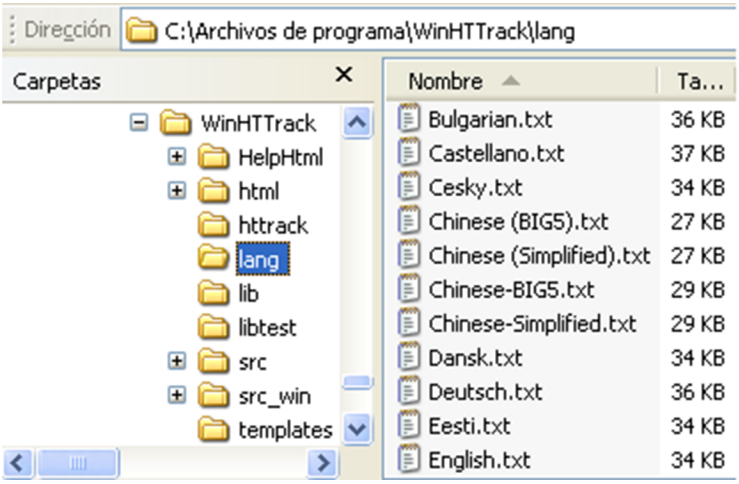
Así, por ejemplo, el programa de software libre para la aspiración de sitios y contenidos web HTTrack está bien internacionalizado, porque se diseñó tomando en consideración desde el principio la posibilidad de localizarlo luego –o eso parece cuando se analizan los archivos que lo componen.

HTTrack

Como se menciona en el módulo «Los sitios y el contenido web» y en sus ejercicios prácticos, HTTrack puede resultar muy útil para descargarse, aspirándolos total o selectivamente, los contenidos de un sitio web estático para, por ejemplo, analizarlos y preparar una oferta de servicios de localización.

Con el objetivo de facilitar el proceso de localización, su autor externalizó de antemano los recursos localizables y los alojó en un archivo de texto. De este modo, para localizar la interfaz del programa, basta con **duplicar el archivo (de texto sin formato) y traducir las cadenas de texto** que contiene, teniendo la precaución de intentar que, en lo posible, la longitud de los textos traducidos no sea (mucho) mayor que la de sus correspondientes originales.

A la izquierda, estructura de carpetas y archivos del programa HTTrack. A la derecha, extracto del archivo localizado \WinHTTrack\lang\Castellano.txt.



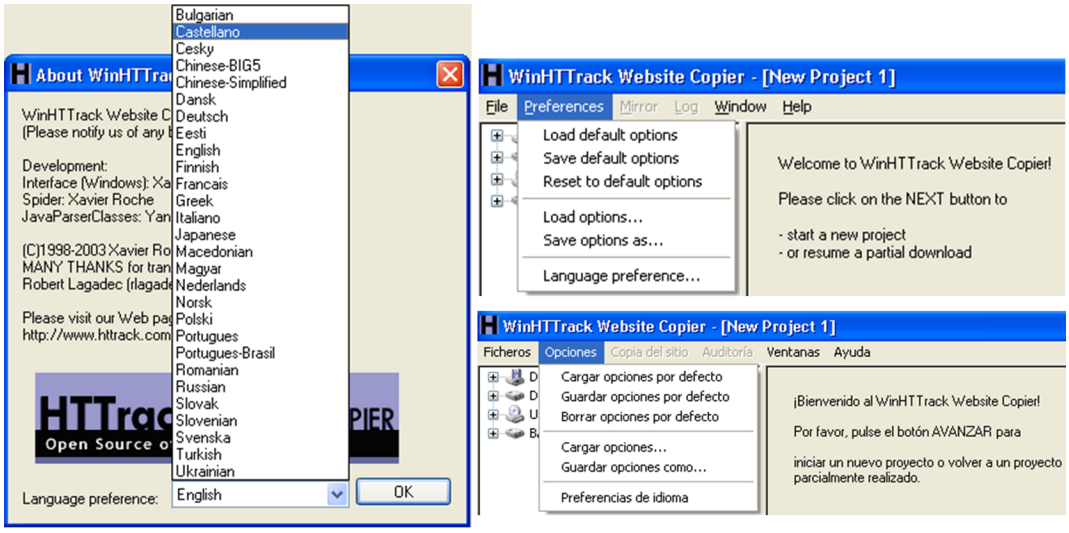

```
735 Nuevo proyecto\tCtrl+N
736 &Open...\tCtrl+O
737 &Abrir...\tCtrl+O
738 &Save\tCtrl+S
739 &Guardar\tCtrl+S
740 Save &As...
741 Guardar &como...
742 &Delete...
743 &Borrar...
744 &Browse sites...
745 &Explorer sites...
```

En este ejemplo, se observan algunos de los elementos característicos de código informático que sigue intercalado en el texto traducible, y que debe tratarse con sumo cuidado durante el proceso de localización, como se explica en distintos subapartados de este módulo y se ilustra en sus ejercicios prácticos.

En este caso, los elementos localizables son la segunda copia de cada cadena de texto.

Esta manera de diseñar un programa no solo **facilita enormemente el proceso de localización** –y reduce de forma drástica sus costes–, sino que también agiliza el cambio de lengua de su interfaz. Así, en HTTrack es posible instalar una única versión del programa y luego elegir el idioma en el que se prefiera visualizar su interfaz. Las instrucciones de la aplicación buscan en la subcarpeta `\lang` los textos de la lengua elegida y los cargan de manera dinámica, sin tener que volver a compilar el programa ni instalar o disponer de versiones diferentes de la aplicación para cada idioma.

A la izquierda, cuadro de diálogo del programa HTTrack para elegir el idioma de la interfaz. A la derecha, extracto de la interfaz de HTTrack original en inglés y localizada al español.



Lo más común, sin embargo, es que los recursos localizables se integren junto con el código en uno o varios **archivos compilados** (por lo general, en formato EXE o DLL). En tal caso, puede optarse por una segunda estrategia –que se explica más abajo– o por extraerlos *a posteriori*, por ejemplo, a un **archivo en formato RC**. Un archivo de recursos en formato RC es un documento de texto sin formato (*text-only*) que puede abrirse y editarse con infinidad de programas: desde un editor de textos hasta casi cualquier herramienta de traducción asistida o de localización.

Resource Hacker

En el ejemplo del ejercicio práctico P05, el editor de recursos Resource Hacker cuenta con funciones para extraer, respectivamente, un determinado recurso (menú Action > Save [X] as *.rc file), una categoría concreta (Action > Save [X] resources...) o incluso la totalidad de los recursos del programa (Action > Save all resources), para localizarlos con la herramienta que se prefiera y reinsertarlos luego en su ubicación original.

