revista.tradumàtica
tecnologies de la traducció

101-107

El software libre en la caja de herramientas del traductor

Gonçalo Cordeiro
Proyecto WordForge

RESUMEN

El objeto de este artículo es dar a conocer una serie de experiencias de traducción en un entorno profesional apoyado exclusivamente en el uso de software libre. Para ello, me referiré a las principales herramientas utilizadas, desde un punto de vista práctico.

Palabras clave: traducción, localización, software libre, código abierto, FLOSS

RESUM *(El programari lliure en la caixa d'eines del traductors)*

L'objecte d'aquest article és el de traslladar una sèrie d'experiències de traducció en un entorn professional recolzat exclusivament en l'ús de programari lliure. Per això, em referiré a les principals eines utilitzades des d'un punt de vista pràctic.

Paraules clau: traducció, localització, programari lliure, codi obert, FLOSS

ABSTRACT *(Open Source software in translator's workbench)*

The purpose of this article is to transfer a number of experiences of translation in a professional environment exclusively supported in the use of free software. To do this, i will refer to the main tools used, from a practical point of view.

Keywords: translation, localization, free software, open source, FLOSS

1. Introducción

Existe una cierta confusión en relación a lo que se entiende como «software libre». En este artículo se entiende como tal el liberado bajo las condiciones descritas por la Free Software Foundation (FSF, 1996). Existen otras varias alternativas que comparten algunas de las condiciones, como el «software de código abierto». De hecho, podemos encontrar cada vez más referencias al «software libre y de código abierto» (FLOSS), un término que comprende dos filosofías parcialmente compatibles: la de la FSF y la de la Open Source Iniciative (OSI). Se recomienda la comparación de ambas a través de sus respectivas definiciones.

A pesar de que frecuentemente se hace referencia a ambos modelos desde el punto de vista de la gratuidad (Castellanos, Copoví y otros, 2011), los aspectos centrales son a) la libertad que se le concede al usuario para ejecutar, estudiar, redistribuír y modificar el software; y b) la posibilidad de «cerrar» el código para distribuír versiones privativas.

Se podría decir que los dos extremos de los modelos de desarrollo están ocupados por el software libre y el privativo. Entre ambos existe una multitud de opciones posibles de acuerdo con las libertades que se le conceden al usuario. Previamente a la utilización de un programa o biblioteca (librería), es necesario que conozcamos la licencia bajo la que está liberado, cuáles son las restricciones que se pudiesen derivar y aspectos como la posibilidad de modificar el código o utilizarlo para fines comerciales. Un usuario medio no intentará modificar el código, pero la solución a un problema puede ser la aplicación de un «parche» o instrucciones para editarlo, para lo que necesitaremos acceso al código.

Por otra parte, cuando alguien se inicia en el uso de software libre o de código abierto, es frecuente la búsqueda de programas equivalentes a los utilizados. Si bien en muchos casos existen tales equivalencias (no exentas de divergencias en su uso o funcionalidades), por lo general es preferible redefinir las herramientas necesarias considerando las tareas y el flujo de trabajo: en el ámbito del software libre es frecuente encontrar herramientas orientadas a tareas específicas, frente a las aplicaciones multifuncionales más habituales en entornos privativos. Esta tendencia al uso de pequeñas herramientas que se encadenan entre sí, entronca con la tradición GNU/Linux de considerar un sistema operativo como un metatexto (Cramer, 2000; Hervada-Sala, 2010). De hecho, es posible leer o modificar con un editor de texto una gran parte de los objetos que componen uno cualquiera de estos sistemas, al tiempo que existen multitud de «pequeñas» utilidades para efectuar las operaciones de búsqueda, filtrado, ordenación, substitución etc. ejecutadas habitualmente en el procesamiento de textos. El estudio de estas utilidades (grep, sed, sort, uniq, cat etc.; v. Lindberg, 2007) es imprescindible tanto para mejorar nuestras habilidades como para la comprensión de los recursos disponibles.

El uso de sistemas operativos libres (GNU/Linux) no es imprescindible para ejecutar software libre. De hecho las herramientas más utilizadas están desarrolladas en lenguajes multiplataforma (especialmente Java y Python). No obstante, es recomendable poseer una instalación GNU/Linux para acceder a utilidades como las descritas.

