

Desarrollo de una nueva versión de gtranslator: reactivando una comunidad de Software Libre

Pablo José Sangiao Roca
Master on Free Software Caixanova
psanxiao@gmail.com

2 de diciembre de 2008

©2008 Pablo José Sangiao Roca.
Este documento se distribuye bajo la licencia Creative Commons 2.5 con reconocimiento, compartir igual. Más detalles sobre esta licencia en:
<http://creativecommons.org/licenses/by-nc-sa/2.5/es/>

Resumen

Un apartado muy importante para la difusión del Software Libre es, sin duda, la localización de los programas. Además de permitir que los usuarios puedan hacer uso de las aplicaciones adaptadas a su lengua y costumbres, facilita también que personas con perfiles no técnicos, puedan colaborar en las comunidades de Software Libre mediante la realización de traducciones. Para facilitar esta labor existen algunas aplicaciones como son Kbabel dentro del proyecto KDE y gtranslator en el proyecto GNOME. Esta última se encontraba en un estado de desarrollo muy pobre, abandonada incluso por los traductores de GNOME y con una comunidad a su alrededor prácticamente inexistente. En este artículo se mostrará el trabajo que se realizó para desarrollar una nueva versión de esta herramienta, que la coloque en un punto de referencia para la comunidad de traductores. Además se indicarán las decisiones y propuestas realizadas para intentar dinamizar la comunidad, así como un análisis del estado de la misma a través de información obtenida a partir del sistema de control de versiones y listas de correo del programa.

1. Introducción

Los proyectos de Software Libre que tienen éxito, suelen tener una característica común: una gran comunidad detrás. Esto permite que, a pesar de que la mayoría son comenzados y mantenidos por personas en su tiempo libre, crezcan rápidamente gracias a las pequeñas contribuciones de mucha gente. Además tener una comunidad grande implica que se pueda dar el fenómeno del relevo generacional. Mucha gente que inicia un proyecto o colabora en él durante un tiempo, finalmente por diferentes motivos termina por desvincularse del mismo, o por lo menos de participar activamente. Es necesario en ese momento

que nuevas personas se hagan cargo del proyecto, produciéndose así un relevo generacional que garantice la continuidad del mismo en el tiempo.

En el caso que nos ocupa, gtranslator estaba en uno de esos momentos de cambio, donde la gente que se ocupaba de su mantenimiento no podía seguir adelante con el mismo por diferentes motivos, y tampoco existía en ese momento una comunidad detrás lo suficientemente grande y madura como para asumir el cambio. Debido a esto, las últimas versiones apenas aportaban novedades y poco a poco el proyecto fue perdiendo atractivo para los usuarios. Por contra, herramientas similares como Kbabel, perteneciente al proyecto KDE, iban ganando funcionalidades y convirtiéndose en herramientas completas de ayuda a los traductores en su labor.

En este punto se decide entrar a forma parte del proyecto, participando activamente en el desarrollo de una nueva versión que ofrezca las mismas funcionalidades que herramientas similares, y que además permita a nuevos desarrolladores incorporarse al proyecto, para lo cual se piensa en un código más moderno y modular y sobre todo fácil de entender y de poder modificar.

A lo largo de los siguientes apartados se mostrará:

- El proceso de entrada en la comunidad siguiendo el *onion model*, hasta llegar a forma parte de la misma como desarrollador oficial e incluso hacerse cargo del mantenimiento.
- Un análisis del proyecto a lo largo del tiempo a través de los datos obtenidos del sistema de control de versiones y listas de correo, mediante el uso de herramientas especializadas.¹
- El proceso de desarrollo de la nueva versión, incluyendo las tecnologías empleadas y las nuevas funcionalidades añadidas.

2. Introducción en la comunidad

La entrada en la comunidad alrededor de gtranslator se hizo siguiendo el *onion model*. Este modelo es característico de los proyectos de software libre y se cumple dentro de GNOME como se puede ver detalladamente en [3]. De manera general, este modelo asume que el primer contacto con un proyecto de Software Libre se realiza a través de su página web, o como usuario del programa. A partir de aquí se puede utilizar la lista de correo del proyecto para consultar dudas e interactuar con otros miembros de la comunidad. Cuando se usa el programa se detectan errores, el siguiente paso es ayudar a la comunidad a resolverlos mediante el envío de informes de fallo. Para esto la mayoría de proyectos suelen usar una herramienta de seguimiento de errores, en el caso de GNOME se emplea *Bugzilla*. Cuando se tienen habilidades de programación se puede usar esta misma herramienta para enviar parches con soluciones para esos errores y así ganar méritos para ser incluido entre los desarrolladores del proyecto. Esto permite tener acceso directamente al sistema de control de versiones para subir las modificaciones directamente en lugar de enviar los parches.