Cuando un profesional de la localización trabaja para una empresa de traducción y no para el cliente final, es muy habitual que, en lugar de enviarle al localizador archivos binarios compilados (en formato EXE o DLL, por ejemplo) para que los modifique con un editor de recursos o con una herramienta integral de localización, se le manden uno o varios archivos en formato RC para que pueda traducirlos, por ejemplo, con un sistema de memorias de traducción. De este modo, el traductor, además de beneficiarse de las ventajas de una herramienta de traducción asistida (aprovechamiento de repeticiones y coincidencias, apoyo para garantizar la coherencia terminológica, fraseológica, estilística, etc.), no tiene por qué contar con los conocimientos técnicos mínimos que requiere el manejo de un editor de recursos o una herramienta de localización.

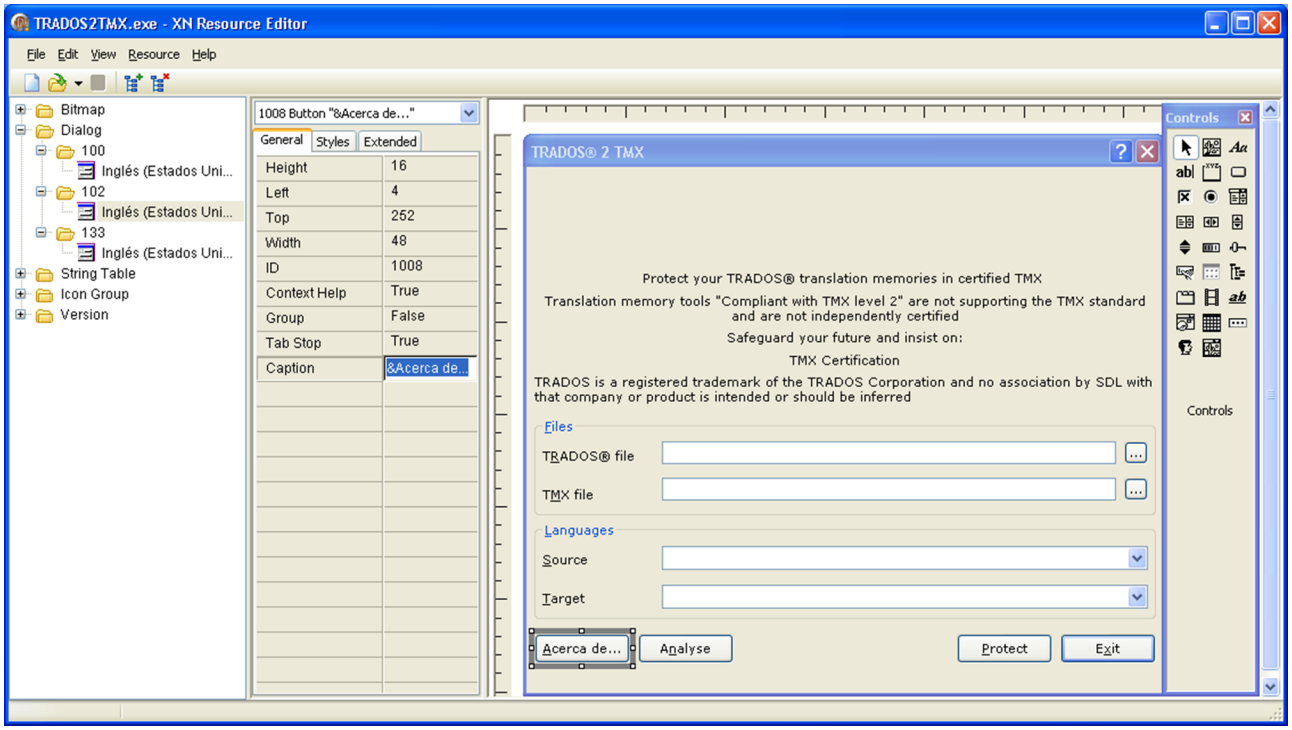
Preparación y traducción de un archivo RC

El procedimiento de preparación y traducción de un archivo RC es análogo al explicado para localizar archivos de JavaScript en el módulo teórico «Los sitios y el contenido web» y el ejercicio práctico P05.

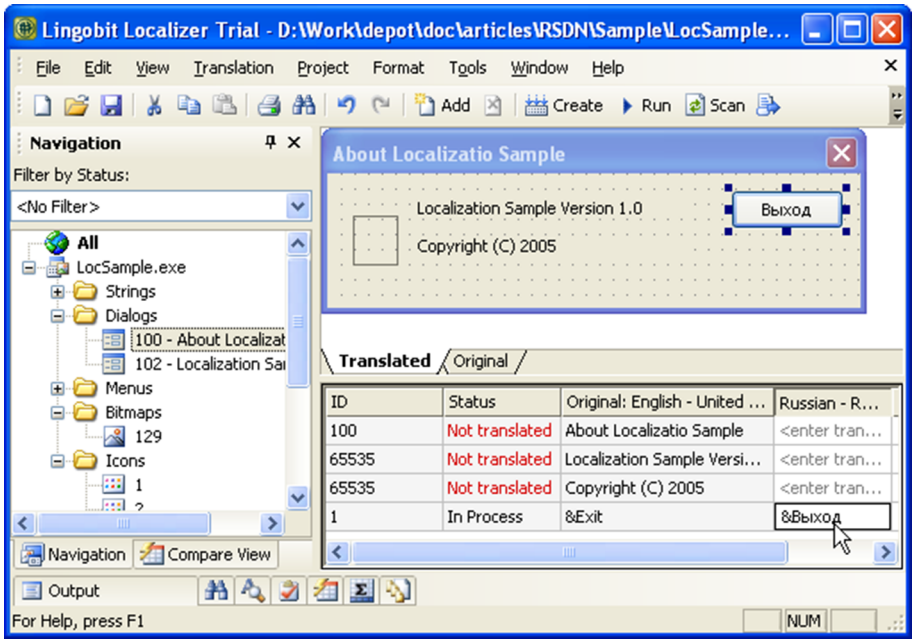
3.2. Estrategia 2: localización de los recursos en su contenedor original

Una segunda estrategia se decanta por **intervenir directamente sobre el producto compilado**, traduciendo los textos en su contenedor original (con un editor de recursos o una herramienta especializada de localización), y comprobando visualmente y retocando sobre la marcha la versión localizada. Esta segunda posibilidad, algo más cómoda e intuitiva pero también más laboriosa y arriesgada, lleva aparejado para el traductor un mayor grado de subordinación del texto al código y a su entorno visual, amén de exigir una competencia técnica más amplia y el empleo de herramientas especializadas. Hoy existe un sinfín de herramientas que permiten procesar y localizar archivos binarios, tanto editores de recursos (por ejemplo, XN Resource Editor) como programas integrales de localización (por ejemplo, Lingobit Localizer).

Archivo Trados2TMX.EXE abierto en el editor de recursos XN Resource Editor.



Interfaz de la herramienta de localización Lingobit Localizer.



Unos y otros –cuyas características se describen con detalle más abajo– permiten **ver el resultado final en el acto**, a medida que se traduce. A diferencia de lo que sucede cuando se localizan textos previamente extraídos, el marco visual ayuda mucho a contextualizar las cadenas que se han de traducir.