2. Herramientas y estándares

A pesar del interés mostrado a partir de 2005 en el seno de las comunidades de traducción de software libre hacia la adopción de la especificación XLIFF, el peso de la tecnología Gettext (FSF, 1998) ha condicionado el desarrollo de filtros de conversión y herramientas (Fernández García, 2007), pero sobre todo la extensión de su uso fuera de la traducción profesional. De hecho, las primeras implementaciones — las Open Language Tools liberadas por SUN y el editor Transolution — habían sido diseñadas como editores XLIFF nativos, mientras que los desarrollos más recientes — Lokalize, WordForge y Virtaal — siguen manteniendo la compatibilidad con el formato PO de *gettext*.

No obstante, han emergido con fuerza dos herramientas provenientes de otros ámbitos (la industria de la localización y la traducción profesional): Okapi Tools y OmegaT. Se trata de dos desarrollos libres con un amplio soporte de los estándares abiertos recogidos en OAXAL (OAXAL, 2008).

En la práctica, esto significa que además de estar soportados todos los formatos de documentación admitidos por las Okapi Tools y OmegaT (con conversión a XLIFF), es posible trabajar con cualquier conversor que porporcione un objeto PO.

Los formatos de trabajo y filtros disponibibles marcarán en cada caso la estrategia más adecuada a la hora de combinar dos o más de estas herramientas.

3. Traduciendo con software libre

Los siguientes ejemplos pretenden mostrar algunas de las limitaciones y ventajas del software libre aplicado a la traducción de documentos y aplicaciones.

3.1 Sabor a galego

«Sabor a galego» fue una campaña promovida en el 2008 por la Concellaría de Normalización Lingüística del Ayuntamiento de A Coruña que buscaba la dinamización del gallego en el sector de la hostelería. Los cerca de 100 establecimientos que se adhirieron a esta campaña, remitieron los materiales en diversos formatos: PDF, XLS, DOC, DOCX etc. Los documentos originales fueron remitidos en español y gallego. La fase de traducción era previa a la de diseño y maquetación, de modo que lo decisivo no era el formato de los documentos sino su entrega a los diseñadores en una forma manejable.

Al contrario de lo que sucede en la localización al gallego, los glosarios y sobre todo las memorias de traducción para documentación no están públicamente disponibles; de modo que en un primer momento se reunieron todos los materiales disponibles de trabajos anteriores, para confeccionar una memoria en el formato TMX. La aplicación utilizada fue bitext2tmx. Este programa tiene la desventaja de no permitir guardar fases intermedias del proceso de alineación, por lo que la tarea se ha de completar antes de generar el fichero TMX; sin embargo se trata de una aplicación estable y que genera una salida conforme al estándar.

Los filtros de Okapi son compatibles con Microsoft Office 2007, con lo que cabría la posibilidad de transformar a esta versión los diferentes ficheros. Sin embargo, su conversión a texto simple facilita las tareas de traducción automática y conversión a XLIFF, por lo que se utilizó el conversor en lotes JODConverter; excepto para la conversión de PDF a texto, realizada mediante la utilidad pdftotext (parte de Xpdf).

Se realizaron pruebas para comprobar la viabilidad de usar el motor Apertium para la pretraducción de español a gallego. Lamentablemente, tanto la calidad de los originales como lo específico del léxico impidieron obtener buenos resultados.

Para la generación de glosarios se procesó la recopilación de documentos existentes, que se fue alimentando con cada nuevo fichero traducido. La detección de candidatos para el glosario se apoyó en dos herramientas: el analizador TextStat para obtener las listas de palabras y el extractor Lexterm para identificar las equivalencias. Existe una aplicación más potente para el análisis de texto y concordancias: AntConc, pero su licencia únicamente permite su uso para fines personales o de investigación sin finalidad lucrativa. Tampoco TextSTAT está liberado bajo una licencia FLOSS, aunque se limita a una cláusula de exención de responsabilidad. Por otra parte, a pesar de que Lexterm se ejecuta razonablemente bien en GNU/Linux bajo Wine, su rendimiento presenta algunos problemas, particularmente con ficheros grandes. Para la elaboración final del glosario se utilizó el visualizador de concordancias Koyori.

En el paso de creación del paquete de traducción, podemos seleccionar un fichero XLIFF genérico que podremos trabajar con alguna de las aplicaciones mencionadas en el segundo apartado de este artículo; o bien un conjunto de directorios y ficheros específicamente diseñado para ser traducido con OmegaT.

3.2 La localización de eFront

La plataforma eFront es un sistema de e-learning. La localización de la versión *community* respondió a necesidades internas, pero una vez finalizado el trabajo se informó a un responsable del desarrollo, quien integró la versión en gallego en el repositorio oficial.

