UNIVERSIDAD
REY JUAN CARLOS

Máster Universitario en Software Libre

Curso Académico 2011/2012

Proyecto Fin de Máster

Translations in Libre Software

Autor: Laura Arjona Reina

Tutor: Dr. Gregorio Robles

Agradecimientos

A Gregorio Robles y el equipo de Libresoft en la Universidad Rey Juan Carlos,
por sus consejos, tutoría y acompañamiento en este trabajo
y la enriquecedora experiencia que ha supuesto estudiar este Máster.
A mi familia, por su apoyo y paciencia.

Dedicatoria

Para mi sobrino Darío

Contents

List of Figures

List of Tables

Summary

Libre software licenses allow to modify the program and distribute the derived work, so users of libre software may translate it to their desired language or dialect.

In this report we will analyze this legal freedom and its technical viability, provided by accessing the source code and the use of different internationalization guides, programs and platforms that libre software developers have created to help software translation.

These tools, particularly web translation frameworks, together with manual and style guides, promote the involvement of 'non-technical users' in the development project, allow to carry multiple translation projects to many languages at the same time, and promise to enhance the quality of translations. The main goal of this work is to provide a detailed view of the translation process in libre software, showing a variety of cases of projects that carry out localization tasks, the different tools that may support that work, and the people that are involved in translations in libre software projects.

Resumen

Las licencias de software libre permiten modificar el programa y distribuir la obra derivada, por lo que los usuarios de software libre pueden traducirlo al idoma o dialecto que deseen.

En este trabajo analizaremos esta libertad legal y su viabilidad técnica, propiciada mediante el acceso al código fuente y el uso de diferentes guías de inernacionalización, programas y plataformas, que los desarrolladores de software libre han creado para ayudar a la traducción del software.

Estas herramientas, en particular los entornos de traducción web, junto con los manuales y guías de estilo, promueven la implicación de 'usuarios no técnicos' en el desarrollo del proyecto, permitiendo llevar a cabo múltiples proyectos de traducción a muchos idiomas a la vez, y prometiendo mejorar la calidad de las traducciones.

El principal objetivo de este trabajo es proporcionar una visión detallada del proceso de traducción en software libre, mostrando una variedad de casos de proyectos que llevan a cabo tareas de localización, las diferentes herramientas que apoyan ese trabajo, y las personas que están implicadas en la traducción de proyectos de software libre.

Chapter 1

Introduction

1.1 Terminology

1.1.1 Internationalization, localization, translation

Software internationalization is the process to prepare a program to be adapted to different languages, regional or cultural differences without engineering changes (source code remains the same) [Con10]. This process involves tasks as separating the text strings that are showed to the user in the different interfaces so they can be translated, supporting the different character sets, using certain libraries to manage date, currency or weights and measures entry verification according to the different formats, and many other details. Internationalization is sometimes referred to by the numeronym i18n (as in: "i", followed by eighteen more letters, and then "n").

Localization is the process of adapting the software to a particular target market (a *locale*) [Con10]. This process involves tasks as translating the text files to the desired language or dialect, providing translated documentation with images (screen shots) in the target language, adapting graphics and colors, adopting local currencies, using proper forms for dates, measures and many other details. Localization is sometimes referred to by the numeronym l10n (as in: "l", followed by ten more letters, and then "n").

As we can see, translation, that is, adaptation of information in text form to a target *locale*, is part of the localization process, but not all of it.

1.1.2 Free, libre, open source software

The usage and distribution terms of any software can be read in its license.

Every software product need a written license, because software works are covered by copyright law, and this law states that all the rights (the right to use, to distribute, to modify, to link that software with others, etc) belong to the author or copyright holder. This usually apply not only to the code of the program, but also to its end-user documentation (manuals, guides).

Developers of proprietary software use licenses to state what the user can do with the program (usually, few things: for example, to use it, maybe distribute it without changes, maybe distribute it but free-of-charge). Normally they write in “negative-language”: they state all the things that the user cannot do (for example “you can use this program only for educational purposes, professional use is not allowed”).

On the other side, developers of free software state in their licenses the rights or freedoms that provide to the end-user.

The concept of **free software** was conceived in 1983 by Richard Stallman, when he started the GNU Project to develop GNU: a complete free software system, upward-compatible with Unix, and bringing back the cooperative spirit that prevailed in the computing community in earlier days: to make cooperation possible once again by removing the obstacles imposed by the owners of proprietary software [Fouc].

The **Free Software Foundation** was created to advocate for free software ideals as outlined in the **Free Software Definition** [Fou12b], which states that for a program to be said that it is free (as in freedom) software, its license should include four basic freedoms:

- Freedom to use the program, for any purpose
- Freedom to study and adapt the programs (modify)
- Freedom to distribute the program to others
- Freedom to distribute to others the modified versions of the program

It is possible to provide the four freedoms and state other clauses or restrictions in the license not affecting those freedoms; and the software will still be free software. One example can be a the license with a clause stating that if you distribute the program, modified or not, it must been distributed within the same license terms. This is what is called “copyleft”, and the most known copyleft licenses are the General Public License version 2 [Foub], and the General Public License version 3 (GPLv3) [Foua], which are widely used in the free software communities. Other free software licenses that are more "permissive" and not considered copyleft are the Apache v2.0 license, or the BSD license.

In 1998, the Open Source Initiative (OSI) was founded, coining the **Open Source Definition**, with a different writing but similar legal effects than the definition of free software, while focusing its philosophy in how these technologies, licenses, and models of development can provide economic and strategic advantages [Ini12].

In practice there are small differences between "free software" and "open source software". However the term "open source" has been historically rejected by part of the community, claiming that it misses the meaning of freedom. On the other side, other part of the community, mainly the business supporters behind the Open Source Initiative, were reluctant to use the term "free" because of the confusion between the meaning of "freedom" and the meaning of "no cost".

In order to avoid the confusion of the term "free", but keeping the meaning of freedom, "liberation", the term **libre software** has gained popularity in the last years (and with it, the acronym **FLOSS**: Free, Libre, Open Source Software). In this document, "libre software" will be used to refer to any code that conforms either to the definition of "free software" (according to the Free Software Foundation) or "open source software" (according to the Open Source Initiative).

Software licenses and localization

The "limitations" of privative licenses to end-users of the software creates a situation in which, without permission from the copyright holder, it is illegal to translate the software to other languages different than the provided in the software release. It is also forbidden to make internationalization improvements to the software, if certain parts of it need to be modified in order to be suitable for localization. In addition to this, while there are cases of "privative software" translated to other languages by users (for example, the translation of **Twitter**, which is managed in the website `http://translate.twttr.com` and open to external contributions), the common case (of privative software) is that software development company or organization controls the translation process, whether providing themselves the translations, or subcontracting them to a professional translation company.

The free software licenses create a totally different scenario for software localization. As any modification of the software is allowed, end-users may translate the software to their desired language or dialect, without need of permission from the copyright holder. This legal viability is usually viewed as an advantage of libre software versus proprietary alternatives, both by users and by developers, so, although it is not a requirement and we can find free software created under a closed development model, it is common that both communities collaborate in the translation processes: for developers, is a way to increase the dissemination of the program, attract new users and new possible contributors to improve the project. For users, is a way to guarantee the availability of the software in their language, despite of the interests of the developer group or company.

In figure 1.1 we can find a diagram summarizing this two different scenarios.

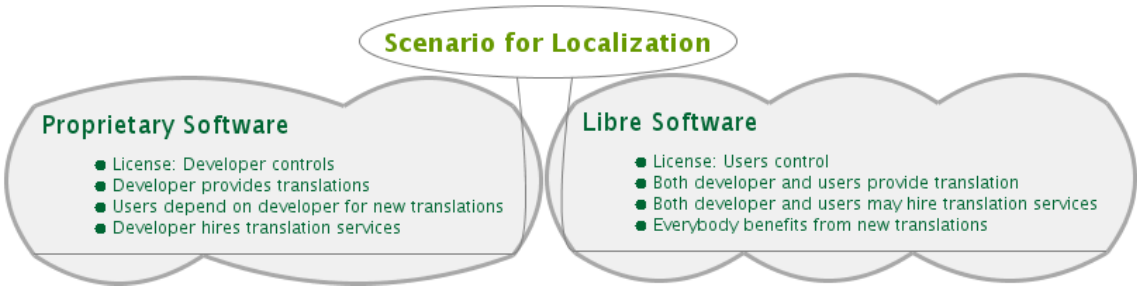

Scenario for Localization
Proprietary Software
License: Developer controls
Developer provides translations
Users depend on developer for new translations
Developer hires translation services
Libre Software
License: Users control
Both developer and users provide translation
Both developer and users may hire translation services
Everybody benefits from new translations

Figure 1.1: Software Licenses and Localization

1.1.3 Free culture, freedom definition, creative commons

As explained above, software is considered a cultural work, and after the success of the legal hacking to copyright laws to bring freedom to the software users, a free culture movement was born, which pretends to extend that freedoms to the "users" or "receivers" of other kind of cultural work (such as literature, films, multimedia contents…). A wider definition of free, libre cultural works has been designed with the name of **Freedom Defined** [Def08] and on the other side, a set of licenses known as **Creative Commons** [Com12] have been written, in order to show to the authors or copyright holders their possibilities for retaining certain rights while bringing more freedom to their audience (for redistributing the work, modify it, translate it, or make business with it).

Although Creative Commons is a set of 6 different licenses (a three-layer design of the attribution (CC-BY), non-commercial (CC-NC), non-derivative (CC-ND) and share-alike (CC-SA) components that can be combined) plus one more, CC-0, equivalent to public domain, only three of them (CC-BY, CC-BY-SA and CC-0 for public domain) are considered "free" according to the Freedom Defined definition (or the philosophy behind the Free Software Movement represented on the Free Software Foundation). However, we cannot avoid the great advance that they represent (if we compare them with the restrictions of the "standard copyright" application) and their role in spreading the word of openness and free access to culture.

Being the free software licenses best adequate to guarantee the freedoms to the user of computer programs, Freedom Defined and Creative Commons licenses may also be used (and in fact, they are used) within the scope of software projects, licensing with them for related work as documentation, artwork, multimedia components or derived knowledge from translation work as translation memories or glossaries.

1.2 About this document

1.2.1 Structure of the document

In this chapter I am introducing the main concepts needed to understand the framework of localization in libre software projects. In chapter 2 the goals of this work are presented. Chapter 3 is devoted to analyze how localization and translation is performed in libre software projects, giving some brief examples of each phase and the tools and tasks involved. Chapter 4 shows the benefits that localization brings for the libre software projects, its stakeholders and the society in general, and chapter 5 explains some of the problems and challenges to face when "going international". In addition to the already presented examples in former chapters, detailed case studies are analyzed in chapter 6. Finally, conclusions are drawn in chapter 7, including lessons learned and future work.