3.3. Combinación de varias estrategias

No es infrecuente que se den **secuencialmente las dos estrategias** comentadas. Así, un fabricante de software podría entregarle a un (gran) proveedor de servicios lingüísticos multilingües (MLV) los archivos compilados de un programa (acompañados o no de los archivos fuente a partir de los cuales se han generado), de modo que este extraiga los recursos localizables, los procese para transferírselos a sus traductores en algún formato ofimático (TXT, CSV, RTF, XLS, etc.) o de intercambio (XLIFF), y vuelva a reinsertarlos en su contenedor original una vez localizados.

Tampoco es descartable que **las dos estrategias se combinen de manera simultánea** en un mismo proyecto de localización. Así, podría decidirse extraer el grueso de los textos de un determinado programa (los mensajes informativos y de error, habitualmente agrupados en la categoría `String Table`, por ejemplo) para subcontratarle su traducción a un profesional autónomo (por ejemplo, con un sistema de memorias de traducción); y localizar dentro de la propia empresa los recursos menos voluminosos y más supeditados a su entorno gráfico, como los menús (categoría `Menu`) y cuadros de diálogo (categoría `Dialog`), con un editor de recursos o una herramienta de localización.

Aunque en el mercado comercial resulta ciertamente insólito, **en el ámbito del software libre** sí se dispone a menudo de los archivos en código fuente (*source code*), a partir de los cuales se generó el producto compilado. En tal caso, la localización puede hacerse directamente en estos archivos, que posteriormente se compilan en la versión final del programa en formato binario.

4. Las herramientas esenciales

En programación, encontramos una serie de pautas de diseño y de patrones sintácticos y semánticos convencionalmente aceptados. Sin embargo, es tal la diversidad de lenguajes y son tantas las maneras de estructurar un programa informático, que hay muchas formas de localizarlo e infinidad de herramientas para hacerlo. A continuación, se describen las principales y sus características, atendiendo tanto a su **tipología** como a la **evolución histórica** del sector de la localización.

Lenguajes de programación

Como ilustración de la diversidad de lenguajes, véanse, por ejemplo, algunas de las exhaustivas enumeraciones recogidas en la Wikipedia (https://es.wikipedia.org/wiki/Anexo:Lenguajes_de_programación).

Antes de que existieran herramientas específicamente diseñadas para localizar o de que la traducción asistida hubiera alcanzado su actual estado de madurez, era obligado recurrir a **las mismas herramientas que utilizaban los propios programadores**, como los entornos de programación o los editores de recursos. Dado que esto exigía ciertos conocimientos de programación, a menudo la traducción acababa haciéndola un técnico con competencias lingüísticas, o bien los textos se extraían mediante algún proceso automatizado y se enviaban al traductor en algún formato ofimático o de intercambio para que los tradujera. Este no contaba la mayoría de las veces con ningún tipo de información visual ni funcional que contextualizara los textos que traducía.

Hoy la situación ha cambiado, y no solo existen **herramientas especializadas para localizar**, sino que los **programas de traducción asistida** ya incorporan las funciones necesarias para procesar los formatos más habituales en un proyecto de localización. Así, lo más frecuente es que el traductor utilice un sistema de memorias de traducción o –si sus conocimientos y su presupuesto se lo permiten– una herramienta integral de localización. En cualquier caso, siempre es aconsejable conocer y contar con otros **programas complementarios** que, con los anteriores, se describen a continuación.

4.1. Los entornos de programación

Un entorno de programación (*programming environment*) es la herramienta que los programadores utilizan para diseñar, codificar y compilar aplicaciones, la cual recibe comúnmente la denominación de **CASE (*Computer-Aided Software Engineering*)** y cubre una o varias de las fases del desarrollo de software. En general, se trata de programas complejos para cuya utilización es preciso contar con unos conocimientos avanzados de informática y mínimos de programación.

No son, por tanto, programas pensados para traducir, pero sí permiten acceder a las entrañas del código informático para modificar los textos que este pudiera contener, y así poder localizarlos. Otras ventajas de este tipo de herramientas son **la contextualización visual y funcional** de los textos que se han de

localizar (tal y como se ven al ejecutar en tiempo real el programa localizado), y la posibilidad de modificar los recursos en un entorno gráfico (WYSIWYG), por ejemplo, para reajustar las dimensiones o la ubicación de los controles que pueden integrar un determinado recurso (botón, cuadro de texto, casilla de verificación, lista desplegable, etc.).

Sin embargo, desde el punto de vista del localizador, los entornos de programación presentan como gran inconveniente el hecho de no **contar con ninguna de las funciones habituales en una herramienta de traducción asistida** (para reaprovechar traducciones anteriores, mantener la coherencia, etc.), que tan indispensables resultan hoy en un proyecto de localización, sobre todo, cuando se localizan sucesivas versiones de un mismo producto o aplicaciones similares.

Con estas herramientas, se ejerce un control total sobre el programa que se está localizando y se puede intervenir en todas las etapas del proceso de localización (rediseñando, si fuera necesario, alguno de sus elementos). Sin embargo, su empleo pasa necesariamente por poseer, además de conocimientos especializados, todos los materiales utilizados para desarrollar el producto en cuestión. Y esto no es siempre posible por razones comerciales, legales o logísticas.

A título de ilustración, a continuación aparecen las capturas de una versión del célebre juego *Space Invaders*, mientras se localiza con un entorno de programación. En este caso, se trata de un programa escrito en el lenguaje BASIC, que puede modificarse y compilarse, por ejemplo, con la herramienta gratuita Emergence Basic Development Environment.