Para completar una versión de eFront es necesario traducir un fichero principal de idioma en PHP y diversos módulos que confieren funcionalidades adicionales. Las cadenas a traducir están almacenadas en el original bajo el siguiente formato:

```
define("_LANGUAGE","Language");
define("_PASSWORD","Password");
```

El formato esperado de un fichero traducido sería:

```
define("_LANGUAGE","Idioma");//Language
define("_PASSWORD","Contrasinal");//Password
```

Lo que se traduce es el segundo argumento de cada *define*, y se espera que el segmento traducido aparezca incluído como un comentario en PHP (//).

Existe, como en este caso, la posibilidad de utilizar herramientas *ad hoc* disponibles en linea, pero normalmente presentan la desventaja de, por ejemplo, no poderles aplicar una memoria de traducción o un corrector ortográfico. No hay tampoco extractores de contenido traducible que garanticen una total conformidad con el estilo empleado para su internacionalización.

Bajo estas circunstancias, lo más recomendable es crear una regla de extracción propia utilizando la aplicación Rainbow (parte del Okapi Framework). El trabajo básicamente consiste en crear un proyecto en Rainbow y asignarle un nuevo filtro de extracción basado en uno de los existentes, por ejemplo el disponible para expresiones regulares. Para este formato en concreto, lo substancial es definir una expresión regular para nuestro filtro:

```
(\s*define)\("_(.*?)("\s*,\s*"|"\s*;\s*")(.*?)"\);\s*?.*?$
```

y asignar los grupos que serán extraídos y su colocación.

El aspecto de un filtro personalizado para Rainbow sería el siguiente:

```
#v1
useLD.b=false
localizeOutside.b=true
startString="
endString="
extractOuterStrings.b=false
useBSlashEscape.b=true
oneLevelGroups.b=true
regexOptions.i=40
mimeType=text/plain
ruleCount.i=1
rule0.ruleName=efront-original
rule0.ruleType.i=1
rule0.expr=^(\s*define)\("_(.*?)("\s*,\s*"'|"\s*;\s*")(.*?)"\);\s*?.*?$
rule0.groupSource.i=4
rule0.groupTarget.i=-1
rule0.groupName.i=-1
rule0.groupNote.i=2
rule0.preserveWS.b=true
rule0.useCodeFinder.b=false
rule0.sample=define("_WEEKLY" ,
"Weekly");$0a$define("_MONTHLY","Monthly");$0a$define("_YEARLY"
;"Yearly");$0a$define("_DAYSCONDITIONAL",
"Day(s)");$0a$define("_WEEKSCONDITIONAL",
"Week(s)");$0a$define("_MONTHSCONDITIONAL",
"Month(s)");$0a$define("_YEARSCONDITIONAL", "Year(s)"); //teste$0a$if
(defined('MODULE_HCD_INTERFACE') && MODULE_HCD_INTERFACE == 1) {$0a$
define("_THENEWFORMOFADDITIVELEARNING","eLearning and Human Capital
Development");$0a$ define("_AUTOMATICUSERACTIVATION","Automatic user
activation");$0a$ define("_USERSEXPORT","Export users");
rule0.codeFinderRules.count.i=0
rule0.codeFinderRules.sample=
rule0.codeFinderRules.useAllRulesWhenTesting.b=false
```

Este filtro, que podemos guardar en texto simple como «okf_regex@efront.frmp» se puede importar y examinar en la propia aplicación para ver los resultados esperados. Como se aprecia, contiene una serie de ejemplos extraídos del fichero original que presentan pequeñas variaciones: uso inesperado de espacios, comillas dobles o simples etc.

A igual que en el caso anterior, podemos generar el tipo de XLIFF deseado mediante Okapi.

Este mismo procedimiento puede ser aplicado a cualquier formato de texto para el que podamos identificar uno o varios patrones. Si no resulta posible identificar los patrones de un modo fiable, siempre cabe la posibilidad de aplicar un marcado manual para delimitar el texto traducible; únicamente deberemos suprimir nuestros marcadores una vez hayamos traducido y reconvertido nuestro fichero al formato original. Esto abre la puerta a soluciones personalizadas cuando nos enfrentamos a un formato no soportado.

3.3 Otros formatos

Hemos visto que los editores en software libre tienden a la adopción de estándares abiertos. Por ello, la traducción de documentos complejos o en formato propietario, como presentaciones PowerPoint o ficheros de InDesign, no siempre es posible.