La secuencia de entrada en la comunidad de gtranslator se hizo siguiendo este modelo. El primer paso fue hablar con el mantenedor del programa en ese momento y comunicarle la intención de colaborar en el desarrollo de una nueva

¹<http://forge.morfeo-project.org/projects/libresoft-tools/>

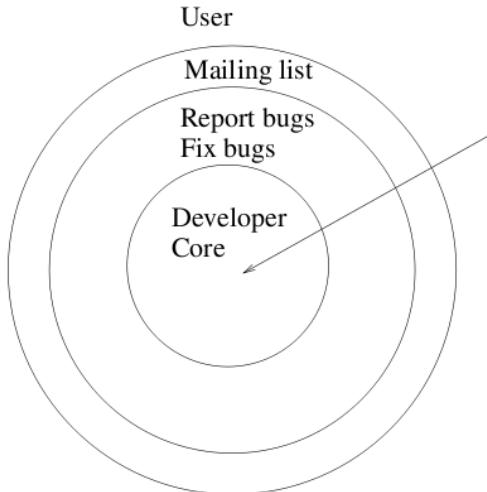


Figura 1: The Onion Model

versión. Viendo que en realidad, su labor en esos momentos era más bien testimonial, para evitar la muerte total del proyecto, ya que no disponía de tiempo para dedicarle se vió la posibilidad de dar ese relevo generacional comentado anteriormente. Para ello se seguieron una serie de pasos. Los proyectos de Software libre suelen funcionar por meritocracia, es decir, es necesario contribuir de manera activa al proyecto antes de poder tener voz a la hora de tomar decisiones.

El primer paso consistió en enviar un mensaje a la lista de correo del proyecto. Este primer mensaje sirvió como presentación y como declaración de intenciones, comentando algunas ideas para el desarrollo de la nueva versión. Al mismo tiempo se comenzó a trabajar en la versión estable del proyecto resolviendo algunos de los *bugs* más importantes enviando las soluciones al *Bugzilla*. Esto permitió generar la suficiente confianza por parte del mantenedor y los méritos para solicitar una cuenta en el sistema de control de versiones como desarrolladores oficiales del proyecto.

Posteriormente se preparó una nueva *release* con las soluciones aportadas sobre la versión estable y se comenzó a desarrollar la nueva versión. En Octubre de 2007 se asumió oficialmente el mantenimiento del programa en GNOME.

3. Análisis del proyecto

En este apartado se pretende analizar la evolución del proyecto y de su comunidad en el tiempo, desde su comienzo en 1999 hasta hoy. Este análisis se centrará en el sistema de control de versiones y las listas de correo del proyecto. En el Software Libre, debido al uso de este tipo de herramientas para facilitar el desarrollo en comunidad, es posible recavar mucha información y hacer minería de datos para estudiar diferentes comportamientos en el proyecto.

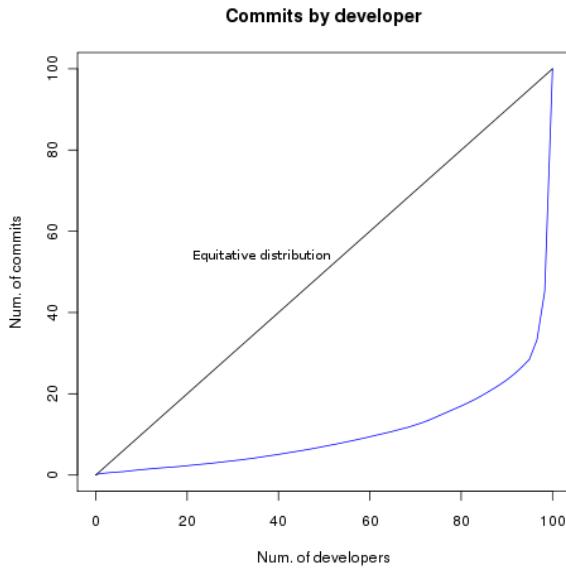


Figura 2: Curva de Lorentz: gtranslator

En el caso que nos ocupa, se analizó el sistema de control de versiones, en este caso *Subversion* y las listas de correo. Para ello se utilizaron herramientas específicas: *cvsanaly* y *mailingliststats*, ambas desarrolladas por el grupo *Libresoft*.²

Los resultados se analizarán prestando atención a la evolución del proyecto en los dos últimos años, tratando de ver si hay signos que inviten a ser optimistas en la recuperación de la comunidad.