Four Appendix are provided: Appendix A is a list of libre software tools to help localization (the ones mentioned in former chapters, and others listed for further reference). A set of interviews to translators is provided in Appendix B, and my own experience participating in the localization of several projects is included in Appendix C. Finally, Appendix D explain the possibilities of getting information about how localization is managed by mining the software repositories, using the project Tux Paint as example.

1.2.2 Scope

Internationalization and localization are complex processes involving many different aspects. This document will focus on explaining some tools, frameworks and workflows followed by different libre software projects, without entering in technical details as how the used tools are developed, their complete feature list, particularities of target locales as text orientation or character sets... With respect of localization, we will focus on translation (that is, localization of texts). The intention is not thoroughly compare the different approaches for internationalization and localization, but to show the wide diversity available (and not only available, but susceptible of improvements because of their nature of being libre software) for carrying out internationalization and localization tasks.

1.2.3 Methodology

I have gathered the information presented in this work from different sources. On one side, we have the public available sources as websites of the projects, documentation, mailing lists for translators, and related literature showed in the Bibliography. On the other side, I have contacted personally with some free software translators, and also provided my own experience contributing to the translation to Spanish of several libre software projects.

Chapter 2

Goals and objectives

2.1 Goals

The main goal of this work is to understand how translation is carried out in libre software projects, the different tools available for each need of the process, and why localization and translation is a key aspect for the sustainability of the libre software projects.

Another goal is to present personal stories related with the participation in the localization of libre software projects, so any person thinking about getting involved in this kind of task may get a clear image of what to expect and ensure a successful and gratifying experience.

2.2 Objectives

In order to achieve these goals I have set the following objectives:

1. Explain the different phases of localization process.

2. Analyze the benefits and counterparts of using open and collaborative development models for translation and localization.

3. Provide case studies of libre software projects and tools

4. Present the experience of certain individuals involved in localization tasks.

2.2.1 Explain the phases of localization process

- Define each phase of localization, describing tasks, tools and people involved.
- Present new trends, if any, for each phase.
- Provide several examples of the different approaches.

2.2.2 Analyze the benefits and counterparts

- Analyze the benefits for the libre software community.
- Analyze the benefits for other stakeholders and society in general.
- Explore the possible vulnerabilities and threats to take into account.

2.2.3 Provide case studies of libre software projects and tools

- Gather in-depth information about how localization is carried out for several libre software projects.
- Show traditional, widely used tools for localization
- Introduce new tools that are becoming trend in localization processes

2.2.4 Present personal experiences

- Gather personal experiences of people involved in translation teams of libre software projects.
- Present the information in a comprehensive way, thinking on people that are not involved (yet) in libre software projects.

Chapter 3

Localization in libre software projects

3.1 Localization workflow

There is not a standard, accepted workflow for localization followed by all the libre software communities, nor by the professional translation communities (the object of translation being software or not).

OASIS, the Organization for the Advancement of Structured Information Standards, approved in December 2009 the Open Architecture for XML Authoring and Localization (OAXAL), encouraging the development of an open Standards approach to XML Authoring and Localization [ftAoSISO09].

Among many other processes and proposals, OAXAL sets up a suitable translation workflow, comprising the following steps:

- Identification of the text to be localized
- Segmentation of the text into sentences if required
- Matching of the sentences with previous translated versions of the document if possible
- Translation of the text
- QA/Post-editing of the translated text
- Merging/recreation of the original document in the target language

Figure 3.1 reproduces this workflow.

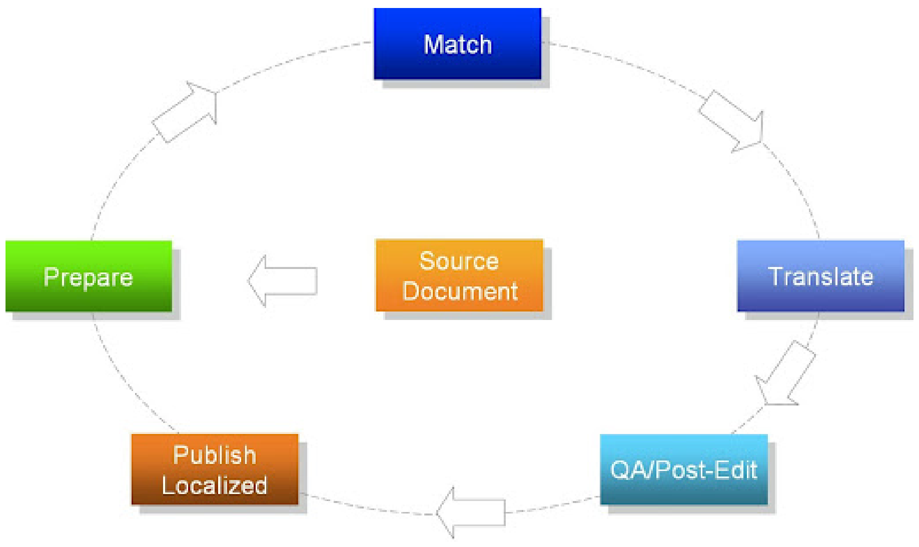


Figure 3.1: Document Localization Life-cycle

While OAXAL open specification is still to be widely accepted in the localization world and business environments, it represents a comprehensive, efficient, and cost-effective model regarding the authoring and translation aspects of XML publishing. The localization workflow may be applied in the libre software communities in order to easily integrate the work of professional translators, the same than other parts of developments are also getting professionalized.

We can adapt the different phases of the localization life-cycle to the process of localizing a software project so we can obtain the following workflow:

- Prepare: Identification of the objects to be localized
- Internationalize: Separate texts from source code, set up the accepted type of documents for translation.
- Localize(1): Match the objects to be translated with previous translated versions of the project if possible
- Localize(2): Translation of the pending strings
- QA/Post-editing of the translated objects
- Publish: Update the original project integrating the target language

Although not all the libre software projects care well about all the process, we will see in the following sections how each phase of the process can be carried out and some examples of successful cases.

3.2 Prepare: Defining objects to be localized

A software project is not only the source code, there are many artifacts susceptible of translation or localization. The first step of the localization workflow is to define the objects to be localized:

- Source code: hard-coded strings, comments, interface-related strings (this may come in .po files, .php, .ini, .xml files...)
- Text files that "come" with the binary release: README, license, disclaimers, changelog, credits files.
- Documentation: this can be available in text files, DOC, ODT, RTF, HTML files, Wiki pages, PDF...
- Images. audio, video subtitles
- Marketing tools: flyers, banners to put in websites for promotion
- Other

The people involved in this task is the whole community: users may suggest that the software needs to be translated to a new language, or complain about incomplete translations, or translated text but not screenshots for example. Internationalization team (i18n-team in advance) should receive that user feedback and communicate it to the development team and localization teams, prioritize and coordinate the steps involved in this task. The main tools used for defining the localizable objects are the communication channels of the project: mailing lists, issue tracker, website (where at least a brief list of the localizable objects is needed along with the help to possible contributors). The results of this task is, as explained before, a detailed list of the objects to be localized.

3.2.1 Some examples

The KDE desktop environment, in its localization page[1], states that the localizable objects for this project are the Applications and the Documentation. More information about KDE internationalization and localization process can be found in section 6.4.2.

The Debian community in its internationalization page[2] defines which kind of objects are localizable within their project:

- Installation system (the Debian installer)
- Debian documentation
- Debian web pages
- Debian wiki pages
- Debian specific packages

Since Debian is a GNU/Linux distribution, which basically is a collection of an operating system, software components and applications adapted to be used altogether in a specific hardware architecture, it encourages people willing to translate other kind of objects to contribute their suggestions or translations to the *upstream* projects. More information about Debian internationalization and localization process can be found in section 6.4.3.

3.3 Internationalize: Setting up the i18n platform

A project's internationalization team is a bridge between developers and translators.

On the "developers" side, the internationalization team reviews all the artifacts susceptible of localization and suggests **accepted formats** for all of them, and also creates or suggests **coding guidelines** in order to ease the localization task. For example, if the project includes images, it is easier to change the text included in the images if they are in a vector format as SVG, better than in a "photographic format" as JPG.

[1]`http://l10n.kde.org/`

[2]`http://www.debian.org/international/index.en.html`

On the “translators” side, the internationalization team creates and/or manages the **common infrastructure used by all the localization teams**: documentation for translators, communication channels, coordination, translation server or platform, and development of specific tools.

Internationalization team also decides about the timing of the translation process, according it with the release schedule.

In addition to i18n-team, other contributor roles in the project may be needed to carry out this task: infrastructure team to set up the platform or mechanism to keep up to date internationalization files, and developers, to discuss and accept the coding guidelines.

The tools used in this task are the communication channels, tools as *gettext* to generate i18files such as POT templates, scripts or robots to periodically run to generate the templates so they are up to date all the time.

The products derived from this task are the i18n files (such as POT templates for example), scripts, documentation for developers and translators.

3.3.1 Used and accepted file formats for localization

There are a number of standards to be applied to the task of translation and localization, that determine the use of certain file formats to ensure interoperability, that is, the ability to use different tools to do the translation work.

If we look at the complete process of localization in general (not only related to software localization), considering the OAXAL framework [ftAoSISO09], there are certain standard file formats used in localization, as tmx, tbx, and xliff.

However, these standards are quite recent and the authors of libre software were worried about the localization from the very beginning. So, much before the OAXAL framework was designed, other kind of file formats became *de facto* standards as the “PO” (Portable Object) file format for software translations.

The GNU Project [Fouc] took the tool Gettext [Fou10], which first implementation is from Sun Inc[3] and used it to help the internationalization of the GNU project software. Gettext explores the source code and extracts all the “printed” strings into a POT file.

With time the use of PO files have become a *de facto* standard for libre software internationalization. There are other file formats used to hold the translatable strings; one of them is

[3] http://compgroups.net/comp.unix.solaris/History-of-gettext-et-al

the XLIFF (XML Localization Interchange File Format), which version 1.2 was approved by OASIS in February 2008 [ftAoSISO08]. XLIFF consist in an XML specification to hold translatable strings along with metadata. This format is very used in professional translation and not only in software internationalization.

Nowadays we can find multiple file formats for internationalization of software, documentation and other elements in any libre software development project. For example, Android Operative System uses "Android Resources" to make easy the localization of Android applications. More information can be found in `http://developer.android.com/guide/topics/resources/localization.html`.

In table 3.1 there is a summary of the characteristics of different internationalization formats [Tra12].