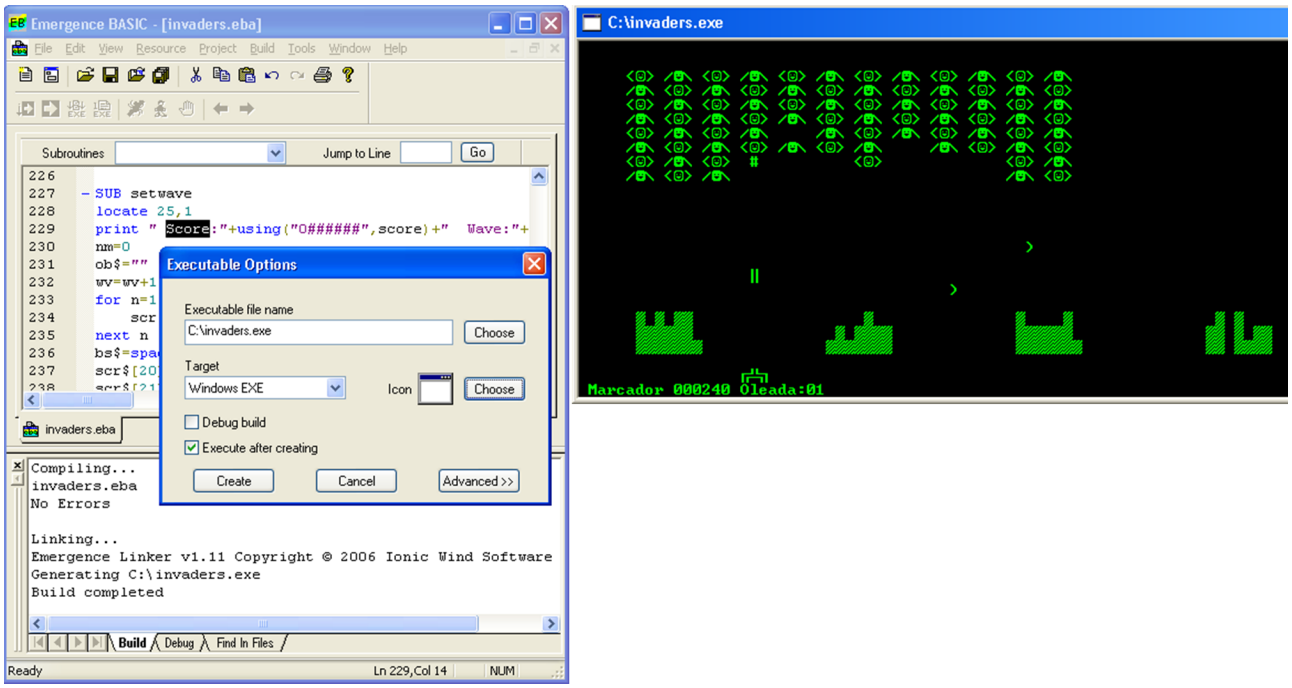
Emergence Basic Development Environment

Esta herramienta (que puede descargarse gratuitamente de www.ionicwind.com) incluye una nutrida colección de ejemplos de pequeños programas que el estudiante puede utilizar para familiarizarse con la interfaz y las funciones de un entorno de programación.

Nota

Familiarizarse con el proceso y las dificultades de la localización del software de un producto informático, es precisamente uno de los objetivos del ejercicio práctico P06, en el cual se propone localizar una pequeña aplicación (CompleteWordCount), programada en Visual Basic for Applications, utilizando, para ello, un editor de este lenguaje.

A la izquierda, localización del código fuente (*source code*) y compilación en un entorno de programación. A la derecha, programa localizado y compilado (invaders.EXE) ejecutándose en tiempo real.



Para alguien que se dedique –o pretenda dedicarse– de lleno a la localización, puede ser aconsejable **familiarizarse mínimamente, al menos, con algún entorno CASE**, para conocer los entresijos de la programación de aplicaciones, aunque sea de manera superficial. Saber cómo está construido internamente un programa puede convertirse en una enorme ayuda a la hora de localizarlo, con independencia de la estrategia y las herramientas que luego se empleen para hacerlo.

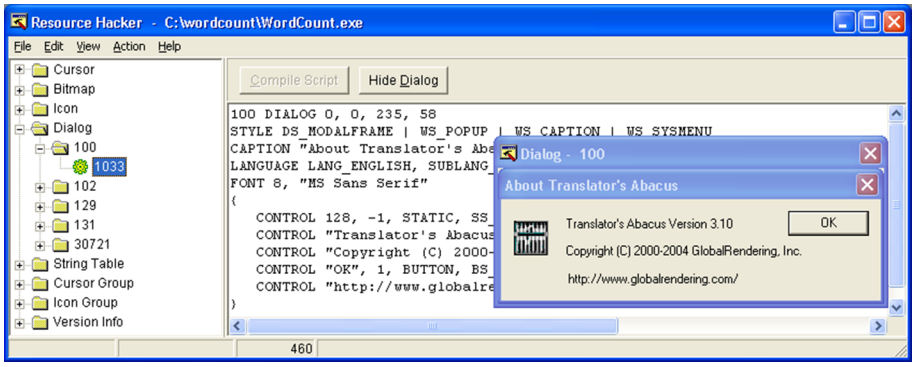
4.2. Los editores de archivos binarios y de recursos

Otra categoría de herramientas –emparentadas con las anteriores, aunque de menor entidad– con las que es posible localizar los archivos que integran una aplicación son los editores de recursos, programas que permiten modificar tanto archivos compilados en formato binario (principalmente EXE o DLL) como archivos de recursos (en formato RC).

Tampoco estas son herramientas diseñadas para localizar aplicaciones, pero –como su propio nombre indica– permiten **acceder a los recursos de un programa para modificarlos**, que es de lo que se trata al fin y al cabo. Como ya se ha explicado, al abrir un archivo compilado, el editor (Resource Hacker, en la siguiente imagen) clasifica sus recursos en diferentes grupos o categorías, según se trate de menús (categoría `Menu`), cuadros de diálogo (categoría `Dialog`), mensajes de texto (categoría `String Table`), iconos (categoría `Icon`), etc., de

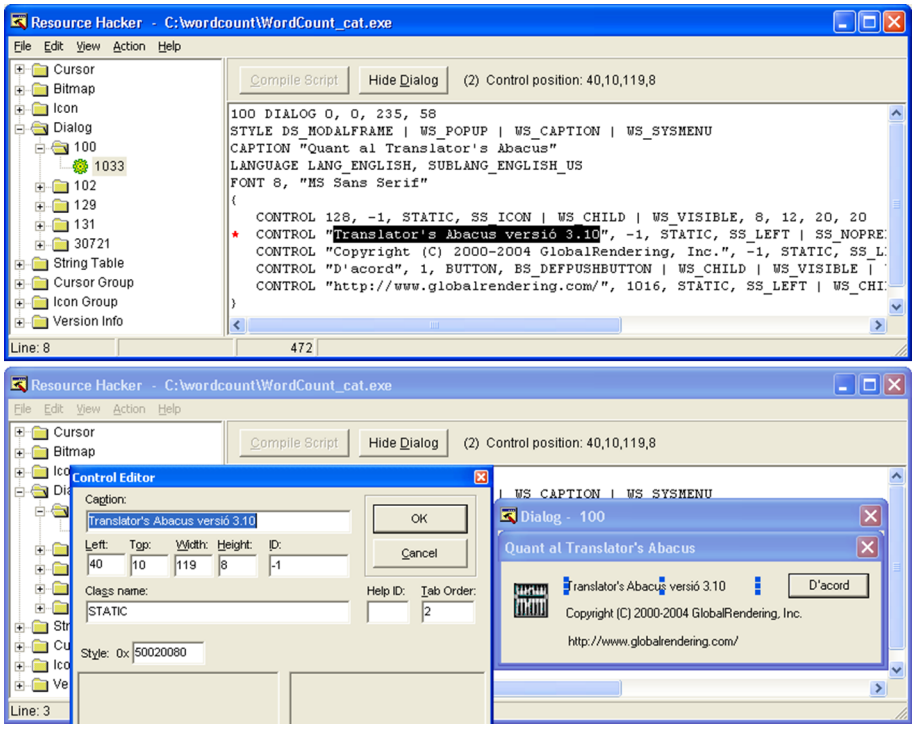
manera análoga a como lo hacen las herramientas integrales de localización. Cada recurso tiene asignado un número que lo identifica inequívocamente, y que no debe alterarse en ningún caso.

