UNIVERSIDAD DE SALAMANCA
FACULTAD DE TRADUCCIÓN Y DOCUMENTACIÓN
GRADO EN TRADUCCIÓN E INTERPRETACIÓN
Trabajo de Fin de Grado

# RAINBOW

## La herramienta de Okapi para gestores de proyectos de localización

Jon Fontán Calzada

Dirigido por Emilio Rodríguez Vázquez de Aldana

Salamanca, 2017

# Resumen

Entre las herramientas de que dispone el gestor de proyectos de localización destaca Rainbow, integrada en el marco Okapi, porque ofrece las principales utilidades de localización de forma gratuita y multiplataforma. Además, trabaja con los formatos estandarizados de la industria de la localización. Explicamos una selección de sus potencialidades más destacadas: la conversión a XLIFF y su posproceso, la adaptación y modificación de filtros, la pseudotraducción, la alineación basada en identificadores y la pretraducción. Para subrayar el enfoque empírico, todas las utilidades aquí analizadas se aplican a ejemplos reales.



# Palabras clave

Okapi, Rainbow, localización, L10n, internacionalización, i18n, traducción, gestión de proyectos, ingeniero de localización, TAO, pseudotraducción, alineación por identificadores, pretraducción, filtros HTML, filtros PROPERTIES, estándares XML.



# Abstract

Among the many tools available to localization project managers, Rainbow stands out. It is a free cross-platform toolbox integrated in the Okapi Framework that features the main localization utilities. In addition, it works with the standardized formats set by localization industry. We detail a selection of its most prominent features such as the creation of XLIFF translation packages and their post-processing, the adjustment and modification of filters, the pseudo-translation, the id-based alignment and the pre-translation. In order to emphasize the empirical approach of this document, every utility analyzed here is applied to a real-life example.



# Key Words

Okapi, Rainbow, localization, L10n, internationalization, i18n, translation, project management, localization engineer, CAT, pseudo-translation, id-based alignment, pre-translation, HTML filters, PROPERTIES filters, XML standards.

# Índice

# Índice de figuras



# Índice de anexos

# 1 Introducción

## 1.1 Motivación y justificación del análisis

En el Grado en Traducción e interpretación se nos prepara para que seamos capaces de gestionar proyectos de traducción y localización. Así, en las asignaturas de Gestión de proyectos y de Prácticas de traducción nos enfrentamos a la dificultad principal que abordan los gestores: coordinar a traductores que trabajan con herramientas distintas y formatos muy diversos. Por tanto, pronto nos queda patente que las herramientas informáticas lejos de ser una mera ayuda son imprescindibles.

Ante el auge de la localización, las asignaturas de la carrera relacionadas con la Informática básica, los Recursos tecnológicos para la traducción y la Localización cobran más relevancia año tras año. Mediante este documento exponemos cómo las competencias técnicas y de gestión son fundamentales para el buen gestor de proyectos. Profundizamos en los proyectos de localización porque son los que más retos técnicos plantean. Así, entre otras cuestiones, se abordará aquí mediante Rainbow la conversión de diversos formatos al estándar XLIFF, la alineación de ficheros, la pseudotraducción, la pretraducción y la adaptación de filtros. Aunque por razones de espacio no se trate al completo en este documento, nos gustaría destacar que el abanico de potencialidades que ofrece Rainbow es mucho más amplio y comprende, por citar algunos ejemplos, el renombrado de archivos, la extracción de terminología y las labores de control de calidad.

## 1.2 Hipótesis de partida y objetivos

La hipótesis de partida es que profundizar en las utilidades y potencialidades de Okapi Rainbow, una herramienta informática gratuita y multiplataforma, facilita y mejora la labor del gestor de proyectos de localización. Con este fin, en este documento vamos a mostrar las herramientas de una parte fundamental del Paquete Okapi. Como el abanico es muy amplio y completo, dado el alcance de este Trabajo de fin grado, nuestro objetivo consistirá en explicar la siguiente selección de utilidades, que justificamos posteriormente en el tercer apartado de este documento:

- El proceso de conversión mediante Rainbow de un fichero al estándar bilingüe XLIFF para poder trabajar con él en cualquier herramienta TAO.

- La conversión de dicho XLIFF para, previo paso por una TAO, devolverlo al formato original en la versión en lengua destino. Es decir, la conversión de vuelta (o «back-conversion»). En la herramienta elegida este paso es denominado «post-processing» y nosotros lo traducimos como «posproceso».
- La pseudotraducción.
- La alineación basada en identificadores para generar un TMX.
- La pretraducción a partir de un TMX.
- La adaptación de los filtros de Rainbow a las características de los ficheros en formato PROPERTIES.
- La modificación mediante reglas de los filtros de Rainbow para los ficheros de ayuda HTML.

Este conjunto de utilidades en las que ahondaremos en los siguientes apartados se ha aplicado a tres programas informáticos en explotación con el fin de subrayar el enfoque real del trabajo. Además, cada uno de ellos tiene el texto que localizar en ficheros distintos, con lo que conseguimos abordar más contenido y mostrar al máximo la versatilidad de la herramienta Rainbow.

Así, el primer experimento lo realizamos sobre una aplicación en JAVA cuyo texto traducible está en ficheros en formato PROPERTIES. Para el segundo caso nos centramos en una aplicación para Android en la que el texto traducible está en ficheros XML con especificaciones Android. Por último, trabajamos con los ficheros de ayuda HTML de una aplicación para Windows.

## 1.3 Metodología

Para la redacción de este documento se ha seguido el siguiente proceso:

1. *Revisión, búsqueda y selección de fuentes teóricas* para fundamentar el marco conceptual sobre el que se construye la parte práctica de este análisis. El proceso de documentación se ha centrado libros como *Tradumàtica* (Martín-Mor, Piqué y Sánchez-Gijón 2016), *Localizing Apps: A practical guide for translators and translation students* (Roturier 2015) y *Translation and Web Localization* (Jiménez-Crespo 2013), entre otros. El objetivo era consolidar y ampliar los

conocimientos adquiridos en la asignatura de Localización para así plantear experimentos mediante Rainbow que simulen encargos reales de localización.

2. *Profundización en la parte experimental de la documentación*. Se ha prestado un especial énfasis a la documentación técnica del programa Okapi Rainbow, donde se enumeran y resumen las principales utilidades de la aplicación. La documentación sobre los formatos estándares XLIFF y TMX también ha sido exhaustiva.
    a. Selección del conjunto de utilidades más destacadas. Por ejemplo, la alineación basada en identificadores.
    b. Selección de aquellas otras utilidades que, subordinadas a las anteriores, pueden resultar útiles. Por ejemplo, la pretraducción.
3. *Parte experimental*. Tras explorar las diversas utilidades y revisar la documentación técnica de la herramienta Okapi, el siguiente paso fue elegir un conjunto de aplicaciones reales cuyo texto que traducir estuviera contenido en archivos de diferentes formatos. El objetivo que perseguíamos no era otro que seleccionar una muestra lo más variada posible. Tras la selección de las aplicaciones objeto de estudio y análisis, el último paso ha sido comprobar con Rainbow que podíamos resolver los problemas que nos planteamos.

## 1.4 Estructura del documento

Este documento consta de cuatro grandes apartados:

1. *La introducción.* En este primer apartado se justifican la hipótesis de partida, la motivación, los objetivos y la estructura del documento. Además, se expone la metodología de trabajo.
2. *El marco teórico o conceptual*. Se corresponde con los apartados dos y tres de este documento. En este bloque se abordan conceptos fundamentales como la localización y la internacionalización desde un punto de vista teórico. Después, se justifica la elección de Rainbow como herramienta de trabajo. Por último, se analizan desde una perspectiva teórica los formatos XLIFF y TMX, con los que se trabajará en la parte práctica.
3. *La experimentación práctica.* Se aplica lo expuesto en el marco teórico para alcanzar los objetivos propuestos en la introducción. Se corresponde con los apartados cuatro, cinco y seis. En ellos se analizan detalladamente las utilidades

y potencialidades más destacadas de Rainbow (la conversión a XLIFF y el posproceso, la pseudotraducción, la adaptación de filtros para PROPERTIES y la modificación mediante reglas de filtros para HTML) y se aplican a ejemplos actuales de aplicaciones reales para demostrar su utilidad.

4. *Las conclusiones*. Se corresponde con el séptimo apartado. En él se expone cómo se han alcanzado mediante la experimentación práctica los objetivos inicialmente formulados en la parte introductoria.

# 2 Marco teórico-conceptual

## 2.1 La localización y la internacionalización

La mejor forma de definir la *localización* es exponer en qué se diferencia de la *traducción*, con la que se suele confundir. Johann Roturier lo expone con claridad:

> The term localization is often used to describe a process that encompasses more than the translation of a single digital document (say, a Microsoft Word document) using a Computer Assisted Translation (CAT) tool […]. Other linguistic (and not linguistic) activities are often, if not always, required to adapt an application to the needs of people whose main language is not the same as the one in which the product or service was originally developed[1] (Roturier 2015, 9).

Miguel A. Jiménez-Crespo define la localización como «a complex communicative, cognitive, textual and technological process by which interactive digital texts are modified to be used in different linguistic and sociocultural contexts […]»[2] (Jiménez-Crespo 2013, 20).

Por consiguiente, la localización es más amplia que la traducción en tanto que engloba un abanico mayor de acciones. Sin embargo, localizar *software* que no ha sido preparado para ello puede ser difícil y extraordinariamente costoso. Por ello surgió el concepto de la *internacionalización*:

> [Internationalization is] the process of generalizing a product so that it can handle multiple languages and cultural conventions without the need for redesign. Internationalization takes place at the level of program design and document development[3] (Roturier 2015, 55).

Por tanto, cuando se desarrolla *software* con la vista puesta en su distribución internacional, en el proceso de programación se siguen los estándares de internacionalización que van a facilitar su posterior localización a los mercados

[1] El término localización se suele emplear para describir un proceso que abarca más allá de la traducción de un solo documento digital (como, por ejemplo, un archivo de Microsoft Word) empleando una herramienta de Traducción Asistida por Ordenador (TAO) […]. Otras actividades lingüísticas (y no lingüísticas) también son necesarias con frecuencia, si no siempre, para adaptar la aplicación a las necesidades de las personas cuya lengua materna es distinta de aquella en que se desarrolló el producto originalmente. (Traducción propia).

[2] Un proceso complejo en los aspectos comunicativos, cognitivos, textuales y tecnológicos mediante el que textos digitales interactivos se modifican para que funcionen en distintos contextos lingüísticos y socioculturales […]. (Traducción propia).

[3] [La internacionalización es] el proceso por el que se genera un producto preparado para soportar distintos idiomas y convenciones culturales sin requerir para ello ser rediseñado. La internacionalización se realiza al nivel del diseño del programa y del desarrollo de documentación. (Traducción propia).

elegidos. Por suerte existen organismos internacionales que establecen directrices para normalizar este proceso. LISA (Localization Industry Standards Association) fue la asociación comercial que fijó los estándares de la industria de la localización a nivel mundial hasta el año 2011. Entonces pasaron a ser mantenidos por GALA (Globalization and Localization Association).

En su página web, LISA publicó un glosario que permite comprobar cómo la localización y la internacionalización se complementan. Hoy en día su página web ya no existe y solo es accesible mediante WaybackMachine, una base de datos que contiene copias de seguridad de páginas web que permite acceder a versiones anteriores. Como las definiciones de LISA son muy exhaustivas y están en inglés, se reproduce a continuación una adaptación propia de su contenido en forma de tabla.

| **Localización** | **Internacionalización** |
|---|---|
| El proceso por el que se modifican productos o servicios para adaptarlos a las necesidades de distintos mercados. Se abrevia como L10n (Adaptado de LISA 2011). | El proceso por el que se simplifica la localización o creación de distintas versiones de un producto. Se realiza a nivel técnico, mediante el diseño de la aplicación y la programación del código. Trata de garantizar que se siguen los estándares internacionales. Se abrevia como i18n (Adaptado de LISA 2011). |

**Figura 1. Definiciones de LISA de la L10n y la i18n (adaptadas).**

El *gestor de proyectos de localización* es la persona responsable de gestionar un equipo de traductores para localizar un producto. Según el modelo de M. Quirion (citado en Jiménez-Crespo 2013, 173), las habilidades del gestor de proyectos de localización se dividen en cuatro subtipos de competencias: las traductológicas, las técnicas, las de gestión y las específicas del campo de la localización.