3.3.2 Collaborative models and tools to approach l10n tasks

The concept of free software was born to retrieve the spirit of collaboration, so it is very common, specially in the last years, to find libre software projects that build a collaborative web infrastructure to carry out localization tasks. The advantages of **crowdsourcing** for the project include [CS12]:

- Lower user access barrier to localization, and receive the user experience that deals everyday with the software
- Lower the costs of translation
- Translation work and integration is fast
- The process enhances user identification and involvement with the project
- Peer review and editing of translation is built into the system

It is task of the internationalization team to study and decide if the localization of the project is going to be driven by a crowdtranslation platform or other kind of tools and mechanisms.

In the first case, some projects decide to use one an external web based collaborative environment for all the localization teams to be coordinated (for example, OpenStack project is

Name	**File extension**	**Notes**
Android Resources	.xml	XML based format. 3 types of entries: string, string-array and plurals.
Apple strings files	.strings	UTF-16
Desktop files	.desktop	Configuration files describing how a program appears in menu, etc. It is widely used by KDE and Gnome
Gettext based formats	.po, .pot	Widely used in libre software projects. Many tools to convert to/from PO files
Java Properties	.properties	ISO-8859-1
Joomla! INI files	.ini	Joomla localization files
Maker Interchange Format	.mif	Markup language for Adobe's FrameMaker documentation.
Mozilla DTD	.dtd	List of entities that need localization before used in XUL files.
PHP files	.php	3 types: PHP Array, Define and Alternative Array
Plain text	.txt	One text file for each translated language.
Property list files	.plist	XML based, used to store serialized objects, in Mac OS X, iOS, NeXTSTEP, GNUstep.
Qt Linguist	.ts	XML based
Subtitle formats	.srt, .sub, .sbv	SubRip, SubViewer and Youtube captions
Wiki markup	.wiki	Syntax and keywords used by the MediaWiki software and other Wiki packages
Windows installer projects	.wxl	XML based, used to build .msi installation files
Windows resource files	.resx	.NET and Windows Phone applications
HTML, XHTML	.html, .xhtml	An HTML parser may extract the text to translate.
XLIFF	.xlf, .xliff, .xml	XML Localization Interchange File Format, XML-based, created to standardize localization.
YAML	.yml, .yaml	YAML translation dictionaries

Table 3.1: Internationalization file formats

discussing about which platform to use: Pootle, Launchpad or Transifex, for handling the localization of the documentation, and includes some comparative charts and assessment in their wiki page about translations[4]). The approach can be to use an external server (hosted outside the project) where the chosen platform is running (this case will be discussed in next sections as does not require internationalization effort), or to deploy an instance of that software using one of the community servers, in which case the infrastructure team of the project, in coordination with the internationalization team, should care that the server is up to date and running all the time, in order to offer the service to the localization teams.

On the other side, some projects decide to develop their own web-based infrastructure, which is usually also released as free software, and depending on the focus of the platform (more oriented to the project for which was created, or more standardized), may be used later for other projects too.

3.3.3 Some examples

The Android Open Source Project includes a *Localization document*[5] explaining the process that Android developers should follow in order to make their applications localizable. There is also available the *Hello, L10N* tutorial[6] providing an example of how to build a simple localized application that uses locale-specific resources [RR12].

The KDE internationalization build system[7] basically consists in a script running in the KDE source code repository server, which extracts the localizable strings in the KDE source code and generates the localization templates periodically. This way developers only need to care about writing code with localization in mind, and localizers will always find an update template with the last strings to translate as developers introduce changes in the system.

Many different and popular projects decide to host their translation in an external service. For example Wordpress uses the official Pootle translation server[8], and the FreedomBox Foundation and the Fedora community use Transifex[9] for the localization of their websites.

[4]http://wiki.openstack.org/Translations
[5] http://developer.android.com/guide/topics/resources/localization.html
[6] http://developer.android.com/resources/tutorials/localization/index.html
[7]http://techbase.kde.org/Development/Tutorials/Localization/i18n_Build_Systems
[8]http://pootle.locamotion.org/
[9]http://transifex.net

About project-tailored web based localization platforms, we can find Damned Lies[10] which is the translation platform developed and used by the GNOME project, or the Drupal localization website[11] which is used in the Drupal CMS project.

3.4 Localize: Setting up the l10n platform

Generally there are independent teams for each language and they agree on using locale-specific tools, formats, guidelines... Each localization team is responsible to translate, review, and upload changes to the source code repository. The tools used vary from traditional tools as Poedit, or plain text editors, to sophisticated web platforms that integrate translation and revision processes, and sometimes even automatically committing the changes to the SCM repository.

The products result of this task may be localization files, translation guidelines, glossaries, language-specific conventions...

3.4.1 Web platforms for supporting translation

As I explained before, in the last years a number of web based translation platforms have been developed in order to get the advantages of internet based collaboration, automate certain tasks and also lower barriers to external contributions (crowdtranslation).

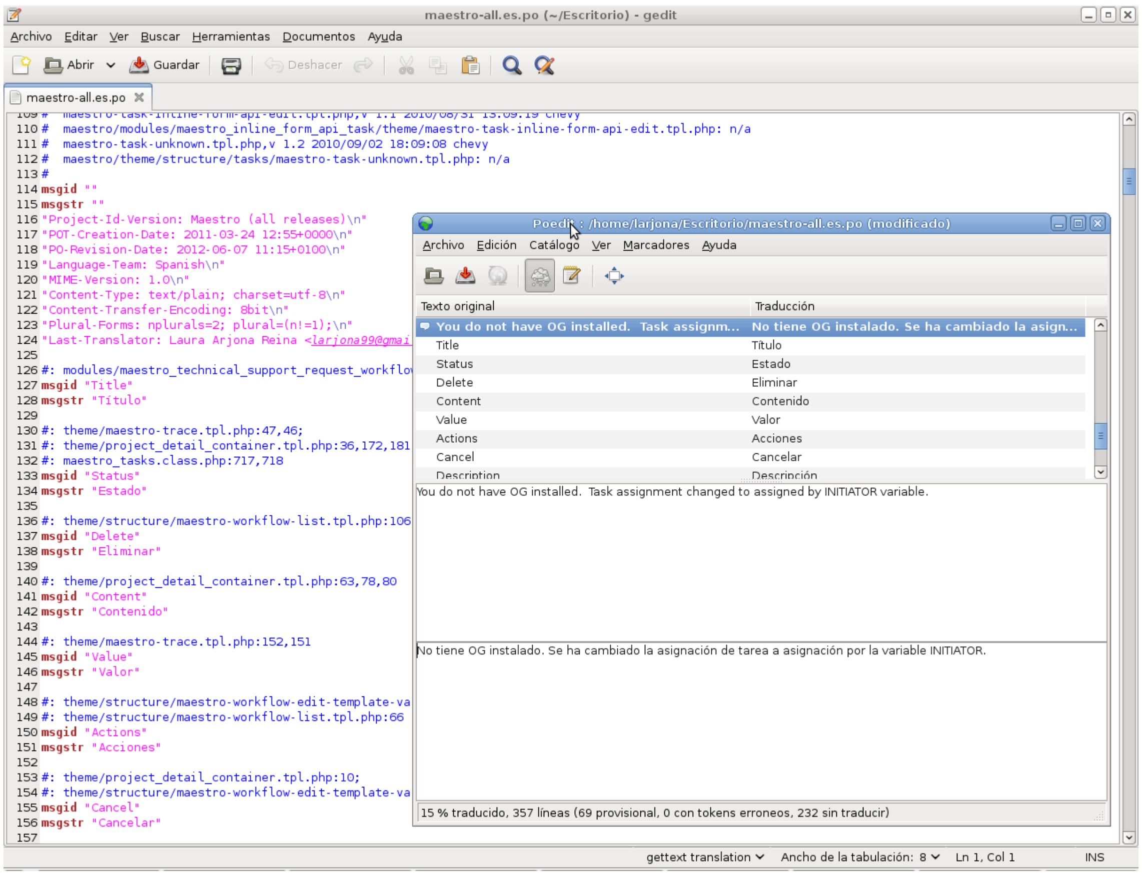
Despite of the internationalization team of the project recommending or not to use one of these web platform, each localization team is usually free to use the tools that they want. For these reasons, sometimes we find that some language teams decide to use a certain, local tool, and other localizations of the same project are driven in a (usually external) web based platform for translations.

Sometimes this situation is found in early stages of the project, and once that several tools are tried the internationalization team together with the different language teams decide if recommending one of the available tools or keep the l10n teams to choose depending on their circumstances.

[10]http://l10n.gnome.org/
[11]http://localize.drupal.org

3.4.2 Some examples

Poedit[12] is a cross-platform editor por `.po` files (gettext catalogs). It allows to configure the tool with the information related to the translator (name and email) and the environment, and every time a file is translated and saved, that information is included in the file. In figure 3.2 we can see a file opened for translation with a text editor, and the same file opened with Poedit.


maestro-all.es.po (~/Escritorio) - gedit
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
Abrir Guardar Deshacer
maestro-all.es.po
Poedit : /home/larjona/Escritorio/maestro-all.es.po (modificado)
Archivo Edición Catálogo Ver Marcadores Ayuda
gettext translation Ancho de la tabulación: 8 Ln 1, Col 1 INS

```
#  maestro-task-inline-form-api-edit.tpl.php,v 1.1 2010/08/31 13:09:19 chevy
#  maestro/modules/maestro_inline_form_api_task/theme/maestro-task-inline-form-api-edit.tpl.php: n/a
#  maestro-task-unknown.tpl.php,v 1.2 2010/09/02 18:09:08 chevy
#  maestro/theme/structure/tasks/maestro-task-unknown.tpl.php: n/a
#
msgid ""
msgstr ""
"Project-Id-Version: Maestro (all releases)\n"
"POT-Creation-Date: 2011-03-24 12:55+0000\n"
"PO-Revision-Date: 2012-06-07 11:15+0100\n"
"Language-Team: Spanish\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=utf-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=(n!=1);\n"
"Last-Translator: Laura Arjona Reina <larjona99@gmai

#: modules/maestro_technical_support_request_workflo
msgid "Title"
msgstr "Título"

#: theme/maestro-trace.tpl.php:47,46;
#: theme/project_detail_container.tpl.php:36,172,181
#: maestro_tasks.class.php:717,718
msgid "Status"
msgstr "Estado"

#: theme/structure/maestro-workflow-list.tpl.php:106
msgid "Delete"
msgstr "Eliminar"

#: theme/project_detail_container.tpl.php:63,78,80
msgid "Content"
msgstr "Contenido"

#: theme/maestro-trace.tpl.php:152,151
msgid "Value"
msgstr "Valor"

#: theme/structure/maestro-workflow-edit-template-va
#: theme/structure/maestro-workflow-list.tpl.php:66
msgid "Actions"
msgstr "Acciones"

#: theme/project_detail_container.tpl.php:10;
#: theme/structure/maestro-workflow-edit-template-va
msgid "Cancel"
msgstr "Cancelar"

```

Texto original	Traducción
You do not have OG installed. Task assignm...	No tiene OG instalado. Se ha cambiado la asign...
Title	Título
Status	Estado
Delete	Eliminar
Content	Contenido
Value	Valor
Actions	Acciones
Cancel	Cancelar
Description	Descripción

You do not have OG installed. Task assignment changed to assigned by INITIATOR variable.