Editor de recursos Resource Hacker.



Estas herramientas suelen ofrecer, en paneles o ventanas, dos versiones del contenido de cada recurso: una textual, no compilada (que recoge el código fuente a partir del cual se genera, por ejemplo, un cuadro de diálogo), y otra gráfica, ya compilada (que muestra el cuadro de diálogo en sí, tal y como lo percibe el usuario cuando ejecuta el programa). Así, es posible **modificar los recursos tanto en modo textual** (sobrescribiendo el texto traducible, que en este ejemplo va delimitado por dobles comillas), **como gráfico o WYSIWYG** (seleccionando cada control y modificando el valor de su propiedad `Caption`).

Localización de recursos en modo textual y en modo gráfico.



En los dos casos, como se explica en los ejercicios prácticos P06, P07 y P08, es posible asimismo **reajustar la posición o las dimensiones de un control** si fuera necesario (porque el texto traducido no tuviera cabida en el control o porque, al traducir, se hubiera producido algún solapamiento o truncamiento, etc.).

Soluciones lingüísticas

No obstante, como ya se ha mencionado, es siempre recomendable optar por soluciones lingüísticas (abreviación, sinonimia, reformulación, etc.) antes que técnicas o visuales (como el denominado *resizing*), con el objetivo de alterar la integridad del código lo menos posible y reducir así el riesgo de provocar problemas de funcionamiento.

También es habitual que estas herramientas incluyan **funciones para exportar**, global o selectivamente, los recursos de un programa, de modo que, una vez extraídos en formato RC, pueden localizarse con una herramienta de traducción asistida o un editor de textos, como se explica en los apartados de este módulo dedicados a las estrategias de localización de software.

Por lo demás, **los inconvenientes y las ventajas de los editores de recursos** coinciden con los mencionados en el subapartado anterior para los entornos de programación. Así pues, su empleo puede resultar recomendable para familiarizarse con la localización de software e incluso para traducir profesionalmente aplicaciones de escasa envergadura, pero quizá sus carencias desde el punto de vista del localizador aconsejen no emplear editores de recursos en grandes proyectos multilingües de localización, salvo como herramientas complementarias.

4.3. Las herramientas integrales de localización

Lo que en esta asignatura se denomina «herramienta integral de localización» –que en inglés recibe distintas denominaciones como *localisation tool*, *toolkit*, *suite* o *solution*– es una aplicación que combina las principales funciones de los editores de recursos con las prestaciones habituales en los programas de traducción asistida y, en particular, las que atañen a la gestión de memorias de traducción y glosarios. Además, estas herramientas suelen incluir distintas funciones de validación y control que permiten detectar y corregir las anomalías funcionales que suelen producirse durante el proceso de localización, para garantizar la integridad y la calidad del producto final localizado. Se trata, por tanto, de aplicaciones especializadas de cierta complejidad que se emplean de forma casi exclusiva en el sector de la localización, para el que han sido concebidas.

Herramientas de localización

Existen distintos repertorios de herramientas de localización, entre los cuales pueden consultarse, por ejemplo, el portal Electonline del Localisation Research Centre, las secciones de reseñas de productos (Magazine > Product Reviews) o de recursos (Resources > Industry Resources > Translation Tools) de la revista Multilingual Computing and Techonology o el apartado «Tools Links» del sitio web de OpenTag.

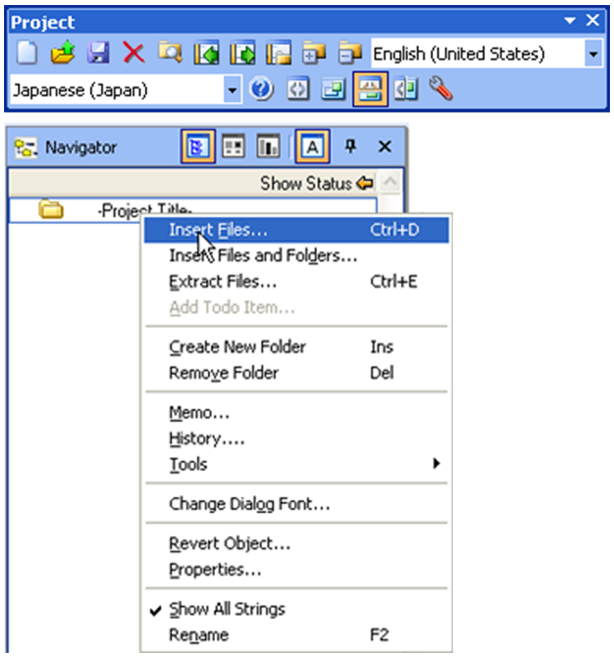
En las herramientas integrales de localización, **el flujo de trabajo** de un proyecto de localización suele dividirse –con ligeras variaciones que atañen a su interfaz y terminología– en las siguientes **fases**: creación y preparación del proyecto y sus materiales; exportación de paquetes o subproyectos para que sean traducidos externamente; traducción y localización con el apoyo de materiales de referencia; reinserción en el proyecto marco de los paquetes o sub-

proyectos exportados; comprobación, corrección y validación de los elementos localizados; y preparación de la versión final localizada del producto para su entrega al cliente.

Para poder localizar los archivos que componen un producto informático con una herramienta integral de localización, el primer paso consiste en **crear un proyecto**, definir sus parámetros (idiomas de origen y destino, ubicación de los archivos del proyecto y de los materiales de referencia, etc.) e incluir en él todos los archivos que se pretende localizar.

En Alchemy Catalyst, por ejemplo, puede crearse o abrirse un proyecto eligiendo la opción al efecto al ejecutar la aplicación, o con la barra de herramientas o el menú correspondiente si la aplicación ya se está ejecutando.

Creación y apertura de proyectos en la herramienta de localización Alchemy Catalyst.



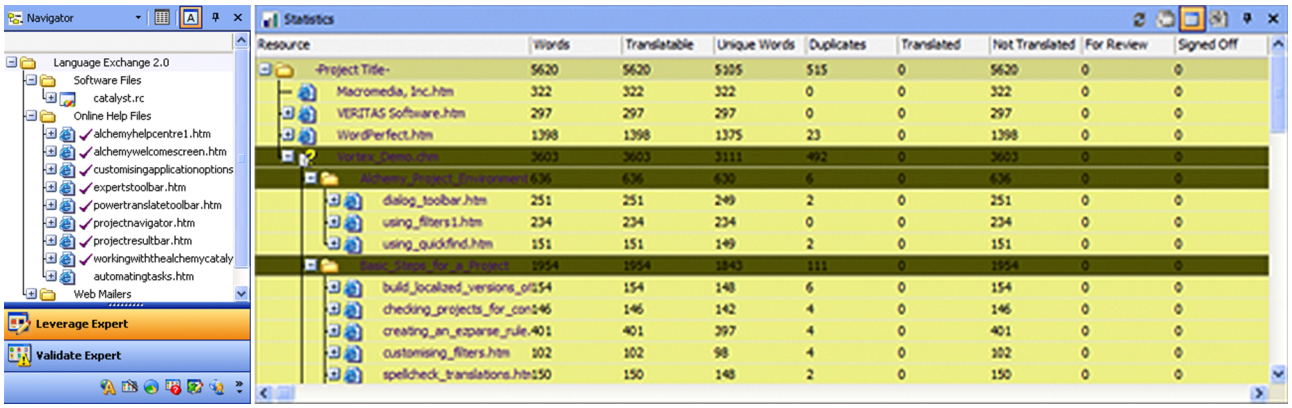
En otras herramientas, el procedimiento es muy similar. Algunas permiten integrar en el mismo proyecto **archivos en varios formatos** (que corresponden a los componentes o módulos del producto) y generar **versiones localizadas a varios idiomas**. Así, por ejemplo, pueden incorporarse al mismo proyecto tanto los archivos de la interfaz de un producto como los que conforman su sistema de ayuda, para localizarlos a varias lenguas. De este modo, se centralizan todos los materiales que conforman el proyecto de localización en un