3.1. Análisis del Subversion

El sistema de control de versiones de gtranslator, *Subversion*, está alojado en el de GNOME y disponible en la dirección: *svn://svn.gnome.org/svn/gtranslator*. Este análisis se realizó con *cvsanaly* sobre la rama de desarrollo(*trunk*).

El primer dato interesante que se puede obtener es el número de desarrolladores que aportan al proyecto (*committers*) y el número de aportaciones de cada uno de ellos (*commits*). Con estos datos podemos hallar la curva de *Lorentz* para comprobar si la distribución del trabajo es homogénea. El resultado se muestra en la figura 2.

En los proyectos de Software Libre es habitual que se cumpla el principio de Pareto o regla del 80:20. En la mayoría de ellos el 80 % del trabajo es realizado por el 20 % de los contribuidores. En este caso los datos confirman este principio. En la figura 2 se muestra la curva de *Lorentz*. La curva que muestra la distribución de las contribuciones al proyecto se aleja mucho de la distribución equitativa confirmando que el 80 % de las contribuciones (*commits*) son realizadas por el 20 % de los contribuidores (*committers*).

²<http://www.libresoft.es>

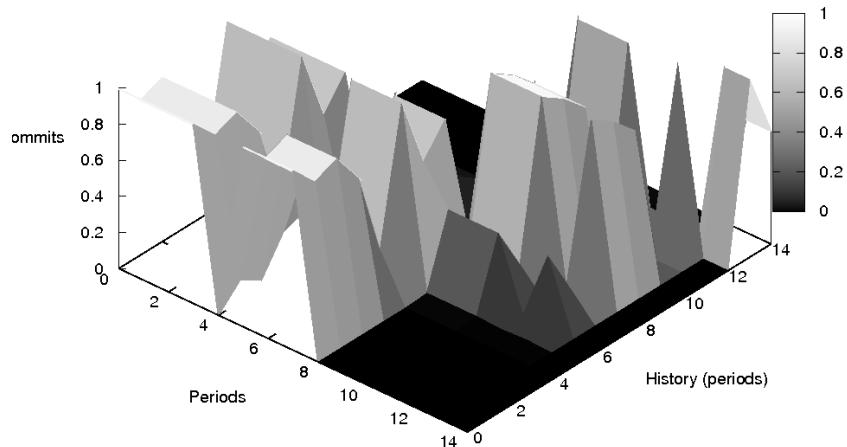


Figura 3: generational change-over

Con *cvsanaly* es posible ver si ha habido relevo generacional en algún momento durante la vida del proyecto. Utilizando el plugin de generaciones, es posible obtener un gráfico como el de la figura 3, que muestra la evolución en el tiempo de los grupos de contribuidores (*committers*).

En la figura se pueden ver tres etapas bastante claras en la vida del proyecto, o dicho de otra manera parece que ha habido dos relevos generacionales. Las zonas más oscuras y sin picos de la figura implican que no ha habido actividad por parte de ese grupo de *committers*. Después de la primera de estas zonas vemos unos picos, que indican que comienza la actividad de un nuevo grupo de contribuidores. Estos primeros picos son más pequeños que los siguientes, además coinciden con otros más grandes del grupo anterior, lo que quiere decir que el relevo se llevó a cabo de forma gradual. El nuevo grupo se incorpora al proyecto contribuyendo con nuevos *commits*, pero es el antiguo grupo el que sigue realizando la mayor parte del trabajo. A medida que avanzamos en el gráfico los picos del primer grupo descienden hasta desaparecer, a la vez que los del segundo grupo aumentan considerablemente, esto es signo claro de que se ha producido el relevo generacional.

El segundo relevo es el que se produce con motivo del trabajo comentado en este artículo. Es un relevo mucho más brusco y no tan progresivo como en el caso anterior. Esto viene motivado porque apenas se estaba realizando trabajo anteriormente. El grupo anterior había dejado ya la actividad de desarrollo y solo había *commits* de mantenimiento. Se puede ver en la figura que el nuevo grupo que se incorpora realiza un trabajo constante, sin picos, esto es el periodo de trabajo para el desarrollo de la nueva versión.