No tiene OG instalado. Se ha cambiado la asignación de tarea a asignación por la variable INITIATOR.

15 % traducido, 357 líneas (69 provisional, 0 con tokens erroneos, 232 sin traducir)

Figure 3.2: A gettext catalog opened with a text editor, and with Poedit

Virtaal[13] is a more modern translation tool, which allows working with XLIFF files [MW11]. In figure 3.3 we can see an XLIFF sample file opened with a text editor and with Virtaal.

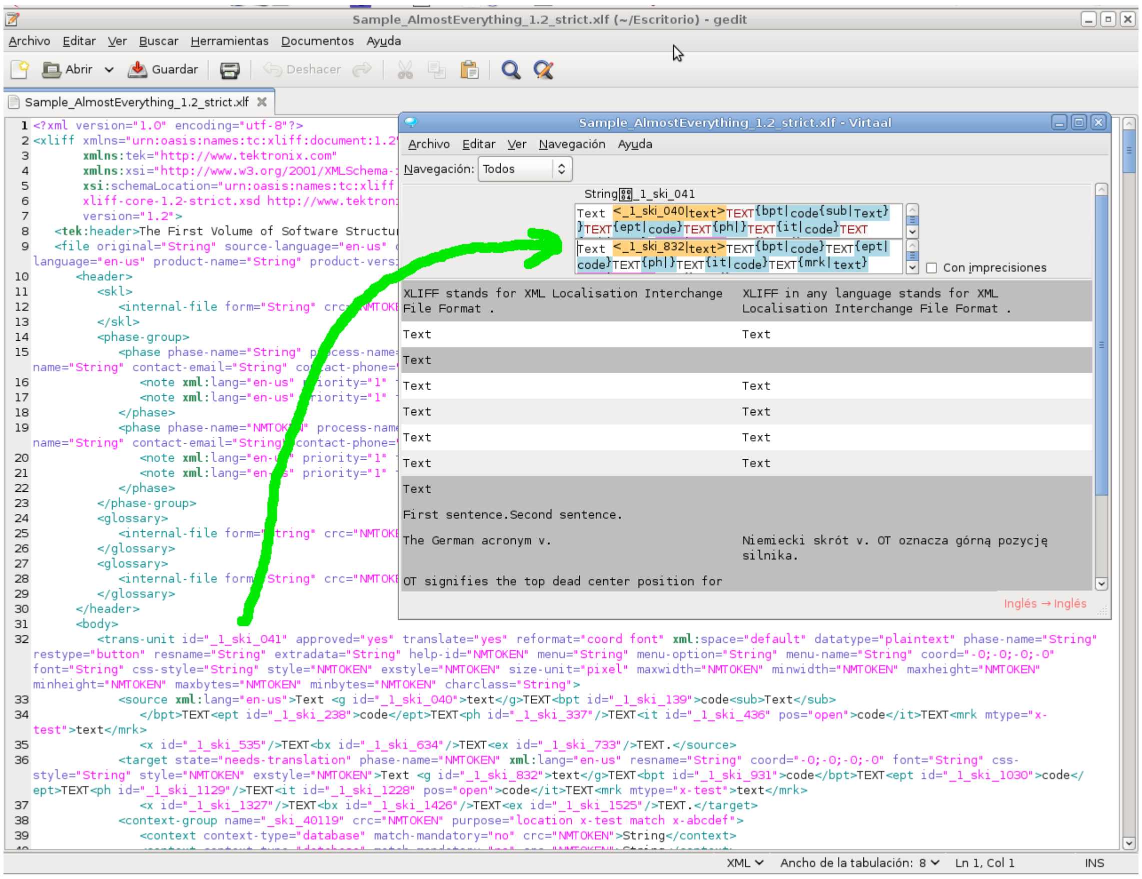
Probably the older example of a web based translation platform is **Pootle**[14], which is built on top of the Translate Toolkit[15], a set of scripts and tools to help localizers carry out their

[12]`http://www.poedit.net/`

[13]`http://translate.sourceforge.net/wiki/virtaal`

[14]`http://translate.sourceforge.net/wiki/pootle/`

[15]`http://translate.sourceforge.net`


Sample_AlmostEverything_1.2_strict.xlf (~/Escritorio) - gedit
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
Abrir Guardar Deshacer
Sample_AlmostEverything_1.2_strict.xlf

```
<?xml version="1.0" encoding="utf-8"?>
<xliff xmlns="urn:oasis:names:tc:xliff:document:1.2"
       xmlns:tek="http://www.tektronix.com"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-i
       xsi:schemaLocation="urn:oasis:names:tc:xliff
       xliff-core-1.2-strict.xsd http://www.tektroni
       version="1.2">
   <tek:header>The First Volume of Software Structu
   <file original="String" source-language="en-us" c
language="en-us" product-name="String" product-vers
      <header>
         <skl>
            <internal-file form="String" crc="NMTOKE
         </skl>
         <phase-group>
            <phase phase-name="String" process-name=
name="String" contact-email="String" contact-phone="
               <note xml:lang="en-us" priority="1"
               <note xml:lang="en-us" priority="1"
            </phase>
            <phase phase-name="NMTOKEN" process-name
name="String" contact-email="String" contact-phone="
               <note xml:lang="en-us" priority="1"
               <note xml:lang="en-us" priority="1"
            </phase>
         </phase-group>
         <glossary>
            <internal-file form="String" crc="NMTOKE
         </glossary>
         <glossary>
            <internal-file form="String" crc="NMTOKE
         </glossary>
      </header>
      <body>
         <trans-unit id="_1_ski_041" approved="yes" translate="yes" reformat="coord font" xml:space="default" datatype="plaintext" phase-name="String"
restype="button" resname="String" extradata="String" help-id="NMTOKEN" menu="String" menu-option="String" menu-name="String" coord="-0;-0;-0;-0"
font="String" css-style="String" style="NMTOKEN" exstyle="NMTOKEN" size-unit="pixel" maxwidth="NMTOKEN" minwidth="NMTOKEN" maxheight="NMTOKEN"
minheight="NMTOKEN" maxbytes="NMTOKEN" minbytes="NMTOKEN" charclass="String">
            <source xml:lang="en-us">Text <g id="_1_ski_040">text</g>TEXT<bpt id="_1_ski_139">code<sub>Text</sub>
               </bpt>TEXT<ept id="_1_ski_238">code</ept>TEXT<ph id="_1_ski_337"/>TEXT<it id="_1_ski_436" pos="open">code</it>TEXT<mrk mtype="x-
test">text</mrk>
               <x id="_1_ski_535"/>TEXT<bx id="_1_ski_634"/>TEXT<ex id="_1_ski_733"/>TEXT.</source>
            <target state="needs-translation" phase-name="NMTOKEN" xml:lang="en-us" resname="String" coord="-0;-0;-0;-0" font="String" css-
style="String" style="NMTOKEN" exstyle="NMTOKEN">Text <g id="_1_ski_832">text</g>TEXT<bpt id="_1_ski_931">code</bpt>TEXT<ept id="_1_ski_1030">code</
ept>TEXT<ph id="_1_ski_1129"/>TEXT<it id="_1_ski_1228" pos="open">code</it>TEXT<mrk mtype="x-test">text</mrk>
               <x id="_1_ski_1327"/>TEXT<bx id="_1_ski_1426"/>TEXT<ex id="_1_ski_1525"/>TEXT.</target>
            <context-group name="_ski_40119" crc="NMTOKEN" purpose="location x-test match x-abcdef">
               <context context-type="database" match-mandatory="no" crc="NMTOKEN">String</context>
```

XML Ancho de la tabulación: 8 Ln 1, Col 1 INS
Sample_AlmostEverything_1.2_strict.xlf - Virtaal
Archivo Editar Ver Navegación Ayuda
Navegación: Todos
String 1_ski_041
Con imprecisiones
Inglés → Inglés

```
Text <_1_ski_040|text>TEXT{bpt|code{sub|Text}}TEXT{ept|code}TEXT{ph|}TEXT{it|code}TEXT
Text <_1_ski_832|text>TEXT{bpt|code}TEXT{ept|code}TEXT{ph|}TEXT{it|code}TEXT{mrk|text}
```

XLIFF stands for XML Localisation Interchange File Format .	XLIFF in any language stands for XML Localisation Interchange File Format .
Text	Text
Text	
Text	Text
Text	Text
Text	Text
Text	Text
Text	
First sentence.Second sentence.	
The German acronym v.	Niemiecki skrót v. OT oznacza górną pozycję silnika.
OT signifies the top dead center position for	

Figure 3.3: An XLIFF sample file opened with a text editor and with Virtaal

tasks. Pootle allows online translation, work assignment, gives statistics and includes some revision checks. **Launchpad**, which is a complete software forge including a powerful web based translation system, was developed by Canonical to handle the Ubuntu development, and initially released under a privative license, but later liberated under the GNU Affero General Public License, version 3. **Transifex** is a newer platform which enhances the translation memory support, and Weblate is the last one coming to the neighborhood, developed by Michal Cihar (PHPmyAdmin project leader), and introducing a better integration with the Git control version system.

In section 6 several web based translation platforms are studied in-depth.

3.5 Maintenance and Quality Assurance

This is not the last step, but several tasks that affect the whole process, including the revision of translations (in l10n teams, before committing the changes), bug submitting and tracking for translation suggestions or problems, extending the number of different kind of objects to be localizable, or targeting new languages.

The people involved in this task is the whole software community: users, l10n teams, i18teams and developers.

The results of this task are not new files, but the improvement of the existing products, and the update of the documentation and information related to i18n and l10n.

3.5.1 Some examples

The Translate Toolkit[16] is a collection of useful programs for localization, and a powerful API for programmers of localization tools. The Toolkit includes programs to help check, validate, merge and extract messages from localizations. Among them we can find:

- **poconflicts** - extract messages that have conflicting translation
- **pofilter** - filter PO files to find common errors using a number of tests
- **pogrep** - find strings in PO files

Pology[17] is a Python library and collection of command-line tools for in-depth processing of PO files. Some of the prominent elements of Pology functionality include examination and in-place modification of collections of PO files (the **posieve** command), and custom translation validation, by rules written by users for particular languages and projects, and applied in various contexts. Pology distribution contains internal validation rules for some languages and projects, and more can be contributed at any time.