único contenedor que facilita su gestión. Para poder crear proyectos de localización, suele ser preciso disponer de la **versión completa de la herramienta**. El formato de los archivos de proyecto suele ser exclusivo de cada herramienta y, por tanto, incompatible con el de otras.

Por lo general, el **área de trabajo** de estas herramientas se subdivide en varios paneles que cuentan con funciones y barras de herramientas especializadas en las distintas tareas de un proyecto de localización. Lo habitual es que haya, al menos, un navegador para la gestión global del proyecto y sus materiales; paneles para la traducción en modos textual y gráfico, así como para la gestión y el reaprovechamiento (*leverage*) de memorias de traducción, glosarios y otros materiales de referencia; y algún módulo, sección o asistente con funciones como las de seudotraducción, validación y control de calidad.

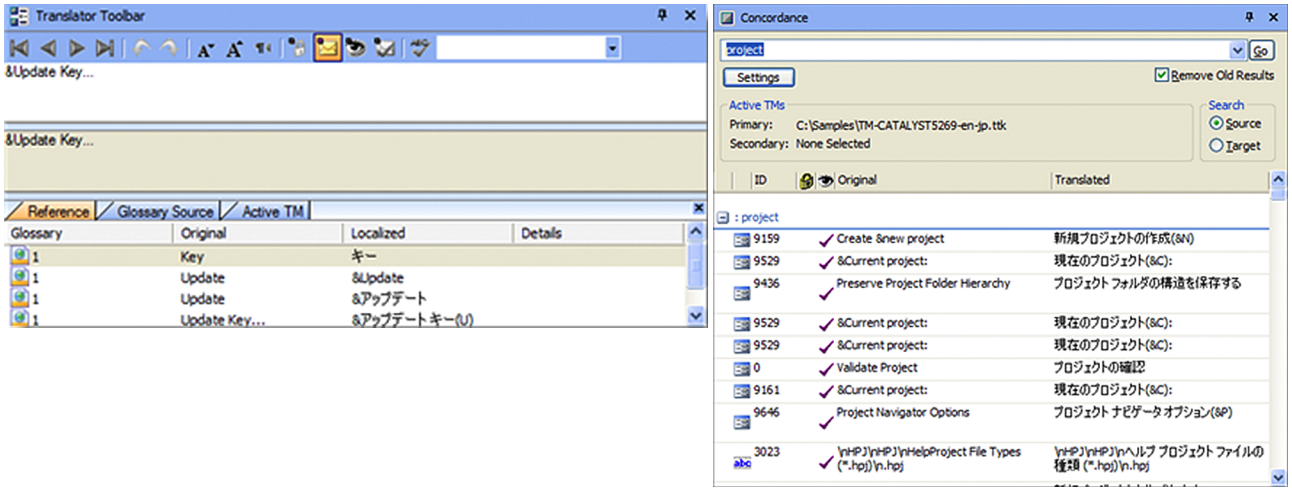
Por ejemplo, el área de trabajo de Catalyst se compone de distintos paneles, ventanas y barras de herramientas para la gestión del proyecto y sus archivos.

A la izquierda, navegador del proyecto. A la derecha, panel de estadísticas del proyecto.



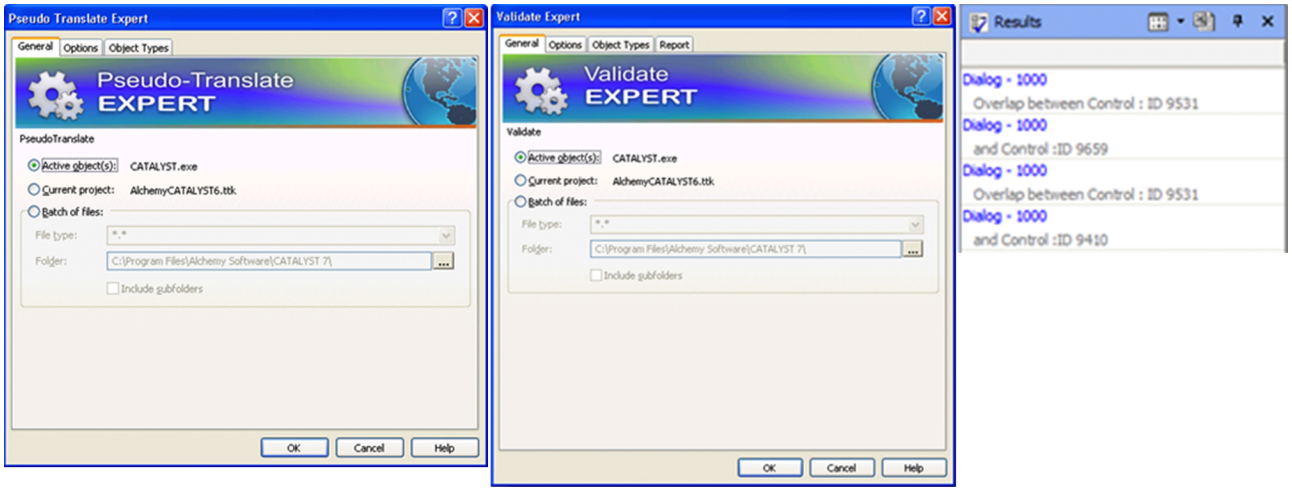
También incluye varios paneles más para la traducción y gestión de materiales de referencia, como memorias de traducción y glosarios.

A la izquierda, ventana y barra de herramientas de traducción. A la derecha, ventana de concordancias.