Así, el gestor de proyectos de localización debe ser una persona con amplios conocimientos técnicos. Por esta razón, también recibe el nombre de *ingeniero localizador*. El que se opte por una u otra denominación depende del énfasis del proyecto concreto: *ingeniero localizador* presume un proyecto donde la competencia instrumental técnica es la más destacada, mientras que *gestor de localización* prevé encargos donde la competencia traductológica incide más (Jiménez-Crespo 2013, 175).

En la sección práctica de este documento se exponen tres ejemplos de localizaciones. Todos se realizan sobre las últimas versiones de aplicaciones razonablemente populares que se han programado siguiendo los criterios de internacionalización de LISA y GALA con el fin de facilitar su posterior localización. Esto se traduce en que su arquitectura prevé futuras labores de localización y, para simplificarlas, externaliza la parte traducible de lo que es el código lógico en sí mismo.

Tomemos uno de los tres ejemplos contemplados en este documento para ilustrarlo: una aplicación Android preparada para su internacionalización. La lógica y los mensajes están físicamente unidos tras la compilación pero separados dentro de la APK.


**Figura 2. Arquitectura general de las aplicaciones Android.**

La extensión «.apk», se deriva de las siglas en inglés de *Android Application Package* (en castellano Aplicación empaquetada de Android). Es una variante del formato JAR de Java y funciona como un archivo comprimido en ZIP en tanto que se puede descomprimir e inspeccionar empleando un software como 7-Zip o Winzip.

El que la aplicación fuera programada buscando facilitar su futura localización implica que, de acuerdo con la *Guía API* para desarrolladores de Android, contiene un subdirectorio «res/» donde reúne los recursos localizables o *resources*.

> Resources are text strings, layouts, sounds, graphics, and any other static data that your Android application needs. An application can include multiple sets of resources, each customized for a different device configuration[4] (Desarrolladores de Android 2017a).

[4] Los *resources* son las cadenas de caracteres, los diseños, los sonidos, los elementos gráficos y cualesquiera otros recursos estáticos que su aplicación Android requiera. Una aplicación puede incluir varios sets de recursos, cada uno de ellos adaptado a una distinta configuración del dispositivo. (Traducción propia).

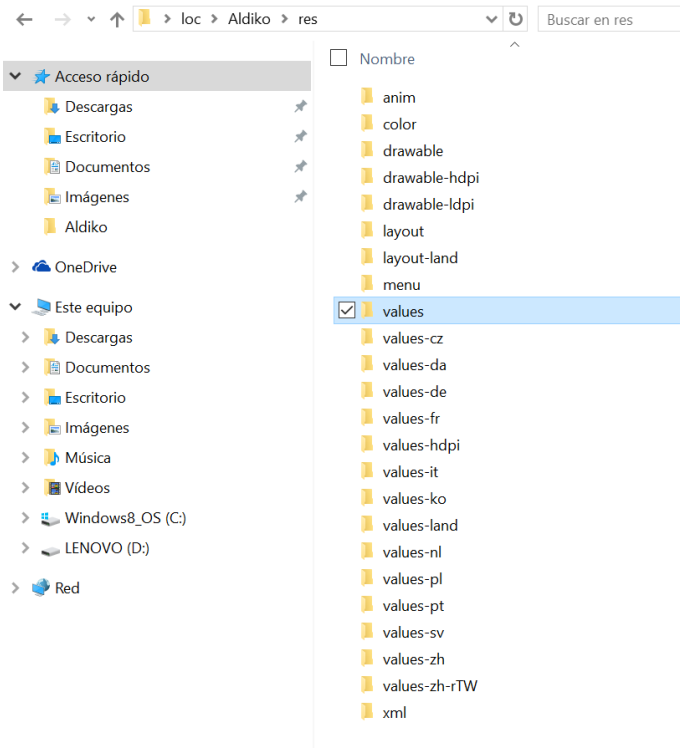



**Figura 3. Ejemplo de una APK preparada para la localización.**

Así, al descompilarla nos encontramos con distintas subcarpetas dentro del directorio «res/» que contienen los elementos localizables de la aplicación antes citados: el texto que se muestra al usuario, las imágenes, los elementos de la interfaz, etc. Como se puede comprobar en la fig. 3, esta aplicación está perfectamente estructurada de tal forma que la localización sea sencilla, rápida y segura: al tener los ficheros traducibles separados en ficheros XML ordenados por carpetas, evita que alguien modifique por error una parte del código que no debería. En este ejemplo, para localizar la aplicación al castellano europeo el localizador debe crear un subdirectorio de «values» (que será «values-es» para España) donde introducirá los ficheros XML traducidos.

Gracias a esta arquitectura pensada para la internacionalización, cuando un usuario ejecute la aplicación, Android automáticamente sabrá qué recursos debe cargar en función de la configuración del dispositivo. Así, si se ejecuta en un móvil configurado en España, cargará el subdirectorio «values-es»; si está en Portugués, «values-pt»; si está en italiano, «values-it»; y así respectivamente, siempre que dicha configuración del país tenga un subdirectorio en «res/».

## El auge de la localización

A medida que crece el número de productos tecnológicos que se distribuyen por todo el planeta, también lo hace la necesidad de localizarlos porque los usuarios muestran su preferencia por el *software* adaptado a su cultura (Roturier 2015, 4). Por tanto, desde una perspectiva económica, las empresas y los particulares que producen contenidos deben tener siempre presentes las directrices establecidas para la internacionalización de productos porque ello agilizará y abaratará la posterior localización de los mismos a terceros mercados. Así, se ha demostrado que el retorno de la inversión hace rentable localizar porque permite además aprovechar nuevas oportunidades empresariales (*Íbid*., 5).

Más allá de la perspectiva mercantil, desde un punto de vista humanista resulta imprescindible adaptar el contenido a las necesidades de culturas en principio menos fuertes para evitar la imposición de valores y lenguas. Además, cabe destacar que en muchos países la legislación directamente exige la labor de localización. Recordemos que la localización es un concepto más amplio que la traducción: consiste en la adaptación total del producto al mercado destino y, en algunos casos, esto incluye cambiar aspectos técnicos. Por ejemplo, la legislación de la Unión Europea es particularmente exigente con la protección de datos.

## 2.2 Trabajar con Okapi Rainbow

Rainbow es una de las seis aplicaciones que forman el Marco Okapi (Okapi Framework). En su página web, Okapi define su marco como «a free, open-source and cross-platform set of components and applications designed to help your localization and translation processes»[5] (Okapi Framework 2017a). Todas las aplicaciones del proyecto tienen en común que emplean y fomentan el uso de los llamados «open standards» (estándares abiertos), cuyo uso defiende la industria de la traducción y la localización.

Así, en este documento nos hemos centrado en Rainbow porque, al formar parte del Marco Okapi, es una herramienta gratuita, de código abierto y multiplataforma que además trabaja con estándares. Para elaborar la parte práctica hemos empleado la última versión estable disponible de Rainbow, la v.6.0.32 también denominada M32

[5] Un conjunto de componentes y aplicaciones gratuitas, de código abierto y multiplataforma diseñado para ayudarle en los procesos de localización y traducción. (Traducción propia).

(JFrog 2017a), que fue actualizada por última vez el 31 de enero del 2017. Durante el proceso de redacción de este documento una nueva versión, la M33, ha pasado a estar disponible a partir del día 8 de mayo de 2017 (JFrog 2017b).

Entre las muchas potencialidades de Rainbow destacan la creación de paquetes de traducción en formatos estandarizados, la conversión entre formatos, la extracción de terminología, la alineación basada en identificadores, la búsqueda y reemplazamiento, el renombrado, la pseudotraducción, la pretraducción y la modificación y adaptación de filtros. Como la lista completa de utilidades es aún más amplia, en este documento nos hemos centrado en aquellos aspectos que creemos especialmente relevantes por la frecuencia con que se emplean en los encargos reales de localización ya que, dada su relativa complejidad, requieren ser explicados.

Las utilidades de Rainbow se construyen a modo de «pipelines», término traducible como «canales». Consisten en listas de tareas formuladas a partir de acciones simples, denominadas «steps», que están perfectamente definidas. Es decir, algunas de las utilidades antes citadas (por ejemplo, la pseudotraducción) son «pipelines» predefinidas compuestas por «steps» mediante los que se va avanzando paso a paso hacia el objetivo buscado. Además, los «steps» se pueden ejecutar por separado y Rainbow permite crear «pipelines» personalizadas.

Uno de los aspectos más destacables de Rainbow es que fomenta el uso estándares. Según exponen en su página web, esto persigue un doble objetivo (Okapi Framework 2016a):

1. Evita que la información quede sujeta a formatos que son propiedad de empresas privadas.
2. Permite ampliar el número de herramientas con que traducir y localizar ya que estos formatos son reconocidos por un abanico más extenso de aplicaciones.

En el marco de la gestión de proyectos, este segundo aspecto de Rainbow es especialmente destacable porque facilita el trabajo en equipo. Es decir, posibilita trabajar con ficheros para los que algunas herramientas TAO no disponen de filtros adecuados. Así, gracias a Rainbow un gestor de proyectos de localización podrá coordinar un equipo de traductores que trabaja con herramientas TAO distintas y

formatos de archivo diferentes porque será capaz de pasar los ficheros a los estándares como, por ejemplo, XLIFF (XML Localization Interchange File Format) para los archivos bilingües; TMX (Translation Memory Exchange) para las memorias de traducción; o TBX (TermBase eXchange) para las bases de datos terminológicas.

## 2.3 El formato XLIFF

OASIS es el consorcio sin ánimo de lucro que fomenta el desarrollo, la convergencia y la adopción de los estándares abiertos. Creó XLIFF en 2002 y lo define así: «XLIFF is a specification for the loss-less interchange of localizable data and its related information»[6] (OASIS 2008a). John Roturier lo define en términos más sencillos: «XLIFF, the XML Localization Interchange File Format, is a type of XML document which is used to exchange information during a localization project»[7] (Roturier 2015, 34).

La principal ventaja de XLIFF es que «it is tool-neutral, has been formalized as an XML vocabulary, and features an extensibility mechanism»[8] (OASIS 2008a). Tal y como explica OASIS en su página web, es un formato que creó para adaptar el proceso de trabajo de traducción de tal forma que se evitaran los siguientes cuatro problemas:

1. La interoperabilidad insuficiente entre herramientas de traducción y localización.
2. La falta de soporte en el proceso de trabajo de las labores de localización.
3. La necesidad de herramientas que lidien con multitud de formatos.
4. La gran cantidad de formatos intermedios.

Gracias a XLIFF, estos problemas tuvieron respuesta. La existencia de un estándar redujo la dependencia de los formatos especiales privados, aumentó el control de los localizadores sobre su trabajo al aumentar su conocimiento del formato con que trabajaban y mejoró la interoperabilidad entre herramientas. Todos estos aspectos se ilustrarán en detalle en el punto siguiente de este documento.

[6] XLIFF es la especificación establecida para el intercambio sin pérdidas de información localizable y sus datos asociados. (Traducción propia).
[7] XLIFF, el formato de intercambio de archivos de localización en XML, es un tipo de documento XML que se emplea para intercambiar información durante los proyectos de localización. (Traducción propia).
[8] No depende de ningún programa informático, se ha formalizado a modo de un vocabulario XML y presenta un mecanismo de extensibilidad. (Traducción propia).

## La localización sin el formato XLIFF

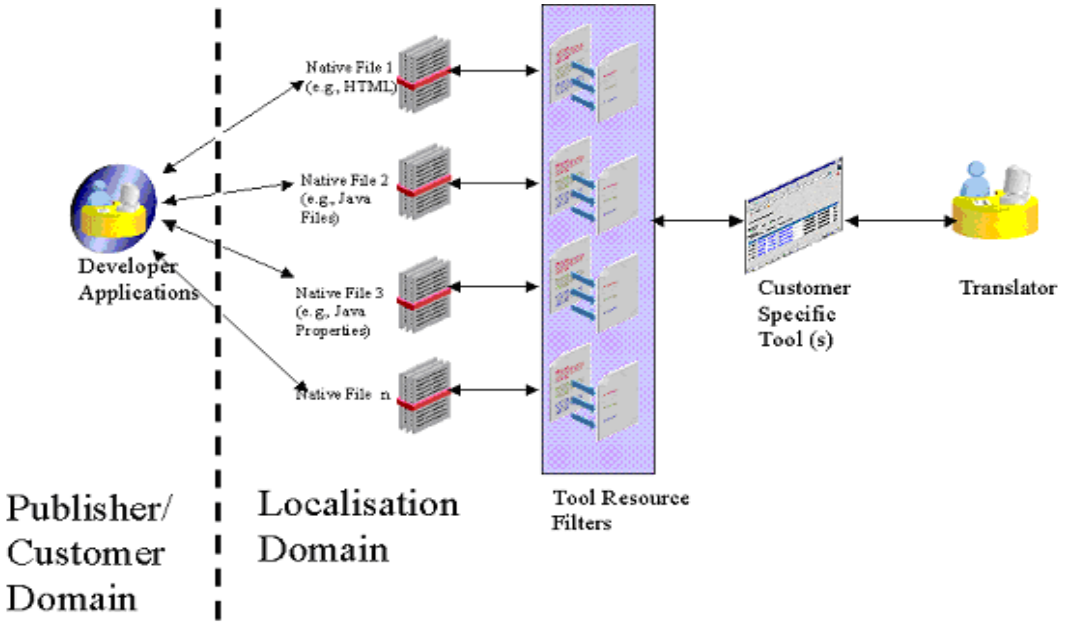



**Figura 4. Localizar sin XLIFF (extraído de OASIS 2008a, 11).**

Tal y como lo expone OASIS, el proceso general de un proyecto de localización consiste en que «the translatable content ("resources") of an application, database or website is first extracted, translated or modified for a given language or market and finally rebuilt or redeployed»[9] (OASIS FAQ 2008a, 11).

