UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA

PROXECTO DE FIN DE CARREIRA
DE ENXEÑERÍA TÉCNICA EN INFORMÁTICA DE XESTIÓN

Reenxeñería de Gtranslator: Unha aplicación para a tradución de Software Libre

Autor: Pablo José Sangiao Roca

Director: Juán José Sánchez Penas

Titor: Ramón Miñones Crespo

A Coruña, Setembro de 2008

Especificación

Título do proxecto:	**Reenxeñería de Gtranslator: Unha aplicación para a tradución de Software Libre**
Clase:	Proxecto clásico de enxeñería.
Nome do alumno:	Pablo José Sangiao Roca.
Nome do director:	Juán José Sánchez Penas.
Nome do titor:	Ramón Miñones Crespo.
Membros do tribunal:	
Membros suplentes:	
Data de lectura:	
Cualificación:	

Á miña avoa... que se foi antes
de poder ver o soño
do seu neto cumprido.

Agradecementos

Nada disto sería posible sen a axuda de moita xente. Non sempre a axuda ten por que ser visible e moitas veces é dificilmente cuantificable. Nalgúns momentos, unhas simples verbas de ánimo son toda a axuda que se precisa para seguir cara adiante.

Por iso gustaríame dar as grazas a toda a xente que fixo posible que este proxecto se fixera realidade.

En primeiro lugar o meu director, Juán José Sánchez Penas, por tódala túa axuda e o teu apoio durante todo este tempo.

Tamén o meu titor, Ramón Miñones Crespo, pola túa axuda e confianza no proxecto.

Os meus pais... non teño palabras para agradecer todo o que me destes.

Os meus amigos, por estar sempre aí cando me fixéchedes falta.

A Noelia, por non deixar nunca que me rendese, nin sequera nos peores momentos.

E a toda a xente que por un ou outro motivo, nun ou noutro momento aportou algo a esta cabeciña.

A todos, ¡¡Mil Grazas!!

Resumo do proxecto

No mundo do *Software Libre* a *localización* dos programas faise mediante ficheiros de texto con extensión *.po*. Estes ficheiros conteñen todas as cadeas de texto que, a través da interface gráfica, lle son amosadas ó usuario. Ademais das cadeas de texto na linguaxe orixinal conteñen etiquetas para facer a tradución e información acerca de cada unha das mensaxes de maneira que se podan empregar directamente para a *localización* do programa. Todo isto fai que a súa manipulación con editores de texto tradicional por parte de usuarios non técnicos sexa bastante complexa.

Por todo isto xorden editores específicos para este tipo de ficheiros que por unha banda tratan de extraer as cadeas orixinais que se deben traducir, de todo o resto de información asociada para que esta sexa transparente ó tradutor e por outra banda tentan incluír funcionalidades que axuden ó usuario na súa labor de tradución.

Dentro de GNOME, unha das plataformas de escritorio máis populares entre o software libre, o programa para a edición de arquivos de tradución chámase *Gtranslator*. Nos últimos anos o seu desenvolvemento foi case nulo, ata o punto de que as comunidades de tradutores que se encargan de facer a tradución do propio proxecto GNOME ás diferentes linguas, deixaron de usalo no beneficio de outras ferramentas de proxectos similares, como pode ser o caso do Kbabel, por teren un nivel de desenvolvemento moito maior e ofrecer moitas máis funcionalidades de axuda ó tradutor.

O obxectivo deste proxecto era impulsar o desenvolvemento do *Gtranslator* para levalo a un nivel que lle permita ser tido en conta como unha opción válida para os tradutores. Este proxecto basease no análise da actual versión estable do programa e

a súa mellora ademais do deseño e implementación dunha nova versión que utilice as tecnoloxías actuais do proxecto GNOME ofreza unha serie de funcionalidades básicas que faciliten a tarefa de tradución os usuarios. Existe ademais unha compoñente menos técnica pero non por iso menos importante nun proxecto destas características, como é a de tentar reactivar a comunidade tanto de usuarios, dos que poder recoller a realimentación necesaria para mellorar o programa, como a de desenvolvedores para garantide a continuidade do proxecto no tempo.

Palabras clave

- Software Libre
- GNOME
- Gettext
- Ficheiro PO
- GObject
- GTK+
- Localización
- Internacionalización

Índice de contidos

Índice de Figuras

Índice de Táboas

Capítulo 1

Introdución

1.1 Motivación e obxectivos

Dentro do proxecto GNOME o *Gtranslator* é a ferramenta que se utiliza para a edición de ficheiros *PO*. É unha ferramenta moi necesaria para facilitar a labor de tradución os voluntarios xa que de forma xeral, destes son usuarios non técnicos e como imos ver no capítulo 2 a edición de estes ficheiros cun editor de texto convencional é unha labor que pode resultar complexa.

O estado do *Gtranslator* antes de comezar este proxecto era o de unha ferramenta que estaba bastante desactualizada. Tanto o seu mantemento como o seu desenvolvemento encóntranse nun nivel de actividade bastante baixo. Isto fai que tanto a comunidade de desenvolvedores como a de usuarios sexa moi pequena. Os propios tradutores do proxecto GNOME vense obrigados a ter que utilizar aplicacións similares de outros proxectos como pode ser o caso da ferramenta Kbabel dentro do proxecto KDE. Esta ferramenta, por exemplo, prové os tradutores de moitas funcionalidades adicionais ademais da simple edición de ficheiros que facilitan a labor de tradución. Por isto o principal obxectivo deste proxecto será o de conseguir levar o *Gtranslator* a un nivel de desenvolvemento similar ao das demais ferramentas similares e poda ofrecer os tradutores un conxunto de funcionalidades que facilite o

seu traballo máis alá de ser un simple editor de ficheiros. Ademais como segundo obxectivo derivado directamente do principal está o intentar conseguir reactivar a comunidade detrás do proxecto tanto de desenvolvedores coma de usuarios.

Máis en detalle os principais obxectivos serán:

- Documentar a aplicación para os desenvolvedores de maneira que podan entrar a colaborar minimizando o esforzo de aprendizaxe e familiarización co código.
- A creación dun manual de usuario que inclúa explicacións sobre o uso de tódalas novas funcionalidades.
- Facer unha análise da ferramenta para ver os seus principais problemas a nivel de eficiencia e usabilidade e tratar de aportar unha solución para os mesmos.
- Implementar novas funcionalidades que axuden os tradutores na súa labor como por exemplo un sistema de memorias de tradución que permita a reutilización de traducións anteriores ou un sistema de perfís que permita ter contas con diferente información sobre a lingua ou o tradutor.

1.2 Fases do proxecto

No mundo do *Software Libre* normalmente hai sempre como mínimo dúas versións dos programas, a versión estable, isto é versión de produción nun proxecto comercial, é a que usan normalmente os usuarios do programa e a que se inclúe nas distribucións de *GNU/Linux*, e por outra banda está a versión de desenvolvemento ou *trunk* que é a versión na que traballan os desenvolvedores para implementar as novas funcionalidades e que se converterá na seguinte versión estable.

Isto non significa que na versión estable, unha vez que foi lanzada, non haxa traballo nela por parte dos desenvolvedores. existe o que se denomina traballo de mantemento que consiste en aplicar pequenos parches que resolvan erros detectados no programa despois do lanzamento da versión estable. Neste proxecto dánse os dous

tipos de traballo, o de mantemento da versión estable e o de desenvolvemento dunha nova versión do *Gtranslator*.

A primeira fase está ligada a versión estable do *Gtranslator* isto é a versión 1.1.7 do programa. Nesta primeira fase o obxectivo será o de analizar o estado actual da ferramenta para ver por un lado que se necesita facer para mellorala seguindo os obxectivos fixados e por outro lado arranxar os seus problemas máis importantes e lanzar unha nova versión de mantemento, a 1.1.8, que serva de ponte para que os usuarios a podan utilizar mentres se fai o desenvolvemento da nova versión que será 2.0. Esta primeira fase serve tamén como entrada no proxecto.

En xeral os proxectos de *Software Libre* cumpre o chamado modelo da cebola. Segundo podemos ver en [9] o proceso de entrada dun desenvolvedor no proxecto GNOME segue este modelo. Na súa primeira etapa o desenvolvedor arranxa erros do programa mediante o envío de parches o mantedor do programa ata que se consegue o nivel de confianza necesario para ser aceptado como desenvolvedor oficial do programa, xa que no eido do *Software Libre* predomina a meritocracia. Nesta primeira fase do proxecto téntase obter esta confianza por parte do actual mantedor do *Gtranslator* para poder ter liberdade a hora de tomar decisións para o desenvolvemento da seguinte versión.

A segunda fase do proxecto é o desenvolvemento da nova versión do *Gtranslator*. Neste caso será a versión 2.0 do proxecto. Tomando como base a primeira versión farase un análise do seu estado actual para tentar ver cales son os cambios necesarios e as funcionalidades que haberá que implementar para conseguir unha ferramenta que este o nivel doutras similares.

1.3 Estrutura da memoria

A estrutura da memoria ven marcada polas fases do proxecto. Permite seguir como foi o desenvolvemento do proxecto dende os pasos iniciais de coñecemento de como

funciona a tradución no mundo do *Software Libre* e o estado do programa que se quere mellorar, o contacto coa comunidade de GNOME e o desenvolvemento da nova versión do programa e as funcionalidades.

No capítulo 2 móstrase o funcionamento da tradución no mundo do *Software Libre* e o estado actual das ferramentas que se usan para levala a cabo, incluíndo ó do *Gtranslator*.

No capítulo 3 explícase o método de traballo que se segue no proxecto así como as metodoloxías de desenvolvemento de *software* que se usaron no mesmo.

No capítulo 4 detállase o traballo de mantemento realizado na versión estable do *Gtranslator*, a resolución dos erros máis importantes que se levou a cabo e o análise que se fixo da ferramenta nese caso.

No capítulo 5 explícase con detalle a parte do proxecto que ten que ver co desenvolvemento da nova versión do *Gtranslator*, os pasos que se seguiron no análise, deseño e implementación da nova versión, as decisións que se tomaron e o proceso de implementación das novas funcionalidades.

Capítulo 2

Estado da Arte

2.1 Tradución de *Software Libre*

Á tradución de *Software Libre* é un proceso tan importante como o do desenvolvemento da aplicación. De pouco serve que a interface da aplicación sexa moi clara senón está no idioma co usuario entende. É por isto que no eido do *Software Libre* se lle dea unha grande importancia a tradución.

Ademais é unha posibilidade de colaboración para aquelas persoas que non teñen unha formación técnica no desenvolvemento de *software* pero queren participar nun proxecto de *Software Libre*.

2.1.1 Como se traballa

O traballo comeza por parte do desenvolvedor da aplicación, quen terá que preparar o seu código e xerar os ficheiros que posteriormente deberán ser traducidos. Esta preparación do código debe ir máis alá dunha mera separación das cadeas de texto para que despois podan ser traducidas, xa que dun idioma a outro cambia a orde alfabética, o signo da moeda e a separación dos decimais, por exemplo. Polo tanto o desenvolvedor debe facer unha *internacionalización* do código.

A partires de aquí comeza o traballo dos tradutores, en plural, xa que normal-

mente a tradución non é un traballo individual. Os tradutores adoitan a xuntarse en equipos de traballo onde hai xente dedicada á tradución en si, outra xente que se encarga das revisión e un coordinador que se encarga da organización e o reparto do traballo. Unha vez que a tradución está lista envíase normalmente o desenvolvedor e os mantedores dos paquetes. Pero o traballo non remata aquí, xa que é normal que a aplicación teña novas versións, polo que haberá que facer actualizacións.

2.1.2 ¿Que é *GNU Gettext*?

Xa vimos que son os desenvolvedores os encargados de comezar o proceso de tradución preparando o código e creando os ficheiros necesarios. Para axudarlles na súa labor existe *GNU Gettext*.

GNU Gettext ofrece ó desenvolvedor un conxunto de utilidades e documentación que minimizan os problemas que poden xurdir na *internacionalización* do código. Estas utilidades inclúen:

- Un conxunto de convencións sobre como escribir os programas para que soporten un catálogo de mensaxes.
- Unha nomenclatura propia para os directorios e ficheiros do catálogo de mensaxes.
- Unha biblioteca para parsear e crear os ficheiros que conteñen as mensaxes traducidas.

Aínda que se empregase xa antes a palabra *internacionalización* imos tratar de explicar agora un pouco máis a fondo o seu significado e o de outra palabra que sempre a acompoña: *localización*. Cando falamos de *internacionalización* estámonos a referir a operación pola cal un programa é capaz de soportar múltiples linguaxes. Os desenvolvedores empregan varias técnicas para internacionalizar os seus códigos, algunhas das cales foron estandarizadas. *GNU Gettext* é un deses estándares.

Á *localización* dálle a un programa que fose previamente internacionalizado toda a información necesaria para poder soportar os as diferentes linguaxes e as súas convencións culturais. Para que un programa saiba en que linguaxe debe mostrar tódalas mensaxes ó usuario existen unha serie de variables de entorno denominadas *LOCALES*. Ditas variables poden ser fixadas mediante ficheiros de configuración únicos para todo o sistema ou poden ser modificadas libremente polo usuario. Isto permite unha grande flexibilidade xa que se pode trocar a lingua na que se mostra un programa na execución simplemente antepoñendo o LOCALE correspondente. Os aspectos que se deben ter en conta a hora de establecer un LOCALE son:

Caracteres e Codesets O codeset máis empregado na maior parte dos países anglosaxóns é o *ASCII*. Sen embargo moitos caracteres que son necesitados por varios *LOCALES* non funcionan ben con esta codificación. O *ISO 8859-1* ten moitos dos caracteres especiais que necesitan a meirande parte das linguas Europeas.

Moeda Os símbolos da moeda varían dun país a outro, ademais da colocación do dito símbolo con respecto as cantidades.

Datas O formato da data varía entre os *LOCALES*. Á meirande parte dos países anglosaxóns empregan o formato Mes/Día/Ano, sen embargo nos países latinos usamos o formato Día/Mes/Ano. Ademais o formato da hora pode variar de hh:mm a hh.mm ou outras formas.

Números Os números teñen diferente representación dependendo do LOCALE, por exemplo:

```
12,345.67       Inglés
12.345,67       Galego
 12345,67       Francés
```

Mensaxes É o apartado máis obvio dentro dos *LOCALES*. É onde *GNU Gettext* prove o que os desenvolvedores e usuarios necesitan para de maneira sinxela cambiar a linguaxe na que o *software* se comunica co usuario.

Imos ver cal é o proceso que debe seguir un desenvolvedor para conseguir a *internacionalización* do seu código axudándose das ferramentas que proporciona *GNU Gettext*. En primeiro lugar hai que importar as bibliotecas correspondentes para poder facer uso das funcións de *GNU Gettext*.

```
#include <libintl.h>
```

Unha vez feito isto, o segundo paso é facer que cando se execute o programa faga uso das variables de entorno *LOCALES*, das que falábamos anteriormente. O código para iso debería ser algo parecido a isto:

```
int main (int argc, char * argv[])
{
...
setlocale(LC_ALL, " ");
bindtextdomain(PACKAGE, LOCALEDIR);
textdomain(PACKAGE);
...
}
```

LC_ALL inclúe a meirande parte das categorías de LOCALE. Pódense empregar outras en caso dalgunha necesidade concreta como por exemplo o uso da "cedilla", ç, que é normal no francés por exemplo pero non así no inglés. PACKAGE e LOCALEDIR débense de establecer en ficheiros de configuración ou no Makefile. Á hora de preparar as cadeas de texto para ser traducidas hai que ter especial coidado ó seguir unhas pautas para facilitar a labor de tradución.

- Un decente estilo de Inglés

- Sentencias enteiras
- Utilizar cadeas de texto formateadas no lugar de concatenacións.
- Evitar marcas inusuais

Cuns exemplos poderemos entender mellor todo isto:

```
printf ("File %s is %s protected", filename, rw ? "write" : "read");
```

O problema desta liña é que non hai sentencias enteiras, vemos dous verbos write e read que aparecen sós. Isto hai que tentar evitalo xa que o futuro tradutor necesita un contexto para poder traducir con coherencia. A liña correcta sería:

```
printf (rw ? "File %s is write protected" :
             "File %s is read protected", filename);
```

Outro exemplo:

```
printf ("Locale charset \"%s\" is different from\n", lcharset);
printf ("input file charset \"%s\".\n", fcharset);
```

Neste caso o tradutor vería e traduciría dúas cadeas separadas, pero nada lle indicaría que deben ir unidas formando unha soa frase. Por iso sería mellor unilas cun só *printf* así:

```
printf ("Locale charset \"%s\" is different from\n\
input file charset \"%s\".\n", lcharset, fcharset);
```

Unha vez arranxados os problemas estilísticos debemos marcar as cadeas que queremos que sexan traducibles, isto é, as mensaxes coas que a aplicación se comunicará co usuario. A forma de facelo é encerrar a mensaxe entre parénteses e antepoñer a palabra clave `gettext`, así:

```
gettext ("Cadea traducible")
```

Moitos paquetes usan '_' como palabra clave no lugar de `gettext`, xa que é habitual que as palabras clave sexan cortas e non obstrutivas para o resto do código. No caso de que o paquete non soporte '_' como palabra clave é fácil definilo manualmente facendo:

```
#include <libintl.h>
#define _(String) gettext (String)
```

Una vez feitos todos estes pasos xa temos o programa preparado para a súa *internacionalización*. Agora debemos facer uso da ferramenta *xgettext* que se encargará de percorrer todo o noso código buscando as cadeas que marcamos, anotando a liña na que aparecen e xerando así o ficheiro de texto *POT* inicial que posteriormente será empregado polos tradutores para realizar o seu traballo. Cabe destacar que *xgettext* non crea un ficheiro con extensión *POT*, senón *PO*, polo que debemos renomealo a man. O motivo é que cando se creou *xgettext* a extensión *POT* aínda non era empregada.

2.1.3 ¿Que hai nun ficheiro *PO*?

Como xa vimos, a responsabilidade no proceso de tradución do desenvolvedor podemos dicir que remata cando xera o ficheiro *POT* que contén tódalas mensaxes que se deben traducir. Este ficheiro é en realidade unha plantilla sobre a que se crean os ficheiros *PO*, di o seu nome *PO* Template. Cando se empeza a tradución dun programa a unha determinada lingua o que hai que facer é crear un ficheiro *PO* correspondente a esa lingua, por exemplo, para o galego chamaríase `gl.po`. Este ficheiro gárdase dentro dun directorio chamado "po" xunto con tódolos demais ficheiros das linguas as que foi traducido o programa, empregando para todos eles como nome un código de dúas letras que identifica a lingua. A creación dun ficheiro *PO* pódese facer de forma manual simplemente copiando o ficheiro *POT* e renomeando

de maneira automática facendo uso da outra ferramenta do paquete *GNU Gettext*, *msginit*. Un exemplo de uso sería:

```
$ cd PACKAGE-VERSION
$ cd po
$ msginit
```

Imos ver agora o contido deste ficheiro *PO* con detalle. O primeiro que nos atopamos é unha cabeceira que contén os seguinte campos:

Project-Id-Version Este é o nome e a versión do paquete.

Report-Msgid-Bugs-To Este campo é enchido pola ferramenta *xgettext*. Contén unha dirección de email ou unha URL onde se poden reportar erros sobre as cadeas sen traducir.

POT-Creation-Date Data de creación da plantilla, tamén é enchido por *xgettext*.

PO-Revision-Date Este campo non é necesario que se encha manualmente. Debe ser enchido polo editor de ficheiros *PO* cando se garde o ficheiro.

Last-Translator Nome e dirección de email do último tradutor que modificou o ficheiro.

Language-Team Nome en Inglés da lingua á que se traduce e dirección de email ou páxina web do equipo do que o tradutor forma parte.

Content-Type Información sobre a codificación dos caracteres do idioma.

Content-Transfer-Encoding Débese establecer en 8bits.

Plural-Forms É un campo opcional. Só se necesita se o ficheiro *PO* contén formas plurais.

A cabeceira séguenlle as entradas. Cada unha desas entradas dun ficheiro *PO* ten a seguinte estrutura:

```
ESPAZO EN BRANCO
# COMENTARIOS DO TRADUCTOR
#. COMENTARIOS AUTOMÁTICOS
#: REFERENCIA DA MENSAXE
#, MARCAS (traducción dubidosa, ...)
msgid CADEA ORIXINAL
msgstr CADEA TRADUCIDA
```

Cada unha das cadeas de texto que foran marcadas no código fonte como cadeas traducibles aparecen agora aquí despois da palabra clave *msgid*, que indica que o que ven despois é unha cadea orixinal que se mostrará por pantalla ó usuario, a cal débese traducir. Inmediatamente despois aparece a palabra clave *msgstr*, e unha cadea valeira. Esta cadea é a que debemos encher coa tradución.

Este proceso pódese facer editando o ficheiro con calquera editor de textos, aínda que é máis cómodo empregar algunha das ferramentas específicas que existen para elo e que veremos máis adiante. Como sabemos as aplicacións soen evolucionar ó longo da súa vida xa que os desenvolvedores soen sacar novas versións melloradas cada certo tempo. Nestas novas versións é moi probable que aparezan novas mensaxes que deben ser traducidos. Para non ter que voltar a traducir o ficheiro *PO* completo existe outra ferramenta de *GNU Gettext*, *msgmerge*, que vainos permitir reutilizar as cadea de texto xa traducidas. Si a cadea variou algo dende a primeira versión, *msgmerge* marcará a cadea como dubidosa (fuzzy en Inglés) e será o tradutor o que teña que revisala.

2.2 Ferramentas para a tradución

Acabamos de ver que a base da tradución de aplicacións de *Software Libre* son os ficheiros *PO*, e que estes ficheiros pódense editar con calquera editor de ficheiros de texto, pero, para facilitar a labor dos tradutores fóronse creando ó longo do tempo programas orientados á edición deste tipo de ficheiros, son os chamados editores de ficheiros *PO*. Aquí falaremos do *Gtranslator* e do Kbabel que son os editores de ficheiros *PO* dos escritorios Gnome e Kde e faremos tamén unha mención a Rosetta, unha nova aplicación de este tipo que ademais ten a novidade de ser online.

2.2.1 KDE e Kbabel

Kbabel é un conxunto de ferramentas para a edición e o manexo de ficheiros *PO*. É un editor moi potente no que destacan dúas aportacións: o xestor de catálogos e a posibilidade de usar dicionarios.

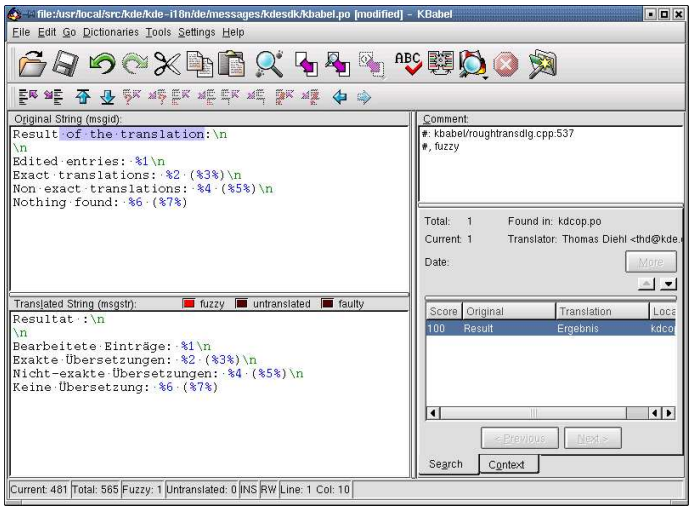

Figura 2.1: Interface principal de Kbabel

Xestor de catálogos O xestor de catálogos de Kbabel permítenos xestionar dunha forma cómoda e ordenada todos os nosos proxectos de tradución. Unha vez aberto e despois de esperar a que lea todos os directorios, móstranos o es-

tado da tradución de todos os ficheiros. Danos a información sobre cada un dos ficheiros, como o número de mensaxes totais, as que están sen traducir, o número de cadeas marcadas coma dubidosas, etc. Á nivel de directorio móstranos cunha icona se todos os ficheiros están completamente traducidos ou polo contrario conteñen cadeas sen traducir ou dubidosas. Facendo "dobre clic" sobre calquera dos ficheiros automaticamente ábrese o editor de Kbabel para que podamos modificalo.

Dicionario de Kbabel Kbabel posúe tamén un dicionario que nos permite comprobar as diferentes traducións que fixeron outros tradutores dalgún dos termos. Podemos facer búsquedas en distintos tipos de dicionarios:

- *PO* auxiliar
- Base de datos de tradución
- Compendio *PO*
- Compendio TMX

Os compendios son grandes catálogos de mensaxes traducidas que se poden xerar coa ferramenta, *msgcat*, a partir de unha serie de ficheiros *PO* ou mediante TMX, que é unha linguaxe que cumpre as especificacións de XML e que ten como propósito proporcionar un estándar para o intercambio das memorias de tradución.

A outra gran ferramenta de Kbabel e a máis importante é o editor propiamente dito. Consta de varios espazos na ventá principal. A metade esquerda é onde se leva a cabo o proceso de tradución, na parte de arriba preséntanse as cadeas orixinais nun recadro non editable, mentres que na parte inferior aparece un recadro editable onde se debe teclear o texto traducido.

Na parte da dereita móstrase todo tipo de información que facilita a labor do tradutor. Na parte superior aparecen os comentarios que o desenvolvedor puidera

escribir con aclaracións para os tradutores. Na parte inferior contén a súa vez, unha serie de pestanas con distintas opcións, entre as que podemos destacar a de "Contexto PO", onde se pode ver un extracto do contido do ficheiro onde se encontra a cadea a traducir. Na parte inferior, temos a barra de estado que mostra a información sobre o número de cadeas totais do ficheiro, as que están sen traducir, as dubidosas e na que nos encontramos.

En resumo, trátase dunha ferramenta moi potente que facilita nunha gran medida o traballo dos tradutores e que ofrece moitísimas funcionalidades complementarias á tradución. É por isto polo que actualmente é sen dúbida a ferramenta máis empregada pola grande maioría de equipos de tradución.

2.2.2 GNOME e Gtranslator

Gtranslator é o editor de ficheiros *PO* de GNOME. A día de hoxe o seu nivel de desenvolvemento é moito menor que o do Kbabel, referencia no mundo da tradución de *Software Libre*. O seu desenvolvemento encóntrase bastante estancado, pero iso é algo que se intentará mellorar con este proxecto.

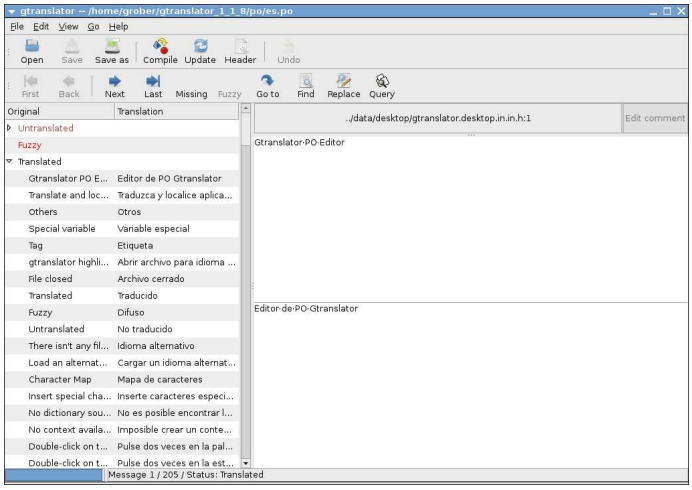

Figura 2.2: Interface principal de *Gtranslator*

Analizándoo con respecto a Kbabel podemos darnos conta dos problemas e do que hai que mellorar. Por exemplo, falábamos do cómodo que resulta en Kbabel facer

búsquedas sobre traducións anteriores dunha mensaxe e todas as diferentes opcións onde as podíamos facer. Sen embargo no *Gtranslator* non hai nada parecido, o único que nos ofrece é a posibilidade de facer unha tradución automática usando un buffer en Xml onde podemos ir almacenando as traducións que imos facendo. O problema é que as traducións automáticas non soen funcionar de maneira axeitada.