En resumen, el primero de los relevos es lo que se desea para un proyecto de

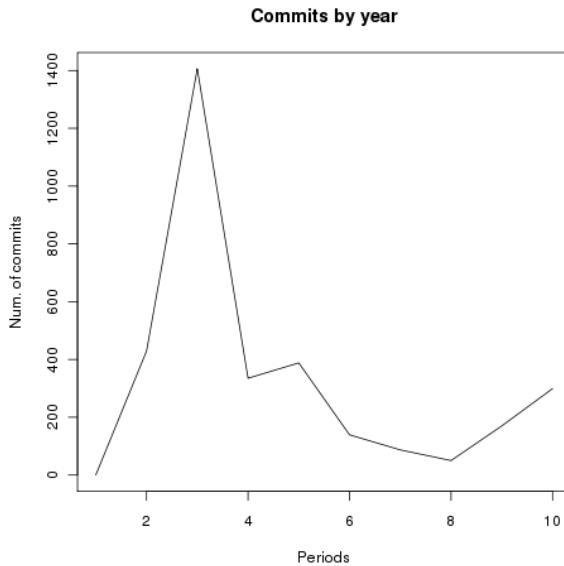


Figura 4: número de commits por período

Software Libre, los nuevos desarrolladores se incorporan al proyecto de manera progresiva, tomando contacto poco a poco con el proyecto. Poco a poco estas aportaciones se van haciendo más importantes hasta que alcanzan la madurez necesaria para hacerse cargo del proyecto. En este momento se realiza el relevo de manera natural, sin que el proyecto se resienta por ello. En el segundo caso el relevo viene motivado porque el proyecto está en un momento de crisis, donde no hay nadie en la comunidad que pueda o quiera hacerse cargo del mismo, hasta que aparece algún desarrollador dispuesto a hacerse cargo o el proyecto queda en el olvido. Este tipo de relevo no es el más deseable ya que implica que los nuevos desarrolladores deben hacerse cargo de todo de manera brusca.

Para ver cual fue el volumen de trabajo a lo largo de la vida del proyecto y comprobar su estado actual, se divide el proyecto en períodos de una año y se obtiene el número total de *commits* que se han hecho en cada uno de esos períodos. La gráfica que sale se puede ver en la figura 4.

Se puede ver que en los dos períodos iniciales el número de *commits* es muy alto, esto es normal, ya que son los inicios del proyecto y todo está por hacer. A partir de ahí en los dos períodos siguientes el número de *commits* tiende a estabilizarse, pero a partir del quinto comienza a desdencere de manera considerable hasta llegar al séptimo, donde el número de commits durante ese período fue tan sólo de 50. Este es el punto que coincide con el segundo relevo generacional comentado anteriormente. A partir de ese momento y hasta la actualidad se puede observar una línea ascendente en el número de *commits*.

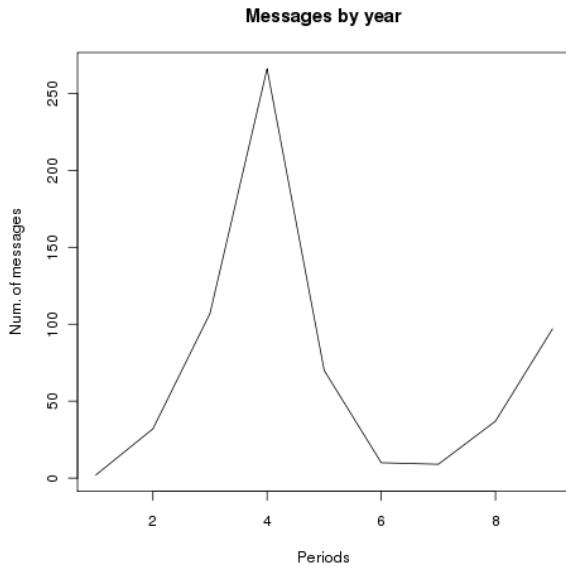


Figura 5: número de mensajes por período

3.2. Análisis de las listas de correo

En el caso del gtranslator existe una única lista de correo: *gtranslator-list@gnome.org*. Para realizar el análisis se utilizó la herramienta *mailingliststats*. Con ella es posible analizar todo el tráfico de una lista de correo y obtener una base de datos sobre la que hacer consultas, para obtener información sobre el proyecto y los miembros de la comunidad.

Al igual que en el caso del *Subversion* un primer análisis consiste en dividir el análisis en períodos de un año y obtener el número de mensajes por año en la lista de correo.