La figura 4 ejemplifica cómo se realiza dicho proceso sin emplear el formato estándar XLIFF. En primer lugar, un desarrollador programa un código que después entrega a un ingeniero localizador. Por tanto, la parte de la izquierda de la imagen ilustra cómo el programador no realiza ninguna tarea de internacionalización ni localización: simplemente presenta el código a un localizador mediante ficheros en su formato nativo.

Después, el ingeniero o gestor localizador emplea múltiples filtros para interpretar los recursos localizables de los archivos originales. Con frecuencia tendrá que emplear varias herramientas distintas o adaptadas específicamente al cliente. Finalmente, distribuye los ficheros a un equipo de traductores para que los localice. Por supuesto, una vez estos completen su trabajo, el ingeniero localizador deberá realizar el proceso de reconversión de todos los ficheros.

[9] El contenido traducible (los «recursos») de una aplicación, una base de datos o un sitio web, en primer lugar se extrae, después se traduce o modifica para adaptarlo a un determinado idioma o mercado; y finalmente se reconstruye o redistribuye. (Traducción propia).

Como bien indica OASIS en su página web; este modelo de trabajo era muy reactivo en tanto que cada vez que un formato cambiaba o se introducía uno nuevo, el localizador debía aprender su funcionamiento y, además, asegurarse de que disponía del filtro adecuado para trabajar con él. Asimismo, cualquier error en las múltiples conversiones intermedias de formatos podía echar a perder el resultado final de la localización.

## La localización con el formato XLIFF

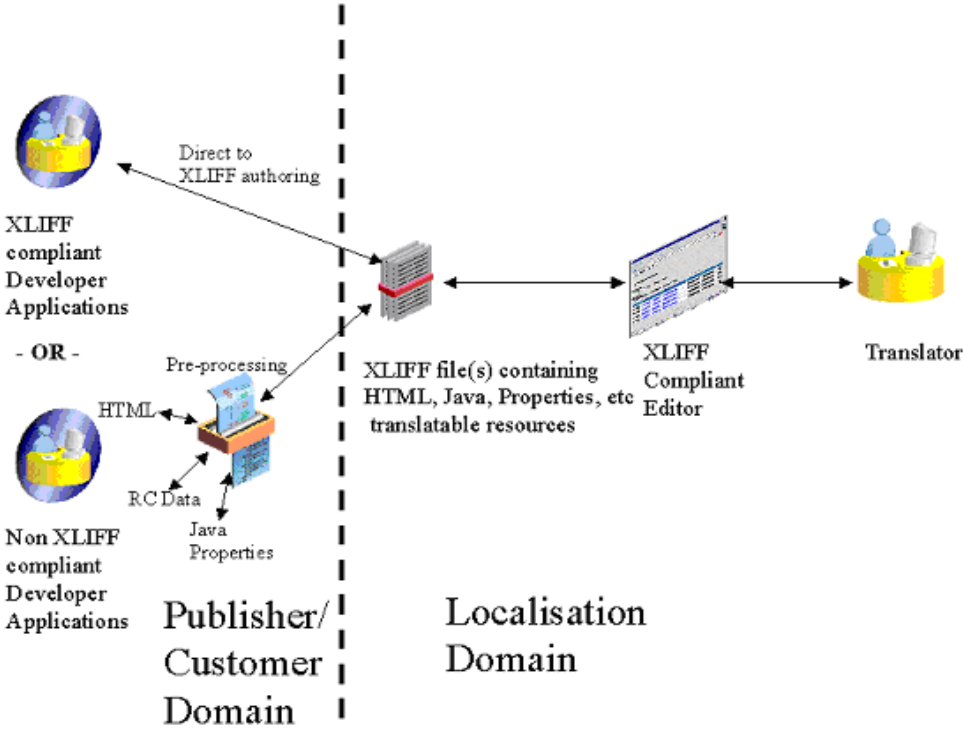



**Figura 5. Localizar con XLIFF (extraído de OASIS 2008a, 12).**

La figura 5 muestra simultáneamente las dos formas más frecuentes en que el formato XLIFF facilita la gestión de proyectos de localización.

En la parte superior izquierda se muestra el proceso de localización que inicia un desarrollador al enviar directamente al gestor de proyectos los archivos convertidos a XLIFF. Dichos ficheros XLIFF pueden contener información proveniente de ficheros HTML, JAVA, PROPERTIES, etc. Después, el traductor solo tiene que emplear la herramienta TAO de su elección para traducir el XLIFF y enviárselo de vuelta al gestor de tal forma que este, sin necesidad de cambiar el formato, se lo devuelva al programador.

En la misma imagen, partiendo de la esquina inferior izquierda, se muestra un proceso ligeramente más complejo porque añade un paso más: el programador envía los archivos en el formato original y hay un pre-procesador que los convierte a XLIFF. Como se ha ilustrado anteriormente en este documento, Okapi Rainbow resulta idóneo para realizar este paso intermedio. Una vez convertidos a XLIFF, el resto del proceso es igual que en el caso anterior. La reconversión de vuelta a los formatos originales también puede realizarse con Rainbow.

Así, la aplicación Rainbow permite convertir cualquier archivo para el que tenga filtro al formato XLIFF. Por tanto, facilita al ingeniero o gestor de proyectos de localización trabajar con un estándar, lo que implica el proceso será más rápido, eficiente y seguro.

Rainbow emplea por defecto la versión 1.2 de XLIFF, publicada en 2002, que denomina «Generic XLIFF» porque se convirtió en el formato estándar de OASIS en febrero del 2008 (Schnabel, Hackos y Raya 2015, 15). Además, también dispone de filtro para la versión 2.0 de XLIFF, publicada en el 2014. Sin embargo, esta última aún está en formato beta, es decir, experimental, y no está lista para entornos de explotación. Por esta razón, en los ejemplos de este documento hemos empleado la v.1.2. Cabe señalar que en el año 2016 OASIS empezó a trabajar en el borrador de la versión 2.1.

## 2.4 El formato TMX

LISA (Localization Industry Standards Association) publicó el estándar TMX en 1997. Toma su denominación de *Translation Memory eXchange format* (traducible como Formato de intercambio de memorias de traducción). «TMX es un lenguaje que cumple las especificaciones XML y cuyo propósito es proporcionar un estándar para el intercambio de las memorias de traducción» (Gómez 2001).

Por tanto, basta con exportar la MT (Memoria de Traducción) a este formato estandarizado para que sea importable a cualquier TAO con filtro para TMX. Hoy en día todas las herramientas populares de TAO como SDL Trados, MateCat o MemoQ disponen filtros para TMX.

> The purpose of the TMX format is to provide a standard method to describe translation memory data that is being exchanged among tools and/or translation vendors, while introducing little or no loss of critical data during the process[10] (GALA 2011).

Con la desaparición de LISA, TMX pasó a ser un formato bajo la licencia Creative Commons 3.0, que permite modificar, compartir y hacer uso comercial del mismo siempre que se atribuya la autoría original a LISA. La última versión operativa del estándar es la 1.4b (GALA 2011) y es con la que trabaja Okapi Rainbow (Okapi Framework 2016a).

[10] El objetivo del formato TMX es proporcionar un método estandarizado mediante el que describir el contenido de las memorias de traducción que se está intercambiando entre herramientas y proveedores de traducciones. Gracias al formato TMX se logra que se pierda poca o ninguna información crítica durante el proceso. (Traducción propia).

# 3 Justificación de las utilidades de Okapi Rainbow aquí exploradas

El abanico de potencialidades que ofrece Rainbow es vasto y, además, dado que, como ya se ha indicado en apartados anteriores de este documento, permite subdividir los *pipelines* en *steps*, las utilidades se pueden recombinar. Así, por motivos de espacio, en este documento solo abordaremos algunas de ellas: las que consideramos más útiles para los estudiantes del Grado en Traducción e interpretación. Estas utilidades son las que les facilitarán su futura labor como gestores de proyectos de localización.

Las utilidades de Rainbow abordadas en este documento son las siguientes:

A. **La creación de paquetes de traducción**. Consiste en convertir los distintos formatos al estándar XLIFF para facilitar que se puedan traducir y revisar mediante cualquier TAO.
B. **El posproceso de paquetes de traducción**. Es el final de un proyecto de traducción, a menudo llamado «back-conversion» o «conversión de vuelta»: consiste en revertir los ficheros del formato bilingüe XLIFF al formato original en el que se encontraban los archivos en la aplicación pero ya en lengua destino.
C. **La pseudotraducción**. Simula el aspecto que tendrá un documento una vez traducido. Resulta muy útil para realizar la triple comprobación expuesta en el apartado 4.1 del presente documento.
D. **La alineación basada en identificadores**. Se emplea para crear un archivo bilingüe (generalmente un TMX) a partir de dos monolingües. La sincronización de los textos origen y el meta se realiza mediante el identificador de cada una de las unidades textuales de los archivos de origen.
E. **La pretraducción**. Es la búsqueda y sustitución automática de los segmentos de un texto que coinciden total o parcialmente con los segmentos almacenados en una MT. El resultado de esta tarea es una primera versión de la traducción y permite ahorrar mucho trabajo a los traductores y aumentar la consistencia terminológica.
F. **La adaptación de los filtros para PROPERTIES**.
G. **La modificación mediante reglas de los filtros para HTML**.

Por su naturaleza, algunas de las potencialidades anteriores están pensadas para ejecutarse de forma conjunta. Así, en este documento se han agrupado en tres

experimentos que maximizan su utilidad. Como uno de nuestros objetivos es mantener siempre un enfoque práctico, hemos empleado las últimas versiones de tres aplicaciones reales distintas para, mediante experimentos basados en encargos reales, mostrar las utilidades de que dispone Rainbow para cada caso particular.

Los tres experimentos que aquí abordamos son los siguientes:

1. El primero se corresponde con el cuarto apartado de este documento. El objetivo del experimento es pseudotraducir una parte del propio programa Rainbow. Aquí se exponen contextualizadas la creación de un paquete de traducción, la pseudotraducción y el posproceso de dicho paquete. Se trabaja en todo momento con el formato PROPERTIES (porque es en el que emplea la aplicación real Rainbow) y se adaptan los filtros nativos de Okapi Rainbow para optimizar la labor de localización.
2. El segundo experimento queda expuesto en el quinto apartado del presente documento. El objetivo propuesto es trabajar con un par de ficheros XML en idiomas distintos de la versión antigua de una aplicación Android para pretraducir la versión nueva. Se exponen aquí la alineación basada en identificadores, con el fin de generar un TMX, y la pretraducción, que trabajará a partir de dicho TMX.
3. El tercer experimento se corresponde con el sexto apartado de este documento. Tiene un carácter más técnico que los anteriores y aborda la modificación mediante reglas del filtro para capturar la parte traducible de los ficheros de ayuda HTML de Microsoft Windows.

El análisis que aquí realizamos no es exhaustivo por motivos de espacio. Por tanto, tratamos de abarcar lo máximo posible y, con este fin, nos centramos en aquellos aspectos de las utilidades que pueden plantear alguna dificultad a los futuros gestores de proyectos: la adaptación de los filtros a las características concretas de cada archivo, las peculiaridades de los formatos y algunas cuestiones un poco más complejas que se derivan de la configuración avanzada de Rainbow.