Gtranslator é simplemente un editor, que nos permite traballar sobre un único ficheiro *PO*, non é posible ter unha organización de proxectos como tiñamos en Kbabel gracias o seu xestor de catálogos.

2.2.3 PoEdit

PoEdit é un editor de ficheiro *PO* multiplataforma. Está construído en C++ empregando o *toolkit wxWidgets* que permite crear aplicacións gráficas para diversas plataformas (Win32, Mac OS X, *GTK+*, X11). Aínda que polo seu deseño pode ser multiplataforma, ata o momento só está dispoñible para entornos *Unix* con *GTK+* e Windows.

Entre as súas principais características podemos destacar:

- Presenta tódalas mensaxes nunha forma moi compacta por medio dunha lista, que permite navegar por todas elas de forma moi cómoda.
- Como xa se comentou é multiplataforma.
- Soporta formas plurais.
- Resalta os espazos en branco nas mensaxes.
- Resalta e mostra as mensaxes dubidosas e sen traducir ó principio da lista.
- Compila de forma automática os ficheiros .mo.
- Actualiza a cabeceira dos ficheiros *PO* de forma automática.
- Permite escanear código fonte para buscar cadeas traducibles.

- Intégrase ben cos escritorios GNOME e KDE.
- Ten un sistema de memorias de tradución automático.
- Permite a edición de comentarios.
- Inclúe un xestor de proxectos.

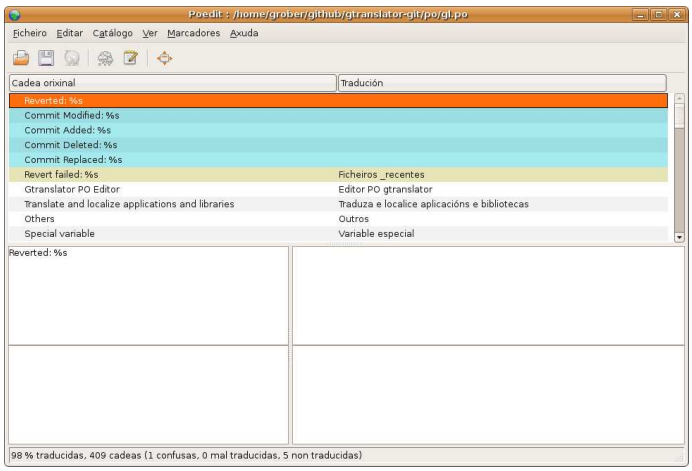

Figura 2.3: Interface principal de PoEdit

2.2.4 Tradución online: Rosetta

O amparo de Ubuntu, unha das distribucións de ámbito global máis importante na actualidade, nace *Rosetta*. *Rosetta* é o nome dun sitio web, creado para traballar na tradución de *Ubuntu*. O primeiro paso para comezar a traballar con *Rosetta* é rexistrase como tradutor, creando unha conta de usuario onde debemos dar a nosa dirección de correo e uns poucos datos máis. Despois teremos que configurar a nosa conta elixindo o nosa lingua para traducir. Aquí é recomendable unirse ó equipo de tradutores da nosa lingua o que se fai pinchando en *Ubuntu Translators*, sen embargo esta inclusión non é inmediata, xa que se necesita pasar por un proceso de aprobación por parte dos administradores. Vexamos como é o proceso de tradución:

En *Rosetta* hai moitos paquetes que están en proceso de tradución. Estes paquetes poden conter dende unha aplicación grande (como *Aptitude*, por exemplo) ata unha utilidade de liña de ordes, por exemplo *eject*, pasando por algún compoñente oculto do sistema (como *libgphoto*).

Cada un deste paquetes ten un conxunto de mensaxes que son todas as cadeas de texto co programa emprega para comunicarse co usuario. Algúns paquetes poden conter centos de mensaxes e outros só un. Todas estas mensaxes orixinais están en Inglés, polo tanto a nosa tarefa como tradutor será reescribilos ó galego, por exemplo, tentando cumprir estes dous obxectivos:

- O significado debe manterse de forma fiel.
- A mensaxe traducida debe ser clara e soar natural.

Para poder traducir as mensaxes dun paquete, primeiro teremos que atopalo na súa versión correcta. A miúdo atopámonos na súa interface unha lista de paquetes. A barra horizontal de cores ofrécenos información sobre o estado do paquete. Se non atopamos o paquete que queremos traducir no listado, dispoñemos da opción de facer unha búsqueda en toda a base de datos de *Rosetta*, para iso debemos pinchar sobre *any other open source application*.

Cando atopamos o paquete móstrasenos as mensaxes a traducir. A primeira pantalla móstranos as 10 primeiras mensaxes para traducir, a seguinte as mensaxes da 11 á 20 e así sucesivamente. Pinchando sobre *Save & Continue* gardamos as mensaxes traducidas e pasamos a pantalla seguinte. Se non estamos seguros sobre a tradución dalgunha das mensaxes podemos marcar a casela Someone should review this translation.

Capítulo 3

Metodoloxía

3.1 Método de traballo

O traballo a realizar sobre o *Gtranslator* podémolo dividir en dúas fases. Por un lado traballarase sobre a versión estable mediante a realización de parches que corrixan os bugs máis importantes. Despois implementaranse as novas funcionalidades traballando sobre a versión de desenvolvemento tomando cada unha delas coma un proxecto individual e aplicando para o seu desenvolvementos as técnicas que se explican con detalle nos seguintes apartados.

3.2 Técnicas

As técnicas que se presentan a continuación son moi empregadas en proxectos de *Software Libre* xa que se adaptan plenamente a súa filosofía de traballo e presentan unha grande adaptabilidade, o que nos permitirá traballar con ambas ó mesmo tempo de maneira sinxela.

3.2.1 Proceso de desenvolvemento unificado

Para o desenvolvemento deste proxecto seguíronse as liñas definidas pola metodoloxía de desenvolvemento unificado do *software* [2].

O Proceso Unificado é un proceso iterativo e incremental, centrado na arquitectura e guiado polos casos de uso. É un proceso baseado en moitos anos de experiencia no uso da tecnoloxía orientada a obxectos no desenvolvemento de *software* pola compañía *Rational*.

Este proceso guía aos equipos do proxecto na administración do desenvolvemento iterativo de modo controlado, mentres se balancean os requirimento do negocio e os riscos do proxecto.

As súas principais características son:

- É un marco de desenvolvemento iterativo e incremental composto de catro fases: Inicio, Elaboración, Construción e Transición. Cada unha de estas fases a súa vez divídese en diversas iteracións que engaden ou melloran as funcionalidades do sistema de desenvolvemento. Estas iteracións divídense en fases que recordan as definidas no ciclo de vida clásico ou de cascada: Análise de requisitos, Deseño, Implementación e Probas.

- Está dirixido polos casos de uso. Estes empréganse para capturar os requisitos e definir as iteracións. A idea é que en cada iteración se traballe nun conxunto de casos de uso e se desenvolva todo o camiño seguindo as fases de deseño, implementación e probas.

- Céntrase na arquitectura xa que asume que non é posible obter un modelo que englobe tódolos aspectos do sistema. Por iso existen moitos modelos e vistas que definen a arquitectura *software* dun sistema.

- Está enfocado nos riscos, xa que se necesita que os membros do equipo de desenvolvemento identifiquen os riscos que sexan críticos nunha etapa temprana

do ciclo de vida. Nas iteracións os riscos principais débense de considerar primeiro.

Tamén se usou a Linguaxe Universal de Modelaxe ou *UML* [1], producindo con ela diagramas de clases, casos de uso e secuencia. Todos estes diagramas permiten dispor dunha fonte de información estandarizada durante todo o desenvolvemento do proxecto.

3.2.2 eXtreme Programming

A programación extrema ou eXtreme Programming (XP) é un enfoque da enxeñería do software que a diferenza das metodoloxías tradicionais pon máis énfase na adaptabilidade que na previsibilidade. Considera que os cambios de requisitos sobre a marcha son un aspecto natural, inevitable e incluso desexable no desenvolvemento de proxectos. Ser capaz de adaptarse os cambios de requisitos en calquera punto da vida dun proxecto é unha aproximación mellor e máis realista que tentar definir tódolos requisitos ó comezo do proxecto e inverter esforzos despois en controlar os cambios.

Pódese considerar a programación extrema como a adopción das mellores metodoloxías de desenvolvementos de acordo ó que se pretende levar a cabo co proxecto e aplicalo de maneira dinámica durante o ciclo de vida do *software*.

As súas principais características son:

- Desenvolvemento iterativo e incremental: pequenas melloras, unha tras outra...
- Probas unitarias continuas, frecuentemente repetidas e automatizadas. Aconséllase escribir o código da proba antes da codificación.
- Programación en parellas: Suponse que o código é de máis calidade xa que é revisado e discutido mentres se escribe.
- Frecuente iteración do equipo de programación co cliente ou usuario.

- Corrección de tódolos erros antes de aplicar a funcionalidade.
- Refactorización do código, é dicir, reescribir certas partes do código para aumentar a súa lexibilidade e mantebilidade pero sen cambiar o seu comportamento.
- Simplicidade no código: é a mellor maneira de que as cousas funcionen. Cando todo funcione poderase aplicar máis funcionalidade.
- A simplicidade e comunicación son complementarias, canto máis sinxelo sexa o sistema, a comunicación será máis sinxela tamén.

3.3 Ferramentas

O traballo de desenvolvemento en proxectos de *Software Libre* non se entende sen que se desenvolva en comunidade, polo tanto son necesarias ferramentas que faciliten esa colaboración entre desenvolvedores que se atoparán en diferentes partes do mundo. As máis empregadas son estas:

3.3.1 Bugzilla de GNOME

Bugzilla é unha base de datos que serve para que o usuario reporte tódolos bugs ou erros que atope no seu GNOME. Reportar erros é unha tarefa sinxela e que resulta moi útil os desenvolvedores para mellorar as súas aplicacións.

Máis en detalle *Bugzilla* e unha ferramenta das chamadas de seguimento de erros (bugs), orixinalmente desenvolvida e empregada polo proxecto Mozilla, é hoxe en día un dos sistemas de seguimento de bugs máis empregados e foi adoptada por moitos proxectos de *Software Libre*, incluído GNOME.

Bugzilla permite organizar os bugs de varias formas, permitindo o seguimento de múltiples produtos con diferentes versións. Pódense categorizar os bugs de acordo á

súa prioridade e severidade, así como asignar en que versión van ser arranxados. As categorías típicas dos bugs son:

- **Blocker:** Bloquean o desenvolvemento do programa.
- **Critical:** *Crashes* do programa, perda de información.
- **Major:** Perda de funcionalidade no programa.
- **Normal:** A maioría dos bugs.
- **Minor:** Perda de pequenas funcionalidades pero non impiden o uso normal do programa.
- **Trivial:** Pequenos problemas que non afectan ó comportamento do programa, erros de sintaxe nas cadeas por exemplo.
- **Enhancement:** Non son erros propiamente ditos senón peticións de novas funcionalidades para o programa.

Para poder usar o *Bugzilla* é necesario crear unha conta cunha dirección de correo válida. Con isto calquera usuario pode acceder e reportar un erro en calquera aplicación de GNOME. O mesmo tempo o *Bugzilla* serve para que os desenvolvedores envíen os seus parches que ofrecen solucións para os erros.

O fluxo normal que segue un bug é: Primeiro alguén crea un novo bug, fai un reporte explicándoo o fallo e dándolle unha categoría. Despois o mantedor do proxecto revisará o bug e comprobará se é un bug realmente, se é así marcarao como novo e asignarallo a un desenvolvedor, ademais pódeo cambiar de categoría ou asignar a unha versión concreta do programa. O desenvolvedor encargado do bug arranxarao e subirá o parche. Unha vez comprobado que o parche funciona pecharase o bug marcándoo como arranxado. Esta é unha versión reducida do proceso, que pode ter bastantes máis variantes como se ve no seguinte gráfico:

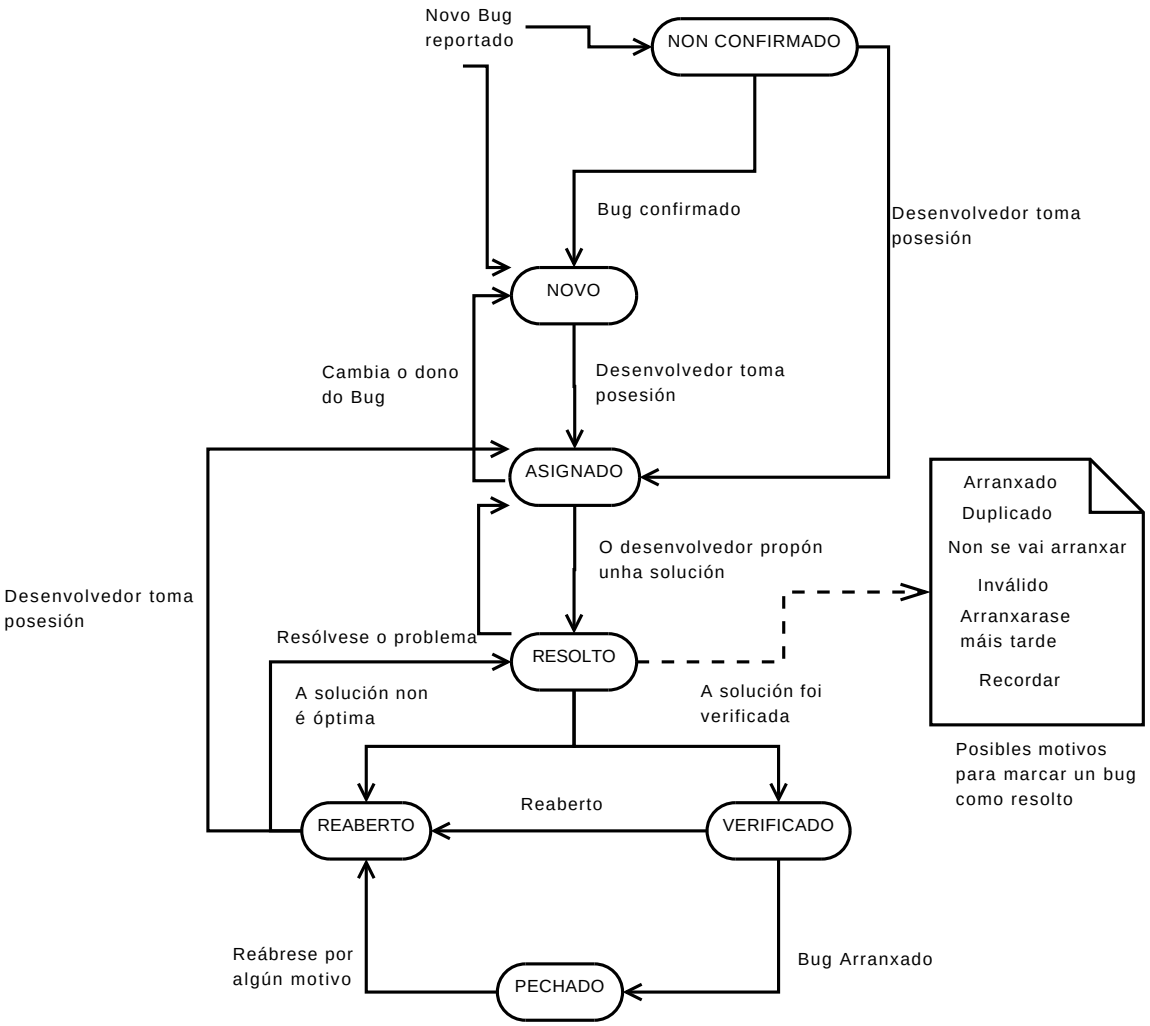

Figura 3.1: Diagrama de fluxo dos estados polos que pode pasar un Bug

O mantedor da aplicación será o encargado de xestionar o *Bugzilla*, modificando o estado dos *bugs* ou pechándoos cando se considera que están solucionados. O acceso ó *Bugzilla* fáise a través dun navegador *web* na dirección: http://bugzilla.gnome.org

3.3.2 *Subversion*

O *Subversion* é o sistema de control de versións de GNOME. É un método que permite a varios desenvolvedores traballar no mesmo código. Mantén información sobre cada un dos cambios que se fan nos ficheiros. A maneira de traballar é a seguinte: Cada desenvolvedor obtén unha versión do código dendo o *Subversion* ó seu equipo persoal. Unha vez coa súa copia traballa no desenvolvemento de maneira illada. Cando leva feitos varios cambios e considera que xa están suficientemente probados, fai un commit, é dicir, envía os cambios feitos no código que tiña na súa máquina persoal o servidor do *Subversion*. Este encárgase de actualizar o código cos cambios que ficho o desenvolvedor e avisarlle no caso de que exista algún problema para que o desenvolvedor o corrixa manualmente e trate de enviar os cambios de novo.

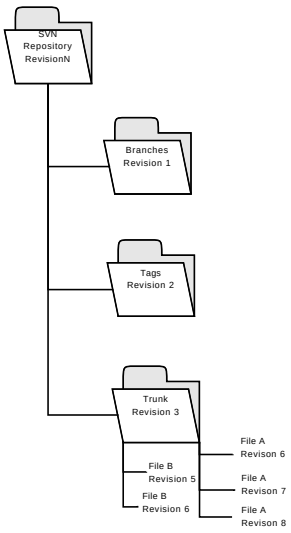

Figura 3.2: Sistema de ficheiros de Subversion

A organización tradicional do *Subversion* é como se pode ver na figura 3.2 un di-

rectorio ou repositorio que contén tres directorios chamados *branches*, *tags* e *Trunk*. Isto ten que ver coa organización tradicional que teñen os proxectos de *Software Libre* pero non ten por que ser así, a estrutura de directorios pódese modificar como se desexe. Na estrutura tradicional o directorio *Trunk* é para traballar na versión de desenvolvemento do proxecto. O directorio *tags* é un directorio onde se van mantendo tódalas diferentes versións do proxecto que se van sacando. Por último o directorio *Branches* emprégase para crear ramas. As ramas son empregadas para facer desenvolvementos en paralelo. Por exemplo cando se plantexan diferentes solucións ou implementacións para unha nova funcionalidades pódese traballar de maneira paralela nas dúas e despois chegado un momento pódese facer un *merge*, é dicir, unha unión das dúas, ou ben descartar unha en favor de outra.

3.3.3 Listas de Correo

As listas de correo son unha parte moi importante para o desenvolvemento e comunidade dentro do GNOME. Estas listas empréganse para discutir sobre as distintas aplicacións, problemas que xorden a hora de programar, distintas solucións ou novas funcionalidades que se queren incluír nun módulo ou simplemente para anunciar a nova versión do programa que se está a desenvolver. Dentro do proxecto GNOME existen listas de correo para tódolos módulos, ademais de listas específicas para os temas da tradución, anuncios de novas versións, organización interna, etc. No caso do *Gtranslator* existe unha lista de correo dentro do proxecto GNOME. Para poder enviar mensaxes é necesario estar subscrito. De non ser así a mensaxe pasará por un proceso previo de aprobación.

A subscrición a lista de correo do *Gtranslator* pódese facer a través da seguinte ligazón:
http://mail.gnome.org/mailman/listinfo/gtranslator-list.

Para enviar un correo a lista a dirección é: **gtranslator-list@gnome.org**

3.3.4 Wiki

Un *wiki* é un sitio web que ten páxinas que poden ser editadas. É moi útil para que os contidos podan ser modificados por por múltiples voluntarios empregando un navegador web. Dentro do proxecto GNOME, así como noutros proxectos de *Software Libre*, son moi empregados os *wikis* para tarefas de coordinación. Os equipos de tradución é habitual que empreguen os *wikis* para levar o control do estado de tradución dos diferentes proxectos.

No caso do *Gtranslator* empregase un *wiki* para anotar ideas sobre melloras futuras do programa, organizar as tarefas pendentes ou por exemplo para facer propostas para un posible cambio de nome do proxecto.

O *wiki* do *Gtranslator* pódese atopar na seguinte ligazón :
http://live.gnome.org/gtranslator

3.3.5 HIG: Human Interface Guidelines

As HIG (Human interface guidelines) son documentos que lles dan os desenvolvedores un conxunto de recomendacións para a construción das interfaces de usuario. Deste xeito conséguense interfaces máis intuitivas e cómodas de usar.

A meirande parte destas definen simplemente unha aparencia común para as aplicacións, centrándose nos usos en particular para un escritorio concreto. Definen políticas as veces baseadas en estudos da iteración persoa-ordenador, pero na maioría dos caos son convencións arbitrarias elixidas polos desenvolvedores da plataforma.

Dan unha serie de regras para o deseño, incluíndo deseño de iconas e estilos das xanelas. Soen definir tamén terminoloxía estándar de aplicación a certos elementos ou accións. Permiten crear un estándar visual para un sistema, no que as aplicacións teñan un aspecto similar e elementos comúns, como poden ser os botóns, as iconas ou por exemplo diálogos.

Algunhas das vantaxes de empregar estas regras son:

- Os usuarios aprenderán mías rápido o uso do programa, porque os elementos da interface terán un comportamento similar ós que xa usaron.
- A aplicación integrarase ben dentro do escritorio e co resto de programas.
- A interface da aplicación seguirá que véndose ben cando o usuario cambie o tema do escritorio.
- A aplicación será accesible a tódolos usuarios incluíndo aqueles con algunha discapacidade.

No proxecto GNOME defínese unha guía deste estilo chamada as GNOME User Interface Guidelines [6]. O longo deste proxecto empregouse esta guía para facer tódolos deseños das diferentes partes da interface, así como de tódolos diálogos.

Capítulo 4

Mantemento da versión actual

Este capítulo trata do mantemento da versión estable do *Gtranslator*. Este proceso é o punto de entrada ó proxecto e serve como toma de contacto para coñecer o seu estado actual.

Comézase o capítulo explicándoo os resultado obtidos ó facer un análise do código actual do programa para logo tratar de ver cales son os problemas máis importantes que presenta, os denominados bugs, como se xestionaron estes e as solucións aportadas para o seu arranxo.

4.1 Análise e documentación do código

Actualmente *Gtranslator* atópase na versión 1.1.7. Esta versión non aproveita todas as posibilidades das ferramentas que ofrece *GNU Gettext*. O seu código está cheo de funcións obsoletas nalgúns casos e noutros feitas *adhoc* porque non existía a correspondente en *GNU Gettext* nese momento. Neste apartado tratarase de analizar o código da versión actual. Por unha parte farase un análise das estruturas que se empregan e por outra analizarase o código a través dos ficheiros nos que se organiza.

4.1.1 Estruturas de datos

As estruturas de datos e as relacións principais entre elas que se empregan pódense ver no diagrama da figura 4.1.

A estrutura *GtrPo* define os ficheiros *PO*. Cando se abre un ficheiro *PO* co programa este crea unha instancia de esta estrutura que o representa. Como xa vimos, un ficheiro *PO* é unha combinación dunha serie de mensaxes e unha cabeceira. A estrutura *GtrHeader* define esta cabeceira establecéndose unha relación entre as estruturas *GtrPo* e *GtrHeader* xa que un ficheiro *PO* ten só unha cabeceira e esta pertence a un só ficheiro *PO*.

Cada unha das mensaxes do ficheiro *PO* represéntase mediante a estrutura *Gtrmsg*. Esta estrutura a súa vez ten unha relación coa estrutura *GtrComment*, que define os comentarios que levan asociados cada mensaxe. Estes comentarios son postos polo programador no caso de que necesite facer algunha apreciación que facilite a tarefa ó futuro tradutor. Unha mensaxe pode ter un comentario ou non e cada comentario está asociado a unha soa mensaxe.

Por último a estrutura *GtrTranslator* define ó tradutor. Utilízase para almacenar os datos do tradutor que formarán parte da cabeceira. En cada execución un tradutor pode ter asociada unha lingua na que está a traducir, para o cal se crea a estrutura *GtrLanguage* e unha relación entre elas.

4.1.2 Estrutura de ficheiros

A estrutura de ficheiros a podemos dividir entre ficheiros principais, aqueles que implementan as estruturas e as funcións principais, e ficheiros auxiliares que conteñen diversos aspectos de configuración ou que se empregan como apoio dalgunhas funcións.

Ficheiros principais

- **parse.h:** Implementa a estrutura *GtrPo* e contén as definicións das funcións para parsear o contido dos ficheiros *PO* no momento que se abren. Ademais

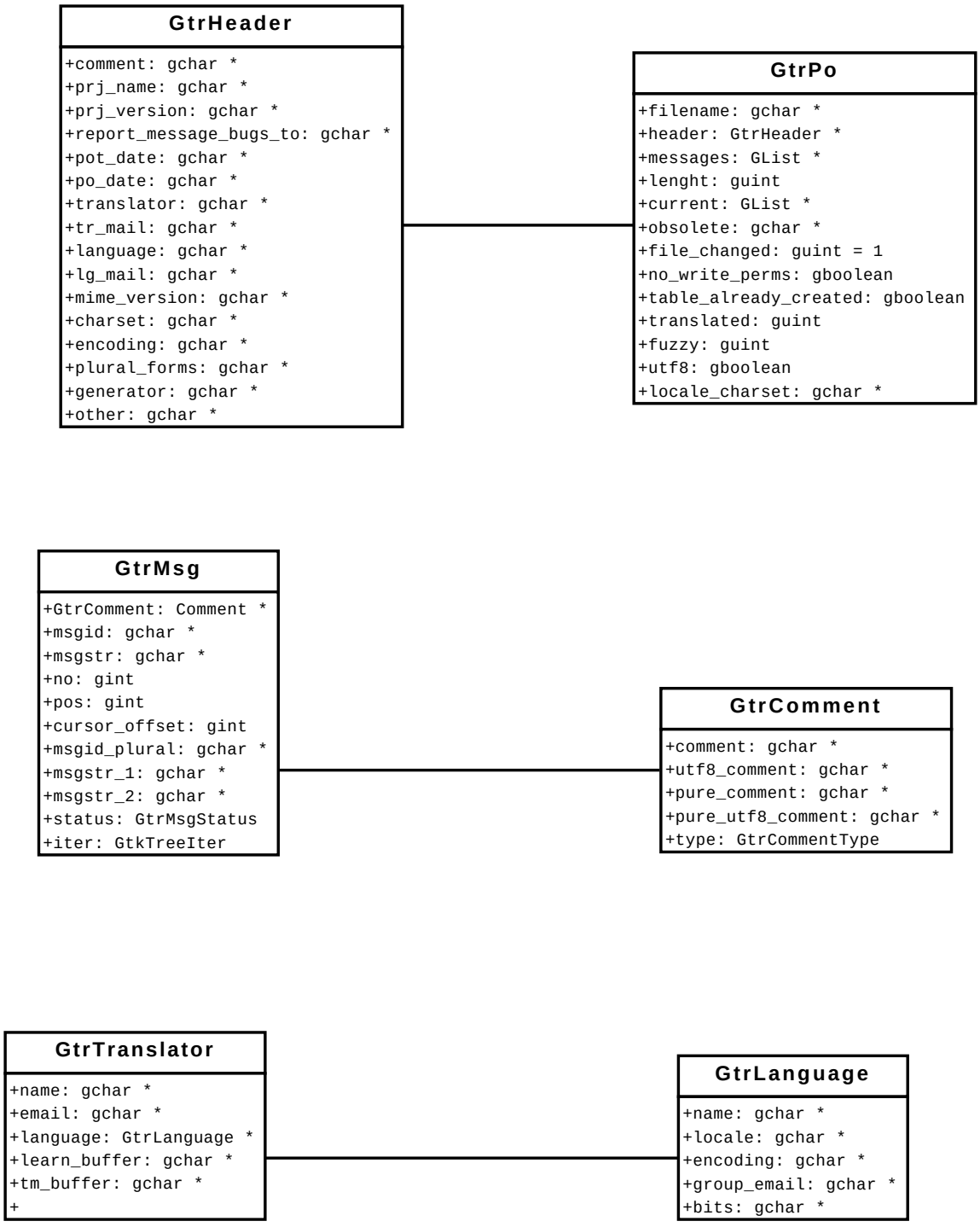

Figura 4.1: Diagrama de estruturas da versión estable

conten as definicións das funcións para gardar o ficheiro, actualizalo e compilalo. A implementación de todas as funcións realizase no ficheiro do mesmo nome con extensión .c.