En la figura 6 se puede ver la gráfica resultante. En primer lugar se puede observar que la evolución es muy similar al caso del *Subversion*, lo que era de esperar, ya que ambas herramientas están muy relacionadas en los proyectos de Software Libre. En los últimos períodos, los que coinciden con el trabajo comentado en este artículo, el número de mensajes a la lista está creciendo considerablemente, además se ve una progresión en alza. Como se comentaba en la introducción, las listas de correo son los puntos de entrada de los nuevos miembros al proyecto. Dado que en estos momentos se está intentando reactivar la comunidad es necesario analizar si este incremento de mensajes en la lista implica también un aumento de participación, es decir, tratar de averiguar cuantos nuevos miembros hay participando en la lista.

Para ello tomamos como referencia la fecha en que el anterior mantenedor nos cedió el mantenimiento del programa oficialmente, esto es, Octubre de 2007 que coincide con el último período de un año analizado, donde el número total de mensajes fue de 97. Haciendo una consulta a la base de datos generada por *mailinliststats* el número de personas diferentes que han enviado algún correo a

la lista en el último período es de 25. Al crear esta base de datos, *mailinglists-tats* tiene un campo *people_ID* autoincremental y que define a cada una de las personas que han enviado un correo a la lista. Con este campo es fácil ver que de esas 25 personas que habían enviado al menos un mensaje durante el último período, 17 de ellas son nuevas. Este dato es importante, ya que aplicando el *onion model*, son 17 nuevos miembros de la comunidad en potencia.

3.3. Análisis con sloccount

Esta es una herramienta que se utiliza para medir las líneas físicas de código fuente (SLOC) de un programa. A continuación se muestran los resultados al medir la última versión estable de gtranslator, la versión 1.1.8 y la nueva versión que se desarollo, la 2.0.

Análisis de la versión 1.1.8:

```
SLOC Directory SLOC-by-Language (Sorted)
11212  src_top_dir    ansic=11212
1568   src_semerkent  ansic=1568
923    top_dir        sh=923
844    data           sh=844
0      autom4te.cache (none)
0      doc            (none)
0      help           (none)
0      man            (none)
0      po             (none)
0      src_pixmaps   (none)

Totals grouped by language (dominant language first):
ansic:      12780 (87.85%)
sh:         1767 (12.15%)

Total Physical Source Lines of Code (SLOC) = 14,547
Development Effort Estimate, Person-Years (Person-Months) = 3.33 (39.91)
(Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Schedule Estimate, Years (Months) = 0.85 (10.15)
(Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers (Effort/Schedule) = 3.93
Total Estimated Cost to Develop = $ 449,318
(average salary = $56,286/year, overhead = 2.40).
```

Análisis de la versión 2.0:

```
SLOC Directory SLOC-by-Language (Sorted)
11401  src_top_dir    ansic=11372,sh=29
8993   plugins        ansic=8962,sh=31
3816   src_dialogs    ansic=3816
3327   src_toolbareditor ansic=3304,sh=23
1954   src_plugin-system ansic=1893,sh=61
1754   src_translation-memory ansic=1754
26     top_dir        sh=26
```

```

0      data          (none)
0      doc           (none)
0      help          (none)
0      m4            (none)
0      man           (none)
0      po             (none)

```

Totals grouped by language (dominant language first):

```

ansic:      31101 (99.46%)
sh:         170 (0.54%)

```

```

Total Physical Source Lines of Code (SLOC) = 31,271
Development Effort Estimate, Person-Years (Person-Months) = 7.43 (89.15)
(Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Schedule Estimate, Years (Months) = 1.15 (13.77)
(Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers (Effort/Schedule) = 6.47
Total Estimated Cost to Develop = $ 1,003,553
(average salary = $56,286/year, overhead = 2.40).

```

Como se puede ver, la segunda versión dobla en líneas de código a la anterior. Además del uso de nuevas tecnologías, como se verá en el siguiente apartado, la mayor parte de este aumento está debido a la inclusión de nuevas funcionalidades. Entre ellas está la creación de un sistema de *plugins* que como se puede ver en el análisis de *sloccount* supone un porcentaje muy elevado de las nuevas líneas de código.

3.4. Desarrollo de la nueva versión

Con el desarrollo de la nueva versión de gtranslator se trató de solucionar los principales problemas de la anterior: código *adhoc*, muy desestructurado y la falta de funcionalidades para ayudar a los traductores. Además la primera versión no usaba muchas de las tecnologías que se ofrecen actualmente en GNO-ME. Incluso no utilizaba la biblioteca que ofrece *gettext*, pensada especialmente para el desarrollo de herramientas de este tipo.

Durante todo el proceso se trató de compartir ideas con la comunidad a través de la lista de correo, del wiki³ y del canal de irc. Se tuvieron en cuenta las opiniones de los traductores ya que serán los futuros usuarios de la aplicación.