# 4 La pseudotraducción, el proceso de creación de un XLIFF y la conversión de vuelta

## 4.1 Definición y finalidad

Las pseudotraducciones son «aquellos textos presentados como traducciones, pero que en realidad no son tales» (Pegenaute 1999). En el contexto de la traducción asistida por ordenador, el significado del término se precisa más: «Pseudo-translation is a procedure which simulates how a translated document will look after translation and how much extra […] work will be required»[11] (SDL Trados Studio 2016).

Así, en el marco de los proyectos de localización, pseudotraducir persigue un objetivo triple:

1. Comprobar la correcta visualización de caracteres.
2. Verificar el espacio para el texto en la interfaz del programa. Es decir: probar el crecimiento o decrecimiento de las cadenas de caracteres.
3. Constatar qué partes de la interfaz de la aplicación están contenidas en el fichero en cuestión.

## 4.2 Ejemplo real de uso de la pseudotraducción

En este documento vamos a exponer la pseudotraducción de un fichero PROPERTIES de la aplicación Rainbow (de código abierto). Nuestro objetivo es realizar la triple comprobación expuesta en el apartado anterior. Así, una vez realizada, como gestores de proyecto sabremos a qué atenernos a la hora de abordar la localización per se.

## 4.3 El formato PROPERTIES

Para ilustrar la pseudotraducción se va a exponer aquí un ejemplo realizado con archivos PROPERTIES. Se ha optado por ellos porque son muy utilizados al permitir almacenar los parámetros configurables de aplicaciones informáticas, como, por ejemplo, en el caso del propio Okapi Rainbow. Además, con frecuencia son empleados con el fin de facilitar la internacionalización y la localización de aplicaciones informáticas y *software*.

[11] La pseudotraducción es un procedimiento que se simula el aspecto que tendrá un documento cuando sea traducido y muestra así cuánto trabajo […] posterior requerirá. (Traducción propia).

## El filtro para PROPERTIES

El filtro necesario para realizar la pseudotraducción dependerá del formato del archivo con el que se trabaje. Así, como en este ejemplo hemos seleccionado un archivo PROPERTIES, el filtro será el específico para archivos con dicha extensión. «The Properties Filter is an Okapi component that implements the IFilter interface for properties files. [...] The implementation is based on the specification found in the Java java.util.Properties class documentation»[12] (Okapi Framework 2016b).

Al tratarse de un fichero PROPERTIES, solo se traduce la parte derecha del símbolo igual y la izquierda, que es el *identificador*, debe permanecer inalterada. Es decir, el identificador lleva asociado un texto que en algún momento se mostrará al usuario en la interfaz del idioma configurado.

Rainbow dispone de un filtro para trabajar con dichos archivos de tal forma que detecta la parte traducible y la porción que no debe modificar. Sin embargo, esto no es suficiente. Es decir, el texto de los menús (por ejemplo) lo puede pseudotraducir sin problema, pero no tiene sentido que pseudotraduzca las *hotkeys* y los atajos porque, de hacerlo, establecería combinaciones de teclas imposibles como, por ejemplo, la combinación «Control+211» en lugar de «Control+1». El texto contenido en el recuadro azul de la figura 6 es el que se puede pseudotraducir de forma normal: el resto de código a la derecha de la igualdad está formado por aceleradores y atajos.

```
utilities.openOutputFolder.text=&Open Last Output Folder\tCtrl+L
utilities.openOutputFolder.ctrl=1
utilities.openOutputFolder.key=L
```

**Figura 6. Fragmento de código extraído del fichero Commands.properties.**

Por tanto, antes de empezar a trabajar, el primer paso es modificar el filtro. Al añadir ficheros PROPERTIES a la *Input List 1*, Rainbow emplea por defecto el filtro okf_properties. Sin embargo, puede resultar interesante para el localizador editarlo si desea que la pseudotraducción solo se aplique al texto per se y no a los aceleradores.

Para editar el filtro se hace clic con el botón derecho, se selecciona *Edit document properties* y, a continuación, se crea uno nuevo a partir del seleccionado puesto que Rainbow no permite modificar mediante la interfaz los filtros predefinidos.

[12] El filtro PROPERTIES es un componente de Okapi que implementa la interfaz IFilter para los ficheros PROPERTIES […]. La implementación se basa en las especificaciones contenidas en la documentación Java de clase java.util.Properties. (Traducción propia).

Al recién creado se le da la denominación deseada y se genera automáticamente a modo de un nuevo archivo en el directorio de trabajo.

Al crear el nuevo filtro, en el apartado *Key filtering* de la pestaña *Options* seleccionaremos las *keys* para indicarle al programa cuáles queremos que pseudotraduzca y cuáles no. En este caso, se le ha marcado que solo trabaje con las cadenas que contienen la expresión «text» de tal forma que, cuando las detecta, pseudotraduce el texto asociado a dicha *key* o cadena.

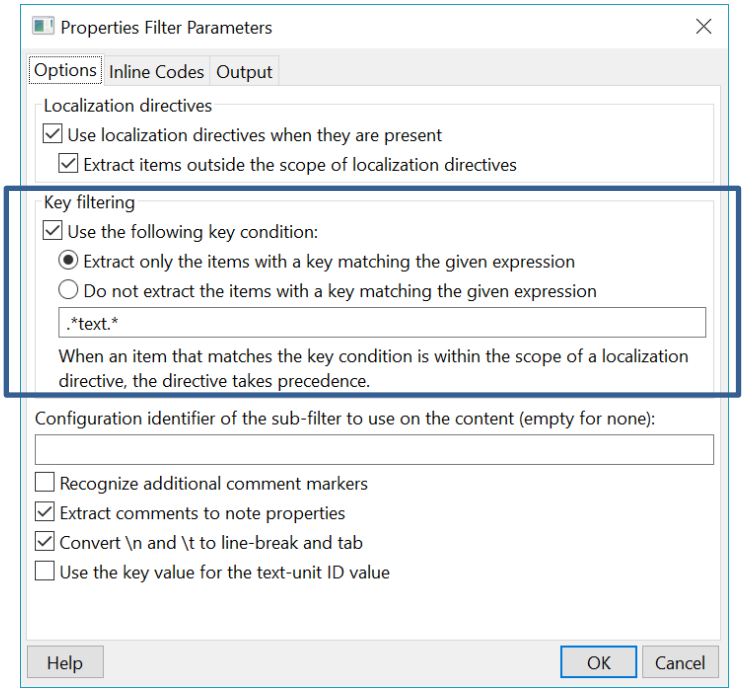



**Figura 7. Configuración del filtro para PROPERTIES.**

La codificación en los ficheros PROPERTIES sigue la norma ISO-8859-1, conocida como el Alfabeto latino 1, que da soporte a los idiomas de Europa occidental. Así, los no contenidos en la misma deben ser introducidos mediante caracteres escapados Unicode. Por tanto, en idiomas como el chino o el árabe, la mayoría de los caracteres estará escapada.

## 4.4 Pseudotraducir con Rainbow

Con el fin de mostrar el funcionamiento de la pseudotraducción, trabajaremos con un fichero PROPERTIES de una copia de la propia aplicación Rainbow; es decir, pseudotraducimos una parte copiada del programa Rainbow mediante Rainbow. Este ejercicio es totalmente real ya que dicha aplicación no está disponible en español. El objetivo es simular con caracteres del alfabeto latino extendido la traducción del texto

contenido en la interfaz del programa para realizar la triple comprobación antes mencionada en el apartado 4.1 de este documento.

Este ejercicio viene justificado por el hecho de que las diferencias gramaticales y léxicas entre idiomas pueden hacer que los textos aumenten de tamaño (como suele ocurrir al traducir del inglés al castellano). Así, el texto localizado podría dar problemas; el más probable, que no entre en los espacios de la interfaz del programa.

Para acceder a la utilidad de pseudotraducción en Rainbow hay que entrar en el submenú *Utilities* y seleccionar «Text Rewriting…» [sic]. Como paso previo se habrá añadido el archivo que se desea pseudotraducir a la *Input List 1*. No es necesario modificar la configuración de los idiomas en la pestaña *Languages and Encodings* para realizar esta tarea.

A continuación se abrirá una ventana titulada *Pre-Defined Pipeline: Text Rewriting*. Los *pipelines* son las cadenas de tareas que se le pueden programar a la aplicación. Así, *Text Rewriting* podría entenderse como una *pipeline* que viene ya predefinida con el programa. Si se selecciona *Text Modification* en el menú de la izquierda, Rainbow permite seleccionar el tipo de cambio de texto que deseamos realizar. Las opciones bajo esta lista permiten añadir caracteres y etiquetas antes y después del texto pseudotraducido.

En la fig. 8, que se muestra a continuación, puede comprobarse cómo mediante el filtro hemos conseguido que Rainbow solo pseudotraduzca el texto.

```
11 file.new.text=[&\u00d1\u00e8w\t\u00c7trl+\u00d1 zzzzz]
12 file.new.ctrl=1
13 file.new.key=N
14
15 file.open.text=[&\u00d8p\u00e8\u00f1...\t\u00c7trl+\u00d8 zzzzz z]
16 file.open.ctrl=1
17 file.open.key=O
18
19 file.save.text=[&S\u00e5v\u00e8\t\u00c7trl+S zzzzz]
20 file.save.ctrl=1
21 file.save.key=S
22
23 file.saveas.text=[S\u00e5v\u00e8 &\u00c2s... zzzzz]
24
25 file.mru.text=[R\u00e8\u00e7\u00e8\u00f1t S\u00e8tt\u00ec\u00f1gs F\u00ecl\u00e8s zzzzz zzzzz zzzzz zz]
26
27 file.clearmru.text=[&\u00c7l\u00e8\u00e5r R\u00e8\u00e7\u00e8\u00f1t S\u00e8tt\u00ec\u00f1gs F\u00ecl\u00e8s L\u00ecst...
   zzzzz zzzzz zzzzz zzzzz zzzzz]
28
29 file.userpref.text=[&\u00dbs\u00e8r Pr\u00e8f\u00e8r\u00e8\u00f1\u00e7\u00e8s... zzzzz zzz]
30
31 file.exit.text=[\u00c9&x\u00ect\t\u00c2lt+F4 zzzzz]
32 file.exit.alt=1
33 file.exit.key=F4
34
35 #========== View menu
36 view.text=[&V\u00ec\u00e8w zz]
37
38 view.inputList1.text=[\u00cf\u00f1p\u00fct L\u00ecst &1\t\u00c7trl+1 zzzzz zzz]
```

**Figura 8. Ejemplo de pseudtraducción con filtro.**

## 4.5 Conversión de PROPERTIES a XLIFF y proceso de vuelta

Este paso tiene como objetivo transformar los archivos PROPERTIES al formato bilingüe estandarizado denominado XLIFF, con el fin de que sea posible localizarlos con cualquier TAO tenga filtro o no para PROPERTIES. Así, para el ejemplo real aquí analizado, esta utilidad nos ofrece dos posibilidades: o bien podemos revisar la pseudotraducción realizada mediante Rainbow o bien podemos generar un archivo bilingüe con Rainbow para después pseudotraducirlo directamente mediante una TAO. En ambos casos, una vez traducido, dicho fichero bilingüe se posprocesará mediante Rainbow para que el contenido del XLIFF recupere su formato original PROPERTIES.

Para realizar este paso resulta muy pertinente establecer en la pestaña *Languages and Encodings* el par de idiomas y la codificación porque así evitaremos problemas más adelante en la herramienta TAO. Una vez establecidos, se hace clic en *Utilities* > *Translation Kit Creation* y se selecciona el documento XLIFF genérico. Rainbow permite llevarlo a la versión XLIFF v2 (en formato beta) pero a efectos prácticos vamos a emplear la versión genérica v1.2 porque sigue siendo la de uso más frecuente. Por ejemplo, hoy día aplicaciones como SDL Trados trabajan por defecto con ella.

Para verificar que el XLIFF se ha creado correctamente, comprobamos que hay un archivo *Manifest* en el directorio de trabajo. Este nos resultará imprescindible más adelante para crear el paquete de retorno del formato bilingüe XLIFF al original (en este caso PROPERTIES). El fichero bilingüe estará contenido en una carpeta creada automáticamente denominada «work».

La ventaja de realizar esta tarea con Rainbow es que el gestor de proyectos evita tener que emplear una herramienta TAO y crear paquetes de proyecto para conseguir el archivo bilingüe. Por consiguiente, gracias a Rainbow puede realizar esta tarea de comprobación de forma rápida y gratuita.

En el Anexo 1 de este documento se muestra una captura de pantalla de una pseudotraducción realizada con SDL Trados (con los parámetros por defecto) y a su lado, la aquí expuesta mediante Rainbow. Como se puede comprobar, los resultados son distintos porque el filtro personalizado creado anteriormente ha funcionado y porque SDL Trados pseudotraduce empleando por defecto palabras al azar del mismo idioma (en este caso el castellano).