Okapi mantiene una gran compatibilidad con los formatos basados en XML, como MS Office 2007, y un filtro en desarrollo para los documentos IDML de InDesign. El paso de PPT o DOC a sus versiones XML puede ser realizada desde el propio MS Office si se dispone de una instalación, aunque como en cualquier proceso de conversión no siempre obtendremos el resultado esperado. Será necesario realizar previamente una serie de pruebas para comprobar que las conversiones funcionan adecuadamente.

4. Conclusiones

En este artículo hemos mostrado algunos ejemplos de traducción con software libre, atendiendo a diversos formatos.

El tratamiento de los diversos formatos de origen suele implicar el uso de varias herramientas a lo largo del proceso. Para determinados estándares de facto el flujo de traducción puede no ser factible usando exclusivamente opciones libres, especialmente si se debe mantener la configuración del original durante la reconversión o generación del documento traducido.

En cualquier caso, tanto la convergencia hacia estándares XML abiertos como la flexibilidad que porporcionan las herramientas disponibles, permiten su uso habitual para las tareas más frecuentes.

Referencias

Herramientas citadas

AntConc <http://www.antlab.sci.waseda.ac.jp/software.html>. Fecha de última consulta: 03.07.11

Apertium <http://www.apertium.org/>. Fecha de última consulta: 03.07.11

bitext2tmx <http://bitext2tmx.sourceforge.net/>. Fecha de última consulta: 03.07.11

JODConverter <http://www.artofsolving.com/opensource/jodconverter>. Fecha de última consulta: 03.07.11

koyori (Bilingual KWIC) <http://www.kl.i.is.nagoya-u.ac.jp/koyori/index.en.html>. Fecha de última consulta: 03.07.11

Lexterm <http://www.linguoc.cat/index_en.html>. Fecha de última consulta: 03.07.11

Lokalize <http://userbase.kde.org/Lokalize>. Fecha de última consulta: 03.07.11

Okapi Framework <http://okapi.opentag.com/>. Fecha de última consulta: 03.07.11

OmegaT <http://www.omegat.org/>. Fecha de última consulta: 03.07.11

Open Language Tools <http://java.net/projects/open-language-tools>. Fecha de última consulta: 03.07.11

TextSTAT <http://neon.niederlandistik.fu-berlin.de/textstat/>. Fecha de última consulta: 03.07.11

Transolution <http://pypi.python.org/pypi/Transolution/0.4b5>. Fecha de última consulta: 03.07.11

Virtaal <http://translate.sourceforge.net/wiki/virtaal/index>. Fecha de última consulta: 03.07.11

WordForge <http://sourceforge.net/projects/wordforge2/>. Fecha de última consulta: 03.07.11

Xpdf <http://foolabs.com/xpdf/>. Fecha de última consulta: 03.07.11

Bibliografia

Castellanos, L., Copoví, M. y otros (2011). Software libre y gratuito para la traducción. <http://tictrad.blogs.uv.es/2011/05/19/software-libre-y-gratuito-para-la-traduccion/>. Fecha de actualización: 05.19.11. Fecha de última consulta: 03.07.11

Cramer, F. (2000). Free Software as Collaborative Text . <http://cramer.pleintekst.nl/essays/free_software_as_text/free_software_as_text.pdf>. Fecha de actualización: 05.11.00. Fecha de última consulta: 03.07.11

Fernández García, J.R. (2007). La traducción del software libre (IV). ¿El momento de cambiar de herramientas? <http://people.ofset.org/jrfernandez/edu/n-c/traducc_4/index.html>. Fecha de actualización: 01.03.07. Fecha de última consulta: 03.07.11

Free Software Foundation (1996). The Free Software Definition. <http://www.gnu.org/philosophy/free-sw.html>. Fecha de actualización: 12.11.10. Fecha de última consulta: 03.07.11

Free Software Foundation (1998). GNU Gettext. <http://www.gnu.org/software/gettext/>. Fecha de actualización: 06.06.10. Fecha de última consulta: 03.07.11.

Hervada-Sala, F. (2010) Unix: A Text-Aware Environment in Text-Oriented Software. <http://u-tx.net/text/case-unix.html>. Fecha de actualización: 06.05.11. Fecha de última consulta: 03.07.11

Lindberg, N. (2007). egrep for Linguists. <http://stts.se/egrep_for_linguists/egrep_for_linguists.html>. Fecha de actualización: 13.01.07. Fecha de última consulta: 03.07.11

OASIS Open Architecture for XML Authoring and Localization Reference Model (OAXAL). <http://www.oasis-open.org/committees/oaxal/charter.php>. Fecha de última consulta: 03.07.11

Open Source Initiative. The Open Source Definition. <http://www.opensource.org/osd.html>. Fecha de última consulta: 03.07.11