- **header_stuff.h:** Implementa a estrutura *GtrHeader* e as definicións das funcións para traballar coa cabeceira. Entre estas funcións están as de ler o contido da cabeceira dende o ficheiro, escribila cando se garda, actualizar algún dos seus campos, etc. Neste ficheiro está tamén a definición da función que crea o cadro de diálogo que se mostra ó editar a cabeceira. Todas as funcións están implementadas no ficheiro header_stuff.c.

- **message.h:** Implementa a estrutura *GtrMsg* que define as mensaxes que hai dentro do ficheiro *PO*. Ademais contén as definicións das funcións que permiten desprazarse a través das mensaxes, actualizar o contido ou modificar o seu estado. A implementación atópase no ficheiro messages.c.

- **menús.c:** Este ficheiro define os elementos da interface gráfica, tanto os menús como as iconas das barras de ferramentas. Relaciona cada elemento co seu callback, é dicir, coa función que se executará ó pulsalo.

- **dialogs.h:** Neste ficheiro defínense as funcións que crea os diálogos para interatuar co usuario. A implementación está no ficheiro diálogos.c

- **prefs.h:** Define a estrutura de preferencias de *Gtranslator*. Contén as cabeceiras das funcións que crean o diálogo de opcións, len os valores de configuración e inicializan ditas opcións cos valores por defecto. As súas implementacións atópanse no ficheiro prefs.c

- **preferences.h:** Encapsula o acceso ó sistema de configuración *Gconf*. Define as funcións que lee e escreben os valores das claves nas preferencias.

- **gui.h:** Este ficheiro abarca todo o necesario para crear a ventá principal do programa. Contén as cabeceiras das funcións que se empregarán como callbacks

para os botóns da ventá así como para os botóns das operacións con texto, cortar, pegar, copiar, etc... Ademais crea unha pequena estrutura que define o *widget* onde se mostran os comentarios asociados a unha determinada mensaxe. As implementacións atópanse no ficheiro gui.c

Ficheiros auxiliares

- **color-schemes.h:** Define unha estrutura para os esquemas de cores da interface. Contén as cabeceiras das funcións para o manexo destes esquemas. A implementación desas funcións está no color-schemes.c.

- **comment.h:** Neste ficheiro defínese unha estrutura para os comentarios.

- **dnd.h:** Contén a función principal para abrir ficheiros por medio da técnica "drag and drop".

- **history.h:** Este ficheiro define a estrutura *GtrHistoryEntry* que se emprega para crear unha lista dos ficheiros abertos recentemente. Ademais contén as definicións de dúas funcións, unha para engadir ficheiros á lista e a outra para obtela. A implementación está no ficheiro history.c

- **languages.h:** Define a estrutura *GtrLanguages* que recolle toda a información dos idiomas os que se traduce. No ficheiro languages.c atópase unha lista de linguas cos seus correspondentes valores para os membros da estrutura *GtrLanguages*.

- **learn.c:** Contén a estrutura *GtrLearnBuffer*. O LearnBuffer é un ficheiro *XML* que se emprega para almacenar as traducións que se van facendo de maneira que poidan ser reempregadas. Este ficheiro conten a implementación das funcións que almacena as traducións no Learbuffer e as que obteñen esa información para levar a cabo a función de autotradución.

- **messages-table.h:** Crea o *widget* que mostra as mensaxes que contén o ficheiro *PO* agrupándoas en tres categorías: traducidas, sen traducir e dubidosas. Este *widget* móstrase na ventá principal de *Gtranslator* durante a súa execución. A implementación das funcións está no ficheiro message_table.c xunto coas estruturas *GtrMessagesTable* que define a táboa cos súas tres categorías e *GtrMessagesTableColors* que contén as cores que representan cada categoría.

- **nautilus-string-h:** Todas as funcións que se empregan para traballar con cadeas. Cóllense do módulo nautilus, o administrador de ficheiros de GNOME.

- **open.h:** Contén a función principal para abrir o ficheiro. En open.c está a súa implementación. Emprégase unha función principal que chama a outras funcións auxiliares segundo o tipo de ficheiro que se quere abrir: ficheiro *PO* normal, compilado, comprimido...

- **replace.h:** Neste ficheiro defínese a estrutura *GtrReplace* que contén a información necesaria para facer un reemprazo dunha cadea. A implementación faise no ficheiro replace.c. Empréganse as funcións de find.h para facer a búsqueda da cadea que se quere reemprazar.

- **runtime-config.h:** Contén a estrutura *GtrRuntimeConfig* que contén a información necesaria en tempo de execución.

- **save.h:** Igual que open.h, contén unha función principal para gardar o ficheiro actual. En save.c está a implementación. Tamén contén unha función para cada tipo de ficheiro que se quere gardar.

- **session.h:** Contén as funcións que se chamarán no caso de que se peche a sesión que se está a executar, no caso de que se faga un sleep ou un restore despois do sleep.

- **sighandling.h:** Contén a función principal que captura calquera sinal que poida facer perder o traballo do tradutor.
- **stylistics.h:** Define unha estrutura de tipo enumerado chamada ColorType que contén os elementos da interface que se van mostrar para chamar a atención do usuario coma por exemplo os caracteres especiais, números, palabras clave, et...
- **translator.h:** Define a estrutura *GtrTranslator*, que conten información sobre o tradutor. Tamén contén as funcións que permiten establecer a lingua para o tradutor e gardar os valores dos membros da estrutura nas preferencias.
- **undo.h:** Contén as funcións que se encargan de facer a acción de Desfacer cambios. Unhas funcións para comprobar se se fixeron os cambios e outra para volver ó estado anterior.
- **update.c:** Contén a función de actualización da interface de usuario, baseada no script my-update.sh que se encontra no directorio "data".
- **utils.h:** Neste ficheiro reúnense unha serie de funcións auxiliares usadas para a implementación doutras funcións. A implementación está no ficheiro utild.c
- **utils_gui.h:** Funcións auxiliares que traballan coa interface gráfica.
- **vfs-handled:** Contén a función para abrir ficheiros a través dun sistema de ficheiros virtuais.

Dependencias

Para analizar as dependencias que teñen uns ficheiros de outros faise uso dunha serie de diagramas que mostran as dependencias entre as funcións e os ficheiros para as operacións máis habituais que existen no *Gtranslator*. O principal para empezar a traballar é abrir un ficheiro, o diagrama correspondente a operación de abrir un ficheiro pódese ver na figura 4.2

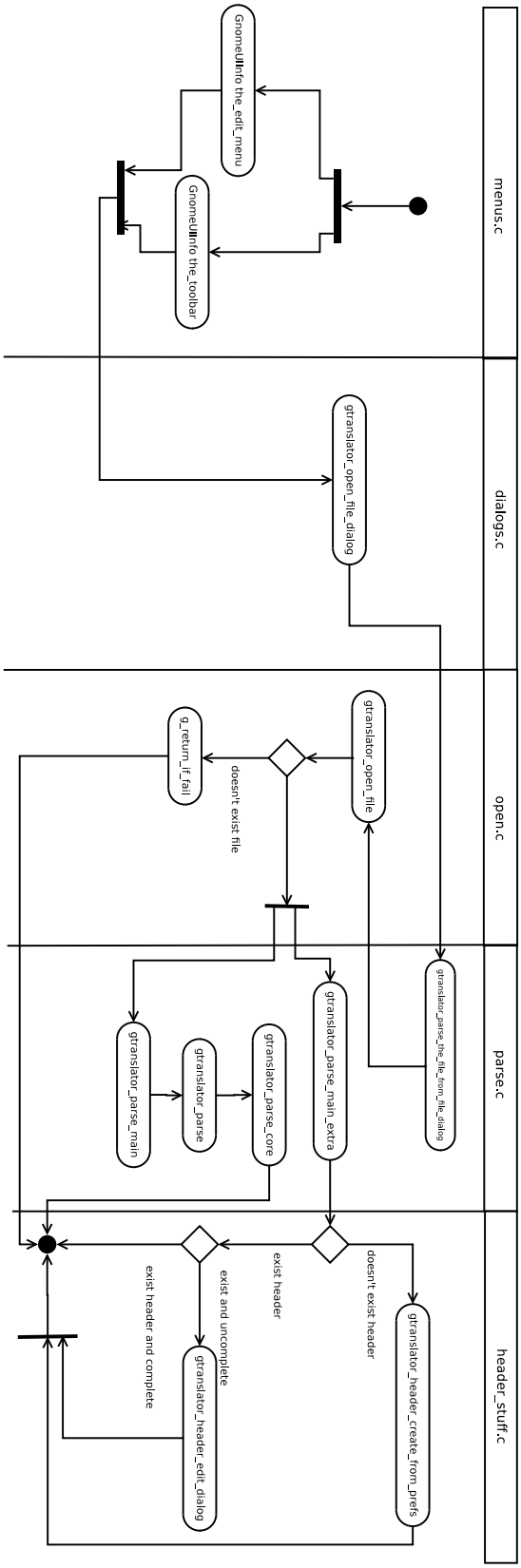

Figura 4.2: Diagrama da operación Abrir Ficheiro

Cando se inicia a operación abrir ficheiro, sexa dende o menú ficheiro ou dende o botón da barra de ferramentas, no ficheiro menus.c chámase a función *gtranslator_open_file_dialog()* que mostra o cadro de diálogo abrir ficheiro para que o usuario seleccione o ficheiro. Esta función chama a *gtranslator_parse_the_file_from_open_dialog()* que se atopa no ficheiro parse.c e ésta encárgase de comprobar se o ficheiro xa está aberto, sé non é así chama a *gtranslator_open_file()* e pecha o cadro de diálogo.

A función *gtranslator_open_file()* atópase no ficheiro open.c e encárgase de comprobar se o ficheiro que se quere abrir existe, se non é así devolve un erro. Se o ficheiro existe chama a *gtranslator_parse_main()* e *gtranslator_parse_main_extra()*, ambas funcións están no ficheiro parse.c. A función *gtranslator_parse_main_extra()* encárgase de comprobar se no ficheiro que se abre existe cabeceira ou non. se non é así chámase a función *gtranslator_header_create_from_preferences()* que se atopa no ficheiro header_stuff.c.Se a cabeceira xa existe pero está incomplenta chámase a función *gtranslator_header_edit_dialog()* que crea un cadro de diálogo para que o usuario complete a cabeceira manualmente. Por outra banda a función *gtranslator_parse_main()* enche os datos dos membros da estrutura *GtrPo* por medio dunha chamada a función *gtranslator_parse()*. Esta función devolve un punteiro a unha estrutura de tipo *GtrPo* a partir dun nome de ficheiro chamando a función *gtranslator_parse_core()* que é quen se encarga de parsear o ficheiro liña a liña almacenando tódalas mensaxes.

En calquera momento da execución do programa pódese necesitar editar a cabeceira manualmente, o proceso podémolo analizar por medio do diagrama da figura 4.3

A través do ficheiro menus.c enlázanse as chamadas tanto dende o menú Editar como dende o botón Header na barra de ferramentas, coa función *gtranslator_header_edit_dialog()* do ficheiro header_stuff.c. Esta función crea o cadro de diálogo que se mostra ó usuario para editar os valores da cabeceira manualmente.

Mediante a función *take_my_options_toggled()*, que se atopa no mesmo ficheiro,

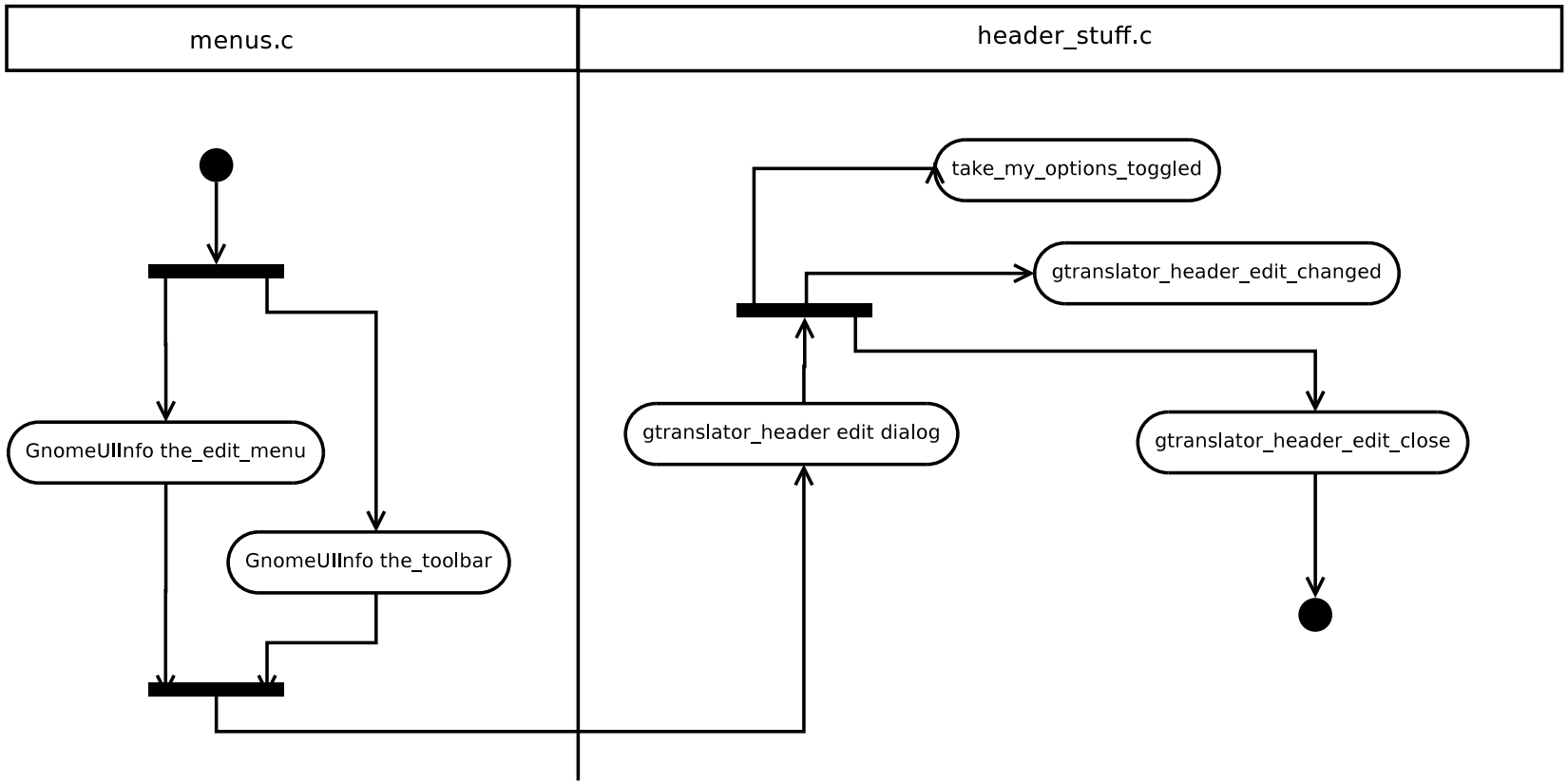

Figura 4.3: Diagrama da operación Editar Cabeceira

compróbase se está activado o botón "usar as miñas funcións para completar as seguintes entradas", se é así, os datos de información sobre o tradutor da cabeceira cóllense das preferencias do usuario. Se edita calquera dos cadros de entrada de texto onde están os valores da cabeceira chámase á función *gtranslator_header_edit_changed()* que se atopa tamén no ficheiro header_stuff.c. Esta función pon a variable global de tipo boolean *header_changed* a TRUE para que cando se peche o cadro de diálogo se actualicen os datos. Esta actualización faina a función *gtranslator_header_edit_close()* que se chama cando se pulsa o botón close. O traballo desta función é actualizar os valores da cabeceira que foran modificados. A función *gtranslator_header_edit_close()* tamén se encarga de destruír o cadro de diálogo.

Outra función que paga a pena analizar é a da búsqueda, xa que debido ó formato dos ficheiros *PO* esta función que pode ser máis ou menos sinxela cando se realiza sobre ficheiros de texto plano, aquí non o é tanto. O diagrama de secuencia correspondente pódese ver na figura 4.4

Dende o ficheiro menus.c enlázanse as chamadas á función buscar tanto dende o menú Editar coma dende a barra de ferramentas coa función *gtranslator_find_dialog()* que se atopa no ficheiro dialogs.c. Dende esta función créase o cadro de diálogo da búsqueda. Compróbase se se introduce unha cadea no ficheiro dialogs.c. Compróbase se se introduce unha cadea de texto no cadro de texto correspondente para realizar á búsqueda, se non é así móstrase unha mensaxe de erro. Se todo é correcto chama a función *gtranslator_find()* que se atopa no ficheiro find.c. Esta función é a principal na búsqueda. Chama a función *Repeat_all()* e comproba o valor que devolve, se é FALSE significa que non se atopou a cadea e polo tanto mostra unha mensaxe de erro por pantalla. No caso de que o valor sexa TRUE faise un *return*. Esto é así porque a función *Repeat_all()* recibe como parámetro outra función: *find_in_message()* que é quen realmente fai a búsqueda na lista de mensaxes do ficheiro. Esta función devolve un 1 se se atopa o string, un 0 se non e un -1 se houbo un erro. No caso de que se atope a cadea chámase tamén a función *gtranslator_message_go_to()* do ficheiro

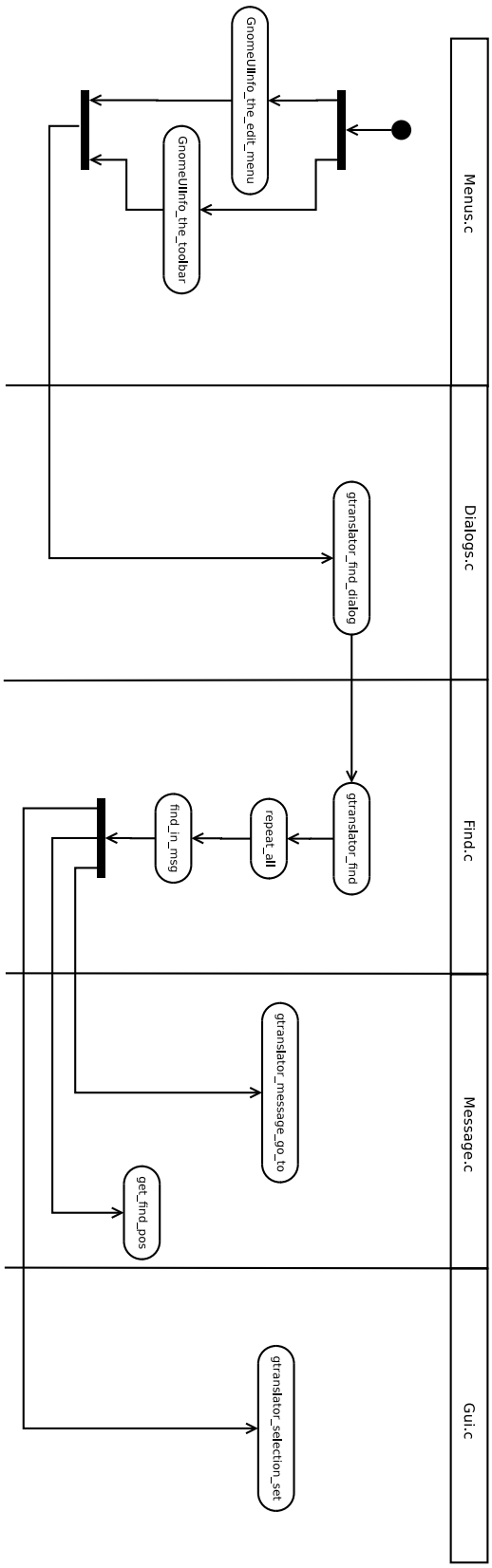

Figura 4.4: Diagrama da operación Buscar

messages.c que mostra a mensaxe onde se atopou a cadea buscada na pantalla e a función *gtranslator_selection_set* do ficheiro gui.d que resalta a cadea buscada dentro da mensaxe.

En canto ás opcións de gardar comezamos analizando a de Gardar Como, o seu diagrama de secuencia pódese ver na figurar 4.5

A función *gtranslator_save_file_as_dialog()* crea o cadro de diálogo de Gardar Como. Esta función que se atopa no ficheiro dialogs.c chámase dende o ficheiro menus.c a través das funcións que se enlazan coa interface dende o menú Ficheiro e dende o botón da barra de ferramentas. Cando se pulsa o botón Aceptar chámase a función *gtranslator_save_file_dialog()* que está no ficheiro parse.c. Esta función obtén o nome co que se quere gardar o ficheiro e chama a función *gtranslator_save_file()* que se atopa no mesmo ficheiro e que realiza todo o traballo para gardar o ficheiro. Primeiro comproba se a extensión co que se quere gardar é a correcta mostrando unha mensaxe por pantalla. Despois chama a función *gtranslator_save_po_file()* que comproba se o arquivo que se quere gardar ten algunha extensión especial, se non continua comprobando se hai permisos para escribir no ficheiro que se quere gardar, en caso de non sexa así mostra un erro por pantalla. Por último escrebe tódalas mensaxes no ficheiro mediante a función *write_the_message()* e os datos da cabeceira mediante a función *gtranslator_header_put()*.

Na operación Gardar, o diagrama de secuencia pódese ver na figura 4.6, empézase chamando a función *gtranslator_save_current_file_dialog()*, que se atopa no ficheiro parse.c. Esta función o que fai é comprobar se houbo algún cambio no ficheiro, se foi así chama a función *gtranslator_save_file()*, que xa vimos na operación Gardar Como.

4.2 Análise e solución de bugs

Todo o análise da sección anterior serve non só para coñecer o funcionamento interno do programa, a súa implementación e como está organizado o código nos ficheiros,

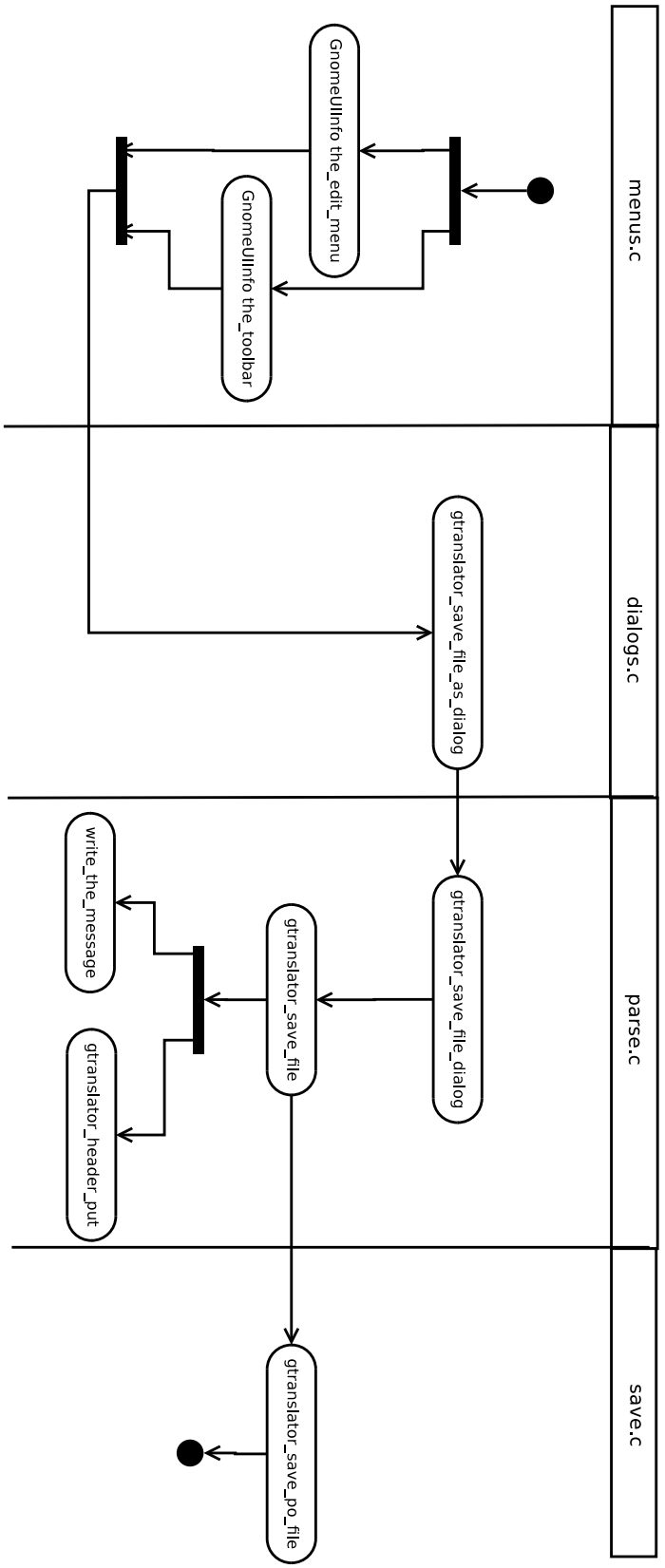

Figura 4.5: Diagrama da operación Gardar Como

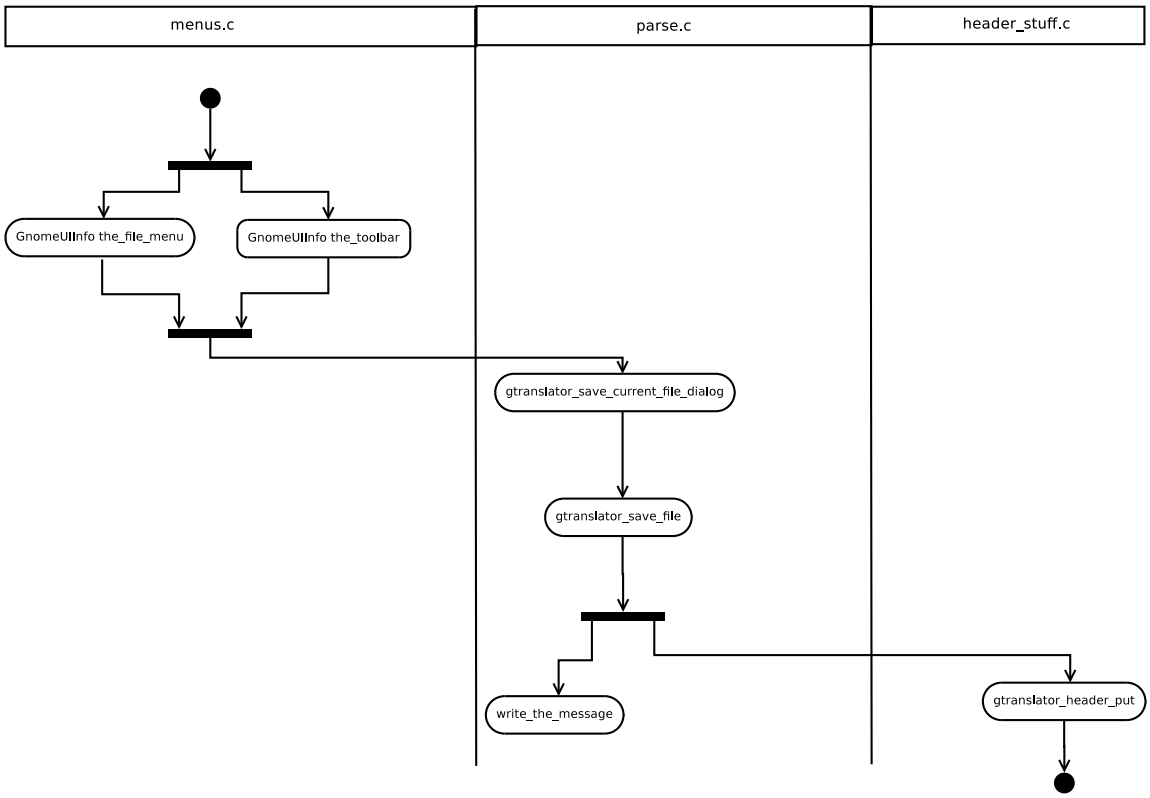

Figura 4.6: Diagrama da operación Gardar

senón que ademais é un bo método de analizar os posibles fallos que poida ter e as partes que carecen da robustez necesaria para que o programa sexa suficientemente estable. Isto xunto coa aportación da comunidade en canto a reporte de erros no programa fará máis doado identificar cales son os puntos onde fixar a atención nos seguintes apartados para o mantemento da versión estable.