Una vez se ha trabajado con el fichero bilingüe en una TAO para realizar o comprobar la pseudotraducción, el siguiente paso es revertir el XLIFF a su formato anterior. En este caso, como hemos pseudotraducido un fichero PROPERTIES de la aplicación Rainbow, el fichero bilingüe en XLIFF que hemos conseguido lo tenemos que posprocesar para que recupere su formato PROPERTIES original.

Esto se logra mediante la utilidad *Translation Kit Post-Processing.* Antes de empezar, debemos asegurarnos de que hemos introducido en la carpeta «work» la versión XLIFF con que hemos trabajado mediante una TAO porque esa será la que se convierta al formato PROPERTIES.

Así, una vez tomada en cuenta dicha consideración, ejecutamos Okapi Rainbow y añadimos a la *Input List 1* el fichero *Manifest* que se nos había generado al crear el bilingüe XLIFF en el paso anterior. Hacemos clic en el menú en *Utilities* > *Translation Kit Post-Processing*. Tras ejecutarlo se nos mostrarán en una nueva ventana los documentos que queremos posprocesar. Continuamos y comprobamos que dentro del directorio de trabajo ha aparecido una nueva carpeta nombrada «done» que contendrá el fichero bilingüe de la carpeta «work» en su formato original, en este caso, PROPERTIES. El fichero ya está listo para ser compilado, firmado e incorporado a la copia de Rainbow. De esta manera, podremos realizar la triple comprobación del punto 4.1 de este documento, que era el objetivo que nos planteábamos conseguir mediante la pseudotraducción.

# 5 La alineación basada en identificadores para generar un XLIFF pretraducido

## 5.1 Definición y finalidad

La alineación consiste en parear los segmentos de dos ficheros en idiomas distintos a un solo fichero en formato TMX. El archivo generado mediante la alineación puede resultar muy útil para traducir o pretraducir automáticamente ficheros.

En la aplicación Rainbow, el TMX se obtiene mediante la utilidad *ID-Based Alignment*, traducible al castellano como Alineación basada en identificadores:

> This utility allows you to create a bilingual output from two monolingual input files. The synchronization between source and target text units is done based on the identifier of each text unit of the input files. Each entry in the input files must have an identifier[13] (Okapi Framework Tutorials, 2017).

Al ejecutar esta utilidad, Rainbow busca segmentos de texto en dos documentos en idiomas distintos en base a los identificadores. Es decir, cuando dos segmentos tengan el mismo identificador, Rainbow alineará el texto que tengan asociado. Por tanto, no importa el orden en que figuren los segmentos en dichos archivos ya que el único criterio para alinearlos son los identificadores.

## 5.2 Ejemplo real de uso de la alineación basada en identificadores

El ejemplo real aquí planteado parte del supuesto de que, como gestores de proyecto, disponemos de los ficheros XML que contienen el texto en castellano y en inglés de versiones antiguas de una aplicación para Android. Ha salido una versión nueva y, para localizarla, queremos sacar partido al texto ya traducido en las versiones anteriores. La forma de hacerlo es alinear dichos ficheros XML en base a sus identificadores mediante Rainbow para obtener un TMX. Este último lo emplearemos después para pretraducir automáticamente (también con Rainbow) y trabajar de forma más eficiente, cómoda y rápida. Además, así respetaremos la coherencia con la terminología anterior.

[13] Esta utilidad le permite crear un archivo bilingüe a partir de dos monolingües. La sincronización de los textos origen y el meta se realiza mediante el identificador de cada una de las unidades textuales de los archivos de origen. Cada entrada en los archivos de origen debe tener un identificador. (Traducción propia).

## 5.3 El formato XML ANDROID

El formato XML, cuyo nombre se obtiene de las siglas en inglés de *eXtensible Markup Language*, es un metalenguaje ideado para representar información de cualquier tipo. Comparte con HTML que ambos son lenguajes de etiquetas o marcas organizadas de forma jerárquica. Sin embargo, HTML es un lenguaje de *presentación* de información mientras que XML sirve para su *representación*; es decir, XML sirve para dotar de estructura a información de cualquier tipo.

Por tanto, los documentos XML contienen gran cantidad de información organizada. En el mundo profesional de la traducción han surgido muchos estándares basados en XML con el objetivo que facilitar el intercambio de ficheros entre distintas TAOs: TMX, XLIFF, TBX, SRX, xml:tm; GMX, etc. Así, es un metalenguaje que funciona a modo de estándar para el intercambio de información estructurada entre plataformas y, por consiguiente, es el formato nuclear de las tareas de localización e internacionalización. De esta manera, resulta crucial que el profesional de la traducción reconozca los elementos traducibles en archivos estructurados en dicho metalenguaje.

Cuando el formato XML se adapta a las aplicaciones de Android, se afirma que es un XML ANDROID. Las especificaciones del formato STRINGS XML están definidas en la *Guía API para desarrolladores de Android*, accesible de forma gratuita por Internet. A continuación se muestra un extracto del código del ejemplo de aplicación Android que se va analizar en el apartado siguiente de este documento.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    […]
  <string name="abc_searchview_description_clear">Borrar consulta</string>
  <string name="abc_searchview_description_query">Consulta</string>
  <string name="abc_searchview_description_search">Buscar</string>
    […]
```

En el ejemplo anterior la primera línea especifica la versión de XML y la codificación del archivo. A continuación, aparece la *etiqueta raíz* «resources»: como este documento contiene recursos *string*, debe contener una etiqueta raíz que los englobe a todos. Después, figura el *nombre de la etiqueta* («string») y el de su *atributo* («name»). El *valor* de cada atributo sería, en cada caso, el elemento de color morado. A

continuación se muestra en color negro *la parte traducible*. Por último, de nuevo de color azul, aparece la *etiqueta de cierre*.

## 5.4 Alinear en base a identificadores con Rainbow

Con el fin de mostrar el funcionamiento de la Alineación basada en identificadores, vamos a parear con Rainbow los segmentos de dos ficheros de *strings* en formato XML ANDROID. Los dos ficheros XML contienen el texto de la versión anterior de la aplicación que nos interesa traducir: uno en inglés y el otro en castellano.

Lo que vamos a lograr mediante la alineación es que los segmentos cuyo valor del atributo sea idéntico se pareen, como por ejemplo los dos que se muestran subrayados en color verde en la figura 9:

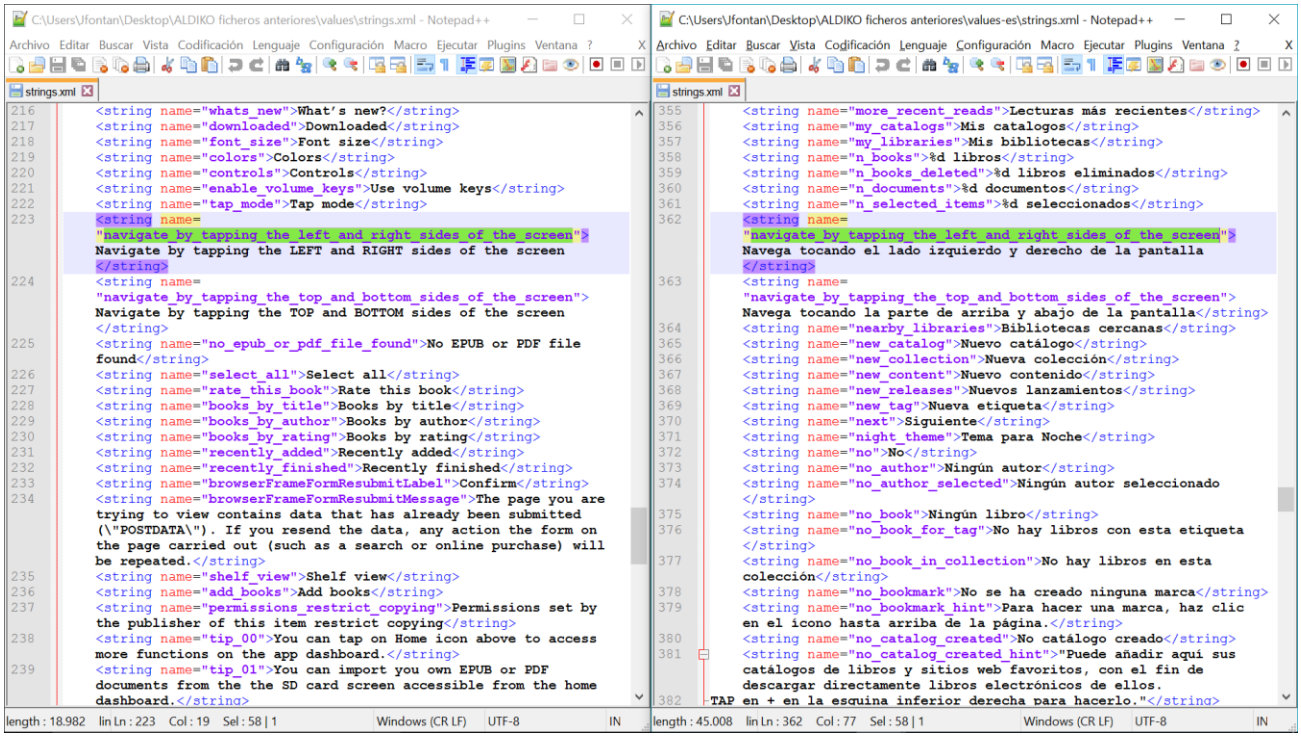



**Figura 9. Dos ficheros XML que alinear.**

Con este fin, ejecutamos Rainbow, añadimos los dos ficheros que vamos a parear (uno en cada *Input List*) y seleccionamos el filtro correcto que queremos que se les aplique. El filtro por defecto es el «okf_xml» pero como sabemos que son XML de Android, le indicamos que use otro filtro más preciso denominado «okf_xml-AndroidStrings». El paso inmediatamente siguiente es establecer los idiomas y la codificación de los archivos en la pestaña *Languages and Encodings*. Hemos comprobado que la codificación de los ficheros es UTF-8 tras abrirlos con Notepad++. Ahora ya está todo listo para ejecutar la alineación y generar el TMX.

Accedemos mediante el menú: *Utilities* > *Id-Based Alignment.* A continuación, se abre una ventana como la que se muestra en la figura 10. En la pestaña *Output* especificamos el directorio en el que deseamos se genere el TMX. Mediante el apartado *Attributes* hemos creado un tercer atributo para indicarle a Rainbow que, al parear los segmentos, indique que en cada segmento que ha sido «Alineado por Jon Fontán Calzada».

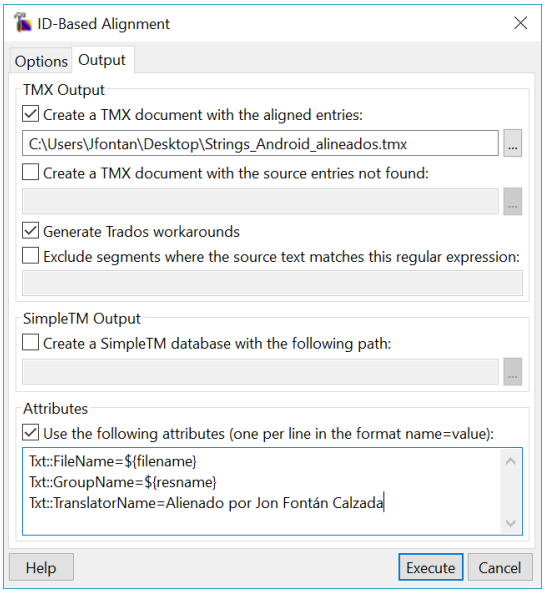



**Figura 10. Configuración de la alineación basando en identificadores.**

Ya solo queda comprobar que la alineación ha sido correcta (ver figura 11). Nótese que solo ha pareado los segmentos que estaban traducidos en las versiones anteriores. Algunos como, por ejemplo, «Cancel» no estaban traducidos y otros contienen errores de puntuación (carecen de signos de interrogación de apertura).

strings.xml | Strings_Android_alineados.tmx

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <tmx version="1.4"><header creationtool="oku_alignment" creationtoolversion
   "en-US" datatype="unknown"></header><body>
3  <tu tuid="41_s0">
4  <prop type="Txt::FileName">strings.xml</prop>
5  <prop type="Txt::GroupName">dont_have_an_account</prop>
6  <prop type="Txt::TranslatorName">Alienado por Jon Fontán Calzada</prop>
7  <tuv xml:lang="en-US"><seg>Don't have an account?</seg></tuv>
8  <tuv xml:lang="es-ES"><seg>No tienes una cuenta?</seg></tuv>
9  </tu>
10 <tu tuid="44_s0">
11 <prop type="Txt::FileName">strings.xml</prop>
12 <prop type="Txt::GroupName">ok</prop>
13 <prop type="Txt::TranslatorName">Alienado por Jon Fontán Calzada</prop>
14 <tuv xml:lang="en-US"><seg>Ok</seg></tuv>
15 <tuv xml:lang="es-ES"><seg>Ok</seg></tuv>
16 </tu>
17 <tu tuid="45_s0">
18 <prop type="Txt::FileName">strings.xml</prop>
19 <prop type="Txt::GroupName">cancel</prop>
20 <prop type="Txt::TranslatorName">Alienado por Jon Fontán Calzada</prop>
21 <tuv xml:lang="en-US"><seg>Cancel</seg></tuv>
22 <tuv xml:lang="es-ES"><seg>Cancel</seg></tuv>
23 </tu>
24 <tu tuid="46_s0">
25 <prop type="Txt::FileName">strings.xml</prop>
26 <prop type="Txt::GroupName">dismiss</prop>
27 <prop type="Txt::TranslatorName">Alienado por Jon Fontán Calzada</prop>
28 <tuv xml:lang="en-US"><seg>Dismiss</seg></tuv>
29 <tuv xml:lang="es-ES"><seg>Anular</seg></tuv>
30 </tu>
```

**Figura 11. TMX resultado de la alineación basada en identificadores.**

## 5.5 Pretraducir a partir de un TMX con Rainbow

Ahora que disponemos del TMX con los segmentos pareados, ha llegado el momento de sacarle partido: vamos a *pretraducir* del inglés al castellano el fichero «strings.xml» de la última versión de la aplicación gracias al TMX que hemos obtenido en el apartado anterior.