3.6 Who are the translators?

In previous sections I have briefly drawn the roles involved in each phase of the localization process: users, localizers (translators), internationalization team, developers. But what kind of people decide to join these internationalization and localization teams? Well, it depends on the project, but we can say that although localization is a very specific task which involves specific workflows and tools, no particular skills are required to work on it.

[16]`translate.sourceforge.net`
[17]`http://pology.nedohodnik.net/`

It is very common that each localization team have their own mailing list or forum where they communicate in the "target" language, so the access barrier for new participants is also lower.

In addition to this, localization is a low-risk task (few critical bugs arise from poor or wrong localization) and it has high granularity: the minimum "translatable" unit is the sentence or sometimes even a single word. All these particularities make localization an attractive start point for people willing to get involved in a free software project.

The motivations and benefits of caring about localization are multiple (see section 4) so we can find translators from very different affiliation:

- **Developers**: Whether reasons are globalization and migratory movements, or the collaborative nature of open development model, which is followed by many libre software projects, it is common to find people participating in the project who know more other languages. Therefore it is also common that the need/opportunity for localization "appears" in a natural way in early stages of development: with little additional effort and many times willingly, developers get involved so their project release is localized to their known languages[18].

- **Users (volunteers)**: They deal with the application each day so they know its interface and the contexts of each message. In many cases they don't know how to work with the localization tools but being localization files text files, this is rarely a handicap in order to contribute translations[19].

- **Public administrations**: they may be interested in promoting the usage of a certain software but they need it to be localized to the regional language. In other cases is a way to promote minority languages, providing software translated to that languages.

- **Professional translators (as individuals)**: for the professional or trainee translators, participating in a libre software community allows them to be in contact with cutting-edge trends in localization as **crowdsourcing - crowdtranslation**, use accessible Computer

[18]For example, Jernej Simončič (Slovene Open Source developer) releases GIMP and GTK+ installers for Windows in English and in Slovene (http://sourceforge.net/projects/gimp-win/)

[19]This is my own situation, see Appendix C

Assisted Translation applications with similar characteristics to the ones used in professional environments, practice with the different objects to be localized, and assume different roles in the translation workflow. They may contribute to free software projects as translation training or to improve their experience portfolio. They may be contracted by companies or public administrations willing a software to be translated to a certain language.

- **Companies**: The software may be part of a complete system or solution developed or deployed by a company, and they want the whole system to be localized to the locale of their target market.

Chapter 4

Benefits

4.1 Benefits for the libre software community

Using libre software licenses and an open development model to carry out localization task brings many benefits for the libre software community.

4.1.1 Extra features with not much extra workload

Developers knowing other languages, specially if they work on the program interface, are able to translate the software to their own language with little extra work. This approach enriches the software project, as each language can be seen as a new feature of the product.

4.1.2 Sustainability: Increase market share and demand for services

Globalization of markets and development of information and communication technologies have meant an increment in the needs of linguistic management for any far-reaching social or business initiative. Having the software translated to many languages helps spreading its use so it is a very powerful asset to gain market share and also increase sustainability of the business since there will be more demand on services related to the software.

4.1.3 Ease the integration of localization made by others

The free software licenses allow the user to modify the program, mix parts of it with other software and vice versa, and they also allow to redistribute that "derived works". In some cases, this makes that localization simply “happens”, sometimes even without the official (*upstream*) software developers being notified. Deploying the infrastructure needed for localization in the original project prevents this kind of “forks” integrating the external contributions and taking benefit of them.

4.1.4 Localization as a way of recruiting

One of the competitive advantages of libre software is based in this open and collaborative development model, which brings more human work to the project, at a relatively low cost (maintenance of the collaborative infrastructure, and coordination of external contributors). Creating and maintaining alive the users and contributors community is a fundamental aspect for many libre software projects. Efforts towards improving project internationalization and create localization helping infrastructures lower the access barrier for contributor candidates, and some of them will become regular contributors in the localization teams and possibly in other areas of the project.

4.1.5 Standardization in localization, for recruiting professional translators

The experience in the development and participation in localization tasks in libre software projects opens new opportunities for professionals of language management. Marcos Cánovas and Richard Samson have studied the mutual benefits of using libre software in translator training [CS12], both for the trainees and for the libre software community.

For the software community, it is logical to think that if a professional (or training) translator decides to collaborate in the localization of a libre software project, she would choose the one that allows her to maximize the benefits for her professional career. Therefore, efforts toward standardization as using standard file formats or promoting workflows and tools similar to the ones in professional translation may serve to attract these professional translators to the libre software communities and this would increase the quality of software localization and as a consequence, the quality and professionality of the project as a whole.

4.2 Business opportunities

Localization of libre software may open business opportunities for companies specialized in translation and localization, or other kind of companies.

4.2.1 Translation companies completing the community work

Companies specialized in translation and localization of software may cover several cases where the community or company developing a certain libre software cannot reach [Par11]:

- Big projects as GNOME need great work due to the volume of translation. In addition to this, it is necessary to maintain different versions for localization (because there are multiple versions in the market). This stable work may be provided by a company.
- Specific applications as OpenBravo may not produce enough synergies to attract volunteer translators, but it has enough number of users (or certain users) that want the software to be translated. A company may offer this work to the company that releases the software.

4.2.2 Services on libre software translation tools

Some of the services that can be deployed on top of libre software translation tools are the following:

- Install, maintain, support translation servers or websites.
- Maintain, adapt and extend linguistic data.
- Starting for a certain language pair, build data on another language pair.
- Develop new tools based on the existent libre software.

Some examples of business with libre software translation tools

- **Acoveo**[1] is a software development company offering several services and among them we can find `www.translate-software.com`, a solution for software localization

[1] `http://www.acoveo.com`

consisting in a Maven plugin and a Jenkins plugin that are offered for free (under the AGPL license) to their customers. The Maven plugin takes care about extracting all the translatable strings of the client's software, and send the strings to www.translate-software.com, where professional translators will localize them, at a certain price per word. The Maven plugin also takes care of retrieving the translated strings as resource files and integrate them back in the client's software. The Jenkins plugin allows the customer to control at any time the localization workflow and progress.

- A similar business is provided by **ICanLocalize**[2], which offers website translation, software localization, Text translations and general translations. For website translations they have developed a Drupal module called "translation management"[3], released under the GPL license, and the Wordpress MultiLingual Plugin[4] (which is licensed as GPL and offered at a certain cost, including the fees for the website content translation).

- **Transifex** is a libre software project that turns on a company offering hosting and support on its platform. It is analyzed in section 6.3.3.

4.2.3 Localization tools as free software: a new market to explore

Deep knowledge about how the localization processes work in software projects, and the needs of software translators may lead to know how the localization processes work "in general", and the needs of "general" translators. This opens a market niche to libre software developers, for creating new translation help tools (for translating software or other kind of works), that can be used in the libre software communities, but also in the professional translation area (competing with privative Computer Aided Translation tools).

[2] http://www.icanlocalize.com
[3] http://drupal.org/project/translation_management
[4] http://wpml.org

4.3 Social benefits

4.3.1 Reduce the digital gap

Localization allows users to interact with software in their own language via an intuitive way, and as consequence, user's proficiency increases. In addition to this, having a software localized lowers the access barrier to certain groups without knowledge of other languages or being under-educated.

4.3.2 Develop and promote minority languages

Minority language activists find libre software a very useful tool, since they can localize whatever libre software to whatever language they want. Having literature and written documents in minority languages has been the traditional way to ensure their survival as repositories of culture; in the information technologies age, localized software is another way of cultural promotion of that languages and language revitalization.

4.3.3 Localization results as free culture works

After any individual or community perform a localization tasks, there is an obvious result of this work: the localization files of the software, in the target language. As the localization is done for a libre software, those files will be under the free software license too, so they can be reused or modified in future versions, and for different projects. But there are other results from the translation work that they can be released as free cultural works or not: **translation memories** (a database that stores paired segments of source and human translated target segments), **glossaries** (definitions for words and terms found in the program user interface), and **corpora** (large and structured set of texts). Releasing these structured linguistic databases with free culture licenses increases dramatically the ease of performing new translations of many different software to the target language, since translators of the new software count with powerful, productivity aids. Some examples of these localization and translation databases released as free cultural work are:

- Galician *Corpus* Mancomun[5]: As result of the Mancomun (the Reference Service Center

[5] http://corpus.mancomun.org/

for libre software in Galicia, promoted by Spanish Galician regional government) effort in translating several applications to Galician language, the Galician corpus Mancomun allows searches of words in English or Galician at any of the stored projects, providing English-Galician pairs of phrases where the text that was searched is found. In addition to this, there is an open API (through XML and XML-RPC) allowing plugin or application development and integration of the Corpus in other kind of computer aided translation software.

- The Atshumato Project[6] was initiated by the South African Department of Arts and Culture, and its developments are done by the Centre for Text Technology (CTexT) at the North-West University (Potchefstroom Campus), in collaboration with the University of Pretoria. The general aim of this project is the development of open source machine-aided translation tools and resources for South African languages [CCfTTC12]. Among them, Atshumato releases Translation memories in the Translation Memory eXchange format (TMX) with Creative Commons Attribution Non-Commercial ShareAlike 2.5 license[7] for the following language pairs: ENG-AFR (English to Afrikaans), AFR-ENG (Afrikaans to English), ENG-NSO (English to Sepedi), NSO-ENG (Sepedi to English), ENG-ZUL (English to IsiZulu), ZUL-ENG (IsiZulu to English).

[6]http://autshumato.sourceforge.net/

[7]As explained in section 1.1.3 this is not a completely "free culture work" but allows quite more freedoms to the receiver than traditional copyright application

Chapter 5

Problems and challenges

“Going international” is not a task exempt from risks or problems. In this chapter I will show some aspects to take into account with respect to internationalization, localization and translations.

5.1 Internationalization problems

Projects that are designed without taking into account an internationalization perspective may need a complete redesign of their interfaces, codebase and architecture in order to make them translatable. This may be a lot of work, unaffordable for a small development team.

Sometimes the interface is designed with the original English locale in mind, and keeping it flexible to fit the translated environment turns in a big challenge for developers or designers. We can think for example the typical “Ok” button, which turns to 7 character text when translated to Spanish: “Aceptar”. Other example is the decision to localize (or not) keyboard shortcuts.

In some cases, certain internationalization criteria are integrated in the project (for example, using gettext for the text messages presented to the user) but not for other parts of the system (date formats, measures...) and this situation derives in problems for the users of non-English locales (they expect a completely localized system and may not deal well in the resulting “mixed” environment).

5.2 Division of work and coordination problems

It is necessary a good communication channel and good coordination between developers and translators to avoid problems and repeating tasks.

For example, when a new release is near, it is common to find that translators make efforts to have all the components of the project translated, while developers try to fix errors that may affect the interface or translatable strings[1].