4.2.1 Análise de bugs

Como xa vimos no apartado 3.3 para xestionar os bugs úsase o Bugzilla. Botando unha ollada a esta ferramenta é co visto despois do análise exhaustivo do código pódense sacar conclusións sobre cales son os bugs máis importantes da ferramenta e polo tanto os que se necesitan arranxar para que a versión estable sexa máis usable e poda serve coma ponte ata rematar o desenvolvemento da nova versión.

Estes son os bugs que se identificaron como máis importantes:

- **Bug 129850:** *Po files are sometimes generated wrongly*

  Este bug ven motivado polo forma de gardar os ficheiros no *Gtranslator*. O que se fai é volver a escribir tódalas mensaxes. O formato que teñen as mensaxes como xa vimos no capítulo 2 é o seguinte:

  ```
  #: ../apt-setup:212
  msgid ""
  "You probably used a CD to install the system, but it is not "
  "currently in the drive. You should insert it and select \\"
  "\"cdrom\\\".""
  msgstr ""
  ```

  Cada liña de texto ten que comezar e rematar con ". A función que garda o ficheiro encárgase de escribilas. Neste exemplo concreto hai unha \ xusto antes das cominllas polo que o parsear o texto pensa que están escapadas e engade outras a maiores para pechar a liña quedando a mensaxe gardada:

```
#: ../apt-setup:212
msgid ""
"You probably used a CD to install system, but it is not "
"currently in the drive. You should insert it and select \\""
"\"cdrom\\\"."
msgstr ""
[more text here]
```

O cal é incorrecto.

- **Bug 323685:** *gtranslator does not update Last-Translator header*

  Neste caso o problema é que o *Gtranslator* non actualiza de maneira correcta o nome do último tradutor que modificou o ficheiro. O campo Last-Translator é un campo da cabeceira do ficheiro *PO* no que se debe mostrar o nome da última persoa que fixo cambios no ficheiro. Esta información débese coller das preferencias do programa e comprobar automaticamente se o nome do tradutor coincide co do campo Last-Translator, se non é así débese cambiar de forma transparente ó usuario.

- **Bug 392323:** Gtranslator *removes all custom headers*

  Hai varios programas de edición de ficheiros *PO* que escreben campos específicos na cabeceira dos ficheiros *PO* para gardar certa información específica para o seu funcionamento. Estes campos no son parte do estándar da cabeceira. O *Gtranslator* cada vez que garda un ficheiro reescribe a cabeceira seguindo o estándar polo que calquera de estes campos son eliminados ou mellor dito escrébense de novo.

- **Bug 414922:** Gtranslator *1.1.6 crashes if I enter \ in search*

  Normalmente a barra invertida \ emprégase para escapar caracteres, é dicir cando se quere facer referencia a un carácter reservado nunha linguaxe por

exemplo emprégase a barra invertida para indicar que se escriba ese carácter pero non se procesa. Cando se fan búsquedas no *Gtranslator* a expresión que se escrebe no diálogo de búsqueda transformase internamente nunha expresión regular polo que se facemos unha búsqueda na que o que queremos buscar é precisamente a barra invertida o programa tratará coma o carácter especial de escapado e non como un simple carácter.

- **Bug 423274:** *find/search does not normalize*

  Este é un problema coa codificación dos caracteres. Internamente o *Gtranslator* manexa tódalas cadeas en formato de codificación UTF-8. Se o usuario non ten o seu sistema configurado neste formato cando introduza unha cadea para buscar no diálogo de búsqueda esa cadea non está en UTF-8 polo que a busca non funcionará correctamente.

- **Bug 423276:** *save as should warn for name conflict*

  Este é un bug crítico. No apartado anterior vimos como é o proceso de Gardar Como. Vimos que se facían comprobacións sobre a extensión do ficheiro que se quería gardar pero non se comprobaba en ningún momento se o ficheiro xa existe e polo tanto non se mostra ningún aviso ó usuario para saber se realmente quere sobrescribir o ficheiro ou non. Simplemente se o nome do ficheiro xa existía o que se fai e machacalo coa conseguinte perda de información.

- **Bug 488835:** *msgctxt support*

  O campo *msgctxt* é unha etiqueta dos ficheiros *PO* que se incorporou nas últimas versións de *GNU Gettext*. Esta etiqueta permite o desenvolvedor especificar un contexto no que está encadrada a cadea orixinal que se debe traducir. No momento de escribirse o programa non existía e como o parseo dos ficheiros está feito totalmente *adhoc* o atopar unha etiqueta estraña, o parseo falla e o que é peor o programa sofre un *crash*.

4.2.2 Contacto ca comunidade

Nun proxecto de *Software Libre* a comunidade é a palabra clave. Un proxecto de *Software Libre* non se pode manter se non hai unha comunidade detrás tanto de desenvolvedores coma de usuarios. No caso dos bugs a comunidade de usuarios resulta de vital importancia. É necesario que eles fagan unha labor de testeo do programa que basicamente consiste no se uso normal. Cando detecten un erro ou un comportamento estraño é moi importante que o reporten a través dunha ferramenta de seguimento de erros coma o *Bugzilla* dentro do proxecto GNOME.

No caso deste proxecto foi moi importante o uso do *Bugzilla* non só para detectar os bugs do programa e melloralos senón tamén como punto de contacto cos usuarios do mesmo. Grazas a posibilidade que ofrece *Bugzilla* de engadir comentarios nun bug púidose interactuar cos usuarios que reportaban os bugs para nuns casos solicitar máis información sobre o bug e noutros para entrar nunha discusión sobre a mellor solución dende o punto de vista do usuario que se podía aportar para amañar o bug.

Na seguinte captura pódese ver un dos bugs comentados no apartado anterior. Como se pode ver a estrutura é similar a dun foro no que se van poñendo comentarios sobre o bug. Neste caso pedíaselle a persoa que reportou o bug un ficheiro concreto que fallase para poder probar o comportamento do programa nese caso.

4.2.3 Solucións aportadas

Neste apartado presentaranse as solucións aportadas para solucionar os bugs comentados no apartado 4.2.1.

Bug 129850: Po files are sometimes generated wrongly

Seguindo o análise do apartado 4.1.2 a escritura das mensaxes no ficheiro cando se garda está no ficheiro parse.c. Mirando o código vemos que existe unha función *append_line* que se encarga de ir engadindo as liñas ó ficheiro. No seguinte trozo de código vemos que sí se comproba o caso de que exista unha \ e despois unha ", pero

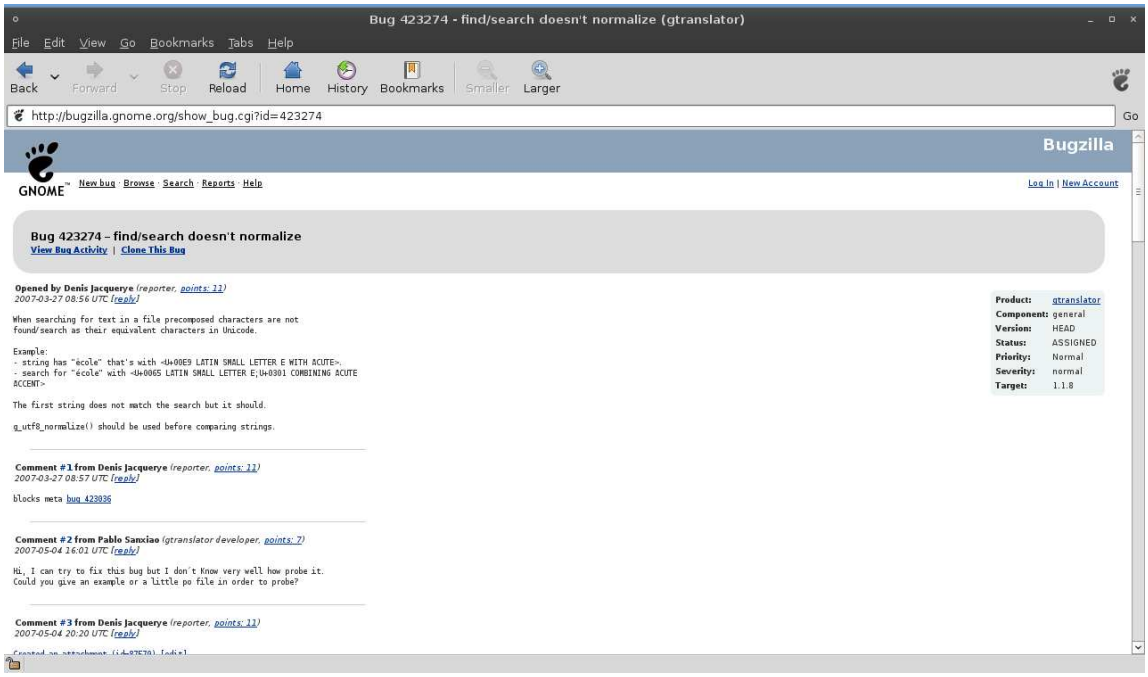

Figura 4.7: Captura dun bug no Bugzilla de GNOME

non se comproba se esa "é en realidade a que marca o fin da liña no lugar de ser un carácter escapado.

```
for (s = 1; s < strlen(tail) - 1; s++) {
  if(tail[s] == '\\') {
    switch(tail[s + 1]) {
    case '\"';
    s++;
    break;
    }
  }
  to_add[d++] = tail[s];
```

A solución é comprobar se a seguinte posición do array despois da "é o final do array o que indicaría que esa "é en realidade a que pecha a liña e non está escapada.

```
for (s = 1; s < strlen(tail) - 1; s++) {
```

```
if(tail[s] == '\\') {
  switch(tail[s + 1]) {
  case '\"':
    if (s+1 != strlen(tail)-1)
      s++;
    break;
  }
}
to_add[d++] = tail[s];
```

Bug 323685: gtranslator does not update Last-Translator header

Neste caso a solución será almacenar nunha variable o valor do campo Last-Translator da cabeceira do ficheiro e facer dúas comprobacións: por un lado ver se está activado o botón "take my options" que indica que o usuario quere que se enchan os campos da cabeceira coa información que previamente introduciu nas preferencias do programa, se é así haberá que comprobar se o valor do campo Translator nas preferencias coincide co valor do campo Last-Translator na cabeceira, senón coinciden haberá que modificalo. Todo este proceso faise no momento de parsear o ficheiro pois é nese momento cando se eche a estrutura header que se almacena en memoria. Traducida a código a solución é como segue:

```
/*
 * If toggle button "take_my_options" is marked then the details
 * about translator must be read from the preferences to update
 * the Last Translator when users save the file.
 */

if (GtrPreferences.fill_header)
  {
```

```
old_translator = ph->translator;
ph->translator = gtranslator_translator->name;

old_tr_email = ph->tr_email;
ph->tr_email = gtranslator_translator->email;

GTR_FREE(ph->language);
ph->language = gtranslator_translator->language->name;
GTR_FREE(ph->lg_email);
ph->lg_email = gtranslator_translator->language->group_email;
GTR_FREE(ph->charset);
ph->charset = gtranslator_translator->language->encoding;
GTR_FREE(ph->encoding);
ph->encoding = gtranslator_translator->language->bits;

if (old_translator && old_tr_email && ph->translator &&
   nautilus_strcasecmp(ph->translator, old_translator) &&
   nautilus_strcasecmp(ph->comment, old_translator))
   {
    gchar   *prev_header_comment;
    gchar   *year;

    prev_header_comment = ph->comment;

    year=get_current_year();

    ph->comment=g_strdup_printf("%s# %s <%s>, %s.\n",
```

```
                                        prev_header_comment,
                                        old_translator,
                                        old_tr_email,
                                        year);

        GTR_FREE(year);
        GTR_FREE(prev_header_comment);
       }
    }
```

E necesario prestar especial atención a liberación da memoria reservada para os punteiros das cadeas de texto para non ter problemas de rendemento por esgotamento de memoria.

Bug 392323: *Gtranslator* removes all custom headers

A solución para este bug consistiu en engadir un campo a maiores a estrutura *GtrHeader* que como se veu no apartado 4.1.2 é a que ten toda a información sobre os campos da cabeceira. Neste novo campo recóllese calquera liña que outras aplicacións podan engadir á cabeceira do *PO*. Con isto ó gardar o ficheiro, cando se volvan escribir os campos da cabeceira deixarase de novo ese campo como estaba para que aínda que o *Gtranslator* non faga uso del, se se volve abrir o ficheiro coa aplicación que o engadiu siga funcionando.

Engadirase un membro a estrutura *GtrHeader* chamado "other" e que será de tipo array de caracteres. A escritura da cabeceira no ficheiro *PO* faise coma se fose unha única cadea de texto, polo que se existe este campo engadirase ó final da cadea antes de escribila no ficheiro:

```
/*
 * If exist other headers that other aplication had written in the po
```

```
  * file, we save it in the new po file.
  */
  if(h->other)
    {
      other_line = g_strdup_printf("\n\%s", h->other);
      prev_msgstr = msg->msgstr;
      msg->msgstr = g_strconcat(prev_msgstr, other_line, "\\n", NULL);
      GTR_FREE(other_line);
      GTR_FREE(prev_msgstr);
    }
```

Bug 414922: *Gtranslator* 1.1.6 crashes if I enter \ in search

Neste caso a solución pasa por escapar a \ na cadea que se introduce no diálogo da busca. Como non existe ningunha función para facer isto dentro da *API* de desenvolvemento de GNOME decidiuse escribir unha.

```
/*
 * Return the same string but with '\' escaped if exits
 */

gchar * gtranslator_utils_escape(const gchar *string)
{
  gchar *result;
  GString *aux;
  gint s;

  aux = g_string_sized_new(strlen(string));

  for(s = 0; s < strlen(string); s++)
```

```
    {
      if (string[s] == '\\')
        {
         g_string_append(aux, "\\\\");
        }else {
        g_string_append_c(aux, string[s]);
      }
    }
  result = aux->str;
  g_string_free(aux, FALSE);
  return result;
}
```

Bug 423274: find/search does not normalize

Para solucionar este problema é necesario converter as cadeas que se escreben no diálogo da busca ó formato UTF-8 que é o que usa o programa internamente.

UTF-8 é unha norma de transmisión de lonxitude variable para caracteres codificados empregando *unicode*. Emprega grupos de bytes para representar o estándar de *unicode* para os alfabetos de moitas linguas do mundo. Emprega entre 1 e 4 byes por carácter, dependendo do símbolo de *unicode*. Por exemplo, necesítase un só byte en UTF-8 para codifica-los 128 caracteres *ASCII* no rango U+000 a U+007F de *unicode*.

En resumo, os bits de carácter *unicode* son divididos en varios grupos, os cales son despois divididos entre as posicións máis baixas dentro dos bytes UTF-8. Os caracteres máis pequenos son codificados cun byte sinxelo que contén o seu valor, correspondendo exactamente cos caracteres de 7-bit dos 128 de *ASCII*. Nos demais casos empréganse de 2 a 4 bytes. O bit máis significativo de todos os bytes da cadea é sempre 1.

Na *API* de desenvolvemento de GNOME ofrecesen algunhas funcións que facilitan esta conversión. A función que se empregou para a conversión foi *g_utf8_normalize* a que se lle pasa a cadea para normalizar e o tipo de normalización desexada. Temos que facer a conversión no ficheiro dialogs.c que é onde se recolle a cadea insertada no diálogo da busca, e no ficheiro parse.c que é onde se lee a cadea do ficheiro. En ámbolos casos a transformación é como segue:

```
find_text_normalized = g_utf8_normalize(find_text,
                                        -1,
                                        G_NORMALIZE_DEFAULT_COMPOSE);
GTR_FREE(find_text);
find_text = find_text_normalized;
```

Bug 423276: save as should warn for name conflict

Como se veu no apartado 4.2.1, este é un bug crítico porque pode supoñer perda de información ó gardar un ficheiro cun nome de outra que xa existe e sobrescribir o primeiro sen preguntarlle ó usuario.

O primeiro paso da solución proposta consiste en facer unha comprobación cando o usuario realiza a acción de Gardar Como de se o nome co que se quere gardar xa existe no sistema de ficheiros e se é así mostrar un diálogo ó usuario para avisándoo da situación e permitíndolle elixir que decisión tomar.

No ficheiro parse.c, na función *gtranslator_save_file_dialog()* insertouse o seguinte código:

```
if (g_file_test(po_file, G_FILE_TEST_EXISTS))
  {
    po->filename = g_strdup(po_file);
    po->filename = g_strdup(po_file);
```

```
    GtkWidget *dialog, *button, *button1;

    dialog = gtk_message_dialog_new (NULL,
                                     GTK_DIALOG_MODAL,
                                     GTK_MESSAGE_QUESTION,
                                     GTK_BUTTONS_CANCEL,
                                     _("The file '%s' already exists,
                                       Do you want overwrite it?"),
                                      po_file);

    button = gtk_dialog_add_button (GTK_DIALOG (dialog),
                                    "Overwrite",
                                    1);

    g_signal_connect (G_OBJECT (button), "clicked",
                      G_CALLBACK (gtranslator_overwrite_file), NULL);

    gtk_dialog_run (GTK_DIALOG (dialog));
    gtk_widget_destroy (dialog);
  }
```

A función *g_file_test* comproba se o nome que se lle quere dar xa existe no sistema de ficheiros, polo tanto se esa comprobación é certa o que se fai é crear un diálogo que se lle mostra o usuario e que ten esta pinta:

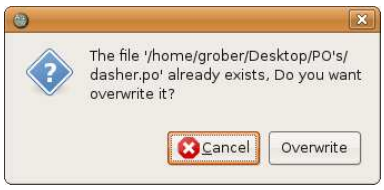

Figura 4.8: Diálogo de aviso cando o nome do ficheiro xa existe

O usuario pode cancelar para escribir un nome do diferente ou pode pinchar sobre o botón sobrescribir se realmente é o que quere facer e o ficheiro gravarase no disco con ese nome sobrescribindo a información do anterior.

Bug 488835: msgctxt support

Neste caso o bug ven motivado pola inclusión dunha nova etiqueta que fornece *GNU Gettext*. O sistema de parseo do ficheiro como xa falamos no apartado 4.2.1 é totalmente *adhoc* co cal a inclusión desta nova etiqueta nos arquivos fai que o parser do *Gtranslator* falle o atopar algo que non espera.

A estrutura que manexa as mensaxes do ficheiro como vimos no apartado 4.1.2 é *GtrHeader*. A solución consiste en engadir un novo membro a esa estrutura de tipo array de caracteres e chamado *msgctxt*.

Despois no parser haberá que engadir unha comprobación para ver se existe a etiqueta no ficheiro e se é así almacenala na estrutura da mensaxe.

```
/*
 * Check if exists msgctxt line. If that's so it'll be written it.
 */

if(msg->msgctxt)
  {
    string = g_string_append(string, "msgctxt \"");
    id = restore_msg(msg->msgctxt);
    string = g_string_append(string, id);
    string = g_string_append(string, "\"\n");
    GTR_FREE(id);
  }
```

4.2.4 Consideracións finais

Todas estas solucións que se aportaron para resolver os bugs foron subidas ó *Bugzilla* en forma de parches para que os usuarios os puideran aplicar e comprobar que funcionan como era de esperar e dean o seu consentimento para dar por pechado o bug.

Despois de ter todos estes bugs arranxados plantéxase a posibilidade de sacar unha nova versión de mantemento do programa. Postos en contacto co actual mantedor do proxecto coméntaselle esa posibilidade ademais da idea de traballar na nova versión 2.0 da aplicación. Esta persoa toma a decisión de traspasar o mantemento do programa ó director e o autor deste proxecto.

Neste punto, pásase de colaborar co proxecto a administralo. A primeira actuación será a de crear unha rama no sistema de xestión de versións (*Subversion*, ver sección 3.3) para a nova versión de mantemento etiquetada como a versión 1.1.8 do programa e aplicar nela tódalas solucións propostas para os bugs arranxados. Tómase a decisión de que esta versión sexa a derradeira da versión 1 do programa e céntrase o traballo a partir de aquí no desenvolvemento da versión 2.0.

Capítulo 5

Desenvolvemento da nova versión

Este capítulo céntrase no traballo de desenvolvemento da nova versión do *Gtranslator*. Faise un análise previo de cales son as actuacións máis importantes cara o redeseño da aplicación tanto na parte gráfica como na parte máis técnica, é dicir no código. Explícanse as decisións de deseño que se tomaron e a súa implementación. Ademais explícanse cales foron as novas funcionalidades que se engadiron a esta nova versión. Finalízase o capítulo co apartado referente as probas e o xeito de levalas a cabo neste tipo de proxectos de *Software Libre*.

5.1 Análise

O primeiro paso para comezar a pensar nas actuacións necesarias para o desenvolvemento da nova versión será facer un análise tomando como referencia a versión actual vendo cales son as súas principais eivas e cales son as partes onde é máis necesario facer os primeiros esforzos de mellora.

5.1.1 Problemas actuais

Algúns dos principais problemas que ten o *Gtranslator* puxéronse xa de manifesto no capitulo 4. Despois de repasar a análise feita anteriormente algún dos maiores

problemas detectados foron:

- **Codificación *adhoc*:** A función principal do programa que, como xa vimos, é a edición de ficheiros *PO*, implica que hai que facer un tratamento destes ficheiros. É necesario parsear o ficheiro cando se abre e obter tódalas súas mensaxes así como a cabeceira. No momento de gardar o ficheiro é necesario escribir de novo os cambios que se fixeron no ficheiro. Toda esta parte básica do programa está feita con funcións *adhoc* que esperan que os ficheiros *PO* teñan un formato concreto e se non é así dará problemas ou non poderá abrir o ficheiro. Actualmente *GNU Gettext* ofrece ademais das ferramentas unha biblioteca chamada *libgettextpo* que ofrece unha *API* concreta para o desenvolvemento de aplicacións que traballen con ficheiros *PO*.

- **Código moi desactualizado:** Seguindo un pouco co apartado anterior a estrutura do código está lonxe de seguir unha metodoloxía ou paradigma de programación o que fai moi complicado o seu mantemento. Está escrito en C usando unha biblioteca desenvolvida polo proxecto GNOME chamada *Glib* (ver apéndice B). As versións actuais de esta biblioteca permiten o emprego de orientación a obxectos en C por medio doutra biblioteca chamada *GObject* e que tenta simular o emprego de obxectos nunha linguaxe de baixo nivel como C. A utilización desta biblioteca posibilitaría que o código fose moito máis sinxelo de manter e non ter que empregar variables globais como se fai actualmente, eliminando tódolos problemas que xorden do uso deste tipo de variables.

- **Falta de funcionalidade:** Mentres que outros proxectos similares, como os que vimos no capítulo 2, ofrecen ó tradutor varias ferramentas adicionais ou funcionalidades que axuden na súa labor, o *Gtranslator* é un simple editor de ficheiros que non ofrece ningunha funcionalidade extra.

5.1.2 Mellora da interface

Na parte da interface os problemas son similares ós anteriores. A interface está feita usando as funcións da *API* de *GTK+* pero non emprega *GObject* co cal o código no ten unha organización e modularidade que dan as clases. Isto é un problema porque fai que non sexa mantible. Calquera cambio, por pequeno que sexa, implicará ter que modificar moito código para encaixalo.

Non emprega ningunha das tecnoloxías actuais existentes en GNOME para traballar coas interfaces gráficas como son *libglade* e o *GtkUIManager*.

Libglade é unha biblioteca que permite integrar no código da aplicación elementos gráficos (*widget*) creados coa ferramenta *glade* que permite crear interfaces gráficas de forma sinxela mediante compoñentes en modo gráfico. Isto resulta moi útil por exemplo para a creación de formularios ou elementos gráficos formados por moitos compoñentes. Ademais permite a división do traballo de deseño puro da interface co traballo de implementación dos controladores que van manexar esa interface.

Por outra banda o *GtkUIManager* permite construír os menús e as barras de ferramentas por medio dun ficheiro *XML*. Ademais permite asociar accións e grupos de accións ós elementos do menú e as iconas da barra de ferramentas. Por exemplo se queremos definir un menú Ficheiro que teña un elemento que sexa Abrir Ficheiro e outro que sexa Saír da aplicación e ademais un botón na barra de ferramentas para realizar a acción de Abrir Ficheiro definimos un ficheiro *XML* como o seguinte:

```
<ui>
  <menubar name="MenuBar">
    <menu name="FileMenu" action="File">
      <menuitem name="OpenMenu" action="OpenFile"/>
      <separator/>
      <menuitem name="QuitMenu" action="Quit"/>
    </menu>
```

```
  <toolbar name="ToolBar">
    <toolitem name="ToolbarOpen" action="OpenFile"/>
  </toolbar>
</ui>
```

As etiquetas *menubar* e *toolbar* indican se o que queremos incluír é un menú ou unha barra de ferramentas e as etiquetas *menuitem* e *toolitem* indican os elementos que se van engadir ó menú e a barra de ferramentas. Vemos tamén que o elemento *OpenMenu* do menú e *ToolbarOpen* da barra de ferramentas teñen asociada unha acción co mesmo nome polo que só teremos que definir unha vez o que queiramos que ocorra cando se execute esa acción e farase así independentemente do sitio no que pinchemos, sexa o no menú ou na barra de ferramentas.

As accións defínense xa no código da aplicación no ficheiro que corresponda. Por exemplo para a acción OpenFile podemos definir o seguinte:

```
static const GtkActionEntry main_window_action_entries [] = {
...

{"OpenFile", GTK_STOCK_OPEN, NULL, <control+O>, N_("Open File"),
 G_CALLBACK (on_open_file)},

...
```

O primeiro elemento é nome da acción que definimos no *XML*, o segundo é a icona que terá asociada, o seguinte está a NULL porque non se emprega, serviría para definir o texto que terá o elemento, neste caso iso vai incluído na icona. O seguinte será o acelerador de teclado asociado para executar a acción. O quinto será a mensaxe que mostrará cando pasemos o punteiro do rato por riba do elemento, e o último elemento será a función que contén o código que se vai executar.

5.1.3 Novas funcionalidades

Un dos aspectos que quedaron claros no capitulo 2 foi que o *Gtranslator* era un simple editor de ficheiros *PO* que non aportaba moito máis os tradutores na súa labor. Por contra as outras ferramentas das que se falaba nese capitulo fornecían ó usuario dunha serie de funcionalidades adicionais que lle facilitaban o traballo.

No caso do *Gtranslator* só podemos falar como funcionalidade da opción de autotradución que permite reutilizar as traducións que se van facendo co programa almacenándoas nun ficheiro en *XML* e posteriormente empregalas para tentar traducir un ficheiro *PO* de forma automática. Este é un intento de aproximación ó que sería unha memoria de tradución, pero o xeito de implementala e o seu funcionamento pode ser moi mellorable. En canto a implementación, gardar tódalas mensaxes nun ficheiro en *XML* non é a mellor opción en canto a rendemento pois cando o ficheiro creza o seu parseo volverase cada vez máis custoso en recursos e tempo. Ademais o facerse a tradución automática implica que o usuario terá que repasar de novo un por un tódalas mensaxes xa que dependendo do contexto unha tradución para a mesma mensaxe pode ser válida nun ficheiro pero non noutro.

Tendo en conta isto as funcionalidades que se implementarán para a nova versión do *Gtranslator* serán:

- **Xestión de diferentes perfís:** Un dos campos das preferencias dos que falamos na versión 1 do *Gtranslator* era o que recollía a información acerca do tradutor e da lingua, información que era usada por exemplo para enches os campos da cabeceira do ficheiro *PO* de xeito automática. Con esta nova funcionalidade o que se pretende e que se podan configurar diferentes perfís ou contas con esa información e elixir en cada momento cal se quere empregar.