Fruto de esta comunicación se diseño una nueva interfaz, partiendo de una propuesta enviada a la lista de correo, que se fue perfeccionando con los comentarios de los usuarios. Para la implementación de la interfaz se utilizó *GtkUIManager* para el diseño de los menús y de la barra de herramientas. El diseño de los diálogos se realizó mediante el diseñador de interfaces *Glade* y se utilizó la biblioteca *libglade* para integrar todos los *widgets* en el código.

Para conseguir un código más ordenado y modular, que permita a los desarrolladores que se interesen por el proyecto, integrarse y empezar a colaborar de forma más sencilla, se portó todo el código a *GObject*. Esta biblioteca de GNO-ME permite el uso de orientación a objetos en C. Para la implemetación de las clases todos los atributos se definieron como privados, accediéndose a ellos sólo

³<http://live.gnome.org/gtranslator>

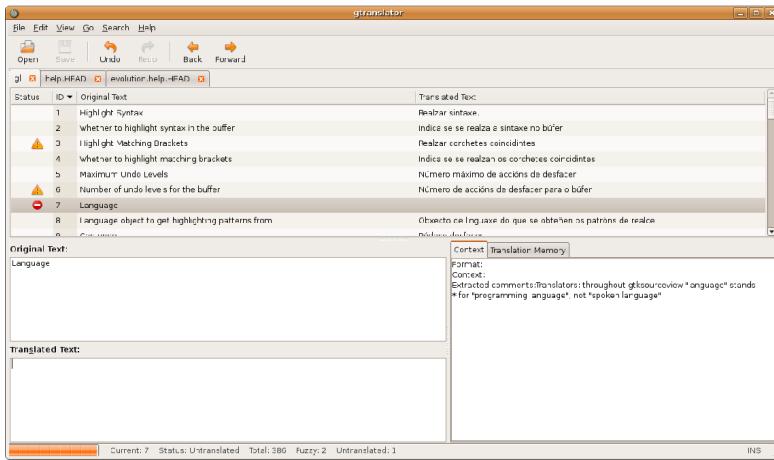


Figura 6: Interfaz principal de la nueva versión

a través de métodos *set* y *get*. Con esto se garantiza la encapsulación, y facilita que se puedan hacer cambios en las clases sin que afecten al resto del código del programa.

El otro problema que tiene la primera versión, es que sólo permite la edición de ficheros de traducción, pero no ofrece ninguna funcionalidad al traductor para facilitar su trabajo, al contrario que otras herramientas similares. Para la nueva versión se desarrollaron una serie de nuevas funcionalidades:

- Gestión de diferentes perfiles con la información del traductor y del idioma. Esta información se utiliza para rellenar la cabecera de los ficheros de traducción. Ahora es posible tener varios perfiles definidos, como si fueran cuentas de usuario y elegir en cada momento cual de ellos se quiere utilizar. Se diseño un diálogo que permite la creación, edición y borrado de los perfiles y elegir cual de ellos se quiere tener activo. Para guardar la información se decidió utilizar un fichero de XML. Para el guardado y parseo de la información de este fichero se utiliza la biblioteca *libxml*.
- Memorias de traducción. Esta funcionalidad es básica en una herramienta como esta. Permite la reutilización de traducciones. Se diseño un diálogo para la creación de las memorias de traducción partiendo de un directorio con fichero de traducción. A partir de estos ficheros se crea una base de datos que contiene las diferentes traducciones que se han hecho de una misma cadena. Se diseño pensando en la posibilidad de tener diferentes *backends* que implementen la creación y almacenamiento y consulta sobre la base de datos.

Para su utilización se diseño un diálogo en la ventana principal del programa que muestra para cada cadena traducible las opciones disponibles en la base de datos, mostrando el nivel de coincidencia con la original. Se pensó así, ya que es más útil que la traducción automática, pues es necesario el contexto para realizar la traducción.