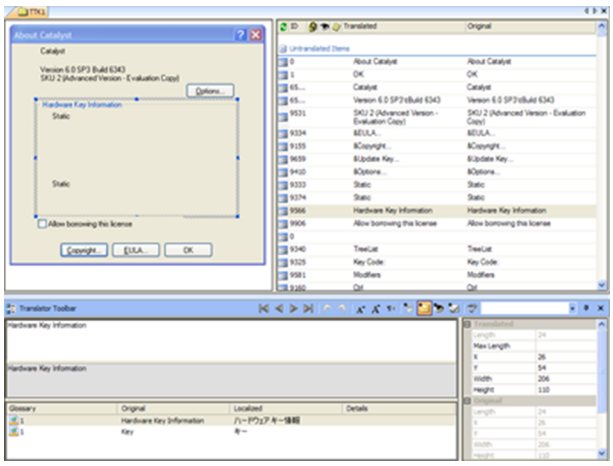
Y ofrece otros para la seudotraducción, la validación y el control de calidad del proyecto.

A la izquierda, experto de seudotraducción. En el centro, experto de validación. A la derecha, ventana de resultados de validación.



Como sucede en los editores de recursos, una herramienta de localización **identifica y distingue** del código informático del programa que se localiza **los elementos susceptibles de ser localizados**, y los diferencia mostrándolos en varios colores o separándolos físicamente en distintos paneles o ventanas, en los que pueden traducirse en **modo textual o gráfico (WYSIWYG)**.

Paneles de la interfaz de una herramienta integral de localización.



A diferencia de los entornos de programación o los editores de recursos, las herramientas integrales de localización han sido **diseñadas para localizar y no para programar**, por lo que toman en consideración los requisitos y la problemática de un proyecto de localización. En este sentido, ofrecen ventajas que palian algunas de las carencias que un localizador encuentra en los entornos de programación y los editores de recursos. Por ejemplo:

Ved también

En el ejercicio práctico P15, se detallan las funciones con las que suelen contar las herramientas de localización.

- Suelen **blindar más el código informático** que acompaña al texto localizable para protegerlo de posibles modificaciones accidentales, además de ofrecer incluso la posibilidad de **bloquear o excluir determinadas cadenas** del proyecto de localización.

- Vienen provistas de distintas **funciones de validación y control de calidad** para comprobar la integridad del programa localizado, comparándolo exhaustivamente con el original.

- Toman en consideración **aspectos específicos de la problemática que entraña localizar** un producto, como la longitud de los textos, el empleo de distintos alfabetos, la direccionalidad horizontal o vertical, los juegos de caracteres, los criterios de ordenación alfabética o de puntuación de miles y decimales, etc.

- Suelen incluir alguna **función de seudotraducción/seudolocalización**, que permite simular *a priori*, en el producto que se va a localizar, los aspectos más característicos de un determinado *locale*, para detectar de antemano posibles problemas de localización.

- Permiten incluir en un mismo proyecto **varias versiones localizables** a distintos idiomas.

- Pueden procesar e incluir dentro de un mismo proyecto **componentes en distintos formatos** (software, ayuda, etc.), lo cual centraliza y simplifica su tratamiento y gestión.

- Permiten **exportar las cadenas de texto** de los recursos localizables a distintos formatos convencionales y/o crear proyectos que pueden localizarse con diferentes herramientas de fácil acceso y gratuitas o de escaso coste (ofimáticas, de traducción asistida o de localización), sin necesidad de hacer una costosa inversión en licencias ni de poseer conocimientos técnicos avanzados.

- Suelen ser compatibles con **los principales estándares y formatos** utilizados de manera normalizada en el sector de la localización (como TMX, TBX o XLIFF).

- Y, sobre todo, cuentan con **funciones propias de los programas de traducción asistida** (o permiten interactuar con ellos), como, por ejemplo, gestión de glosarios y memorias de traducción; análisis estadístico de repeticiones y coincidencias parciales y totales; asignación de estados a los segmentos y filtrado por diferentes criterios; inclusión de comentarios; propagación de segmentos coincidentes; configuración de reglas de segmentación, etc.

Es muy habitual en el sector de la localización que un traductor que carezca de una versión plenamente funcional de la herramienta en cuestión preste sus servicios a un LSP (*Localisation Services Provider*), que sí posee la correspondiente licencia y subcontrata al profesional para encomendarle la localización de un producto o parte de él. Así, el LSP localiza un producto a varias lenguas, y para ello recurre a los servicios de profesionales independientes o pequeñas agencias externas. En una situación como la descrita, el LSP suele crear **subproyectos de localización** para cada lengua de destino y se los envía a sus proveedores. Estos, pese a no contar con una versión completa de la herramienta, pueden utilizar gratuitamente una *demo* o versión gratuita.

Conscientes de esta situación, muchos fabricantes de herramientas comerciales de localización (como Catalyst, Lingobit Localizer, Multilizer o Passolo, entre otras) han diseñado **versiones gratuitas**, comúnmente denominadas *light* o *satellite*, que ponen a disposición de profesionales autónomos y pequeñas empresas de localización (SLV, *Single Language Vendors*) para que puedan localizar con ellas los subproyectos o paquetes preparados por sus clientes, por lo general, empresas multinacionales de localización de mayor envergadura (MLV, *Multi-Language Vendors*). La contrapartida es que estas versiones gratuitas tienen **restringidas muchas de sus funciones** (creación de proyectos, importación, exportación, compilación, validación, etc.), y tan solo permiten abrir un proyecto, traducir sus contenidos localizables (de manera muy similar a como se hace con un programa de traducción asistida) y devolvérselo al

cliente. Esta situación ha venido propiciada, en parte, por el elevado coste de las licencias y por el nivel de competencia técnica necesario para manejar este tipo de aplicaciones especializadas.

Fuera del ámbito comercial, encontramos herramientas de localización de uso privativo o *propietarias*, que han desarrollado, para uso interno y de sus proveedores, tanto los gigantes del mercado del software como las grandes multinacionales de servicios de localización. Tal es el caso, respectivamente, de herramientas como LocStudio y Helium (desarrolladas por Microsoft para los macroproyectos de localización de sus productos), Lingobit Localizer (creada por la empresa de localización Lingobit Technologies) o Idiom WorldServer (originariamente desarrollada por Idiom Technologies, empresa comprada por la multinacional de la localización SDL International, que años antes ya había desarrollado su propia herramienta de localización, SDLinsight).

Mal que nos pese, la inmensa mayoría de las herramientas aquí descritas o citadas han sido diseñadas para las sucesivas versiones del sistema operativo Microsoft Windows, que acapara la inmensa mayoría de los proyectos de localización de carácter comercial. Para el entorno operativo MAC OS de Apple, la herramienta de localización por antonomasia es AppleGlot y sus accesorios, aunque existen –o han existido– otras como Localization Suite, iLocalize, LocFactory Editor, o las de la empresa LocTeam.

4.4. Otras herramientas

Hoy por hoy, prácticamente cualquier herramienta de traducción asistida ya incorpora los filtros o funciones necesarios para procesar y traducir satisfactoriamente no solo los archivos que contienen los recursos localizables del software de un programa informático (en su versión compilada o sin compilar), sino también formatos de intercambio normalizados como XLIFF.