- **Memoria de tradución:** Crear un sistema de memoria de tradución que permita a reutilización das traducións tanto das propias como as feitas por outros tradutores. Buscarase que o sistema de almacenaxe sexa rápido independente-

mente do tamaño das traducións almacenadas. Ademais buscarase un sistema para que sexa o usuario o que poda escoller directamente entre as diferentes opcións en lugar de empregar un sistema de tradución automática.

- **Xestor de catálogos:** Unha ferramenta asociada ó *Gtranslator* que permita xestionar os ficheiros *PO* dos proxectos nos que se está a colaborar facendo tradución e poder ver o seu estado, o número total de cadeas, cantas están traducidas, cantas faltan por traducir, cales están marcadas como dubidosas, etc. Ademais debe permitir abrir os ficheiros que se seleccionen no *Gtranslator*.

5.2 Deseño e Implementación

Neste apartado veremos as decisións de deseño que se tomaron para o desenvolvemento da nova versión tanto da xestión interna do programa como da parte da interface. Ademais explicarase con detalle o proceso de incorporación das novas funcionalidades que se inclúen na nova versión, dende o seu deseño ata a súa implementación.

5.2.1 Análise e deseño

A Linguaxe

A primeira decisión que se tomou foi a de seguir empregando a linguaxe de programación C como base do proxecto. É a linguaxe máis empregada nos proxectos de GNOME, é unha linguaxe de baixo nivel que permite un maior control dos recursos máquina que emprega a aplicación, xa que por exemplo a xestión de memoria faise por parte do programador. Isto pódese ver coma un problema xa que se non se fai ben pode dar moitos problemas na execución do programa, pero sendo coidadosos na reserva e liberación de memoria o rendemento que se pode conseguir movendo grande cantidade de información é importante.

Outra vantaxe é que C é o linguaxe máis portado, habendo compiladores para que case tódalas plataformas imaxinables, o cal pode ser unha vantaxe para tratar de migrar o proxecto no futuro a outras plataformas.

Ademais o feito de dispoñer das bibliotecas *GLib* e *GObject* facilita moito a labor de programación xa que por unha banda como se pode ver no apéndice B *GLib* ofrece unha serie de tipos de datos de máis alto nivel que non están dispoñibles no C estándar.

Por outra banda o recubrimento que da *GObject* permite o uso de orientación a obxectos en C o cal permite unha gran modularidade, facilidade de mantemento e reutilización a hora da implementación.

Os casos de uso

Os casos de uso básicos do proxecto están xa bastante definidos, xa que se trata dunha reenxeñería do programa. A lóxica principal do programa é a mesma ca da versión 1 polo que non é necesario redefinir novos casos de uso, agás para a implementación das novas funcionalidades que non estaban presentes na primeira versión. Nese caso, os casos de uso identificados para elas explicaranse nos apartados correspondentes a cada unha das funcionalidades.

Diagramas de estado

Na operación de abrir un ficheiro o primeiro que se debe comprobar é se xa hai outro aberto, nese caso é necesario crear unha nova pestana para o ficheiro que se vai abrir. A hora de parsealo, primeiro comprobase se a primeira mensaxe é a cabeceira, sendo así parsease por un lado e por outro parseansea tódolos demais mensaxes para obter a lista completa. (Ver figura 5.1)

Para gardar un ficheiro con nome a primeira comprobación será ver se xa existe un ficheiro co mesmo nome no sistema de ficheiros, se é así haberá que mostra unha mensaxe ó usuario para que decida se quere sobrescribir o ficheiro ou quere voltar

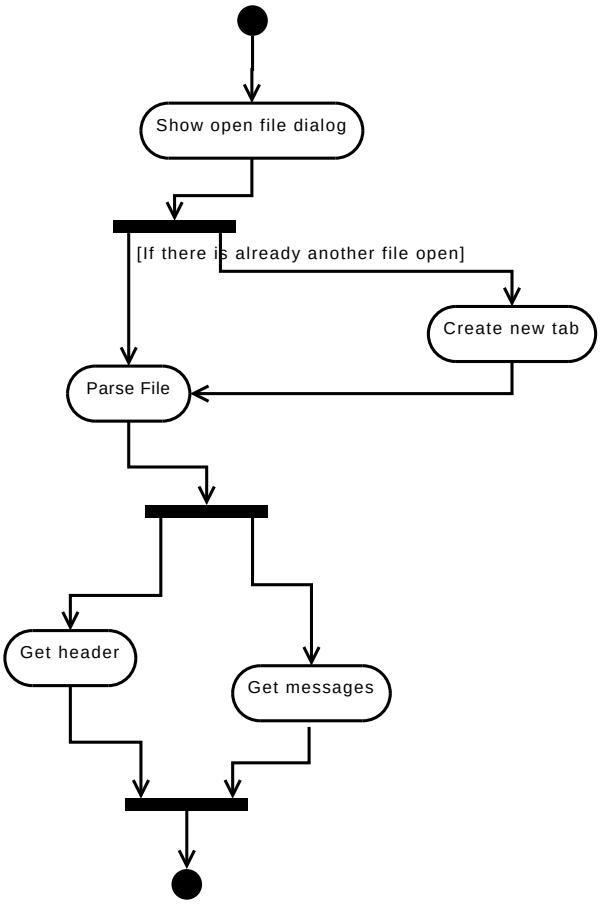

Figura 5.1: Diagrama de estados da apertura dun ficheiro

cara atrás e dar un novo nome.

Despois haberá que comprobar se hai que actualizar a cabeceira cambiando o nome do último tradutor. Unha vez actualizada a cabeceira gárdase na primeira mensaxe do ficheiro e a continuación escrébense a disco o resto de mensaxes do ficheiro. (Ver imaxe 5.2)

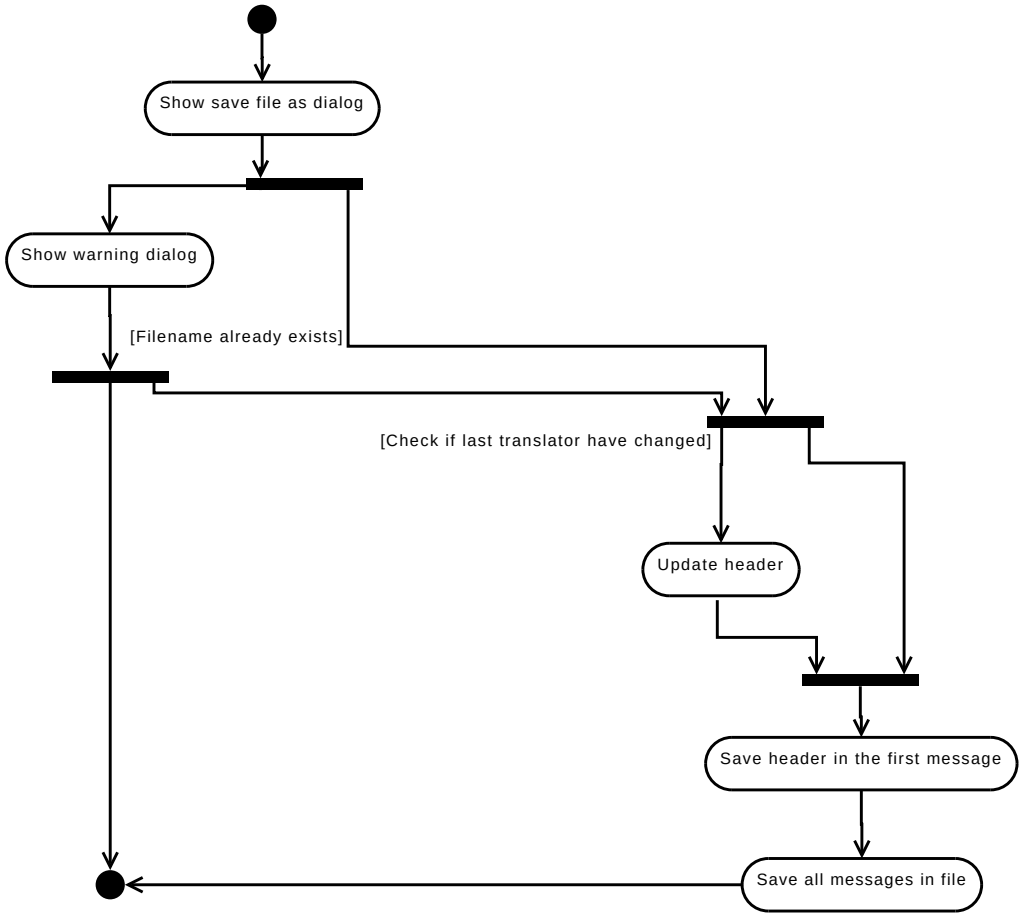

Figura 5.2: Diagrama de estados cando se garda un ficheiro especificando o nome ficheiro

As Clases

O emprego da orientación a obxectos como paradigma de programación para o desenvolvemento da nova versión do *Gtranslator* fai necesario a definición das clases do proxecto e as relacións entre elas. Nalgúns dos casos as clases coincidiran con al-

gunhas das estruturas definidas na primeira versión, pasándoo pola correspondente adaptación a *GObject*, mentres que noutras casos haberá que identificar e definir novas clases.

Unha das decisións de deseño que se tomaron foi a de facer os atributos das clases privados, podéndose acceder a eles só a través dos métodos *set* e *get*. Isto garante a *encapsulación*, de xeito que se podan facer cambios na implementación interna dunha clase sen que iso afecte o resto do programa.

O diagrama de clases simplificado da nova versión do *Gtranslator* quedaría como segue:

Nas figuras 5.4, 5.5 e 5.6 móstrase o diagrama de clases completo con tódolos atributos e os métodos de cada clase. Divídese en 3 figuras distintas tomando como nexo de unión a clase *GtranslatorWindow* que se mostra completa só na primeira figura.

5.2.2 Deseño da nova interface

A hora de facer o deseño da nova interface tívose en conta buscar a funcionalidade para os tradutores máis que o aspecto gráfico. Dende o primeiro momento fixéronse propostas na lista de correo do proxecto e na lista específica para tradutores que existe no proxecto GNOME. A idea era conseguir "feedback" por parte da comunidade de traductores xa que son eles quen van facer uso da aplicación. Na figurar 5.7 móstrase o primeiro intento de definición da nova interface que se enviou as listas de correo para a súa discusión:

Foi un primeiro intento ben recibido nalgúns aspectos pola comunidade pero froito da discusión que se realizou pouco a pouco foise pulindo o deseño ata chegar a versión actual da interface.

Como se falou no apartado sec:mellorainterface os cambios máis importantes para a mellora da interface é a inclusión das novas tecnoloxías que fornece o marco de desenvolvemento de GNOME, *Libglade* e o *GtkUIManager*.

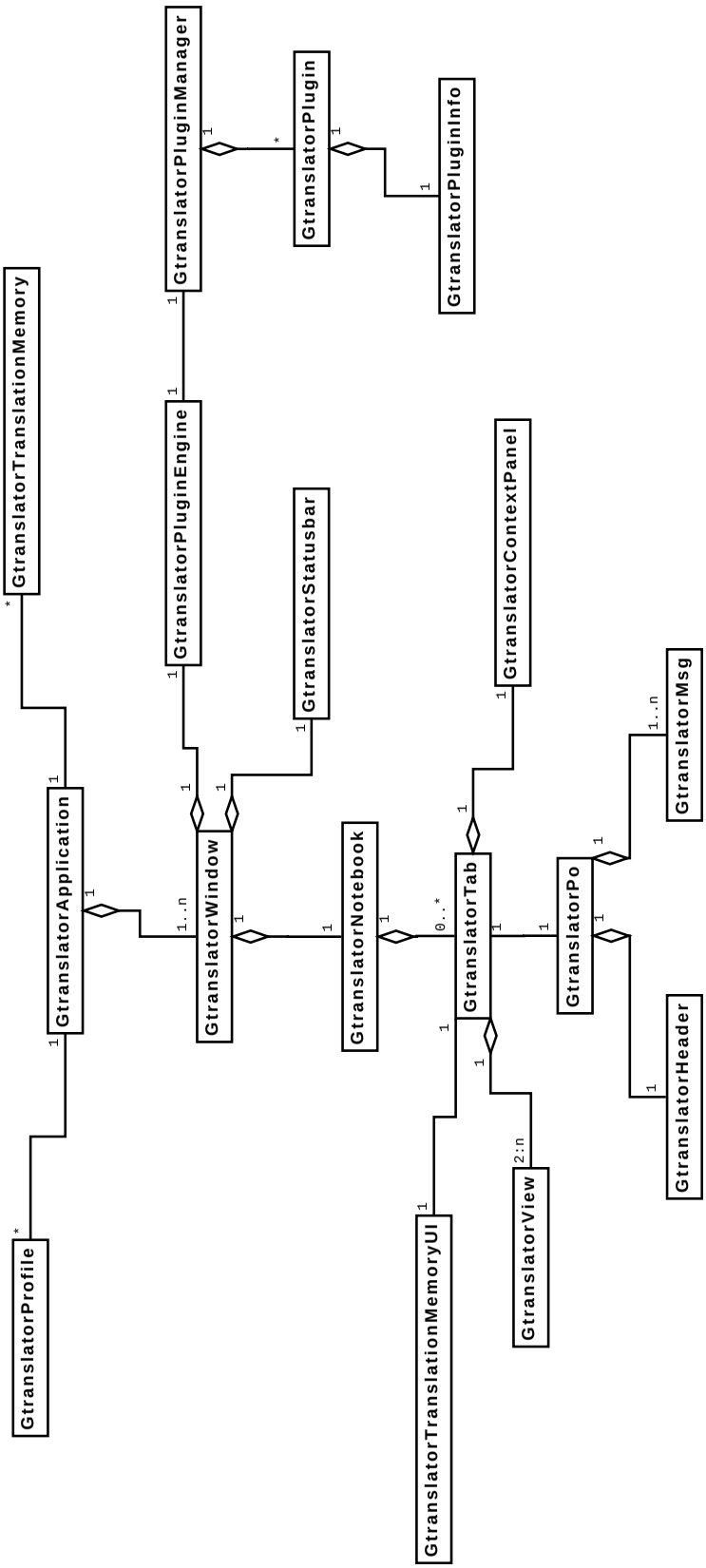

Figura 5.3: Diagrama de clases simplificado

Figura 5.4: Diagrama de clases estendido (Parte I)

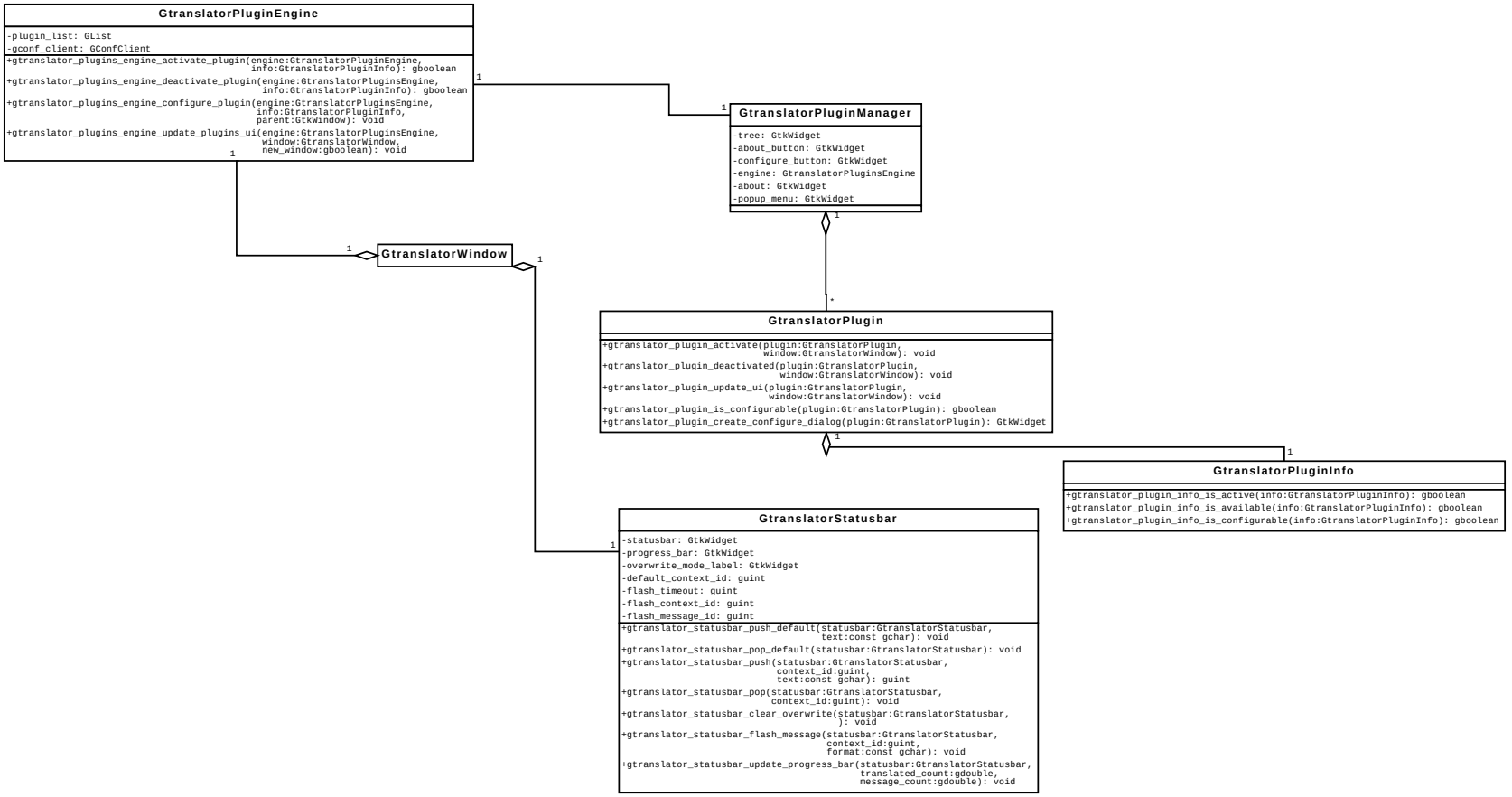

Figura 5.5: Diagrama de clases estendido (Parte II)

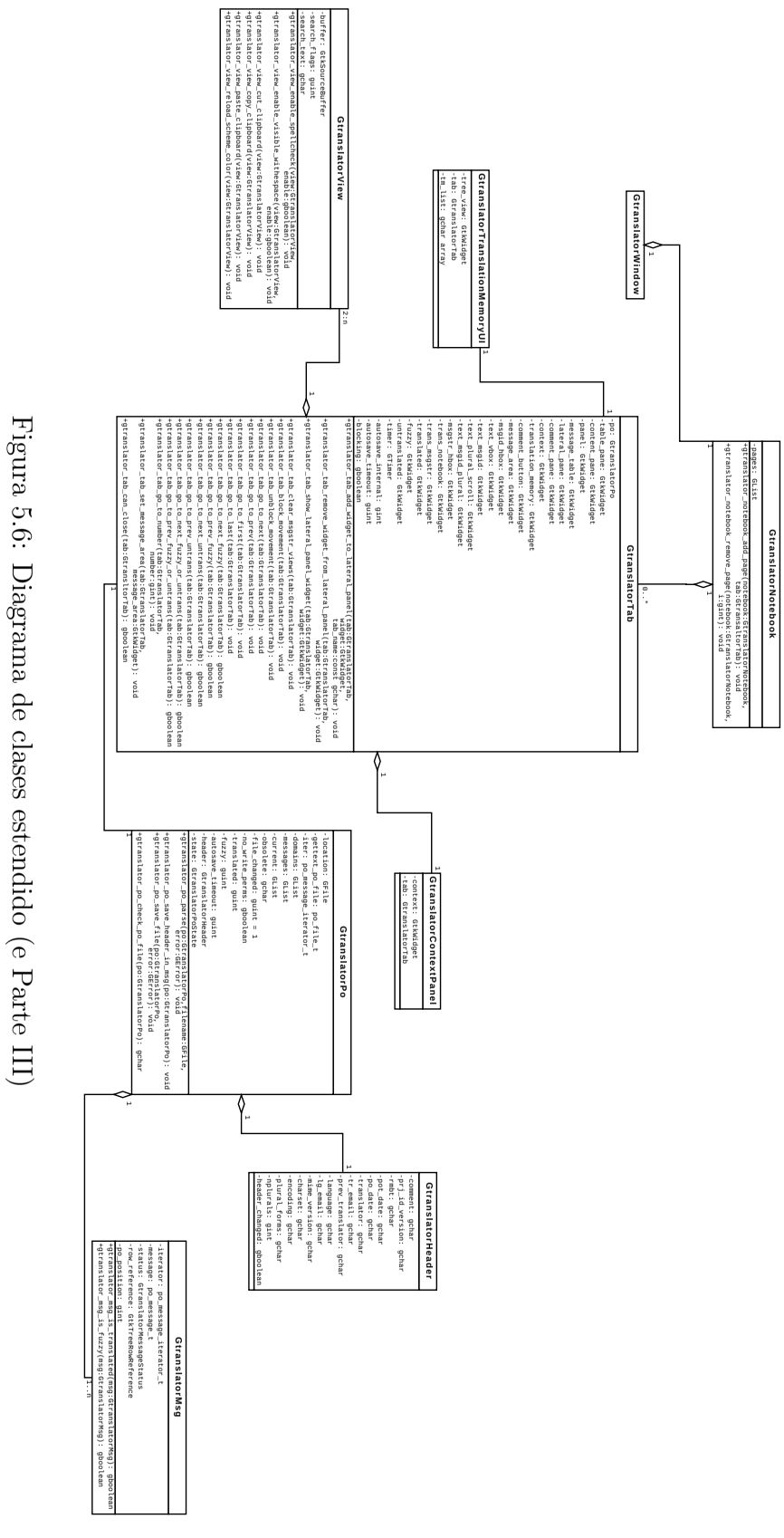

Figura 5.6: Diagrama de clases estendido (e Parte III)

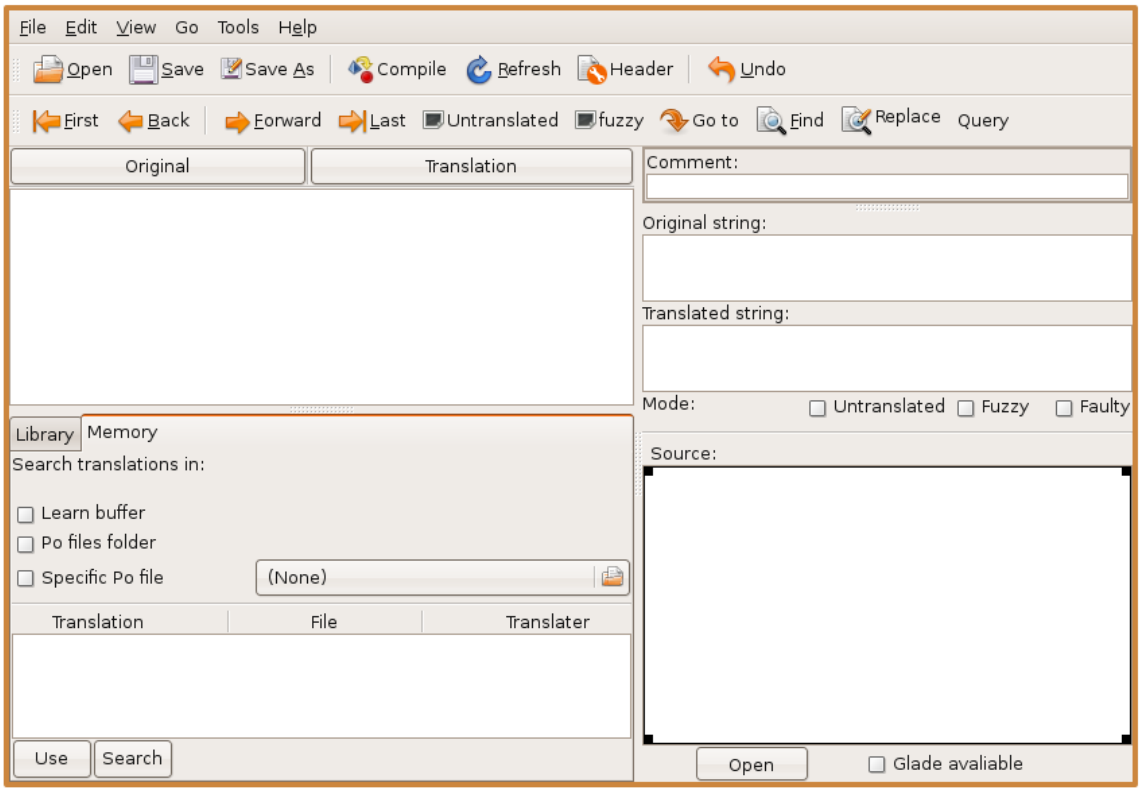

Figura 5.7: Primeiro intento de definición da interface

GtkUIManager

O *GtkUIManager* empregouse para definir tódolos menús da aplicación e barra de ferramentas que agora será só unha no canto das dúas que emprega a primeira versión. Toda a especificación destes menús e da barra de ferramentas faise a través dun ficheiro en *XML* que, como xa vimos, define tódolos elementos do menú e da barra de ferramentas e asócialles unha acción ou grupo de accións.

Todas estas accións defínense na implementación da clase *GtranslatorWindow*. Esta clase ten un atributo privado que é o propio *GtkUIManager*. Desta forma conséguese unha independencia entre a definición gráfica dos elementos do menú e da barra de ferramentas e a implementación das accións que teñen asociadas. A figura 5.8 mostra como quedan os menús e a barra de ferramentas definidas co *GtkUIManager*.

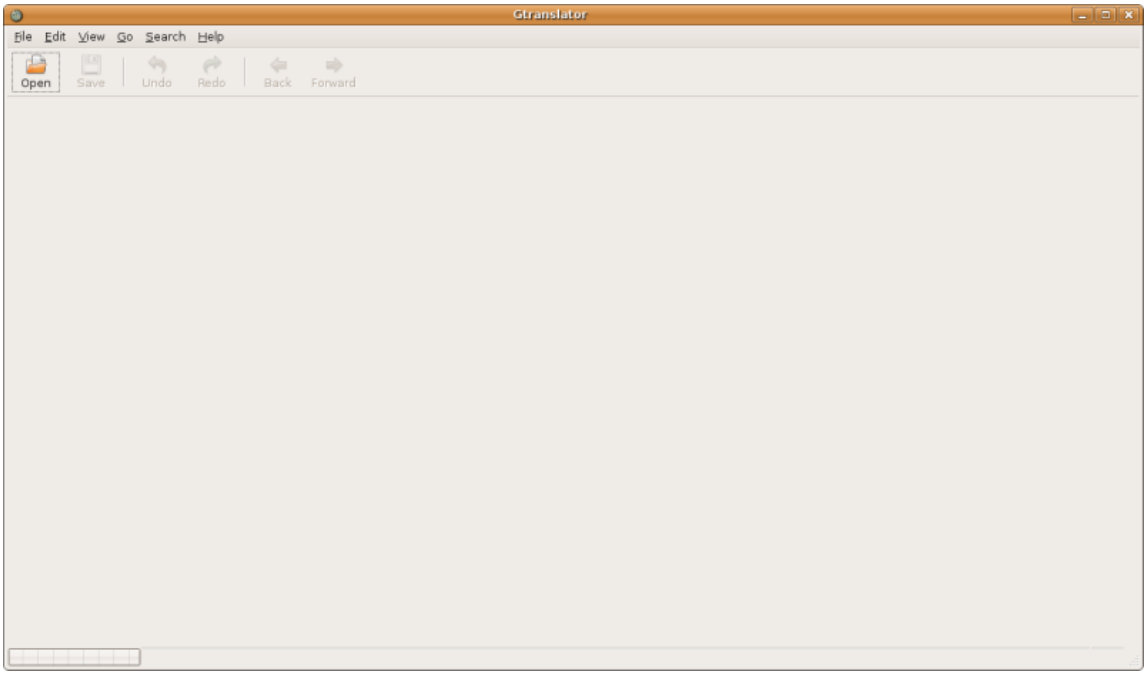

Figura 5.8: Menús e barra de ferramentas definidas con GtkUIManager

Área de traballo

A parte central da área de traballo definiuse como un *GtkNotebook*. Este *widget* de *GTK+* é un contedor que permite ter múltiples pestanas, cada unha delas pode conter a súa vez múltiples *widget*. Isto faise para poder soportar a posibilidade de abrir múltiples ficheiros nunha ventá da aplicación tendo cada un deles nunha pestana diferente. Esta característica portouse doutro proxecto de GNOME, o *Gedit* que é o editor de texto defectivo en GNOME. Partindo da clase *GtranslatorTab* que é a que define cada unha das pestanas créase toda a interface da área de traballo que se divide en tres zonas diferentes (imaxe 5.9).