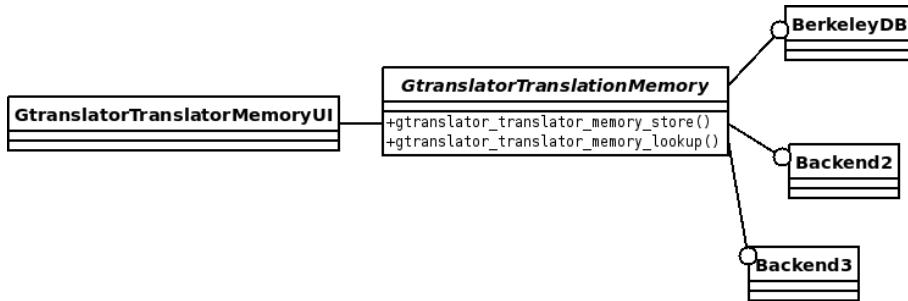


Figura 7: Diseño de las memorias de traducción

- Gestor de catálogos. Se diseño como una herramienta asociada a gtranslator que permite gestionar los ficheros de traducción de los proyectos en los que se colabora. Permite ver de manera cómoda y ordenada todos los ficheros sobre los que se trabaja y conocer su estado: número de cadenas totales, cadenas traducidas, sin traducir, difusas, etc... Además permite abrir cualquiera de estos ficheros en el gtranslator para su traducción.
- Asistente de configuración. Unha asistente para la primera vez que se arranque el programa, ya que es necesario configurar un perfil para poder empezar a trabajar. Este asistente guía al usuario paso a paso para la creación de un primer perfil. Además se puede configurar también la base de datos de la memoria de traducción.

Además de todo el desarrollo de código, se prestó atención también a la documentación, tanto de usuario como de desarrollador. Para la documentación de usuario se empleó GNOME Doc Utils. Este paquete de herramientas contiene utilidades para construir la documentación y todos los ficheros auxiliares, como plantillas de *Docbook* para la construcción del manual por ejemplo.

Una de las mayores ventajas que ofrece el tener los manuales de usuario en el formato que ofrece GNOME Doc Utils, es que gracias a una de las utilidades que incluye, *xml2po* se puede crear un fichero PO que contenga todas las cadenas de texto del manual de usuario.

Para la integración de las GNOME Doc Utils se creó todo el esqueleto necesario modificando los ficheros *Makefile.am* y *configure.ac*. Además es necesario crear un directorio *help* donde es necesario añadir una serie de ficheros, entre ellos el manual de usuario en formato *Docbook*.

Para la documentación de desarrolladores se utilizó *GTK-Doc*. Esta herramienta permite generar documentación para las APIs a partir de comentarios con un formato especial integrados en el código. Genera documentación en formato *Docbook*. Para el desarrollo de nuevos *plugins* se creó una biblioteca del programa de manera que los *plugins* pueda utilizar todas sus funciones. Por este motivo era imprescindible la generación de una buena documentación de desarrollo. Además se completó con la inclusión de diagramas de clase.

4. Conclusiones

En cuanto al desarrollo de la nueva versión, podemos decir que gtranslator ofrece ahora una interfaz muy mejorada respecto a la antigua versión, pensada para los traductores gracias a la discusión llevada a cabo para su diseño. Gtranslator no es ahora un simple editor de fichero de traducción, sino que ofrece una serie de funcionalidades que cubren los aspectos de ayuda a la traducción más importantes. En cuanto a la parte más técnica, destaca el uso de orientación a objetos gracias a la biblioteca *GObject*, que posibilita tener un código más modular y organizado, más fácil de entender y que facilita la programación y el aprendizaje a los nuevos desarrolladores.

En cuanto a los aspectos más relacionados con la comunidad, cabe destacar las múltiples posibilidades que se presentaron debido al estado del proyecto. El que se encontrara en un punto de casi abandono permitió llegar en poco tiempo al núcleo de desarrollo y a convertirse en mantenedor de la aplicación. Esto permitió conocer mucho más a fondo el funcionamiento de una comunidad, así como también posibilitó el aplicar los conocimientos adquiridos sobre el funcionamiento de los proyectos de Software Libre para tratar de reactivar una comunidad.

Como se puede ver en los análisis realizados en apartados anteriores, se ve una clara corriente de ascenso positiva tanto en el número de contribuidores en el *Subversion* como de usuarios en la lista de correo. Además, en los últimos meses se ha incrementado el número de reportes de fallo en el *Bugzilla*, así como también el número de parches, lo que hace pensar que poco a poco se está alcanzando el objetivo de reactivación de la comunidad.

En la introducción se hablaba de que otras herramientas estaban muy por encima de gtranslator en cuanto a funcionalidades. En la siguiente tabla se muestra el estado actual de estas herramientas con respecto a la nueva versión de gtranslator, comparando las funcionalidades⁴ que incluyen cada una de ellas.