La elección de **un sistema de memorias de traducción en lugar de una herramienta integral de localización** responde muchas veces a factores como su inferior precio, la menor capacitación y formación necesarias o la comodidad de poder utilizar –y aprender a utilizar– la misma herramienta para traducir documentos en formatos dispares, incluidos los habituales en un proyecto de localización. Así que, a menudo, el traductor prefiere –o el cliente le impone– emplear la misma herramienta que ya utiliza para otros proyectos convencionales de traducción.

En algunos productos, es tal el volumen de traducción y tan elevado el coste de su localización –por ejemplo, en los sistemas operativos y aplicaciones de mayor implantación–, que en proyectos de localización de gran envergadura se opta a menudo por la utilización controlada de la **traducción automática, casi siempre combinada con la asistida y complementada con la postedición**. Más común, si cabe, es esta decisión en la localización de macrositios web –o, al menos, en algunas de sus secciones menos visibles–, en los que la

localización manual o asistida resultaría económicamente inviable. A menudo, el uso de la traducción automática por parte del traductor autónomo se produce de forma indirecta en la revisión de textos que le llegan pretraducidos.

Aunque –como se ha visto– hoy ya existen infinidad de herramientas específicamente diseñadas para localizar, y la traducción asistida ha alcanzado un notable estado de madurez, en la **caja de herramientas** del localizador no deben faltar algunos **accesorios y utilidades complementarios**. Contar, por ejemplo, con un editor de textos avanzado sigue siendo muy recomendable, por su utilidad para llevar a cabo infinidad de tareas y pequeños retoques. En determinados casos, también es necesario recurrir a un editor hexadecimal para acceder y modificar textos que, por estar incrustados en el código informático (*hard-coded*), no consiguen extraer para su localización la mayoría de las herramientas mencionadas.

Cabe mencionar, por último, el auge experimentado en los últimos años por las herramientas de traducción asistida y localización en el mundo del software libre y de código abierto (FSOSS). Su irregular evolución en este ámbito ha estado jalonada: por un lado, por la apertura de herramientas nacidas en el seno de empresas (como ForeignDesk, de la multinacional LionBridge, que publicó su código en el 2001; o Qt Linguist, complemento de código abierto de la empresa noruega Trolltech para su plataforma de desarrollo de aplicaciones); por otro, por la dispar suerte que han corrido productos e iniciativas, como:

- El paquete gratuito de herramientas y filtros de Enlaso Tools, un clásico del sector, ahora también integrado en la plataforma Okapi Framework, u otros como GNU Gettext, KBabel, Open Language Tools o The Translate Toolkit.

- Proyectos como RosettaWerks u OSTTI - Open Source Translation Technology Initiative.

- Iniciativas como el portal de traducción Pootle.

- Distribuciones para traductores Linguas OS o PCLOS-Trans.

- La macroplataforma KDE Localization.

A las anteriores, se suman distintas **herramientas de traducción asistida** surgidas en este ámbito y que cuentan con funciones para procesar tanto algunos de los formatos habituales en un proyecto de localización, como el formato de intercambio específico del sector, XLIFF.

No cabe duda de que la actividad en **la comunidad del FOSS** es febril, y día a día se traducen cientos de miles de palabras a decenas de lenguas para localizar un sinfín de aplicaciones de todo tipo. Sin embargo, el carácter altruista de la mayoría de las iniciativas y la falta de un aliciente económico siguen pesan-

do demasiado para que los fabricantes de herramientas comerciales apuesten definitivamente por el movimiento, salvo en muy contadas –y casi siempre interesadas– ocasiones.

Resumen

A diferencia de lo que sucede en la traducción de un texto convencional, que puede hacerse de principio a fin sin más, cuando se localiza el software de un producto informático es conveniente **avanzar poco a poco, afianzando los pasos que se van dando**. Así, lo aconsejable –sobre todo, al principio– es localizar uno o varios controles o recursos (o todos los de un mismo tipo), ejecutar el programa para comprobar que todo sigue funcionando correctamente y reanudar luego el proceso de localización tras haber guardado una copia de seguridad provisional de la versión intermedia estable, por si fuera necesario retroceder algún paso. Dependiendo de la entidad del programa que se esté localizando, y de la competencia del localizador, la zancada podrá ser más o menos amplia.

Una vez que el traductor se ha familiarizado con los códigos y convenciones explicados con profusión en este módulo, la dificultad que plantea la localización de software no estriba tanto en la necesaria delimitación formal de código y texto (de la que también se sirven las propias herramientas de traducción asistida y de localización para identificar y extraer selectivamente este sin alterar la integridad de aquel), sino en la frecuente falta de previsión, que provoca que **los programas no se internacionalicen de antemano** o que, en muchos casos, ni se plantee la posibilidad de localizarlos ulteriormente.

Asimismo, es cierto que, amén de la ausencia de internacionalización, por falta de previsión o por mero **desconocimiento de las prácticas recomendables más elementales**, no pocas veces programar de manera eficiente o diseñar con mayor vistosidad está reñido con hacerlo de tal forma que se facilite el complejo proceso de localización, siempre posterior.

Al igual que sucede cuando se localiza un sitio web, **un sencillo truco** para dilucidar si un determinado elemento conflictivo es o no localizable o para averiguar en qué lugar de la interfaz del programa aparece consiste en ejecutar la aplicación que se está localizando constantemente y navegar hasta sus más recónditos rincones, para familiarizarse con ella a fondo. No en vano, el localizador acaba conociendo los productos que localiza casi tan bien como quienes los crearon. Tan consciente acaba siendo el localizador de la cantidad de errores y defectos que por norma encierra cualquier aplicación informática, y de que estos se multiplican durante el proceso de localización, que a menudo acaba haciendo bueno aquello de que «en casa del herrero...», y se resiste a instalar en su ordenador productos localizados, incluso por él mismo.

Bibliografía

Arevalillo, J. J. (2004). «Especial Localización». En: *La linterna del traductor* (núm. 8) [documento en línea]. <http://traduccion.rediris.es/8/index.htm>

Esselink, B. (2000). *A Practical Guide to Localisation*. Ámsterdam/Filadelfia: John Benjamins.

Gouadec, D. (2003). «Le bagage spécifique du localiseur/localisateur. Le vrai "nouveau profil" requis». *Meta* (vol. 48, núm. 4, págs. 526-545) [documento en línea]. <www.erudit.org/revue/meta/2003/v48/n4/008724ar.html>.

Grupo Tradumàtica (ed.) (2002). «La localización / La localització». *Revista Tradumàtica* (núm. 1). Barcelona, UAB [documento en línea]. <www.fti.uab.es/tradumatica/revista/index_01.htm>

Lingo Systems. *The Guide to Translation and Localization* [documento en línea]. <www.lingosys.com/Forms/guide_register_lingo.htm>

Sokoli, R. (2002). «Catálogo de herramientas para la localización de software y de páginas Web». En: *Tradumàtica*. Barcelona: Departament de Traducció i d'Interpretació de la UAB [documento en línea]. <www.fti.uab.es/tradumatica/revista/articles/rsokoli/art.htm o también rsokoli.pdf>