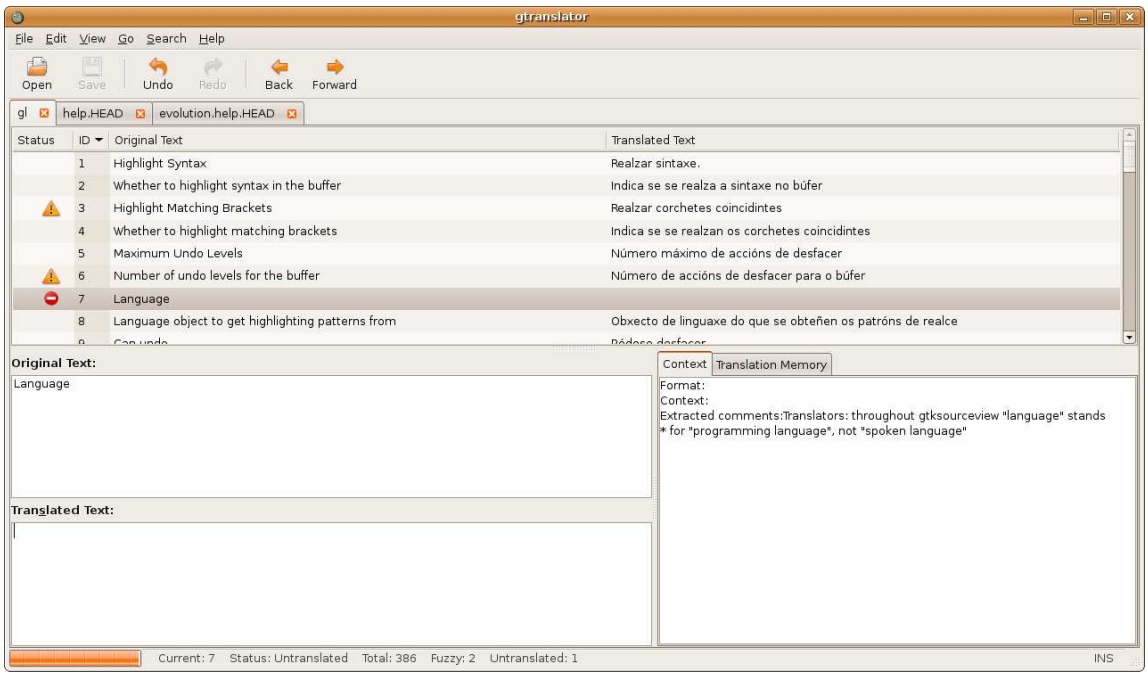

Figura 5.9: Interface definitiva da versión 2.0

A parte superior é a área de mensaxes onde se amosan todas as mensaxes que ten o ficheiro *PO* aberto. Amosa a cadea orixinal e a traducida ademais do identificador da mensaxe e unha columna con iconas que amosa o estado. Está implementada mediante un *GtkTreeView* que é un *widget* moi potente para amosar unha grande cantidade de datos que formen un mesmo conxunto, como unha lista neste caso. Na

implementación definiuse a área de mensaxes coma un atributo da clase *GtranslatorTab*.

A parte inferior está dividida en dúas zonas. Na parte esquerda é onde se fai a tradución das mensaxes. Móstranse dúas caixas de texto multiliña, a de arriba amosa a cadea orixinal e non é editable, a de abaixo é onde se insire a tradución. Baixando a implementación, cada unha destas caixas de texto está definida na clase *GtranslatorView* que está asociada á clase *GtranslatorTab*. Ademais a clase *GtranslatorView* herda da clase *GtkSourceView* de *GTK+*. Esta clase define unha caixa de texto multiliña con capacidades para resaltar texto. A especialización da clase permitiu engadir nova funcionalidade reimplementando os métodos para cortar, copiar e pegar texto, xa que no *Gtranslator* móstranse as mensaxes cambiando os espazos en branco por un símbolo especial para facilitar a visión do tradutor.

Por último a parte inferior dereita é unha área que serve para amosar información de axuda ó tradutor. É unha das cousas que se mantivo dende o primeiro intento de deseño da interface, xa que tódolos usuarios atopárona moi útil. Está implementada coma un *GtkNotebook* para poder ter varias pestanas diferentes. Isto permite cambiar facilmente de vista entre as diferentes opcións. Ademais permitirá engadir novas funcionalidades que mostren información útil ó usuario de forma sinxela.

Na versión 2.0 do *Gtranslator* ten dúas pestanas por defecto, unha que mostra información de contexto sobre a mensaxe e outra que mostra as opcións da memoria de tradución. Cada unha destas está definida coma unha clase asociada a *GtranslatorTab*. A clase *GtranslatorTranslationMemoryUI* define a vista para as opcións da memoria de tradución e verémola con detalle no apartado 5.2.4.

A clase *GtranslatorContextPanel* define a vista de contexto, que mostra información sobre a mensaxe: comentarios que incluíu o desenvolvedor na cadea orixinal, información de contexto, información sobre a linguaxe de programación da que procede a cadea orixinal no caso de ser diferente de C.

LibGlade

O uso de *LibGlade* posibilita a definición dun *widget* complexo de xeito cómodo empregando algunha aplicación de deseño de interfaces como *Glade*. Ademais permite a separación da vista e do controlador de xeito que se podan facer cambios na interface por separado sen que teña un grande impacto no código da aplicación.

No caso que nos ocupa empregouse *LibGlade* para a definición de tódolos diálogos da aplicación. Cada un destes diálogos definiuse coa ferramenta de deseño de interfaces *Glade* na súa versión 3. Ademais para cada diálogo creouse unha clase que contén como atributos tódolos *widget* que forman a vista do diálogo. Desta forma conseguise a independencia entre vista e controlador. A clase fai as labores de controlador e é onde se implementa toda a lóxica de xestión das sinais que lanzan os distintos *widget* da vista.

Para facilitar a integración dos elementos gráficos definidos no ficheiro *.glade* no código da clase portouse unha función do código do *Gedit*. Vexamos un exemplo:

Figura 5.10: Diálogo de edición da cabeceira

Na figura 5.10 vemos o diálogo que se mostra cando o tradutor quere editar a cabeceira do ficheiro *PO*. Este diálogo creouse co deseñador de interfaces *Glade* nun ficheiro chamado *header-dialog.glade*. Por outro lado definiuse unha clase chamada *GtranslatorHeaderDialog* que herda da clase de *GTK+ GtkDialog*.

```
/* Private structure type */
```

```
typedef struct _GtranslatorHeaderDialogPrivate
                GtranslatorHeaderDialogPrivate;

/*
 * Main object structure
 */
typedef struct _GtranslatorHeaderDialog
                GtranslatorHeaderDialog;

struct _GtranslatorHeaderDialog
{
  GtkDialog parent_instance;
  /*< private > */
  GtranslatorHeaderDialogPrivate *priv;
};

/*
 * Class definition
 */
typedef struct _GtranslatorHeaderDialogClass
                GtranslatorHeaderDialogClass;

struct _GtranslatorHeaderDialogClass
{
  GtkDialogClass parent_class;
};
```

Esta clase ten definidos como atributos privados os *widget* que forman o diálogo.

```
struct _GtranslatorHeaderDialogPrivate
```

```
{
  GtkWidget *main_box;
  GtkWidget *notebook;

  GtkWidget *prj_page;
  GtkWidget *lang_page;
  GtkWidget *lang_vbox;

  GtkWidget *prj_id_version;
  GtkWidget *rmbt;
  GtkWidget *prj_comment;
  GtkWidget *take_my_options;

  GtkWidget *translator;
  GtkWidget *tr_email;
  GtkWidget *pot_date;
  GtkWidget *po_date;
  GtkWidget *language;
  GtkWidget *lg_email;
  GtkWidget *charset;
  GtkWidget *encoding;
};
```

Para facer a integración dos *widget* no código da clase a través dos atributos que definimos para esta, empregase a función *gtranslator_utils_get_glade_widgets()* que como se comentou anteriormente foi portada do proxecto *Glade*. A integración faise no método de inicialización da instancia de clase *gtranslator_header_dialog_init()* que se chama cando se mostra o diálogo. A chamada a función *gtranslator_utils_get_glade_widgets()* queda da seguinte forma:

```
gtranslator_utils_get_glade_widgets(PKGDATADIR"/header-dialog.glade",
    "main_box",
    &error_widget,
    "main_box", &dlg->priv->main_box,
    "notebook", &dlg->priv->notebook,
    "lang_vbox", &dlg->priv->lang_vbox,
    "prj_id_version", &dlg->priv->prj_id_version,
    "rmbt", &dlg->priv->rmbt,
    "prj_comment", &dlg->priv->prj_comment,
    "take_my_options", &dlg->priv->take_my_options,
    "tr_name", &dlg->priv->translator,
    "tr_email", &dlg->priv->tr_email,
    "pot_date", &dlg->priv->pot_date,
    "po_date", &dlg->priv->po_date,
    "language_entry", &dlg->priv->language,
    "lg_email_entry", &dlg->priv->lg_email,
    "charset_entry", &dlg->priv->charset,
    "encoding_entry", &dlg->priv->encoding,
    NULL);
```

Neste momento xa temos asociados os *widget* que definimos no ficheiro *.glade* nos atributos da clase *GtranslatorHeaderDialog* co cal xa podemos definir toda a lóxica do controlador na clase separándoa da vista.

5.2.3 Xestión de diferentes perfís

Como se mencionou no apartado 5.1.3 con esta funcionalidade o que se pretende é que o usuario poda definir distintos perfís con diferente información persoal ou da lingua.

Análise e deseño

Nun primeiro análise vese que son necesarios tres compoñentes. Por un lado será necesaria unha interface para configurar os perfís e seleccionar o que queiramos ter activo, ademais será preciso gardar a información dos perfís de xeito persistente e unha lóxica de control para todo isto.

As accións típicas que realizará o usuario serán as de crear un novo perfil, editar ou borrar un xa existente e seleccionar cal quere que sexa o perfil activo en cada momento. (Ver casos de uso na figura 5.11).

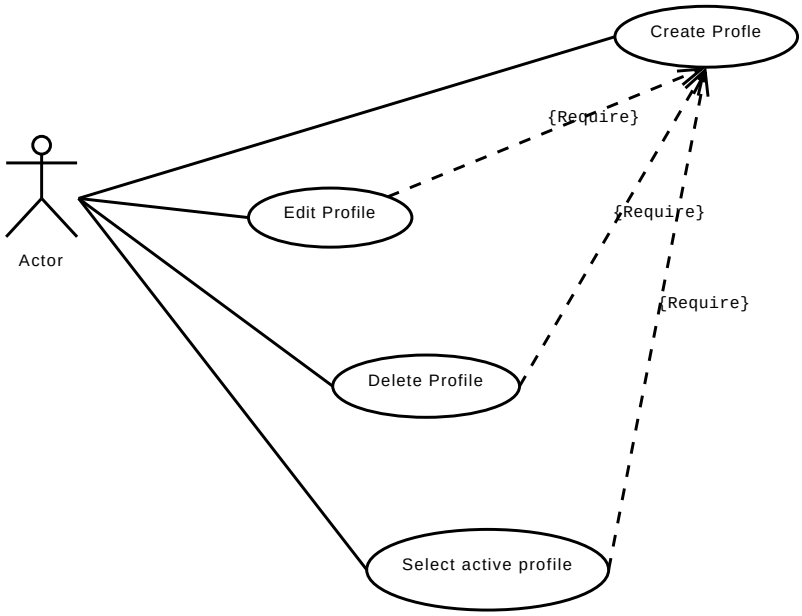

Figura 5.11: Diagrama de casos de uso da xestión dos perfís

Para crear un novo perfil será necesario comprobar se o nome que se lle quere dar xa existe, se é así haberá que notificarllo ó usuario. Ademais se é o primeiro que se crea haberá que facelo activo por defecto. Finalmente gardarase de forma persistente en disco (ver diagrama 5.12). No caso da edición dun perfil a lóxica é igual que á da creación, pero haberá que mostrar o diálogo coa información nos campos correspondentes.

Cando se borra un perfil haberá que comprobar primeiro se é o activo e mostrar

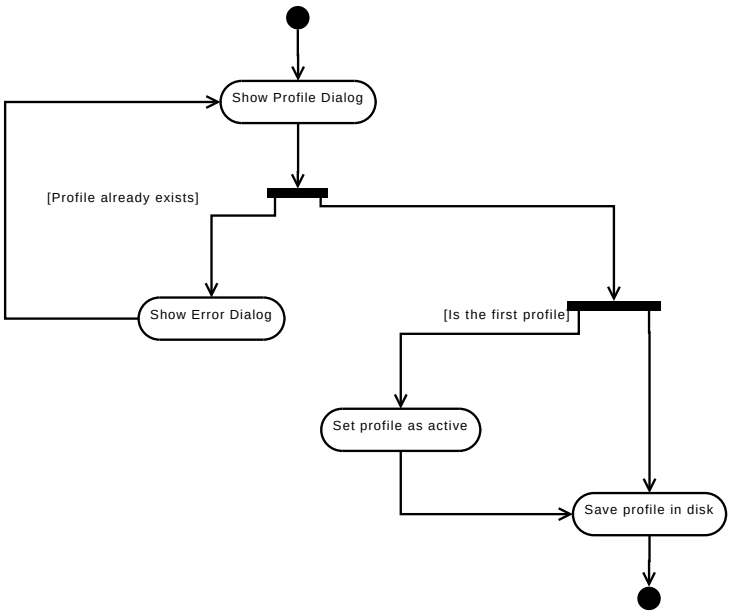

Figura 5.12: Diagrama de estados da creación dun perfil

un diálogo de aviso ó usuario para que seleccione outro como activo primeiro antes de borralo. A hora de seleccionar o perfil activo a política da aplicación é a de ter só un perfil activo en cada momento polo que haberá que cando se marque un novo perfil como activo haberá que desmarcar o que estaba activo anteriormente.

Implementación

Vendo as tres diferentes pezas que son necesarias para o desenvolvemento da nova funcionalidade vese que o patrón MVC (Model View Controler) encaixa bastante ben. Como sabemos este patrón separa os roles de desenvolvemento en tres partes: O modelo de datos por un lado, a vista ou interface gráfica por outro e un terceiro, o controlador que será encargado de recibir e manexar os eventos da vista.

Vista

A implementación da vista fíxose seguindo o esquema descrito no apartado 5.2.2. Neste caso hai que crear unha interface nova, un diálogo para recoller os datos do novo perfil, e modificar unha existente, o diálogo de preferencias.

O diálogo de preferencias é un *GtkNotebook* con varias pestanas. Creouse unha

nova pestana para xestionar os perfís, consistente nun *GtkTreeView* onde se mostran os perfís e cal é o que esta activo, e os botóns para crear, editar e borrar un perfil. (Ver imaxe 5.13).

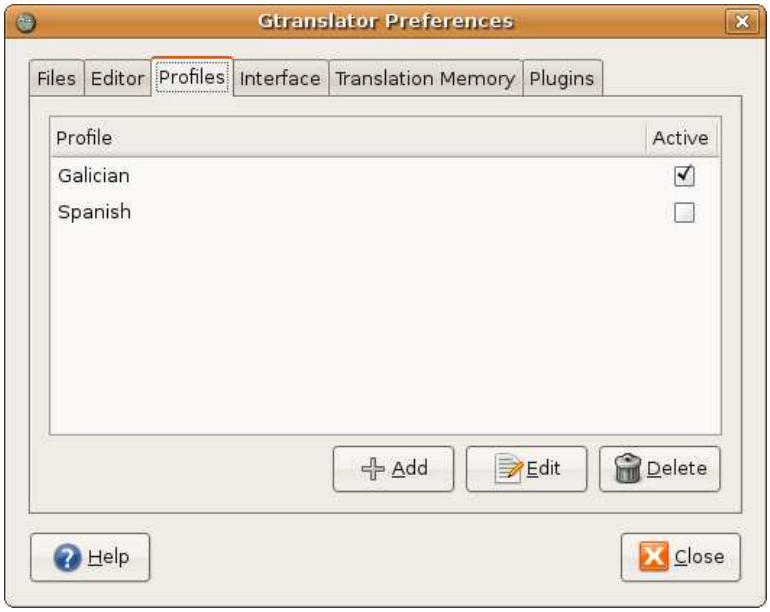

Figura 5.13: Diálogo de preferencias: Pestana dos perfís

O diálogo de creación e edición de perfís (imaxe 5.14) creouse tamén co deseñador de interfaces *Glade* e usando a mesma técnica explicada anteriormente usando *LibGlade*. Creouse a clase *GtranslatorProfileDialog* que define o diálogo dos perfís.

Modelo

O Modelo de datos para esta situación non é complexo, simplemente deberase gardar a información para cada uns dos perfís e a información de cal é o perfil activo en cada momento. Tendo en conta que a situación normal será a de ter poucos perfís xa que como moito se pode dar o caso de que un usuario teña configurados perfís para un par de linguas diferentes ou traballe coma voluntario traducindo para un par de proxectos e queira ter un perfil diferente para cada un deles. O volume de información para almacenar non é moi grande. Ademais cando se queira acceder a información no disco será para coller o conxunto de información dun perfil completo, non se necesita facer búsquedas de partes concretas.

Figura 5.14: Diálogo de creación e edición de perfís

Con todo isto a opción escollida foi a de empregar *XML* para gardar a información nun ficheiro en disco. Para construír o ficheiro e parsear a información que conteña empregouse *Libxml*, unha biblioteca que fornece de moitos métodos para traballar con ficheiros *XML*.

Creouse unha clase chamada *GtranslatorProfile* para ter dispoñible en memoria a información dos perfís a través da instanciación. Nesta clase implementáronse os métodos que gardan e recuperan a información do *XML* no disco: *gtranslator_profile_save_profiles_in_xml()* para gardar os perfís en disco e *gtranslator_profile_get_profiles_from_xml_file()* para recuperalos do disco.

En todo momento mantéñense dúas estruturas en memoria: unha lista con todos os perfís e unha instancia concreta da clase *GtranslatorProfiles* que é o perfil activo nese momento. Durante a execución do programa as operacións de creación, edición e borrado de perfís fanse en memoria e cando se pecha a aplicación escrébese no disco o ficheiro *XML* coa información que conteñen nese momento as dúas estruturas.

Controlador

O controlador está definido polas clases *GtranslatorPreferencesDialog* e *Gtrans-*

latorProfileDialog que se encargan de manexar os eventos cando reciben as sinais que emiten os *widget* dos diálogos, tanto os que veñen da pestana dos perfís no diálogo de preferencias como os que proceden do diálogo de creación e edición dos perfís. Vémolo cun exemplo:

Na función *gtranslator_preferences_dialog_init* que é a que se chama cando se crea unha instancia da clase *GtranslatorPreferencesDialog* é onde se conectan as sinais que emiten os *widget* co seu manexador.

```
g_signal_connect (dlg->priv->add_button,
  "clicked",
  G_CALLBACK (add_button_pulsed),
  dlg);
```

Coa función *g_signal_connect* conéctase neste caso a sinal "clicked", que a emite o botón de engadir novo perfil cando o usuario pincha sobre el co rato, co seu manexador que neste caso é a función *add_button_pulsed*. Nese momento executarase o código desa función:

```
static void
add_button_pulsed (GtkWidget *button,
   GtranslatorPreferencesDialog *dlg)
{
  GtranslatorProfile *profile;
  profile = gtranslator_profile_new ();
  gtranslator_show_profile_dialog(dlg, profile, NEW_PROFILE);
}
```

A función *add_button_pulsed* crea unha nova instancia da clase *GtranslatorProfile* para gardar nela a información do novo perfil que se quere crear e mostra o diálogo de creación de perfís.

5.2.4 Sistema de memorias de tradución

Con esta nova funcionalidade pretendeuse darlles os usuarios a posibilidade de reutilizar as traducións, tanto as feitas por eles mesmos como por outros.

Análise e deseño

Para a primeira versión das memorias de tradución optouse por definir dous xeitos de crear as memorias de tradución. A primeira seleccionando un directorio que conteña ficheiros *PO* que serán procesados para obter as mensaxes traducidas e así crear a memoria. A segunda forma será de xeito dinámica cada vez que se traduce unha mensaxe dun ficheiro almacenarase na memoria de tradución.

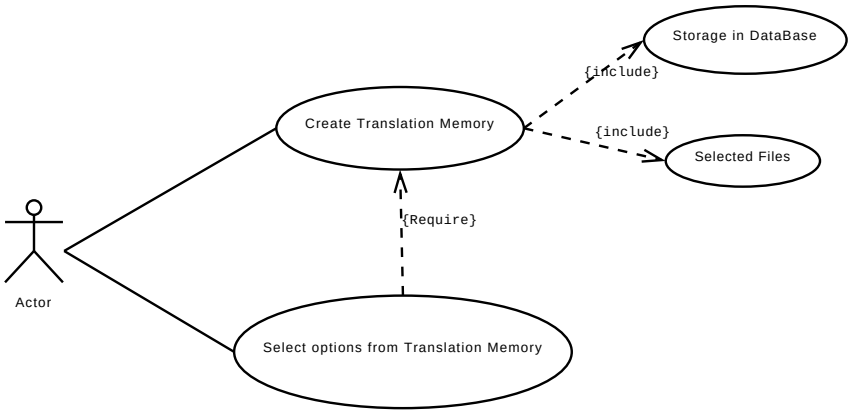

Figura 5.15: Diagrama de casos de uso das memorias de tradución

Implementación

Para a implementación pensouse no patrón MVC (Model View Controller) que encaixa perfectamente cos diferente roles que se definen.

Vista

Creáronse dúas vistas diferentes, por un lado a vista de configuración e creación da memoria de tradución e por outro a de visualización das opcións por parte de usuario.

Na vista de configuración empregouse de novo *LibGlade* modificando o diálogo de preferencias para engadir unha nova pestana onde mostrar as opcións para crear a memoria e engadir novas mensaxes, así como configurar o seu comportamento. (Ver imaxe 5.16)

Figura 5.16: Pestana de configuración das memorias de tradución nas preferencias

Entre as opcións de configuración están a de elixir se se queren ver as opcións da memoria para as mensaxes que xa están traducidas e máis a de elixir se se quere que se empreguen só ficheiros cun determinado nome para crear a memoria. Isto é útil xa que se se quere empregar ficheiros *PO* que forman parte do código fonte dun

programa, estes están nomeados co ?código da lingua. Deste xeito créase de maneira sinxela a memoria con ficheiros para a mesma lingua.

No apartado da vista no que se mostran as opcións da memoria de tradución ó usuario, a decisión foi implementala directamente sobre o código. Creouse unha clase *GtranslatorTranslationMemoryUI* relacionada coa clase *GtranslatorTab*, xa que cada ficheiro *PO* terá as súas propias opcións da memoria de tradución. A vista neste caso consiste nun *GtkTreeView* con tres columnas, unha que mostra o acelerador de teclado asociado a cada unha das opcións, unha barra de progreso que mostra o nivel de coincidencia da opción da memoria coa cadea orixinal e a cadea de texto proposta como tradución. Decidiuse tamén limitar a 10 o número de opcións a mostrar para cada mensaxe.

Para rematar a vista decidiuse empregar o *GtkUIManager* para definir un submenú na barra de menús que contén un ítem para cada unha das opcións da memoria, de xeito que se podan seleccionar tamén desde o menú.

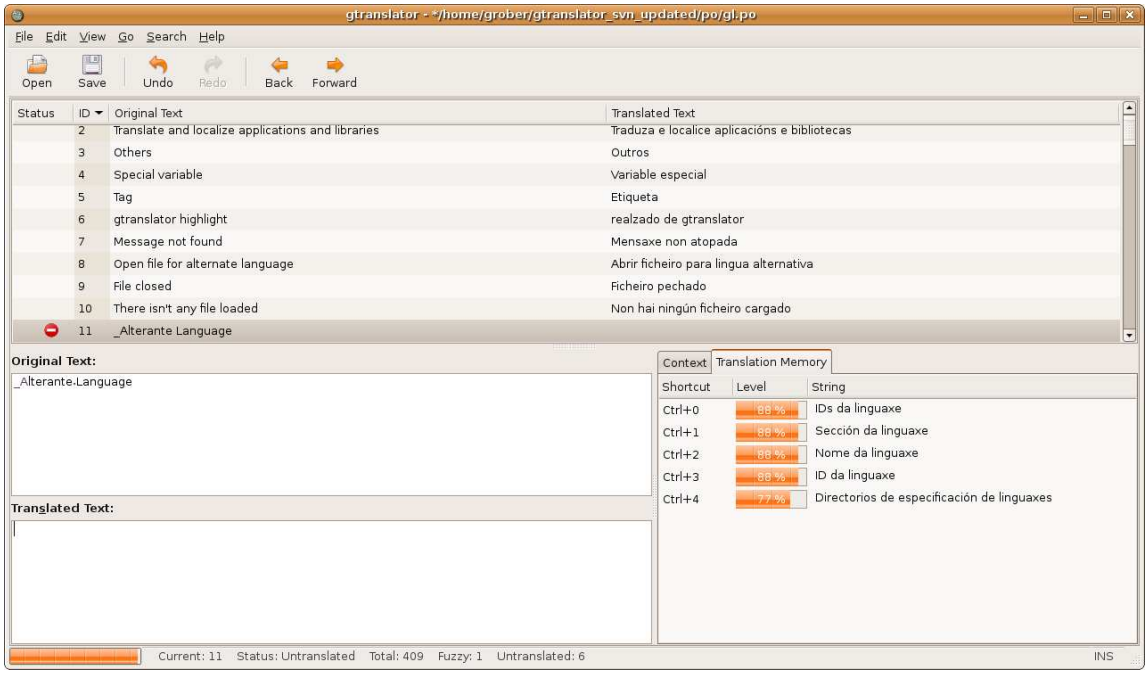

Figura 5.17: Interface de usuario das memorias de tradución no *Gtranslator*

Modelo Para o modelo de datos creouse un sistema de tipo *frontend/backend*

que permita a posibilidade de implementar o acceso a datos de varias formas diferentes. Como resultado existe unha clase chamada *GtranslatorTranlationMemory* que é unha interface. Cada un dos diferentes *backends* implementará esta interface ten dous métodos básicos que son: *gtranslator_translation_memory_store* para almacenar novas cadeas na memoria de tradución e *gtranslator_translation_memory_lookup* que devolve unha lista de opcións para unha mensaxe dada.

De momento existe só un *backend* que implementa a interface *GtranslatorTranslationMemory*. É un sistema creado sobre a base de datos non relacional *BerkeleyDB*. Este sistema de base de datos permite almacenar pares chave/valor coma un array de bytes e soporta a posibilidade de ter múltiples ítems asociados a unha mesma chave. Ten unha arquitectura moi simple comparada con outros sistemas de base de datos, non permite usar *SQL* ó non ser relacional pero encaixa moi ben para a implementación desta funcionalidade xa que o que se necesita e recuperar tódalas cadeas asociadas a unha mensaxe o máis rápido posible. O ter que recuperar conxuntos de datos enteiros non se precisa facer buscas complexas.

A implementación deste sistema fíxose de xeito independente grazas ó emprego do MVC e non corresponde a traballo obxecto desta memoria, senón que foi realizado por outro dos desenvolvedores do proxecto.

Controlador A lóxica de control para o manexo dos eventos e sinais que lanzan os *widget* tanto na configuración da memoria de tradución como na interface do programa está definida na clases clases *GtranslatorPreferencesDialog* e *GtranslatorTranslationMemoryUI*.