5.3 Crowdsourcing and quality of translations

Having a distributed team of collaborators, taking benefit of crowdtranslation, increases the number of languages and strings translated but may introduce inconsistencies or quality problems. Some aspects to take into account:

- Neutral register: people translating to their own language may not be aware that they are using not a "neutral" register but a regional variation of that language.

- Translating without creating glossary and agreeing on terminology may lead to inconsistent translations between the different members of that l10n-team, between the software and other components of a greater solution or distribution, or between releases (when old translators are not available anymore, and new translators introduce new terminology)[2].

- Low access barrier to new contributors may introduce additional effort for reviewers. Since review software translations (some of them consisting in individual words or very short sentences) carry little effort compared with translating them from scratch, it may

[1]Christian Perrier, the Debian translations coordinator, has recently alerted about this situation to the Debian community, with his message "Please consider stopping uploads with *uncoordinated* changes to debconf templates before the release" to the development, release, internationalization, and English localization lists `http://lists.debian.org/debian-i18n/2012/06/msg00046.html`

[2]An interesting initiative to try to help solve this problem is the Open Tran project (`http://open-tran.eu/`), a website that gathers the localization files of 10 big software projects (KDE , GNOME, OpenOffice, openSuSE, Mandriva, Fedora, Mozilla, XFCE, Inkscape and the Debian installer) in more than 100 languages, creating a searchable translation memory where the translator can see how a certain term is translated in those "reference" projects.

not be clear if it is worthwhile to maintain the “crowdsourcing” infrastructure for teams that already have a group of stable translators.

Chapter 6

Case Studies

6.1 Internationalization tools

6.1.1 Gettext

GNU 'gettext' [Fou10] offers to programmers and translators a framework to help other packages produce multi-lingual messages. This framework includes a set of conventions about how programs should be written to support message catalogs, a directory and file naming organization for the message catalogs themselves, a runtime library supporting the retrieval of translated messages, and a few stand-alone programs to massage in various ways the sets of translatable strings, or already translated strings.

History

The definition of the gettext interface comes from a Uniforum proposal. It was submitted there by Sun, who had implemented the gettext function in SunOS 4, around 1990. The GNU project took it and extended it. Nowadays, the gettext interface is specified by the OpenI18N standard[1].

How it works

The gettext manual[2] explains in details all the things needed to do in order to make gettext work. In figure 6.1 there is a general overview of the files handled by GNU gettext and tools acting on

[1]https://wiki.linuxfoundation.org/en/OpenI18N

[2]http://www.gnu.org/software/gettext/manual/gettext.htm

those files.

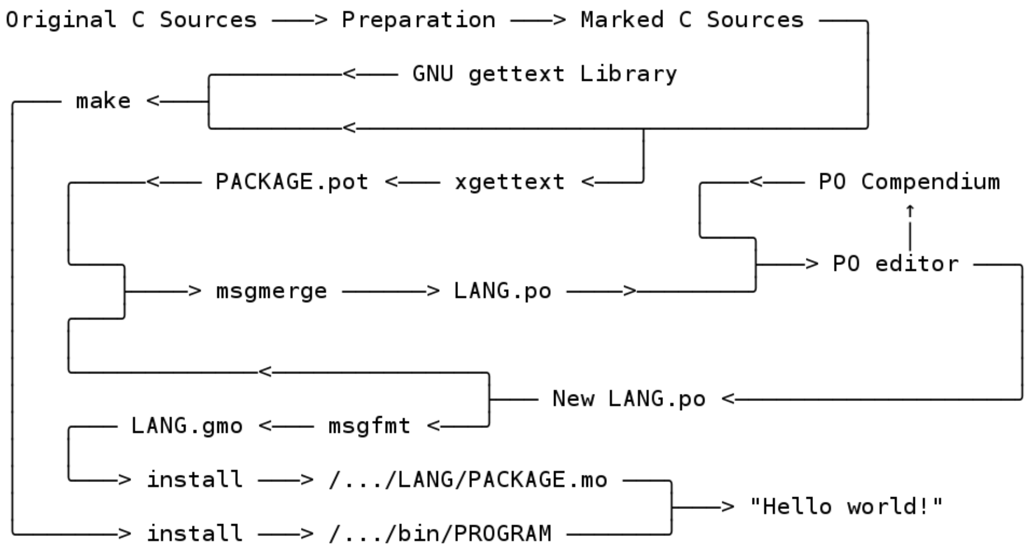


Figure 6.1: Files handled by GNU gettext and tools acting on them

From the point of view of the developer (or the person in charge of internationalize the code), we briefly explain the steps to follow:

- Import the gettext declaration and initialize the locale data.
- Prepare the English strings to make them translatable: use entire sentences, split at paragraphs, usual markup…
- Mark Keywords: all strings requiring translation should be marked in the C sources, to help `xgettext` at properly extracting all translatable strings when it scans a set of program sources and produces PO file templates, and also for triggering the retrieval of the translation, at run time. The canonical keyword for marking translatable strings is `gettext()` (it gave its name to the whole GNU gettext package), but it is usual to define and use the shortcut `_()`. It is possible to "send" comments for translators as parameters of the `gettext()` function, that will be stored in the output `.po` file just above the sentence to translate.
- Make the PO Template File: invoke the `xgettext` program to parse the source code files and create a `domainname.po`, which should be renamed to `domainname.pot`. This file contains all the translatable strings, and will be used as template by all the translators.

From the point of view of the translator, the steps to follow are:

- Creating a New PO File: copy the `.pot` file and rename to `LANG.po`, being `LANG` the target locale. It is possible to automate this with the `msginit` command.
- It is possible to modify the resulting `LANG.po` file with a text editor, or use a specific program to handle gettext catalog files (Poedit, the PO mode of Emacs, Virtaal...).
- The PO file includes a **header**, with some initial comments as `"SOME DESCRIPTIVE TITLE"`, `"YEAR"` and `"FIRST AUTHOR EMAIL@ADDRESS, YEAR"` ought to be replaced with the proper information, as well as other strings as `Project-Id-Version` (version of the package), `Report-Msgid-Bugs-To` (email address or url for bugs on translations), and other. Some strings are filled in automatically by `xgettext` as `POT-Creation-Date`, or by the editor when the file is saved, as `PO-Revision-Date` or `Last translator` or `Language` (if an editor capable to manage the meta information of PO files is used).
- After the header, we can find the original English strings, with the format:

  ```
  msgid "text"
  ```

  and the strings in the target locale, in the format

  ```
  msgstr "text"
  ```

  which can be in different states:

  – Translated Entries: Only translated entries (and not marked as 'fuzzy') will later be compiled by GNU `msgfmt` and become usable in programs. Other entry types will be excluded.

  – Fuzzy Entries: They are preceded by a line with the mark `#, fuzzy`, and usually call for revision by the translator.

  – Untranslated Entries: they are in the format `msgstr ""`

  – Obsolete Entries: they are marked as comment by using the character `#`

- Comment lines may be found before each translatable string, showing the file and line of source code that includes that string, and other messages “sent” to translators by developers.

- When the source code changes, new templates are generated. It is possible to merge the existing translated files with the new strings from the new templates using the `msmerge` command.

- Gettext also includes a suite of commands for manipulating PO files, among them msgcat (concatenates and merges several PO files), msgconv (changes character encoding), msggrep (extracts strings with a given pattern), msgfilter (applies a filter to all translations), msguniq (unifies duplicate translations), msgexec (applies a command to all translations of a translation catalog), and other. These tools can be used to do maintenance and or quality assurance.

The msgfmt command is used to build `.mo` binary translation files from the `.po` gettext catalog files, which will be used at install time to produce the localized interface of the program.

Key elements of success

- GNU Gettext provides a consistent workflow and tools for each phase of the internationalization/localization process. This is nonexistent for other type of localization file formats as XLIFF, where independent tools can be found for some phases but there is not an integrated process.

- The resulting localization files (`.po`) have a simple format (where the “ID” of the translatable strings is the string itself) so they can be easily understood and no special tool is required to work with them, an standard editor is enough.

- Being free software, and used in the GNU project, it became quickly a *de facto* standard.

- Many different programming languages are supported: C, C++, ObjectiveC, PO, Python, Lisp, EmacsLisp, librep, Scheme, Smalltalk, Java, JavaProperties, C#, awk, YCP, Tcl, Perl, PHP, GCC-source, NXStringTable, RST, Glade.

6.2 Translation tools

6.2.1 Text editor extensions

Emacs PO mode

Some tools for working with Gettext in Emacs can be found in the Emacs wiki (http://www.emacswiki.org/emacs/Gettext):

- PoMode: edit .po catalogs
- po-mode+.el: extra features for PoMode, by Gaute Hvoslef Kvalnes (KDE and OpenOffice.org translator into Norwegian Nynorsk)
- MoMode: view .mo compiled catalogs

Vim PO plugin

On the other side, Vim also have a plugin for working with PO files (http://www.vim.org/scripts/script.php?script_id=695):

- po.vim: ftplugin for easier editing of GNU gettext PO files, by Aleksandar Jelenak

6.2.2 Poedit

Poedit[3] is a cross-platform editor por `.po` files (gettext catalogs). It has been introduced in section 3.4.2. It is simple and well known among translators.

6.2.3 Virtaal

Virtaal[4] is a more modern translation tool, which allows working with XLIFF and many other kind of files. It has been introduced in section 3.4.2.

[3] http://www.poedit.net/
[4] http://translate.sourceforge.net/wiki/virtaal

6.2.4 Gtranslator

Gtranslator[5] is the translation files editor in GNOME desktop environment. It allows to open several PO files in tabs, plural forms support, automatic headers update, comments editing, management of different translator profiles, use of Translation Memories, and other, including the fact that it inherits the Gedit (GNOME text editor) plugins system.

6.2.5 Lokalize

Lokalize[6] is the translation files editor in KDE desktop environment. It developed by Nick Shaforostoff, and it is a replacement for former localization tool in KDE, Kbabel.

It allows to open PO files as well as XLIFF files, and to use metadata (insert comments or notes), Translation Memories and Glossaries.

In figure 6.2 we can find a comparison between Poedit, gtranslator and lokalize, from Ohloh.net[7].

6.2.6 Plugins or built-in localization tools

Some programs offer a plugin or tool to ease its own localization, instead of suggesting the user to use an external tool.