	Gtranslator1	Gtranslator2	Kbabel	Lokalize	PoEdit
TM	No	Si	Si	Si	Si
Catálogos	No	Si	Si	Si	Si
Perfiles	No	Si	No	No	No
SCM	No	Si	No	No	No
Diff	No	Si	Si	Si	No
Tags	No	Si	Si	Si	No
Web	No	Si	No	Si	No
Sitaxis	No	Si	Si	Si	Si
Plugins	No	Si	No	No	No
Glosarios	No	No	No	Si	No

⁴TM: memoria de traducción; Catálogos: Gestor de catálogos; Perfiles: Gestión de varios perfiles de usuario; SCM: integración con sistemas de control de versiones; Diff: Diferencias entre distintas versiones de un fichero; Tags: insercción automática de etiquetas; Web: integración con servicios web; Sitaxis: resultado de sintaxis; Plugins: Sistema de Plugins; Glosarios: Búsqueda en glosarios de términos

A la vista de la tabla se puede concluir que gtranslator a día de hoy, no sólo se encuentra al mismo nivel en cuanto a funcionalidades que herramientas similares, sino que algunos casos incluye funcionalidades que el resto todavía no incorpora.

5. Trabajo Futuro

Dentro de las líneas de trabajo que se contemplan para el futuro podemos destacar, por un lado las que se refieren a la gestión de la comunidad y por otro lado los aspectos técnicos a mejorar y la inclusión de nuevas funcionalidades.

En el primer punto surge la idea de seguir un esquema de lanzamiento de nuevas versiones constante en el tiempo. La idea es sincronizar el lanzamiento de nuevas versiones de gtranslator con GNOME, es decir en ciclos de 6 meses. Con esto, incluso se podría planter la posibilidad de intentar que gtranslator fuese incluido como módulo oficial de GNOME.

Dentro de las nuevas funcionalidades a ser incluidas en la versión 2.2 destacan:

- La creación de *bindings* para *Python* que permita la utilización de este lenguaje, más atractivo quizás que C y de más alto nivel. Esto se pensó sobre todo para extender el gtranslator mediante la creación de nuevos plugins, ya que la versión actual tiene un sistema de *plugins* portado de Gedit.
- La integración con glosarios, que permita hacer búsquedas, sobre cuales son las convenciones para la traducción de términos que pueden tener varias traducciones diferentes en un mismo idioma.
- La creación de un sistema de importación/exportación de memorias de traducción entre aplicaciones haciendo uso del estándar TMX.

Sería también muy útil la realización de un informe sobre el estado actual del soporte de accesibilidad en gtranslator. Comprobar si todos los casos de uso puede ser accedidos mediante el teclado. Usar herramientas como accerciser para comprobar si todos los elementos tienen puesto correctamente un nombre y una descripción para ser leídos por las tecnologías de accesibilidad. En caso de detectarse faltas en el soporte deben ser corregidas para lograr un mínimo en cuanto a la accesibilidad de gtranslator.

Por último sería deseable el diseño e implementación de algunas pruebas de unidad para los casos de uso más habituales de la herramienta, así como también de un sistema de integración continua que realice la compilación del programa y ejecute algunos de los tests, generando informes, de manera que puedan ser corregidos lo antes posible.

Referencias

- [1] JESÚS GONZÁLEZ BARAHONA, JOAQUÍN SEOANE PASCUAL, GREGORIO ROBLES, *Introducción al software libre*, 2003. <http://curso-sobre.berlios.de/introsobre>.

- [2] KARL FOGEL, *Producing open source software*, 2005.
- [3] ISRAEL HERRAIZ, GREGORIO ROBLES, JUAN JOSÉ AMOR, TEÓFILO ROMERA AND JESÚS M.GONZÁLEZ BARAHONA, *The Processes of Joining in Global Distributed Software Projects*, 2006.
- [4] MASTER ON FREE SOFTWARE, *Analysis of the concentration of parameters in libre software communities*, 2008. http://gsyc.escet.urjc.es/moodle/file.php/40/Slides/S3_Gini_Roles_Open_BRR/Gini.pdf
- [5] GNOME PROJECT, *Human interface guidelines* <http://library.gnome.org-devel/hig-book/stable/>
- [6] MATTHIAS WARKUS, *The Official Gnome 2 Developer's Guide*, 2004.
- [7] MASTER ON FREE SOFTWARE, *Materiales de las sesiones presenciales*, 2008. <http://gsyc.escet.urjc.es/moodle/course/category.php?id=17>
- [8] GNOME PROJECT, *Developer documentation*, 2008. <http://library.gnome.org-devel/>
- [9] GTK PROJECT, *GTK documentation*, 2008. <http://www.gtk.org/documentation.html>