> Pretraducció: cerca i substitució automàtica dels segments d'un text que coincideixen totalment o parcial amb els segments emmagatzemats en una MT. El resultat d'aquesta tasca és, doncs, una primera versió de la traducció que pot barrejar propostes de segments correctes, incorrectes i segments sense pretraduir quan no se n'ha trobat cap coincidència a la memòria[14] (Martín-Mor, Piqué y Sánchez-Gijón 2016, 56).

Así, mediante este proceso se va a generar un XLIFF que ya va a ser parcialmente bilingüe. Con todo, va requerir un trabajo posterior de posedición y traducción/revisión mediante una TAO. Gracias a que el archivo pretraducido estará en formato XLIFF, el gestor de proyectos podrá enviarlo a su equipo de traductores sabiendo que estos podrán trabajar cualquier herramienta TAO.

El primer paso para pretraducir es añadir el archivo XML que se desea traducir y aplicarle el filtro para Android XML. La configuración de idiomas y la codificación se ajustan como se ha explicado anteriormente en el apartado 7.2.4 de este documento. Después, se ejecuta un proyecto de traducción de ida y retorno: se hace clic en Utilities > Translation Kit Creation y, por las razones antes expuestas en el apartado 7.1.5, elegimos en formato XLIFF.

Antes de ejecutar para iniciar el proceso de creación del bilingüe, debemos indicarle a Rainbow que debe emplear el TMX para pretraducir. Lo hacemos desde el apartado *Leveraging*, seleccionable en la columna de la izquierda (véase la figura 12). Comprobaremos que hay dos pestañas con dicho nombre: es así porque permite realizar dos tareas de este tipo simultáneamente como, por ejemplo, pretraducir y, además, traducir los segmentos no contenidos en el TMX mediante un motor de traducción automática.

[14] Pretraducción: búsqueda y sustitución automática de los segmentos de un texto que coinciden total o parcialmente con los segmentos almacenados en una MT. El resultado de esta tarea es, por consiguiente, una primera versión de la traducción que puede mezclar propuestas de segmentos correctos, incorrectos y sin traducir cuando no se haya encontrado ninguna coincidencia en la memoria. (Traducción propia).

En este caso solo vamos a pretraducir, así que nos aseguramos de que está marcada la opción *Leverage the text units* y, a continuación, indicamos al programa mediante el menú desplegable ubicado inmediatamente debajo que debe emplear un documento bilingüe (*Bilingual File*). Una vez hecho esto, hacemos clic en el botón *Settings* y seleccionamos el TMX y establecemos su codificación (UTF-8). Ejecutamos y comprobamos que en el directorio de trabajo *pack1* se ha creado una carpeta denominada *work* que contiene el archivo pretraducido en el formato bilingüe. En el directorio de trabajo debe figurar además el fichero manifest.rkm, imprescindible para posprocesar el archivo y volver a montar el fichero, una vez traducido al castellano, en su formato original XML para introducirlo en la aplicación.

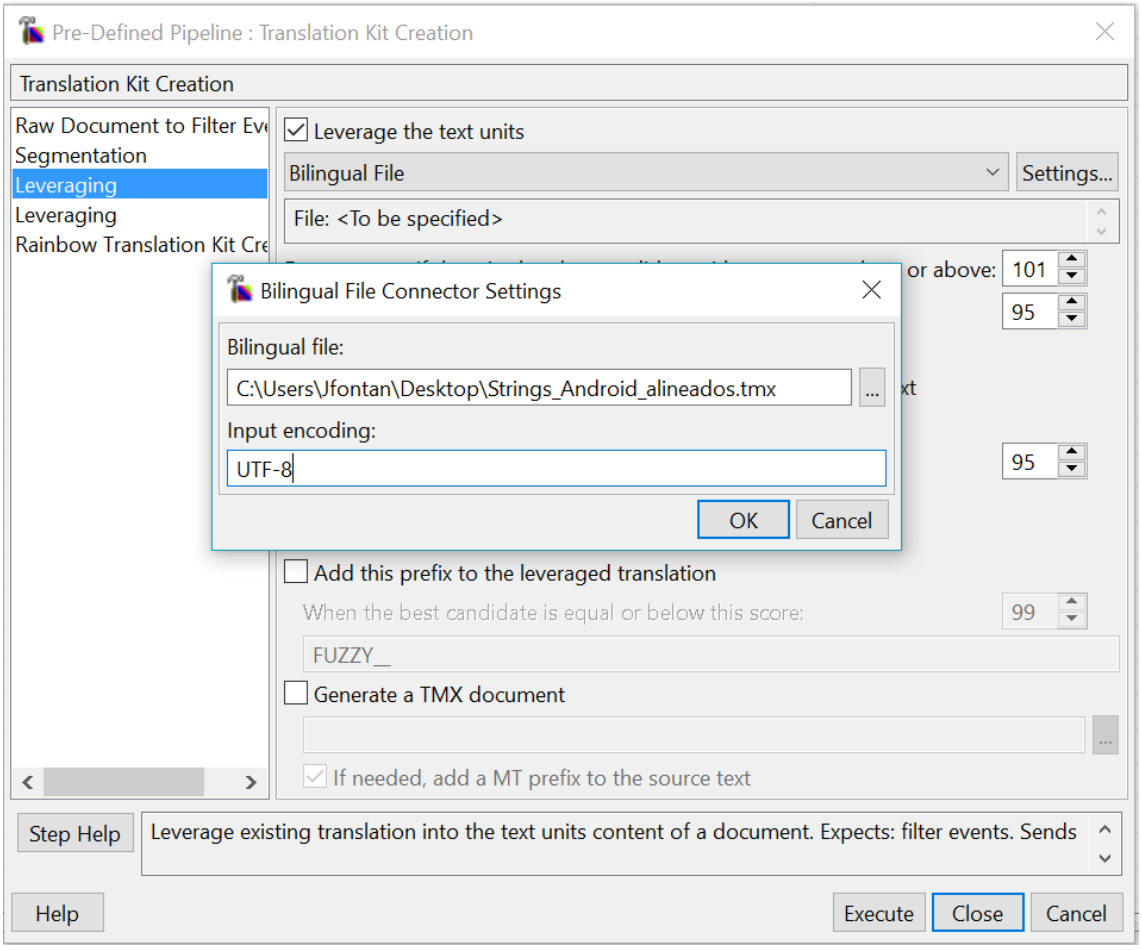



**Figura 12. Importación del TMX para pretraducir.**

En el Anexo 2 se muestra el archivo pretraducido mediante Rainbow con el TMX antes creado. Como se puede comprobar, la práctica totalidad del documento ya está traducida y unos pocos segmentos permanecen en inglés. Gracias a este proceso, el gestor de proyectos de localización ha logrado simultáneamente tres cosas:

1. *Ahorrar tiempo*, al tener una parte muy considerable del texto ya traducida sin necesidad de pasar por una TAO.
2. *Garantizar la coherencia terminológica* de la nueva versión de la aplicación de Android con las anteriores.
3. *Dar libertad a su equipo* para que trabaje con la TAO de su elección, gracias al archivo bilingüe XLIFF.

# 6 La modificación con reglas del filtro para los ficheros de ayuda HTML

## 6.1 Definición y finalidad

Como se ha visto en los apartados anteriores de este documento, con frecuencia resulta interesante o incluso imprescindible adaptar los filtros nativos de las aplicaciones de localización y TAO. Es decir, si se ajusta el filtro al archivo concreto con que se trabaja, se optimizan los resultados de la localización.

## 6.2 Ejemplo real de adaptación de un filtro

En este apartado se mostrará el proceso que seguir para adaptar el filtro nativo de Rainbow para HTML de tal forma que este pase a responder a las características específicas de un Archivo de Ayuda HTML Compilado.

## 6.3 Los ficheros de ayuda HTML Help de Windows

El formato *Microsoft Compiled HTML Help* (Archivo de Ayuda HTML Compilado) «se basa en HTML y otros datos como imágenes y JavaScript.Ayuda HTML 1.0 y fue lanzado en 1997» (Sánchez Pérez 2015, 106). Su extensión de archivo es «.chm». Consiste en un archivo compilado que contiene un índice, una tabla de contenidos y un conjunto de páginas HTML hiperenlazadas a la tabla. Este formato se suele emplear principalmente para la documentación de *software*.

> Microsoft HTML Help is the standard help system for the Windows platform. […] As an information delivery system, HTML Help is suited for a wide range of applications, including training guides, interactive books, and electronic newsletters, as well as help for software applications[15] (Microsoft 2017).

De acuerdo con Microsoft, los ficheros *HTML Help* ofrecen una serie de ventajas destacables sobre aquellos en HTML estándar. Destacan que *HTML Help* permite implementar una tabla que combina los elementos con el índice y, además, se vale de *keywords*. Así, mediante todo esto logran un sistema avanzado de hiperenlaces. Además, al ser un formato compilado, permite comprimir elementos gráficos y otros

[15] El formato Ayuda HTML de Microsoft es el sistema de ayuda estándar para la plataforma Windows. […] Como sistema de envío de información, los archivos de ayuda HTML resultan adecuados para una amplia gama de aplicaciones, incluidos los manuales de instrucciones, los libros interactivos, los boletines informativos electrónicos y las aplicaciones de software. (Traducción propia).

archivos de tal forma que ocupa menos espacio de almacenamiento y resulta más fácil de distribuir por Internet.

La gama de aplicaciones que los emplea es considerablemente amplia porque su uso está muy extendido. Aquí nos vamos a centrar el fichero CHM de 7-Zip (versión 16.04), una aplicación libre que sirve para archivar, comprimir y descomprimir ficheros. Existe una versión posterior de 7-Zip, la 17.00, pero en el momento en que se redacta este documento está en formato beta por lo que se ha preferido trabajar con la última versión estable.

## 6.4 Adaptar el filtro HTML de Rainbow para HTML Help

El fichero de ayuda de 7-Zip se denomina «7-zip.chm». Tras descomprimirlo, obtenemos un directorio que contiene varios archivos y subcarpetas. Entre todos ellos nos centramos en el denominado «7-zip.hhc», que es el que contiene la información del índice que permite navegar por el contenido del fichero de ayuda, es decir, las páginas web.

Dicho fichero en principio funciona como un archivo HTML y en la primera línea de su código se declara que es un documento en dicho formato: «<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">». Sin embargo, como se ha indicado anteriormente, sus características son distintas al HTML estándar. Así, a modo de prueba vamos a generar un paquete de traducción mediante Rainbow aplicándole el filtro por defecto para HTML denominado «okf_html» para comprobar si funciona correctamente o si es necesario adaptarlo.