A primeira destas clases define a implementación dos métodos que manexaran os eventos do diálogo de configuración e creación da memoria de tradución. No caso da creación percorrese o directorio parseando tódolos ficheiro *PO* e obtendo tódalas mensaxes que teñan o estado de traducidas. Para cada unha de elas chámase a función do modelo *gtranslator_translation_memory_store* e pásaselle a cadea orixinal, a tradución e a instancia do *backend* co que se quere almacenar a memoria de

tradución.

A clase *GtranslatorTranslationMemory* é a encargada de xestionar os eventos da interface gráfica da memoria de tradución que se lle mostra ó usuario. Cada vez que se cambia de mensaxe na ventá principal do *Gtranslator* faise unha busca na memoria de tradución para amosar as opcións que hai para traducir esa mensaxe.

Na clase *GtranslatorTab* definiuse unha sinal que se emite cando se remata a edición dunha mensaxe. Este sinal emprégase para almacenar as traducións que se van facendo durante a execución normal do programa na memoria de tradución.

```
signals[MESSAGE_EDITION_FINISHED] =
  g_signal_new("message-edition-finished",
       G_OBJECT_CLASS_TYPE (klass),
       G_SIGNAL_RUN_LAST,
       G_STRUCT_OFFSET (GtranslatorTabClass,
message_edition_finished),
       NULL, NULL,
       g_cclosure_marshal_VOID__POINTER,
       G_TYPE_NONE, 1,
       G_TYPE_POINTER);
```

Cada vez que se cambia de mensaxe na interface a clase *GtranslatorTranslation-MemoryUI* encárgase de recuperar a lista de opcións para esa mensaxe e encher os datos do *GtkTreeView* e actualizar o submenú da barra de de menús. Cando se pulsa nunha das opcións deste submenú ou se emprega un dos aceleradores de teclado que teñen asociados as opcións da memoria insírese a cadea de texto na caixa de texto da mensaxe traducida.

5.2.5 Asistente de configuración

Nas dúas funcionalidades anteriores é necesario facer unha configuración previa para que todo funcione correctamente. Tanto no caso dos perfís como das memorias de

tradución é necesario crealas primeiro para despois poder aproveitar a funcionalidade que ofrecen.

No caso dos perfís é un requirimento, xa que se non existe ningún perfil creado moitas das outras funcionalidades non funcionarán correctamente. Debido a esta restrición decidiuse crear un pequeno asistente que se executa a primeira vez que se arrinca o programa e sempre que se detecte que non está creado o ficheiro de *XML* onde se gardan os perfís. Esta comprobación faise no método de creación da instancia da clase *GtranslatorApplication*.

```
/*
 * If the config folder exists but there is no profile
 */
 profiles_file = g_build_filename (gtranslator_folder,
                                   "profiles.xml",
                                   NULL);
 if (!g_file_test (profiles_file, G_FILE_TEST_EXISTS))
   priv->first_run = TRUE;
 g_free (profiles_file);
```

Desta xeito cando arrinquemos o *Gtranslator* se non existe ningún ficheiro chamado *profiles.xml* no directorio de configuración do programa mostrarase o asistente (Imaxes 5.18 e 5.19).

Aproveitando a creación deste asistente para o caso dos perfís, decidiuse empregar tamén para dar a opción de crear xa a memoria de tradución. (Imaxe 5.20)

A implementación fíxose seguindo o patrón MVC (Model View Controller) no que só se tivo que definir a vista do asistente xa que tanto o modelo como o controlador reaproveitáronse os definidos para a implementación das funcionalidades.

A definición da vista fíxose seguindo o procedemento, xa comentado noutros apartados, creando unha clase *GtranslatorAssistant* que herda da clase de *GTK+*

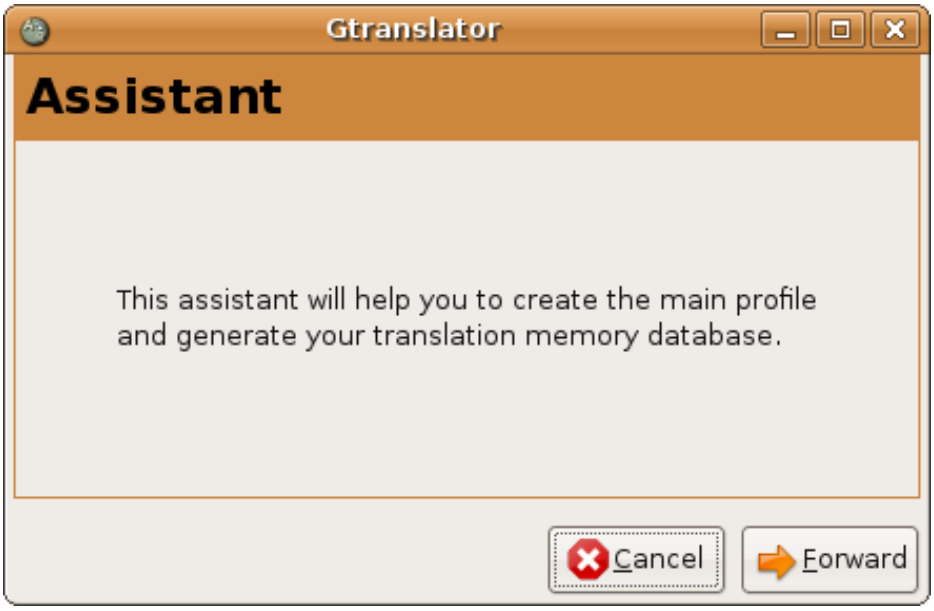

Figura 5.18: Pantalla inicial do asistente

Figura 5.19: Pantalla de configuración dos perfís no asistente

Figura 5.20: Pantalla de configuración da memoria de tradución no asistente

GtkAssistant, un tipo de *widget* especial que fornece *GTK+* para a creación deste tipo de asistentes de configuración.

5.2.6 Xestor de catálogos

Para o xestor de catálogos partiuse da base dun desenvolvemento feito por outro colaborador do proxecto. É unha funcionalidade que está aínda verde e tomouse a decisión de non incluíla na versión 2.0 do programa. Seguirase co seu desenvolvemento para tentar incluila na seguinte *release*, a versión 2.2, se o seu estado está xa máis maduro.

Sen embargo como parte deste proxecto colaborouse no seu desenvolvemento implementando a algunha mellora. O xestor de catálogos é unha xanela que se abre dende o propio *Gtranslator* e que permite crear proxectos con varios ficheiros *PO*. Amosa información sobre o estado de cada un de estes ficheiros a través dun *GtkTreeView*.

A primeira mellora que se fixo foi a de mostrar máis información sobre o estado

dos ficheiros. Orixinalmente só mostraba unha barra de porcentaxe que indicaba o grado de tradución total no que se encontra o ficheiro. Decidiuse conservar ese gráfico pero complementalo con máis columnas que mostren o número de cadeas totais que ten o ficheiro, para que o tradutor teña unha idea clara do tamaño do ficheiro, o número de cadeas que están marcadas como dubidosas, e o número de cadeas que están sen traducir.

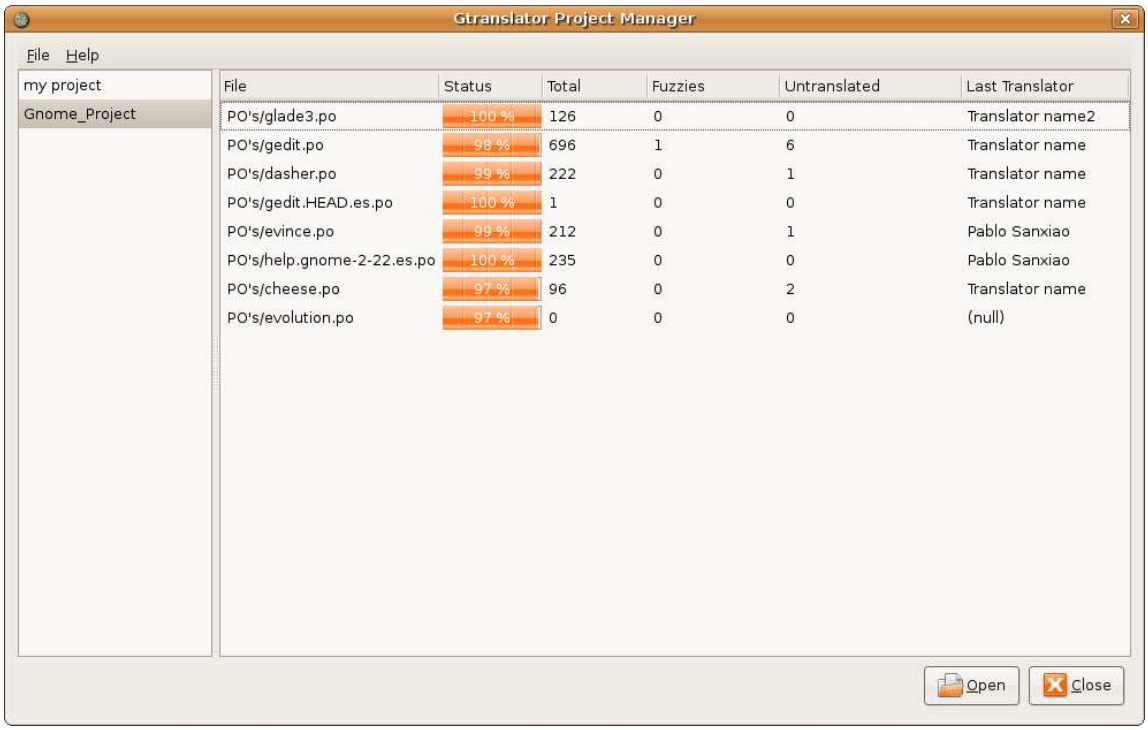

Figura 5.21: Ventá do Xestor de catálogos

A outra mellora que se fixo no xestor de catálogos foi a de posibilitar que se podan abrir varios ficheiros á vez no *Gtranslator* aproveitando a funcionalidade de poder ter varios ficheiros abertos en múltiples pestanas.

En primeiro lugar foi necesario que o *GtkTreeView* permita a selección múltiple nas filas. Isto faise mediante unha propiedade de clase que permite elixir o tipo de selección para as filas.

```
pm->priv->file_treeview = gtk_tree_view_new ();
```

```
selection =
gtk_tree_view_get_selection (GTK_TREE_VIEW (pm->priv->file_treeview));
gtk_tree_selection_set_mode (selection,
                             GTK_SELECTION_MULTIPLE);
```

Ademais engadiuse un botón *Open* para abrir os ficheiros seleccionados no *GtkTreeView*. Cando se emite a evento *response*, o que ocorre cando se pulsa sobre calquera dos botóns ou se pecha a ventá, executarase o seguinte código:

```
static void
dialog_response_handler (GtranslatorProjectManager *pm,
 gint res_id)
{
  GtkTreeSelection *selection;
  GtkTreeModel *model;
  GList *rows_selected = NULL;
  GList *l = NULL;

  switch (res_id)
    {
    case GTK_RESPONSE_OK:
      selection =
      gtk_tree_view_get_selection (
       GTK_TREE_VIEW (pm->priv->file_treeview));
      gtk_tree_selection_selected_foreach(selection,
                                          multiple_row_activated_cb,
                                          pm);
      break;

    default:
```

```
      gtranslator_project_utils_save_config (GTR_PROJECT_MANAGER (pm));
      gtranslator_project_manager_file_quit (NULL,
     GTR_PROJECT_MANAGER (pm));
    }
}
```

Esta función detecta se o evento foi lanzado polo botón *Open* ou polo botón *Close* e de acordo con isto executa un código ou outro. Cando se pulsa sobre o botón *Open* detéctase cales son as filas que están seleccionadas e para cada unha delas chama a función *multiple_row_activated_cb* que se encarga de abrir o ficheiro.

5.2.7 Documentación de usuario

A xestión da documentación de usuario nas aplicacións de GNOME faise a través das GNOME Doc Utils. Este paquete de ferramentas contén dende utilidades para construír a documentación e tódolos ficheiros auxiliares necesarios, coma plantillas de *DocBook* para mostrar o manual empregando a aplicación *Yelp*, que é o navegador da axuda de GNOME.

Unha das maiores vantaxes que aporta ter o manuais de usuario no formato que ofrecen as GNOME Doc Utils é que grazas a unha das súas utilidades, *xml2po*, pódese crear un ficheiro *PO* que conteña tódalas cadeas de texto do manual de usuario. Desta forma conséguese unha plataforma moi potente para traducir o manual de maneira sinxela.

Como parte deste proxecto creouse todo o esqueleto necesario ter o manual de usuario en formato das GNOME Doa Sutil. Ademais creouse o manual en *XML* con información para o usuario sobre as novidades da versión 2 e como empregalas.

Para ter soporte de GNOME Doc Utils na aplicación é necesario ter varias cousas no directorio *help* do código fonte do programa. Por un lado é necesario crear un ficheiro *Makefile.am* que teña un formato como o que segue:

```
include $(top_srcdir)/gnome-doc-utils.make
dist-hook: doc-dist-hook

# The basename of the docbook file. Generally, you want it
# to be the same as your app name
DOC_MODULE = <appname>
#Any entities you use (?)
DOC_ENTITIES =
# If your file is split into different files, list all the files
#that are included here.  Paths are relative to the C/ subdir
DOC_INCLUDES = legal.xml

# List of figures to include.  Paths are relative to the C/ subdir
DOC_FIGURES = \
             figures/first_fig.png
```

Ademais é necesario crear un ficheiro *omf.in* co nome da aplicación para fornecer de metainformación as utilidades que se encargan de xerar a documentación.

```
<?xml version="1.0" standalone="no"?>
<omf>
  <resource>
    <subject category="<categories>"/>
    <type>user's guide</type>
    <relation seriesid="<use the program uuidgen
                         to generate a unique identifier"/>
    <rights type="<License type>" license.version="<version, if desired>"
                   holder="<Writers name>"/>
  </resource>
</omf>
```

O documento *XML* que contén o manual de usuario ten que estar escrito en *Docbook* e conter unha serie de campos obrigatoriamente:

- Un campo *abstract role="description"* no apartado *articleinfo*
- Os campos *author* e *corpauthor* no apartado *articleinfo*
- Un dos seguintes campos: *autor, corpauthor, editor, otercredit ou plublisher* co atributo *role="maintainer"*
- Un elemento *title* no apartado *article* ou *articleinfo*
- Un elemento *date* na derradeira revisión no campo *revhistory*
- Un elemento *revnumber* na derradeira revisión no campo *revhistory*

Despois hai que engadir unha serie de directivas das *autotools*, que son as ferramentas para xestionar a compilación, como segue:

Engadir o ficheiro "help/Makefile" no macro AC_CONFIG_FILES do *configure.ac*. Engadir tamén a macro GNOME_DOC_INIT no ficheiro *configure.ac*. Engadir a liña *gnome-doc-utils.make* no apartado DISTCHECK_CONFIGURE_FLAGS do ficheiro *Makefile.am*.

O manual orixinal debe estar dentro do directorio *help* nun subdirectorio chamado *C*, ó mesmo nivel pódense crear directorios cos ficheiros do manual traducidos a diferentes linguas.

Capítulo 6

Viabilidade

Neste capítulo amósase a planificación, organización e as consideracións que se tiveron en conta para levar a cabo este proxecto, mostrando especial atención as súas particularidades tendo en conta que se trata dun proxecto de *Software Libre*, desenvolvido en comunidade.

Á hora de facer unha estimación económica dun proxecto de desenvolvemento hai varios factores a ter en conta, que teñen como punto de partida os custos de persoal, pero que inclúen outros aspectos que tamén é necesario considerar.

6.1 Acerca da viabilidade

A viabilidade trata de medir a conveniencia dun sistema ou proxecto tendo en conta a relación entre os recursos necesarios para conseguilo ou levalo acabo, e os recursos dispoñibles. Para tratar de analizar isto, hai que ter en conta distintos factores, como por exemplo:

- Custo do *hardware*.
- Custo do desenvolvemento.
- Custo das licenzas do *software* (adquisición e renovación).

Ó ser este un proxecto de *Software Libre*, todo o *software* empregado para o desenvolvemento e o *software* necesario para usar a aplicación resultante deste é tamén libre, polo que se pode ignorar o custo das licenzas, que normalmente supón unha cantidade importante. Ademais as licenzas non libres teñen que ser renovadas de xeito periódico, en moitos casos. para manter o sistema a funcionar.

Sobre o custo de *hardware*, neste caso o resultado deste proxecto é unha aplicación de escritorio, polo que o *hardware* necesario será basicamente un PC con sistema operativo *GNU/Linux*. Os requirimentos que ten que ter este PC non son excesivamente altos. Calquera equipo actual cumpre de sobra con estas características.

6.2 Roles

De xeito xeral, nun proxecto de *Software Libre* son habituais unha serie de perfís que se describen a continuación:

- **Usuario:** Son as persoas que usan o programa.
- **Desenvolvedor:** Son as persoas que fan o deseño e a implementación segundo os requisitos recollidos.
- **Mantedor:** É a persoa que se encarga da organización xeral do proxecto, nalgúns casos, en proxectos moi grandes este perfil pódese subdividir en varias categorías, máis específicas.
- **Desenvolvedor da distribución:** Son desenvolvedores que forman parte do equipo de traballo dunha distribución concreta, a súa labor consiste en integrar as diferentes aplicacións, das que se fan responsables, na distribución. Isto inclúe facer pequenos cambios para adaptalas ó entorno, facer os paquetes ou revisar as licenzas para comprobar se son compatibles.
- **Escritor de documentación:** Son as persoas que se encargan de escribir

toda a documentación dun programa. O manual de usuario, as páxinas man, guías de usuario, etc.

- **Tester:** É a figura das persoas encargadas de facer probas das aplicacións. Normalmente non existe unha figurar concreta que desempeñe este rol de forma illada. Calquera usuario ou desenvolvedor que use o programa pode reportar un erro, realizando así labores de tester.

- **Líder de proxecto:** É a cabeza visible do proxecto. Soen ser persoas de relevancia no mundo do *Software Libre* que conseguen chegar a esa posición por meritocrática. Exemplos: Linus Torvalds, Miguel de Icaza, etc.

- **Tradutor:** Son as persoas que se encargan de face a tradución dos programas.

Neste proxecto participan os perfís de mantedor, desenvolvedor, escritor de documentación e tester. Ademais haberá unha figura máis, habitual nos proxectos máis clásicos de enxeñería que será o de analista/director de proxecto, que se encargará da coordinación e control do proxecto, ademais do análise e recollida de requisitos.

O custo económico asociado a cada un destes perfís son:

Perfil	**Custo por hora**
Analista/Director de proxecto	50 €
Desenvolvedor	25 €
Mantedor	0 €
Probador	0 €
Escritor de documentación	0 €

Táboa 6.1: Custo por hora de cada perfil

Os custos mostrados na táboa 6.1 son aproximacións sobre valores reais de proxectos de desenvolvemento de *software* en Galicia. Nos casos do mantedor, probador e escritor de documentación o custo é 0 porque estes perfís son desenvolvidos por voluntarios que forman parte da comunidade do proxecto.

6.3 Tarefas

Para a identificación das tarefas que se levaron a cabo durante este proxecto, é necesario facer unha distinción entre tarefas de desenvolvemento e tarefas administrativas. Con administrativas referímonos a tarefas de xestión do proxecto e da comunidade típicas do *Software Libre*.

Para facer a planificación temporal das tarefas é necesario ter en conta que se trata de estimacións. A circunstancias dun proxecto destas características poden ser moi cambiantes. Moitas das tarefas son feitas por voluntarios da comunidade, normalmente no seu tempo libre, polo que é moi complicado tratar de saber en todo momento se se van poder cumprir esas tarefas. Este será un risco que haberá que asumir.

TAREFA	**DURACIÓN**
Análise	**7d**
Análise do código da versión 1 e Requisitos da versión 2	5d
Selección dos bugs a resolver	2d
Resolución de bugs na versión 1	**10d**
Deseño da nova versión	**7d**
Deseño da arquitectura	3d
Deseño da interface	2d
Deseño das novas funcionalidades	2d
Desenvolvemento	**45d**
Migración a GObject/*GTK+*	20d
Implementación da nova interface	10d
Implementación das novas funcionalidades	15d
Xestión da comunidade	**12**
Organización e administración do Bugzilla	5d
Integración de parches	3d
Administración da lista de correo	1d
Organización do Subversion	2d
Organización do Wiki	1d
Probas	5d
Documentación	5d

Táboa 6.2: Tarefas do proxecto e estimacións de tempo

Na táboa 6.2 amósase a estimación do tempo para a realización das distintas tarefas.

6.4 Asignación das tarefas

Unha vez identificados os roles que levarán a cabo o proxecto e as tarefas necesarias a realizar, é necesaria a asignación destas a cada un dos perfís.

A asignación pódese ver na táboa 6.3

Perfil	**Tarefa**	**Tempo**	**Custo total**
Analista	Análise	35h	1.750 €
Desenvolvedor	Deseño	35h	875 €
	Desenvolvemento	275h	6.875 €
Probador	Probas	35h	0 €
Mantedor	Xestión da comunidade	60h	0 €
Escritor de documentación	Documentar	35h	0 €
TOTAL:		475h	9.500 €

Táboa 6.3: Asignación de tarefas a cada perfil

Para a estimación do tempo tomouse a xornada laboral como 5 horas por día.

6.5 Consideracións finais

Coma en toda planificación hai que ter en conta que se está a falar en todo momento de estimacións. Os requisitos dos proxectos son cambiantes durante o mesmo, e máis aínda os proxectos de *Software Libre*, que se desenvolven en comunidade.

As vantaxes que ten o facer un proxecto en comunidade mana por exemplo nos custos. Temos tarefas feitas por voluntarios con custo 0. Se se sabe xestionar o proxecto e se consegue ter unha comunidade de certa entidade ó redor do proxecto, pódese dispoñer dun número de recursos importante dispoñibles.

Hai que ter en conta que isto tamén é un risco que debemos asumir. Estanse a planificar tarefas que serán feitas por voluntarios, asumindo que efectivamente as

van a facer e dedicarán as horas estimadas. Se é isto non fose así débense reasignar esas tarefas entre o resto de recursos se é asumible, se non haberá que aumentar o número de recursos o que suporía incrementar o custo do proxecto.

Capítulo 7

Conclusións e Traballo Futuro

Neste capítulo tentaremos de amosar as conclusións sobre o proxecto tomando coma referencia os obxectivos que se marcaron ó comezo. Ademais preséntanse unhas liñas de mellora nas que se pode traballar no futuro do proxecto.

7.1 Conclusións

No apartado de conclusións temos que falar por un lado das conclusións sobre o desenvolvemento, nas que se tentará analizar se o proxecto cumpriu cos obxectivos iniciais e ver como é o estado do *Gtranslator* unha vez rematado este. Por outro lado, nun proxecto de *Software Libre* e coas características concretas que ten este, de desenvolvemento dende a comunidade e para a comunidade, é unha boa idea tratar de ver se se conseguiu dalgunha maneira a reactivación da mesma.

En canto á parte de desenvolvemento, nos aspectos puramente funcionais, o *Gtranslator* ofrece agora unha interface mellorada e pensada para facilitar a labor dos usuarios. Foi moi importante contar en todo momento coa comunidade de tradutores para obter as súas opinións, xa que son eles os que van facer uso do proxecto.

Ademais agora o *Gtranslator* non é un mero editor de ficheiros *PO* senón que ofrece ó tradutor unha serie de funcionalidades que cobren os aspectos de axuda á

tradución máis importantes. É destacable, por exemplo, o sistema de memorias de tradución que facilita o aproveitamento do traballo así como a creación de estándares á hora de traducir certos termos.

Na parte máis técnica do desenvolvemento destaca sen dúbida o cambio de paradigma na programación. O uso de orientación a obxectos no programa foi un avance moi importante que fai que o código sexa moito máis modular e organizado. Ademais a combinación deste paradigma con patróns de deseño como o MVC (Model View Controller) fixo que se puidera minimizar ó acoplamento e reaproveitar código, como se puido ver no apartado 5.2.5.

Púidose comprobar tamén que a solución de *GObject* para fornecer de orientación a obxectos á linguaxe C, facilita moito a programación, permite ter un código limpo e fácil de entender á vez que ofrece as características máis habituais da orientación a obxectos. Todo isto sen perder as cualidades propias dunha linguaxe de baixo nivel.

No apartado de administración ou xestión do proxecto cabe salientar que, como proxecto de *Software Libre*, o seu modelo baséase na comunidade e polo tanto nos voluntarios. A xente que colabora nestes proxectos faino de maneira desinteresada cedendo unha parte importante do seu tempo libre. Foi moi importante coñecer de primeira man o funcionamento destas comunidades, entrando a formar parte desta de maneira escalonada.

O feito de que o desenvolvemento do *Gtranslator* estivese moi estancado antes do comezo deste proxecto, facilitou a nosa entrada, permitíndonos chegar ata o núcleo e converterse en administradores do mesmo. Isto foi unha posibilidade que se presentou que non é moi habitual nos proxectos de *Software Libre* máis consolidados, grazas á cal se aprendeu moito sobre a xestión dos voluntarios nun proxecto.

Por último, destacar que ó remate deste proxecto notouse unha corrente de ascenso positiva tanto no número de usuarios do programa, sobre todo a través do incremento do volume de reportes de erros e peticións de novas funcionalidades no *Bugzilla*, así como un incremento do fluxo de mensaxes na lista de correo. Ademais

uníronse durante o proceso novos desenvolvedores o que abre a esperanza de que este se consolide no tempo, se poidan seguir facendo melloras e sacar periodicamente novas versións.

En resumo, este proxecto permitiu ampliar as capacidades técnicas do autor, aprendendo sobre o desenvolvemento dentro da plataforma GNOME e o uso das librerías de desenvolvemento *GObject* e *GTK+*. Ademais serviu tamén para aprender a xestionar un proxecto de *Software Libre* en comunidade.

7.1.1 Comparativa de funcionalidades entre o *Gtranslator* e outras ferramentas semellantes

No seguinte apartado móstrase unha comparativa das funcionalidades que ofrecen, o tradutor, tanto o *Gtranslator* coma as demais ferramentas semellantes, algunha das cales xa foi mencionada no capítulo 2. Nesta comparativa preténdese ver o resultado que tivo a realización deste proxecto no estado do *Gtranslator* en canto ó nivel de funcionalidade que ofrece, respecto das outras ferramentas existentes.

Na táboa 7.1 pódese ver o resultado desta comparativa. Ademáis do *Gtranslator*, nas súas dúas versións, compáranse as funcionalidades que ofrecen *PoEdit*, *Kbabel* e a nova aplicación que vai sustituír esta última, chamada *Lokalize* e que está xa dispoñible na versión 4 de *KDE*.

Vese cláramente como o programa pasou de ser un mero editor que non aportaba moito máis que a posibilidade de traballar de maneira un pouco máis cómoda cos ficheiro *PO*, mostrándolle ó tradutor só as cadeas, a se converter nunha ferramenta que fornece moitas funcionalidades de axuda a tradución. A nova versión atópase no mesmo nivel, en canto a funcionaliade, que calquera otra ferrenta moderna semellante e incluso ofrece novas funcionalidades que estas aínda non incorporan.

A continuación detállanse os campos da táboa que corresponden as funcionalidades que ofrecen as distintas ferramentas:

- **TM:** Se a aplicación ofrece algún sistema de memorias de tradución.

	TM	Catálogos	Perfís	SCM	Diff	Tags	Búsq. Web	Resal. Sintaxe	Plugins	Glosarios
Gtranslator 1.1.7	Non	Non	Non	Non	Non	Non	Non	Non	Non	Non
PoEdit	Si	Si	Non	Non	Non	Non	Non	Si	Non	Non
Kbabel	Si	Si	Non	Non	Si	Si	Non	Si	Non	Non
Lokalize	Si	Si	Non	Non	Si	Si	Si	Si	Non	Si
Gtranslator 2	Si	Si	Si	Si	Si	Si	Si	Si	Si	Non

Táboa 7.1: Comparativa de funcionaliades entre o *Gtranslator* e ferramentas semellantes

- **Catálogos:** O programa ten un xestor de catálogos asociado para organizar os ficheiros *PO* e ver o seu estado.