Wordpress “Codestyling localization” plugin

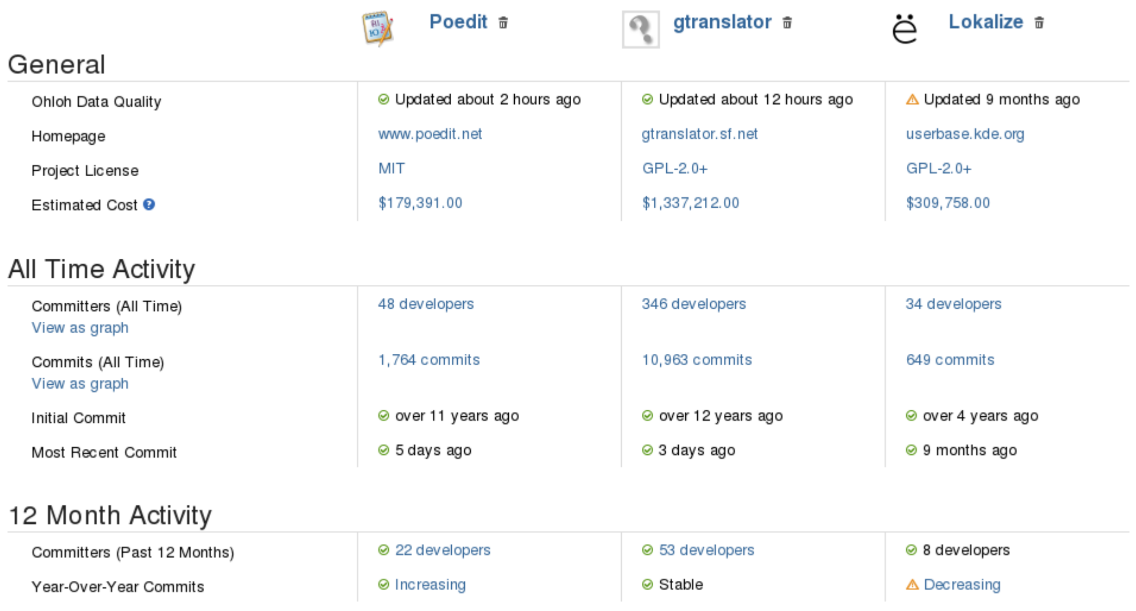
The Wordpress plugin **“Codestyling localization”**[8] allows to generate and manage .po files for Wordpress itself or any installed plugin in the Wordpress Admin Center, without need of an external editor. It has a connection with Google Translate API or Microsoft Translator API to help translation task.

[5] http://projects.gnome.org/gtranslator

[6] http://l10n.kde.org

[7] Ohloh is a public directory of Free and Open Source Software and the contributors who create and maintain it. It belongs to BlackDuck Software (a consultant company dedicated to Open Source Software), and offers data about the projects in its directory taken by analyzing their source code and SCM logs, and gathering opinions from the users in the website.

[8] http://wordpress.org/extend/plugins/codestyling-localization/



	Poedit	gtranslator	Lokalize
General			
Ohloh Data Quality	Updated about 2 hours ago	Updated about 12 hours ago	Updated 9 months ago
Homepage	www.poedit.net	gtranslator.sf.net	userbase.kde.org
Project License	MIT	GPL-2.0+	GPL-2.0+
Estimated Cost	$179,391.00	$1,337,212.00	$309,758.00
All Time Activity			
Committers (All Time) View as graph	48 developers	346 developers	34 developers
Commits (All Time) View as graph	1,764 commits	10,963 commits	649 commits
Initial Commit	over 11 years ago	over 12 years ago	over 4 years ago
Most Recent Commit	5 days ago	3 days ago	9 months ago
12 Month Activity			
Committers (Past 12 Months)	22 developers	53 developers	8 developers
Year-Over-Year Commits	Increasing	Stable	Decreasing

Figure 6.2: Comparison between Poedit, Gtranslator, and Lokalize (by Ohloh.net)

Drupal localization system

The Drupal Content Management System has its own modules for translating the Drupal Core and Drupal modules without using an external tool. Indeed, it is the recommended way for localizing Drupal. This case is analyzed in detail in section 6.3.5.

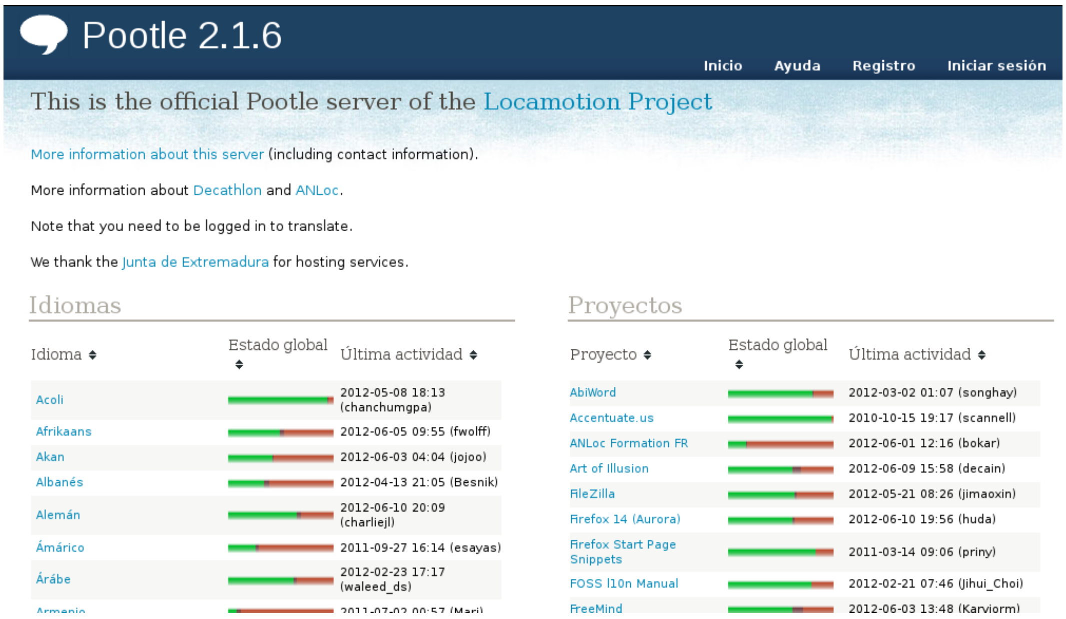
6.3 Web frameworks for crowdsourcing translation

6.3.1 Pootle

Pootle[9] is a libre software web framework developed to help the localization of software projects. It was created by the group “Translate.org.za” and later integrated in the project “Wordforge”. It uses many of the tools available in the “Translation Toolkit”, providing an easy web interface to them, so the translator does not need any knowledge about how these tools work or their syntax to obtain their benefits.

In figures 6.3 and 6.4 we can find a general overview of how a Pootle server looks like.

[9] http://translate.sourceforge.net/wiki/pootle



This is the official Pootle server of the Locamotion Project

More information about this server (including contact information).

More information about Decathlon and ANLoc.

Note that you need to be logged in to translate.

We thank the Junta de Extremadura for hosting services.

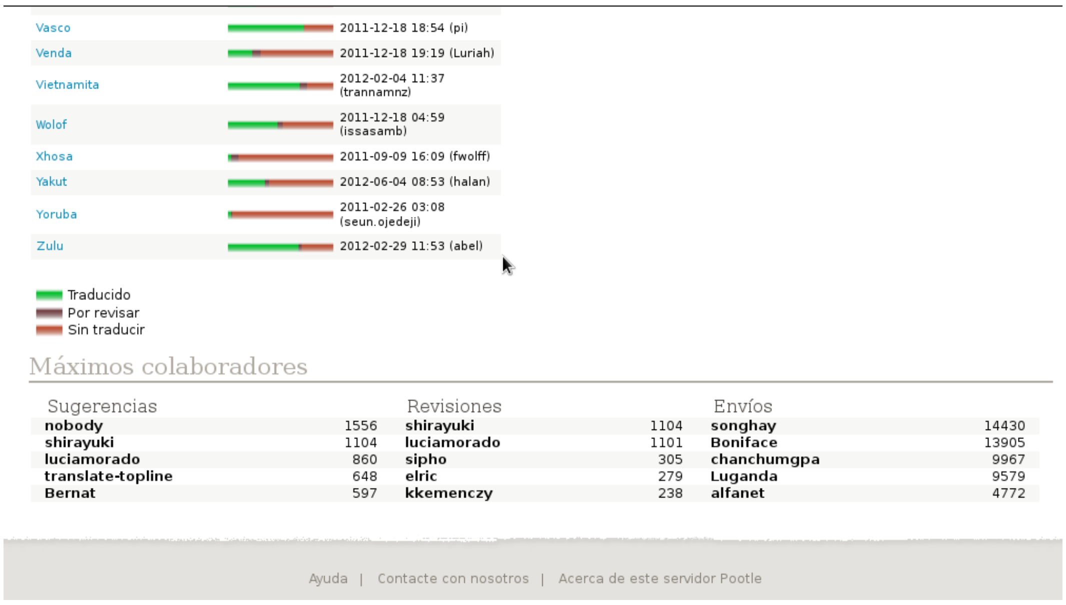
Idiomas

Idioma	Estado global	Última actividad
Acoli		2012-05-08 18:13 (chanchumgpa)
Afrikaans		2012-06-05 09:55 (fwolff)
Akan		2012-06-03 04:04 (jojoo)
Albanés		2012-04-13 21:05 (Besnik)
Alemán		2012-06-10 20:09 (charliejl)
Ámárico		2011-09-27 16:14 (esayas)
Árábe		2012-02-23 17:17 (waleed_ds)
Armenio		2011-07-02 00:57 (Mari)

Proyectos

Proyecto	Estado global	Última actividad
AbiWord		2012-03-02 01:07 (songhay)
Accentuate.us		2010-10-15 19:17 (scannell)
ANLoc Formation FR		2012-06-01 12:16 (bokar)
Art of Illusion		2012-06-09 15:58 (decain)
FileZilla		2012-05-21 08:26 (jimaoxin)
Firefox 14 (Aurora)		2012-06-10 19:56 (huda)
Firefox Start Page Snippets		2011-03-14 09:06 (priny)
FOSS l10n Manual		2012-02-21 07:46 (Jihui_Choi)
FreeMind		2012-06-03 13:48 (Karviorm)

Figure 6.3: Overview of a Pootle server (1)



Vasco		2011-12-18 18:54 (pi)
Venda		2011-12-18 19:19 (Luriah)
Vietnamita		2012-02-04 11:37 (trannamnz)
Wolof		2011-12-18 04:59 (issasamb)
Xhosa		2011-09-09 16:09 (fwolff)
Yakut		2012-06-04 08:53 (halan)
Yoruba		2011-02-26 03:08 (seun.ojedeji)
Zulu		2012-02-29 11:53 (abel)

🟩 Traducido
▬ Por revisar
🟥 Sin traducir

Máximos colaboradores

Sugerencias		Revisiones		Envíos	
nobody	1556	**shirayuki**	1104	**songhay**	14430
shirayuki	1104	**luciamorado**	1101	**Boniface**	13905
luciamorado	860	**sipho**	305	**chanchumgpa**	9967
translate-topline	648	**elric**	279	**Luganda**	9579
Bernat	597	**kkemenczy**	238	**alfanet**	4772

Ayuda | Contacte con nosotros | Acerca de este servidor Pootle

Figure 6.4: Overview of a Pootle server (2)

Features

Pootle allows to organize the translation teams and the translation work, with the following main features:

- It is easy to deploy since it is packaged for many distributions
- Allows many different file formats: bilingual formats (Gettext PO, XLIFF, Qt TS, TBX and TMX) and monolingual formats (Java properties, Mac OSX strings, PHP arrays, Subtitles in many formats)
- It has integration with control version systems, committing changes with a detailed message which includes name of translator and number of translated strings.
- Show real-time translation statistics and link to the translation files for downloading them and translating offline (Virtaal is the recommended tool).
- Several management features as create and manage teams (with goals associated), assign work to various translators, roles and permissions, allow suggestions that need revision.
- Online Translation Editor implementing many of the revision checks using the "Translate Toolkit", machine translation, translation memory and terminology matching, and showing the information that translators need: the location of strings in file, translator or programmer comments, translation context.