Como se puede comprobar en el Anexo 3, aunque a nosotros solo nos interesa que detecte como unidades de traducción el texto traducible, en el bilingüe XLIFF también figuran otros valores de atributo como el tipo de imagen («"ImageType"»), los números de imagen («"ImageNumber"») y la ubicación del fichero («"Local"»). Estos elementos no se deben modificar puesto que hacerlo impediría que el fichero de ayuda funcionara correctamente.

Por tanto, aquí se nos plantean dos opciones: podríamos indicarle al equipo de traductores que no modificara dichos valores al trabajar con el bilingüe o, mejor, podríamos adaptar el filtro para que este solo capturase lo que nos interesa. Así,

optamos por crear una nueva versión bilingüe con Rainbow que solo tenga como unidades de traducción el texto traducible.

Es decir, nuestro objetivo es que el filtro solo capture el texto entre etiquetas cuyo valor del atributo «name» sea exclusivamente «"Name"»: no queremos que capture los valores «"ImageType"», «"Local"» e «"ImageNumber"». Así, la estructura que nos interesa sería la siguiente:

<param name="Name"> value="**texto traducible**">

Por supuesto, este paso podría realizarse mediante herramientas TAO como SDL Trados pero, por los motivos antes expuestos, creemos preferible trabajar con Rainbow. Si hacemos clic con el botón de la derecha sobre el filtro en la *Input List 1*, se abrirá una ventana titulada *Input Document Properties* donde, como se ha indicado en los ejemplos anteriores de este documento, podemos crear un nuevo filtro a partir de uno ya existente. En este caso vamos a trabajar con una copia personalizada del filtro okf_html que denominaremos «okf_html@HTML_help».

Una vez creado, se abrirá una nueva ventana en la que se muestran los parámetros del filtro personalizado de tal forma que los podamos modificar. El fichero estará ubicado en la carpeta de trabajo donde se encuentre el archivo 7-zip.hhc, que vamos a traducir.

Como lo que nos interesa modificar en el filtro es la declaración de qué atributos son traducibles, vamos a «elements», «param» y allí buscamos «translatableAttributes». Nos encontramos con la siguiente declaración:

```
assumeWellformed: false
preserve_whitespace: false
elements:
[…]
  param:
      ruleTypes: [INLINE]
      translatableAttributes: [value]
```

A partir de este ejemplo deducimos que solo se traducen aquellos parámetros que son «value». Es decir, salvo que se establezca expresamente que el atributo es traducible, no lo traduce. Por tanto, tenemos que editar esta última parte del código para definir que solo se extraerá como unidad de traducción cuando el valor del atributo «name» sea exclusivamente «"Name"». El aspecto final del filtro será el siguiente:

```
assumeWellformed: false
preserve_whitespace: false
elements:
  param:
    ruleTypes: [INLINE]
    translatableAttributes:
      value:
      - name
      - EQUALS
      - [Name]
```

En lo que respecta al resto del código original, a efectos de nuestro ejemplo resulta prescindible. Es decir, podemos optar por mantenerlo o simplemente eliminarlo. Así, ya lo tenemos todo listo para aplicar el filtro editado y generar un nuevo fichero bilingüe.

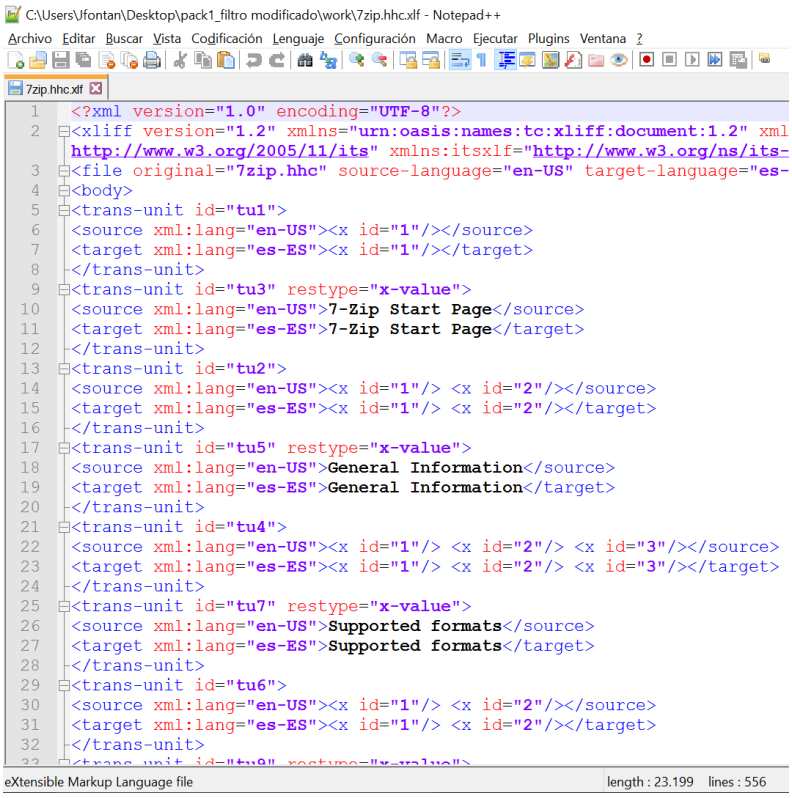



**Figura 13. El resultado de aplicar el filtro personalizado para HTML.**

Como podemos comprobar en la fig. 13, hemos logrado que solo extraiga como unidades de traducción el texto traducible y sus etiquetas. Así, gracias a este filtro personalizado podremos ahorrar tiempo y hacer que nuestro equipo de traductores trabaje de forma más eficiente y segura ya que las partes no traducibles han dejado de estar presentes en el bilingüe y, por tanto, nadie las puede alterar por error.

# 7 Conclusiones

En el apartado introductorio planteábamos la hipótesis inicial de que conocer las utilidades y potencialidades de Okapi Rainbow mejora y facilita la labor de los gestores de proyectos de localización. Mediante los tres experimentos detallados en este documento hemos demostrado empíricamente que cumple dicho objetivo. Para enfatizar el enfoque práctico y subrayar la versatilidad de Rainbow, los experimentos se realizan sobre tres aplicaciones reales distintas que contienen los ficheros traducibles en formatos diferentes.

Mediante el primer experimento detallamos cómo Rainbow permite crear paquetes de traducción en el formato estándar XLIFF y después reconvertirlos a su formato original. Además, abordamos la pertinencia de pseudotraducir un fichero para realizar comprobaciones. En este experimento trabajamos con una aplicación en JAVA cuyo contenido traducible está en archivos PROPERTIES.

Gracias al segundo experimento demostramos que es posible localizar de forma más rápida y eficiente mediante la alineación basada en identificadores. Ilustramos cómo los ficheros TMX generados mediante dicha alineación se pueden emplear posteriormente para crear un XLIFF pretraducido. Además, exponemos cómo este proceso mejora la coherencia terminológica porque el localizador parte de una pretraducción basada en las versiones anteriores de un producto. En este experimento hemos trabajado con una aplicación cuyo contenido localizable está en ficheros XML con especificaciones para Android.

En tercer lugar hemos realizado un experimento con un carácter más técnico que los dos anteriores: modificar manualmente el filtro de Rainbow para HTML. Si bien es cierto que el primer experimento antes expuesto requería adaptar el filtro para archivos PROPERTIES mediante los parámetros para que capturara óptimamente el contenido traducible, este tercer experimento va más allá porque exige trabajar directamente sobre el código. Así, demostramos que es posible aprender a adaptar filtros gracias a la documentación técnica que detalla la estructura y las características de los ficheros de ayuda HTML Help para Windows. Mediante la adaptación del filtro, especificamos qué se traduce y qué no en un fichero de ayuda HTML Help de Windows.

Desde un punto de vista personal me gustaría destacar que, gracias a este Trabajo de fin de grado, he afianzado y ampliado los conocimientos adquiridos en las asignaturas de corte más técnico del Grado en traducción e interpretación como Informática básica, Recursos tecnológicos para la traducción y, especialmente, Localización. Además, he podido contextualizar los experimentos aquí abordados en el mercado real de la traducción gracias a la experiencia adquirida mediante las Prácticas de traducción y a lo aprendido en Gestión terminológica y de proyectos, además de las distintas asignaturas de traducción directa e inversa cursadas a lo largo de la carrera. Por último, me gustaría destacar también a este respecto que la competencia adquirida en las cuatro asignaturas de redacción en lengua castellana ha mejorado sin duda la claridad con que se exponen las ideas en este documento.

Somos conscientes de que el abanico de utilidades y herramientas que ofrece Rainbow es más amplio de lo que aquí abordamos pero este documento está sujeto a las especificaciones de los Proyectos de fin de grado. Por consiguiente, no hemos podido abordar aquí algunos aspectos interesantes como la creación de «Pipelines», que facilitan automatizar conjuntos de utilidades, y la utilidad «Buscar y reemplazar» («Search and Replace»), que es especialmente interesante para editar o construir filtros. Así, su análisis queda pendiente para un futuro proyecto que amplíe el análisis aquí emprendido.

*El «software libre» es una cuestión de libertad, no de precio.*
*—The Free Software Foundation.*

# 8 Bibliografía

*Desarrolladores de Android*. 2017a. “Localizing with Resources”. https://developer.android.com/guide/topics/resources/localization.html

—2017b. “Recursos de strings”. https://developer.android.com/guide/topics/resources/string-resource.html

*Globalization and Localization Association (GALA)*. 2011. “TMX 1.4b”. https://www.gala-global.org/tmx-14b

— 2017. “Industry Standards”. https://www.gala-global.org/resources/industry-standards

Gómez, Josu. 2001. “Una guía al TMX”. *Tradumàtica.* Ejemplar nº0 (digital). http://www.fti.uab.es/tradumatica/revista/num0/articles/jgomez/art.htm

Jiménez-Crespo, Miguel A. 2013. *Translation and Web Localization*. Londres y Nueva York: Routledge.

*JFrog*. 2017a. “Rainbow M32”. https://bintray.com/okapi/Distribution/Okapi_Applications/M32

— 2017b. “Rainbow M33”. https://bintray.com/okapi/Distribution/Okapi_Applications/M33

*Localization Industry Standards Association (LISA)*. 2 de enero del 2011. “Glossary”. Accesible mediante: *Internet Archive: WayBack Machine*. http://web.archive.org/web/20110102003821/http://www.lisa.org/Glossary.108.0.html

Martín-Mor, Adrià, Ramón Piqué y Pilar Sánchez-Gijón. 2016. *Tradumàtica: Tecnologies de la traducció*. Barcelona: Eumo.

*Microsoft*. 2017. “Microsoft HTML Help 1.4”. https://msdn.microsoft.com/en-us/library/windows/desktop/ms670169(v=vs.85).aspx

*Okapi Framework*. 2016a. “Open Standards”. http://okapiframework.org/wiki/index.php?title=Open_Standards

—2016b. “Properties Filters”.
http://okapiframework.org/wiki/index.php?title=Properties_Filter

—2016c. “Rainbow”.
http://okapiframework.org/wiki/index.php?title=Rainbow

—2017a. “Okapi Framework”.
http://okapiframework.org/wiki/index.php?title=Main_Page

—2017b. “Rainbow - Utilities”.
http://okapiframework.org/wiki/index.php?title=Rainbow_-_Utilities

*Okapi Framework Tutorials*. 2017. “Using ID-Based Alignment”.
http://okapiframework.org/help/tutorials/tutorial_01/index.html

*Organization for the Advancement of Structured Information Standards (OASIS).*
2008a. “OASIS XML Localisation Interchange File Format TC: FAQ”.
https://www.oasis-open.org/committees/xliff/faq.php

— 2008b. “XLIFF Version 1.2”.
http://docs.oasis-open.org/xliff/v1.2/os/xliff-core.html#SectionIntroduction

—2014. “XLIFF Version 2.0: OASIS Standard”.
http://docs.oasis-open.org/xliff/xliff-core/v2.0/os/xliff-core-v2.0-os.html

—2016. “XLIFF Version 2.1: Public Review Draft”.
http://docs.oasis-open.org/xliff/xliff-core/v2.1/csprd01/xliff-core-v2.1-csprd01.html

Pegenaute, Luis. 1999. “Pseudotraducciones y engaños”. *El Trujamán.* Septiembre de 1999.
http://cvc.cervantes.es/trujaman/anteriores/septiembre_99/20091999.htm

Roturier, Johann. 2015. *Localizing Apps: A practical guide for translators and translation students*. Londres y Nueva York: Routledge.

Sánchez Pérez, Baldomero. 2015. “Ayuda de comandos en línea”. En *Cuaderno práctico de Windows*. *Sistemas Operativos Monopuestos. Ciclos Formativos de Informática*. Ediciones Lulu. 105-106.