- **Perfís:** Pódense configurar diferentes perfís con información sobre o tradutor e a língua.

- **SCM:** Se o programa ofrece integración cun Sistema de Control de Versións (SCM), por exemplo *Subversion*, co que poder descargar os ficheiros e subilos directamente ó repositorio.

- **Diff:** A aplicación ofrece a posibilidade de ver as diferencias entre duas versións dun mesmo ficheiro. Moi útil para os coordinadores dos grupos de tradución, xa que poden comprobar de forma rápida os últimos cambios que un tradutor fixo nun ficheiro.

- **Tags:** Inserir etiquetas (tags) de forma automática nas mensaxes traducidos.

- **Búsq. Web:** Se a aplicación ofrece a posibilidade de facer búsquedas nalgún servicio Web. Por exemplo a nova versión do *Gtranslator* ofrece a posibilidade de facer búsquedas de palabras ou expresións en *OpenTran*, un servicio Web que ofrece diversas memorias de tradución. O *Localize* tamén permite facer este tipo de búsquedas nas ferramentas de tradución de *Google*.

- **Resal. Sintaxe:** Resaltado de sintaxe para mostrar as etiquetas e palabras reservadas das linguaxes de programación incluídas nas mensaxes.

- **Plugins:** O programa ofrece a posibilidade de extender as súas funcionalidades mediante *Plugins*.

- **Glosarios:** Se a ferramenta permite facer búsquedas en glosarios, escoller o glosario a empregar e a posibilidade de engadir novos termos ó mesmo.

7.2 Traballo Futuro

Dentro das liñas de traballo que se poden contemplar para o futuro do proxecto destacan, por un lado, a actualización da páxina web do proxecto, para permitir os usuarios obter información sobre o estado de desenvolvemento do mesmo, noticias sobre novas funcionalidades e datas das novas versións.

Está previsto tamén facer o anuncio oficial do lanzamento da nova versión resultado deste proxecto, poñéndoa a disposición dos usuarios, para o que será necesario contactar cos mantedores do paquete para incluír a nova versión nas distribucións de *GNU/Linux* máis importantes.

Dentro tamén da parte organizativa do proxecto xorde a idea de seguir un esquema de lanzamento de novas versións constante no tempo. A primeira idea é a de sincronizar as novas versións do *Gtranslator* coas do proxecto GNOME, isto implicaría ter unha nova versión lista cada 6 meses. Ademais con isto poderíase propoñer ó *Gtranslator* para ser incluído como módulo oficial dentro de GNOME.

En canto a traballo máis técnico cara o futuro, como xa se falou no apartado 5.2.6 o xestor de catálogos necesita unha forte revisión para poder ser incluído no *Gtranslator*. É necesario incluír novas funcionalidades imprescindibles para o seu funcionamento básico, xa que por exemplo, permite crear un novo proxecto pero no permite editar un xa existente nin eliminalo.

Ademais hai xa algunhas melloras previstas para ser incluídas na versión 2.2, entre as que destacan:

- A creación de *bindings* para *Python* que permita a utilización desta linguaxe, máis atractiva e de máis alto nivel. Isto pensouse sobre todo para estender o *Gtranslator* mediante a creación de *plugins* xa que a actual versión ten integrado un xestor para eles portado do proxecto *Gedit*.

- A integración cun glosario, que permita facer búsquedas, sobre cales son as convencións para a tradución de termos que poden ter varias traducións dife-

rentes.

- Crear un sistema de importación/exportación das memorias de tradución entre aplicacións facendo uso do estándar *TMX*.

APÉNDICES

Apéndice A

Patróns

Como xa se viu nalgúns puntos desta memoria, o longo deste proxecto empregáronse varios patróns como solución de implementación de varios casos de uso.

A continuación farase unha breve introdución a cada un deles comentando a súa utilidade neste proxecto.

A.0.1 Singleton

O patrón Singleton emprégase na enxeñería do *software* para restrinxir a instanciación dunha clase a un único obxecto. A pesar disto moitas veces é empregado para restrinxir o número de instancias a máis de unha. Este patrón presenta certa controversia xa que hai xente da opinión de que non ten sentido introducir esta limitación nun sistema por considerala innecesaria, sen embargo si que ten sentido cando ter unha limitación no número de instancias dunha clase mellora a eficiencia do sistema.

E moi útil tamén cando se teñen varios fíos e só un deles debe crear a instancia do obxecto. Este problema que de forma máis clásica se arranxaríaa empregando exclusión mutua no método de creación da clase, pódese resolver facilmente usando un singleton.

Nas aplicacións emprégase de maneira habitual nas clases que teñen que controlar

o acceso a un recurso físico único como poida ser o rato, por exemplo.

A lóxica que segue o patrón para garantir a instanciación única é:

- A propia clase é a responsable de crear a instancia única.
- Permítese o acceso global a instancia mediante un método de clase.
- O construtor declárase como privado para que non poida ser instanciado directamente.

No *Gtranslator* emprégase este patrón na clase *GtranslatorApplication* para garantir que só existe unha única instancia desta clase. Desta forma conséguese garantir a exclusión mutua nos ficheiros de configuración que o programa garda no sistema de ficheiros.

A.0.2 Model-View-Controller

Este patrón empregado na enxeñería do *software* divide en tres os roles dunha aplicación. Por un lado están os datos da aplicación, por outro a interface de usuario, e por último a lóxica de control. Este patrón é empregado en moitos sistemas de diferente tipo grazas a súa inadaptabilidade. Como visión xeral pódese empregar en calquera sistema no que haxa unha interface gráfica que emita eventos que serán manexados por un controlador e haxa una sistema de almacenaxe de información.

Cada unha das súas partes defínese da seguinte maneira:

- Modelo: É a representación da información, o sistema de almacenaxe dos datos que normalmente será un sistema de xestión de bases de datos pero pode ser calquera outro tipo de sistema de xestión de información.
- Vista: Normalmente é a interface de usuario, é a encargada de interactuar directamente co usuario.

- Controlador: Responde a eventos, normalmente son accións do usuario emitidas a través da interface. Estes eventos normalmente implican cambios na vista ou no modelo que o controlador se encarga de manexar.

Pódense atopar moitas implementacións diferentes do patrón MVC, pero normalmente o esquema será como segue:

O usuario interactua directamente coa vista a través normalmente da interface de usuario, por exemplo pulsando un botón, metendo texto nun campo, etc. A continuación o controlador recibe por parte da vista un evento que lle notifica a acción solicitada polo usuario. O controlador ten que xestionar o evento normalmente a través dun *callback*.

O controlador accede tamén ó modelo para actualizalo, modificándoo segundo a acción solicitada polo usuario. As tarefas de despregar a interface de usuario recaen todas sobre a vista, e esta a súa vez obtén os datos do modelo para xerala. O modelo non debe ter coñecemento sobre a vista.

Neste proxecto o patrón MVC empregouse por exemplo e como se explica en detalle no capítulo 5 para a implementación das novas funcionalidades de xestión de perfís e memorias de tradución. Nestes casos o usuario interactua co sistema a través da interface gráfica. Os *widget* que compoñen a interface envía sinais que o controlador ten que recoller e tratar por medio dos *callbacks* que ten definidos e asociados a cada unha desas sinais. Cando esas accións implican gardar información ten que acceder o modelo para o seu almacenaxe. Neste caso temos dous tipos de modelos, unha base de datos no caso das memorias de tradución e ficheiros en *XML* para o caso dos perfís.

Apéndice B

GLib, GObject e GTK

B.0.3 GLib

GLib é unha librería de propósito xeral que se emprega para implementar moitas funcións non gráficas. Inicialmente era parte de *GTK+* que é a librería gráfica que se emprega como base do proxecto GNOME. Sen embargo decidiuse separar a parte non gráfica da librería creando *GLib*

Unha das maiores vantaxes de *GLib* é que fornece unha interface independente da plataforma que se este a empregar e que permite co código poida ser empregado en diferentes sistemas operativos. Outro aspecto de *GLib* e que dispón dunha grande gama de tipos de datos dispoñibles para o desenvolvedor.

GLib fornece tipos de datos que non están dispoñibles en C de maneira nativa, coma por exemplo listas enlazadas, táboas hash, listas dobremente enlazadas, árbores binarios, etc.

Ten un tipo de dato para as cadeas de caracteres similar ó de C++ xa que son en realidade *buffers* de texto que medran de maneira automática.

Os segmentos de memoria permiten crear seccións que teñan todas o mesmo tamaño.

Permite o uso de cachés para compartir grandes e complexas estruturas de datos que se usan por exemplo en *GTK+* para os estilos e contextos gráficos, xa que os

obxectos consumen moitos recursos.

Ademais de fornecer varios tipos de datos, *GLib*, tamén dispón de numerosos tipos de funcións. Pódense atopar funcións de manexo de ficheiros, soporte de *internacionalización*, cadeas de caracteres, advertencias, símbolos de depuración, carga dinámica de módulos...

GLib manexa tamén funcións que poden ser chamadas cando o procesador non esta a facer nada na aplicación e chamar funcións nun intervalo de tempo arbitrario.

A documentación completa de *GLib* pódese consultar en:

http://library.gnome.org/devel/glib/stable/

B.0.4 GObject

GLib Object System, ou *GObject*, é unha librería que fornece un sistema de obxectos portable e interoperabilidade transparente multi-linguaxe. *GObject* está deseñado para o seu uso directo en programas escritos en C e a través de *bindings* cara outras linguaxes.

Depende tan só de *GLib* e *libc* e é unha parte fundamental de GNOME. Emprégase en *GTK+*, *Pango*, *Atk* e na meirande parte das aplicacións e librerías de nivel superior de GNOME. Antes de *GTK+* 2.0 o código de *GObject* era parte do código base de *GTK+*.

A liberación de *GTK+* 2.0 fixo que se extraese o obxecto nun sistema de bibliotecas independentes debido a súa utilidade xeral. No proceso a meirande parte que non era parte do *toolkit* gráfico trasladouse a *GObject*, a nova base común. Hoxe en día é empregada por moitos programas non gráficos e aplicacións de servidor.

Aínda que C non é unha linguaxe preparada para a orientación a obxectos de forma nativa o recubrimento que fornece *GObject* é bastante completo e non se botan en falta ningunha das características principais deste paradigma. Entre as posibilidades que ofrece *GObject* están:

- Herdanza: imprescindible en calquera linguaxe orientada a obxectos. Permite

Tipo de dato	Descripción
gboolean	Tipo lóxico estándar. Valores TRUE ou FALSE
gpointer	Un punteiro sen tipo de datos
gconstpointer	Un punteiro sen tipo de datos. O dato non se pode modificar
gchar	Corresponde co tipo carácter de C (char)
guchar	Corresponde co tipo carácter sen signo de C (unsigned char)
gint	Corresponde con tipo enteiro de C (int)
guint	Corresponde co tipo enteiro sen signo de C (unsigned int)
gshort	Corresponde co tipo enteiro corto de C (short)
gushort	Corresponde co tipo enteiro corto sen signo de C (unsigned short)
glong	Corresponde co tipo enteiro largo de C (long)
gulong	Corresponde co tipo enteiro largo sen signo de C (unsigned long)
gint8	Enteiro con signo de 8 bits en calquera plataforma
guint8	Enteiro sen signo de 8 bits en calquera plataforma
gint16	Enteiro con signo de 16 bits en trafulqueira plataforma
guint16	Enteiro sen signo de 16 bits en calquera plataforma
gint32	Enteiro con signo de 32 bits en calquera plataforma
guint32	Enteiro sen signo de 32 bits en calquera plataforma
gint64	Enteiro con signo de 64 bits en calquera plataforma
gfloat	Corresponde o tipo flotante de C (float)
gdouble	Corresponde o tipo flotante dobre de C (double)
gsize	Valor enteiro sen signo resultado do operador sizeof
gssize	Variante con signo do tipo gsize
goffset	Valor enteiro con signo empregado para offsets de ficheiros.
GSList	Listas enlazadas
GList	Listas dobremente enlazadas
GQueue	Pilas e Colas
GHash	Tablas de dispersión
GTree	Arbores binarios balanceados
GNode	Arbores de orden n

Táboa B.1: Tipos de datos de *GLib*

crear clases que herdan funcionalidades doutras que xa existen. Isto permite crear clases máis xenéricas e despois crear outras que engaden máis funcionalidade. En *GObject* isto conséguese mediante estruturas, da seguinte maneira, se queremos que unha clase B herde de A:

```
struct ClaseA {
...
}
struct ClaseAClass {
..}

struct ClaseB {
 struct ClaseA base;
}

struct ClaseBClass {
 struct ClaseAClass base_class;
}
```

- Polimorfismo: que permite definir métodos co mesmo nome pero que se comporten de maneira diferente depende do obxecto dende o que se chamen

- Interfaces: *GObject* permite a definición de interfaces, clases abstractas, e a súa posterior implementación en clases. A maneira de facelo e mediante a definición dunha estrutura que herde de GTypeInterface.

```
struct _GTypeInterface
 {
   GType g_type;         /* iface type */
   GType g_instance_type;
```

```
 };
typedef struct
```

GObject inclúe un sistema dinámico de tipos, unha base de datos na que se van rexistrando as distintas clases. Almacénase o nome da clase e tamén as propiedades asociadas, as funcións de inicialización do tipo, o tipo base do que deriva, etc. Todo isto asociado a un identificador único, GType.

Para poder crear instancias dunha clase, é necesario que o tipo fora rexistrado previamente, ten que estar presente na base de datos de tipos. O rexistro faise usando unha estrutura chamada GTypeInfo. Cando se rexistra un novo tipo é necesaria dar información sobre o tamaño da clase, as funcións de inicialización (construtores), o tamaño das instancias, as normas de instanciación en as funcións de copia. Tódalas clases deben implementar unha serie de funcións, a función de rexistro de clase *my_object_get_type*, unha función de iniciación de clase *my_object_class_init* e unha función de inicialización das instancias da clase *my_object_init*.

B.0.5 GTK

GTK+ é a librería gráfica sobre a que se constrúe a interface gráfica de GNOME. Contén todo o necesario para o desenvolvemento das interfaces gráficas, permitindo crear todo tipo de *widget*, como botóns, etiquetas, caixas de texto, e outros máis complexos como selectores de ficheiros, cores, fontes, caixas de texto multiliña, etc.

Definición de *widget*: En termos de enxeñería do software , un *widget* é un compoñente software visible e persoalizable. E visible porque está pensado para ser empregado nas interfaces gráficas dos programas e persoalizable porque o programador pode cambiar as súas propiedades. Así conséguese a reutilización do *software*. Os *widget* combínanse para construír as interfaces gráficas de usuario. O programador adáptaos sen ter que escribir máis código que o que necesite para definir os valores das propiedades.

GTK+ segue o modelo de programación orientada a obxectos. A xerarquía comeza en *GObject*. Tódolos *widget* herdan da clase *GtkWidget* e esta a súa vez herda de *GtkObject*. A clase *GtkWidget* contén tódalas propiedades comúns ós *widget* aínda que cada un engada logo as súas propiedades particulares.

Os *widget* defínense por medio de punteiros a unha estrutura *GtkWidget*. Os *widget* teñen unha relación pai/fillo entre si. As aplicacións soen estar asociados a un *widget* de tipo ventá de nivel superior que non ten pai. Os demais todos deberán ter un. O *widget* pai chámase contedor. Cando se crea un *widget* non só chega coa creación, é necesario tamén visualizalo. Para crealos existe unha función de tipo *gtk_nome_de_widget_new* e para visualizalos unha función *gtk_widget_show*.

As interfaces gráficas dos programas en GNOME créanse combinando diferentes tipos de *widget*. Establécense retrochamadas (callbacs) asociadas a eventos que ocorren sobre o *widget* e que se denominan sinais. Os *callbacks* son funcións que conteñen código que se executará cando se chamen. O ocorrer un evento sobre un *widget* activarase o sinal correspondente e executarse o código do *callback* asociado a ese sinal.

Para que o sistema funcione correctamente emprégase o que se denomina bucle de eventos, un bucle interno en *GTK+* que vai comprobando os estados dos elementos da aplicación e mostrando os cambios que se produzan. Arráncase coa función *gtk_main* e remátase coa función *gtk_main_quit*. Unha vez arrincando cédese o control da aplicación a *GTK+*. Isto denomínase programación orientada a eventos e é típico en calquera librería de desenvolvemento gráfico.

GTK+ emprega *GDK* para visualizar os widgets. *GDK* é unha *API* de aplicacións que se sitúa por riba da *API* gráfica nativa como por exemplo Xlib ou Win32. Isto permite utilizar as aplicacións escritas con *GTK+* noutras plataformas simplemente portando *GDK*.

Apéndice C

Software Libre e GNOME

Neste apéndice preténdese introducir un pouco de historia sobre os orixenes do *Software Libre* e do proxecto GNOME. Ademais tratarase de mostrar un pouco a filosofía que hai detrás deste modelo de desenvolvemento de *software*.

C.1 Introdución

O *Software Libre* tenta devolverlle ó usuario certas liberdades que o *software* propietario lle nega. Este último a pesar de ser *software* non permite adaptalo as necesidades concretas de cada un, non permite corrixir erros e non permite distribuírllo a outra persoa.

O *Software Libre* tenta garantir ó usuario estas opcións por medio de catro liberdades, expresadas no seu día por *Richard Stallman* na súa definición de que é *Software Libre*:

- Liberdade para executar o programa en calquera sitio, con calquera propósito e para sempre.
- Liberdade para estudalo e adaptalo as nosas necesidades. Isto esixe aceso ó código fonte.

- Liberdade de redistribución, de maneira que podamos colaborar con amigos e veciños.

- Liberdade para mellorar o programa e publicar as melloras. Tamén esixe aceso ó código fonte.

Estas liberdades garántense por medio dunha licenza, como a que se mostra no apéndice D. Nela plásmanse as liberdades, pero tamén restricións compatibles con elas, coma por exemplo dar crédito ós autores orixinais se redistribuímos o programa.

Así pois non estamos a falar de *software* gratuíto, xa que o *Software Libre* pódese vender se así se desexa. Pero hai que ter en conta que a terceira liberdade permite distribuílo sen pedir diñeiro a cambio e sen ter que pedir permiso.

Para tentar explicar as motivacións detrás do *Software Libre* podemos abrir dúas liñas, por un lado as motivacións éticas, abandeiradas pola *Free Software Foundation* e por outro as pragmáticas, abandeiradas pola *Open Source Initiative*.

En canto as éticas, arguméntase que o *software* é coñecemento e polo tanto debe poder ser distribuído sen trabas. A posibilidade de modificar programas é unha forma de liberdade de expresión. Por outra parte, as motivacións máis pragmáticas falan de vantaxes técnicas e económicas. Ademais destas dúas grandes motivacións, a xente que traballa no *Software Libre* pode facelo por outras moitas razóns, que poden ir dende a simple diversión, a mera retribución económica, debida a modelos de negocio específicos.

As consecuencias que ten todo isto para os distintos sectores son:

- Para o usuario final, sexa unha empresa ou un particular, ofrécelle a posibilidade de atopar competencia nun mercado monopolista. Xa non é preciso buscar o produto coa empresa máis grande detrás, senón que se busca o que teña maior aceptación na comunidade, xa que haberá unha gran fonte de información e soporte. Ademais non se depende das estratexias do fabricantes que pode decidir unilateralmente deixar de dar soporte a un produto concreto.

Tamén se pode probar de maneira real o programa antes de decidir se adoptalo coma solución ou non e personalizalo para unhas necesidades concretas.

- Para a administración pública. É un usuario con moitas características especiais, xa que pola súa relación cos cidadáns ten que cumprir unha serie de condicións proporcionando servizos accesibles, garantir a integridade dos datos, privacidade e manténdoos en formatos abertos e estandarizados que non dependan da estratexia dunha empresa en concreto. O uso de estándares e algo habitual no mundo do *Software Libre* pero non así nas empresas.

- Para o desenvolvedor lle supón poder competir e chegar a tecnoloxía punta, aínda sendo pequeno. Pode reaproveitar traballo dos demais, competindo cun produto feito por outros sobre o que fixo algunha mellora. O mesmo tempo, sempre que a licenza sexa *Copyleft*, o competidor copiado tamén se beneficiará da mellora. Cun proxecto ben levado pódese conseguir a colaboración de moita xente. A distribución é barata e global.

- Para os integradores o *Software Libre* é o paraíso. Non hai máis caixas negras. Pode limar asperezas e integrar anacos de programas para conseguir o produto desexado.

C.2 Un pouco de historia

Aínda que o termo *Software Libre* non aparece ata os anos 80, xa na década dos 60 era habitual compartir código. Os programas distribúense co código e non había restricións prácticas. Os programas distribuíanse coas máquinas e non existía o modelo de vender *software* por separado. Cando as universidades ou empresas querían portar un programa cedíaselles o código fonte sen maior problema. Nos 70, IBM anunciou que comezaría a vender o seu *software* por separado, os clientes xa non o podían obter coma unha parte do *hardware*. O *software* comezou a ter valor por si

mesmo. Isto motivou que se pechase e nacese un novo negocio, a venta de *software*.

En 1984, *Richard Stallman*, que traballaba no laboratorio de intelixencia artificial do MIT, abandonou o seu traballo para comezar o proxecto *GNU*. El gozaba coa compartición do código e da tecnoloxía e vía como o seu propio mundo comezaba a consideralo un estraño o non querer firmar acordos de non divulgación. A súa idea era a de crear un sistema operativo completo que fose totalmente libre. Chamouno *GNU*, que é un acrónimo recursivo que significa *GNU* no é *Unix*.

Dende o principio, *Stallman*, estaba preocupado polas liberdades que terían os usuarios. Por iso escribiu a licenza GPL (Licenza Pública Xeral), probablemente a primeria licenza escrita para garantir as 4 liberdades ós usuarios.

Ademais fundou a FSF (Free Software Foundation) para conseguir fondos que dedicar ó desenvolvemento de *GNU* e protexer o *Software Libre*. Sentou as bases éticas do *Software Libre* en documentos coma *The GNU Manifesto* e *Why Software should not have owners*.

O proxecto *GNU* foi concibido como un proxecto moi estruturado que permitía o traballo en grupos pequenos. O método habitual baseábase en grupos de persoas, normalmente voluntarios, que desenvolvían algunha das ferramentas concretas que despois encaixarían completando o sistema completo. En 1990 o proxecto *GNU* estaba moi cerca de ter toda a funcionalidade que ofrecía *Unix*, sen embargo faltaba unha peza clave: o *Kernel*.

En 1991, un estudante finés chamado *Linus Torvalds* publica unha mensaxe onde di a súa intención de facer un proxecto similar a *Minix*. En Setembro libera a primeira versión, e en Marzal de 1994 aparece a versión 1.0. Dende algúns meses antes o *kernel* xa era estable e moitos desenvolvedores xa o empregaban xunto coas ferramentas de *GNU*. Nace así o concepto de distribución *GNU/Linux* que contén o *Kernel* de Linus e tódalas ferramentas do proxecto *GNU*.

C.3 GNOME

O proxecto GNOME ten como obxectivo crear un sistema de escritorio para o usuario final que sexa completo, libre, e fácil de usar. Ademais pretende ser unha plataforma moi potente para o desenvolvedor, o nome é un acrónimo en inglés que quere dicir: *GNU Network Object Model Environment*. Como parte do proxecto *GNU* todo o seu código esta baixo licenza *GNU GPL* ou *GNU LGPL*.

En 1997 discutíase sobre a liberdade da primeira alternativa de escritorio, *KDE*. Nese ano *Miguel de Icaza* viaxa a Redmon para unhas xornadas organizadas por Microfost(TM). Queda marabillado sobre a tecnoloxía de obxectos distribuídos e decide crear, xunto con *Federico Mena*, un escritorio alternativo a *KDE* que sexa totalmente libre.

En 1998 lanzaron a primeira versión, a 0.99, pero a realmente popular foi lanzada en marzo de 1999 bautizada como GNOME 1.0.No ano 2000, despois de un gran debate nas listas de correo creouse a Fundación GNOME. Esta fundación con sede en Boston, ten as seguintes atribucións:

- Coordinar as publicacións.
- Decidir que proxectos son parte de GNOME.
- É a voz oficial, ante a prensa e as organizacións, do proxecto.
- Patrocina conferencias relacionadas con GNOME.
- Representa a GNOME nas conferencias.
- Crea estándares técnicos.
- Promove o uso e desenvolvemento de GNOME.

GNOME conseguiu adentrarse na industria, hai varias empresas que participaron moi activamente no seu desenvolvemento, como casos máis significativos, Ximian, Eazel, RHAD labs de Red Hat e máis recentemente SUN Microsystems.

Actualmente GNOME atópase na versión 2.2, camiño da 2.4. A meirande parte das tecnoloxías que emprega están xa maduras. Destaca o caso de ATK, a biblioteca de accesibilidade. Unha biblioteca de clases abstractas que permite facer aplicacións accesibles, para que persoas con algunha discapacidade podan facer uso de GNOME. Foi desenvolvida principalmente por SUN que se implicou ata o punto de que no equipo de SUN que traballa en GNOME hai unha persoa invidente.

A páxina web do proxecto é: *www.gnome.org*

Apéndice D

Licenza do programa

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our

General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States

should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions. "This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network,

with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code. The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally

available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions. All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law. No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies. You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions. You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this

License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms. You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms. "Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination. You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies. You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients. Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents. A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one

or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom. If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License. Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the spe-

cial requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License. The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty. THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD

THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16. If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

Apéndice E

Acrónimos

API Application Programming Interface.

É a interface que un sistema ou biblioteca fornece a usuarios e desenvolvedores para facer uso das súas funcionalidades.

PO Portable Object.

Son ficheiros que conteñen tódalas cadeas marcadas como traducibles nun programa.

POT Portable Object Template.

Un ficheiro empregado como plantilla para a creación de ficheiros *PO*.

GTK GIMP Toolkit.

É a librería gráfica sobre a que se construe a interface do proxecto GNOME.

TMX Translation Memory Exchange

É un estándar para o intercambio de memorias de tradución entre distintos sistemas.

HIG Human Interface Guidelines

Son documentos que ofrecen ós desenvolvedores regras para construir as interfaces gráficas.

Apéndice F

Bibliografía

Bibliografía

[1] Grady Booch, James Rumbaugh, and Ivan Jacobson. *El Lenguaje Unificado de Modelado*.
Addison-Wesley, 1999.

[2] Grady Booch, James Rumbaugh, and Ivan Jacobson. *El Proceso Unificado de Desarrollo del Software*.
Addison-Wesley, 2000.

[3] Karl Fogel. *Producing Open Source Software: How to Run a Successfull Free Software Project*.
Germania, 2006.

[4] The Free Software Foundation. Manual of gettext.
`http://www.gnu.org/software/gettext/manual/gettext.html`.

[5] The Free Software Foundation. Principles of free software.
`http://www.gnu.org/philosophy/free-sw.html`.

[6] GNOME Project. GNOME human interface guidelines.
`http://library.gnome.org/devel/hig-book/stable/`.

[7] Michel Goosens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.*
Addison-Wesley, 1993.

[8] Jesús M.González Barahona Gregorio Robles and Joaquín Seoane Pascual. *Introducción al* Software Libre.
Germania, 2003.

[9] Juan José Amor Teófilo Romera Israel Herraiz, Gregorio Robles and Jesús M.González Barahona. *The Processes of Joining in Global Distributed Software Projects.*
Universidad Rey Juan Carlos, 2006.

[10] Andrew Krause. *Foundations of GTK+ Development.*
Apress, 2007.

[11] Leslie Lamport. *LaTeX: A Document Preparation System, 2nd edition.*
Addison-Wesley Professional, 1994.

[12] Havoc Pennington. *GTK+/Gnome Application Development.*
No Starch Press, 1999.

[13] Matthias Warkus. *The Official Gnome 2 Developer's Guide.*
No Starch Press, 2004.