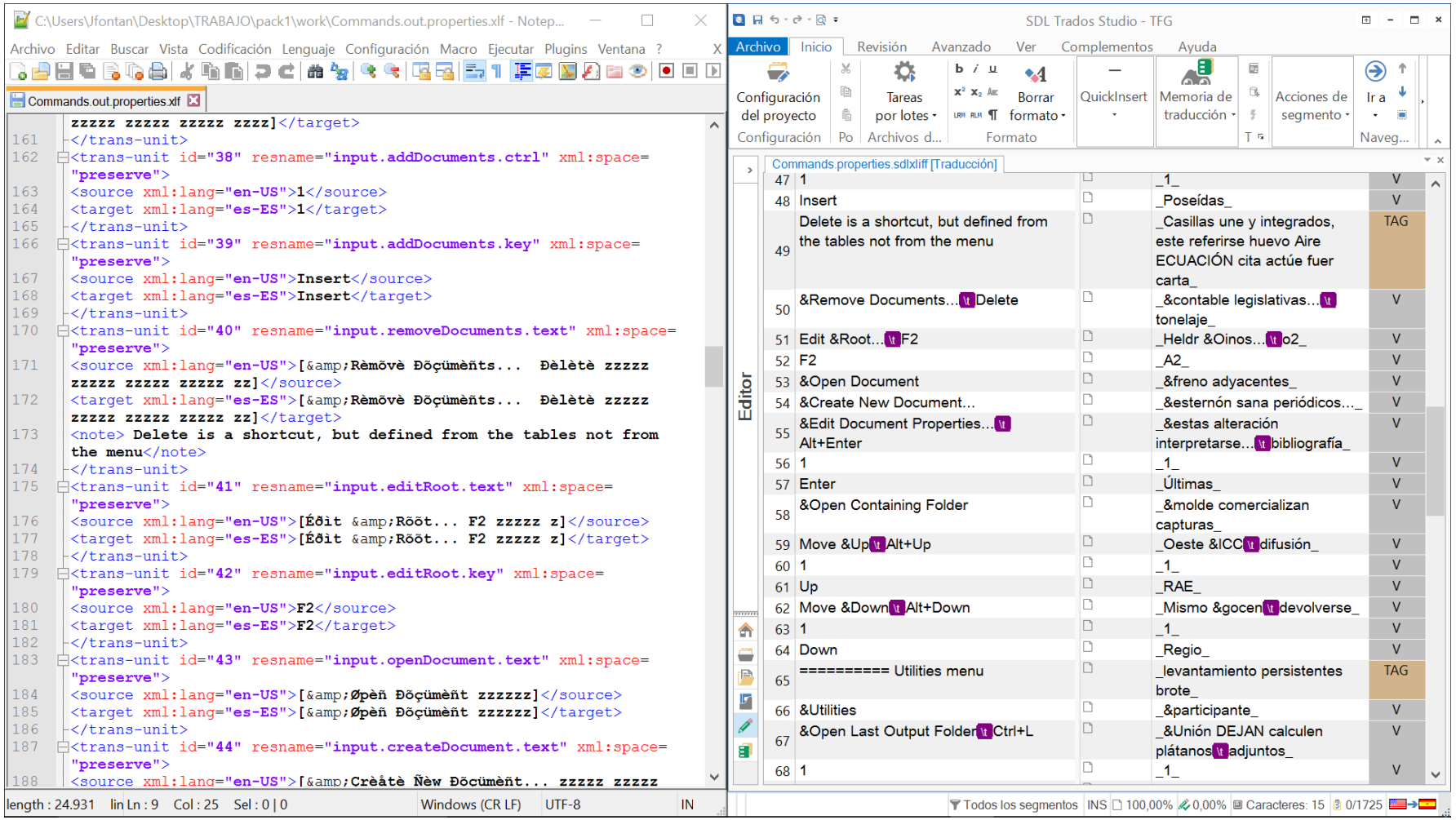
*SDL Trados Studio.* 2016. “About Pseudo-translation”. http://producthelp.sdl.com/SDL_Trados_Studio_2015/client_en/About_Pseudo-translation.htm

Schnabel, Bryan, JoAnn T. Hackos y Rodolfo M. Raya. 2015. *A Practical Guide to XLIFF 2.0.* Dublin. Trinity College.

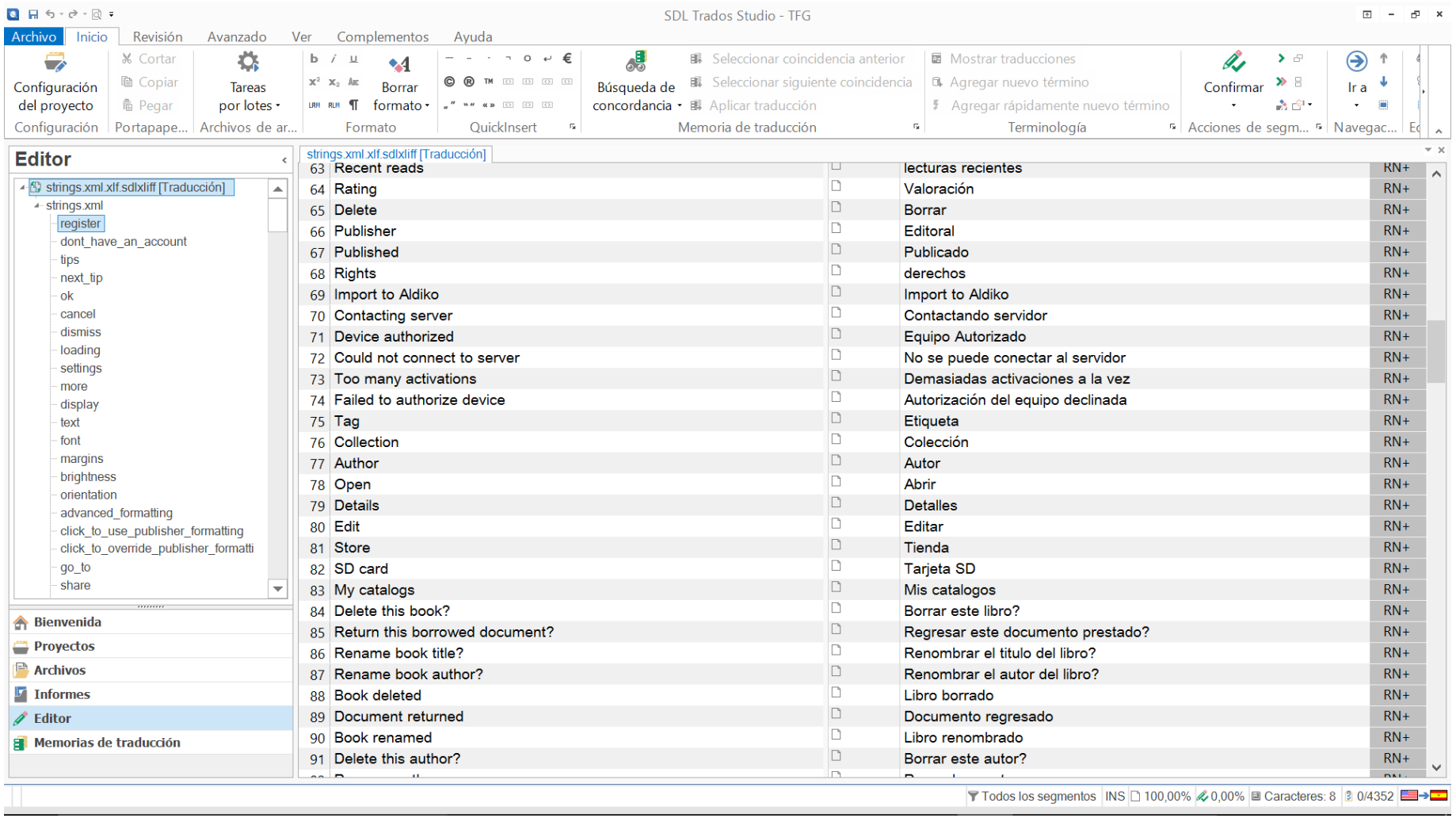
Torres Ripa, Javier. 2013. *Manual de estilo Chicago-Deusto: Edición adaptada al español* [orig. The Chicago Manual of Style, 16th edition]. Bilbao: Universidad de Deusto.

*W3Schools*. 2017. “HTML ISO-8859-1 Reference”. https://www.w3schools.com/charsets/ref_html_8859.asp

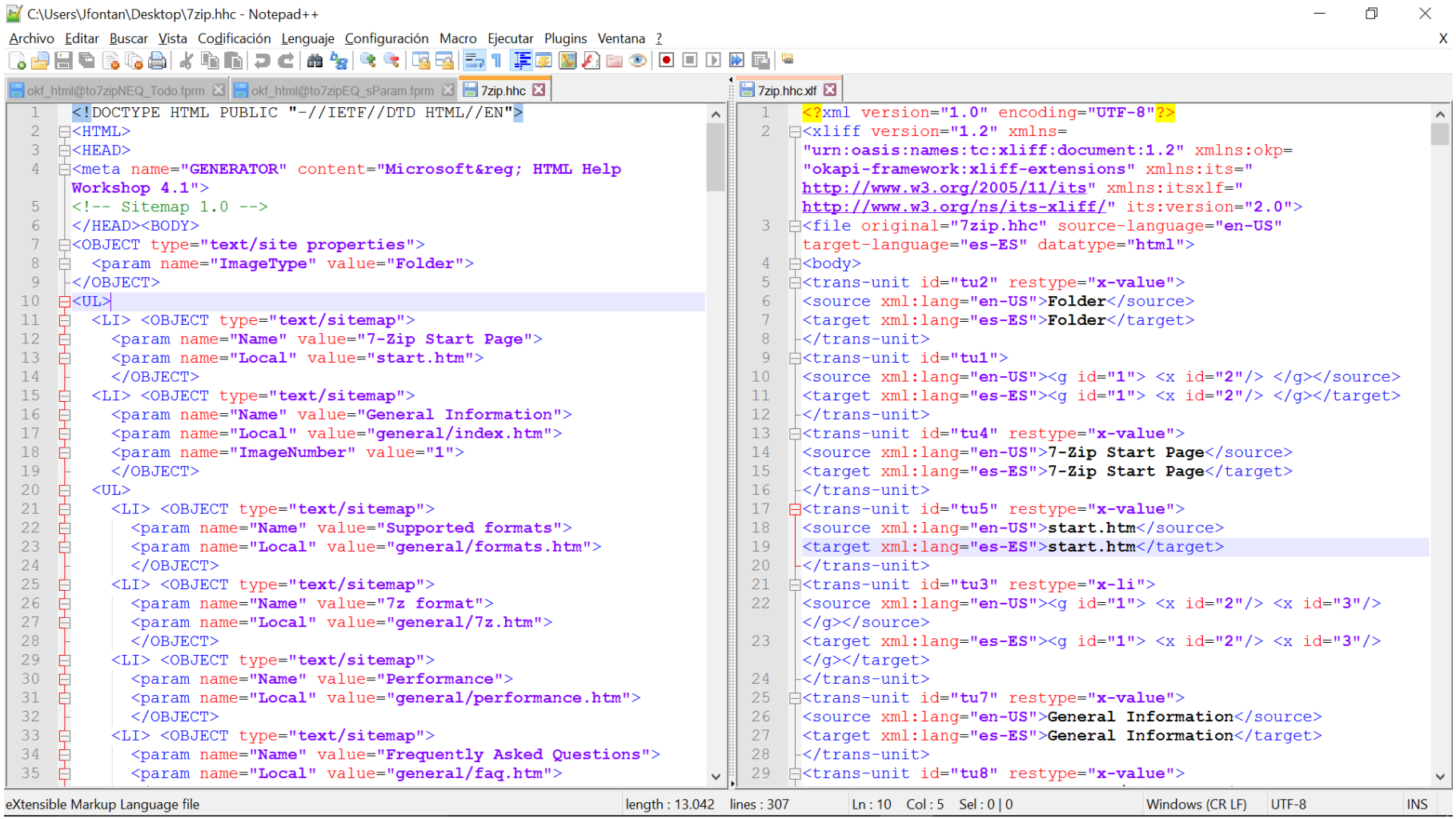
# 9 Anexos




Anexo 1. Comparativa de una pseudotraducción realizada con Rainbow con otra de SDL Trados 2015 con los parámetros por defecto.




**Anexo 2. Una pretraducción realizada con Rainbow abierta en SDL Trados para su revisión.**




**Anexo 3. A la izquierda se muestra el archivo original y, a la derecha, su versión bilingüe resultado de aplicarle el filtro por defecto para HTML.**