Pootle is released under the GNU General Public License (GPL).

Requirements

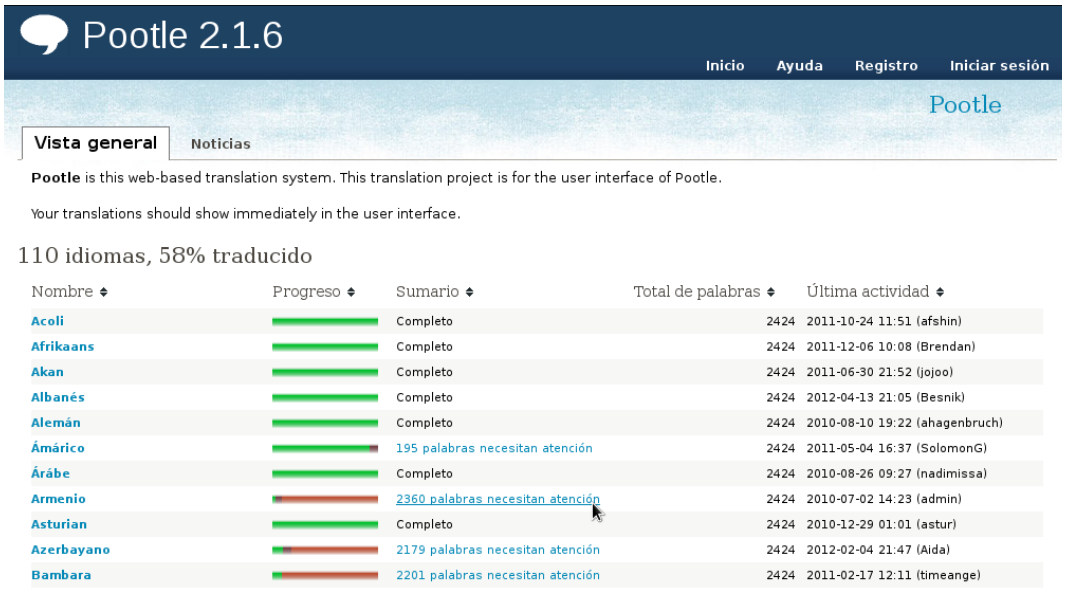
Pootle needs a web server (Apache) and a database server to run (several databases are supported). It is written in Django, and uses the Translate Toolkit. Python, Python database bindings and lxml (Python XLIFF support and HTML sanitation) are also required.

Documentation

There are many documents for the different roles using Pootle in the Pootle website, where a list of external documentation can also be found.

Usage

The official Pootle server is pootle.locamotion.org, where Pootle is translated to 110 languages (see figure 6.5, as well as many other software projects.



Pootle is this web-based translation system. This translation project is for the user interface of Pootle.

Your translations should show immediately in the user interface.

110 idiomas, 58% traducido

Nombre	Progreso	Sumario	Total de palabras	Última actividad
Acoli	🟩	Completo	2424	2011-10-24 11:51 (afshin)
Afrikaans	🟩	Completo	2424	2011-12-06 10:08 (Brendan)
Akan	🟩	Completo	2424	2011-06-30 21:52 (jojoo)
Albanés	🟩	Completo	2424	2012-04-13 21:05 (Besnik)
Alemán	🟩	Completo	2424	2010-08-10 19:22 (ahagenbruch)
Ámárico	🟩	195 palabras necesitan atención	2424	2011-05-04 16:37 (SolomonG)
Árábe	🟩	Completo	2424	2010-08-26 09:27 (nadimissa)
Armenio	🟥	2360 palabras necesitan atención	2424	2010-07-02 14:23 (admin)
Asturian	🟩	Completo	2424	2010-12-29 01:01 (astur)
Azerbayano	🟥	2179 palabras necesitan atención	2424	2012-02-04 21:47 (Aida)
Bambara	🟥	2201 palabras necesitan atención	2424	2011-02-17 12:11 (timeange)

Figure 6.5: The localization of the Pootle software

The webpage http://translate.sourceforge.net/wiki/pootle/live_servers offers a wide list of other projects that use Pootle for managing localizations, among them:

- Python documentation is translated in pootle.python.org
- pootle.librezale.org is used for Basque translation projects
- translate.contribs.org is the Pootle server for localization of the SME Open Source distribution
- Mozilla Verbatim (localize.mozilla.org) is a Pootle server for localizing Mozilla websites and projects.
- Scratch programming language is translated to 50 languages in translate.scratch.mit.edu

- BOINC (Open-source software for volunteer computing and grid computing) server and client software are translated to 85 languages in https://boinc.berkeley.edu/translate
- LibreOffice translations are managed in translations.documentfoundation.org
- F-Droid project uses http://fdroid.org/translate

6.3.2 Weblate

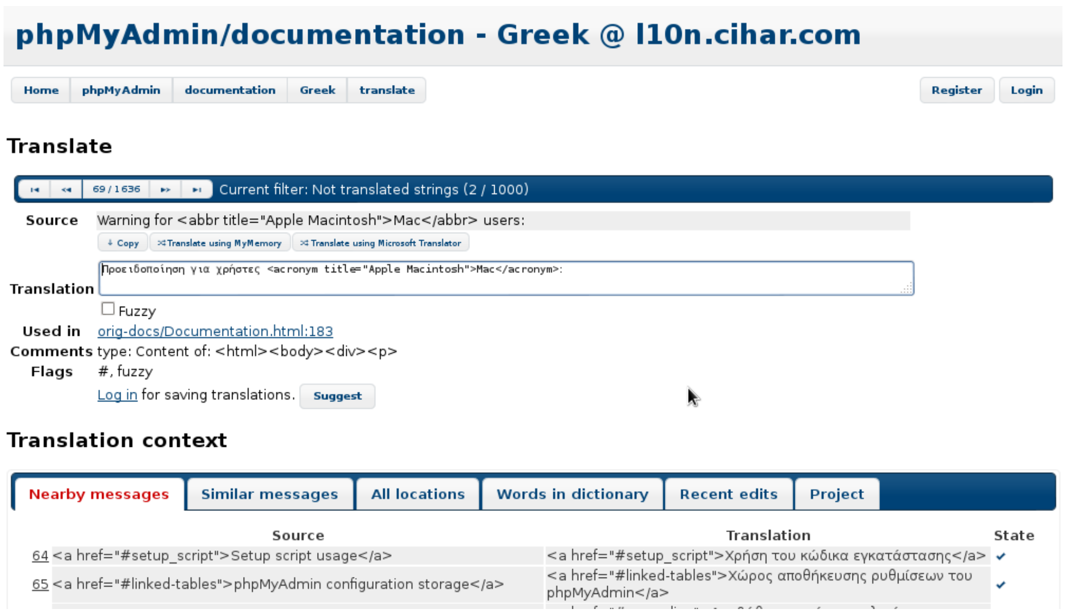
Weblate[10] is a web based translation tool with tight Git integration. It is developed by Michal Čihař (PHPMyAdmin project leader) and licensed under GPL version 3. It pretends to be a replacement from Pootle, taking out many of the less-used options, and improving the coordination with the Git SCM.

Features

The main features of Weblate are the following:

- Easy web based translation, including contextual information (see figure 6.6).
- Propagation of translations across sub-projects (for different branches)
- Every change is represented by Git commit, but on the other side it is possible to configure an option LAZY_COMMITS to group commits from same author into one if possible.
- All changes are committed to Git with correct authorship.
- Allows merging po files or automatically pull upstream changes.
- Wide range of supported translation formats (Gettext, Qt, Java, Windows, Symbian and Android resources as experimental).
- It has consistency checks, although less than Pootle (some of them removed for simplicity).
- It uses Django Administration Interface.

[10] http://weblate.org/



phpMyAdmin/documentation - Greek @ l10n.cihar.com

Translate

Current filter: Not translated strings (2 / 1000)

Source	Warning for <abbr title="Apple Macintosh">Mac</abbr> users:
Translation	Προειδοποίηση για χρήστες <acronym title="Apple Macintosh">Mac</acronym>:
	☐ Fuzzy
Used in	orig-docs/Documentation.html:183
Comments	type: Content of: <html><body><div><p>
Flags	#, fuzzy
	Log in for saving translations.

Translation context

	Source	Translation	State
64	<a href="#setup_script">Setup script usage</a>	<a href="#setup_script">Χρήση του κώδικα εγκατάστασης</a>	✓
65	<a href="#linked-tables">phpMyAdmin configuration storage</a>	<a href="#linked-tables">Χώρος αποθήκευσης ρυθμίσεων του phpMyAdmin</a>	✓

Figure 6.6: Translation of PHPMyAdmin documentation to Greek, using Weblate

Requirements

Weblate needs a web server and a database server to run. It is written in Django, and uses the Translate Toolkit, as Pootle. Other requirements are Python, GitPython, Django-registration and Whoosh (Python indexing and search library).

Documentation

The documentation of the tool is in the website http://weblate.readthedocs.org and mainly includes the usage guide (for translators), installation, configuration and administration instructions (for the i18n team), the Weblate's Web API, and a brief Frequently Asked Questions. There is a demo server deployed in http://demo.weblate.org/.

Usage

Weblate is a very new tool (initial release on February 6th 2012), and as of June 13th 2012 it is used for translating the projects lead by Michal Cihar (10n.cihar.com: phpMyAdmin, Weblate itself and three more projects) and also for the Web-translation for LinuxCNC (computer control of machine tools with Linux), at http://l10n.unpythonic.net/.

6.3.3 Transifex

Features

The main features of Transifex are the following:

- Support of many different file formats: Gettext files (.po, .pot), QT Linguist (.ts), Java properties (.property), Android resources (.xml), Joomla lang packs (.ini), html, xhtml and any strings over API.
- An extensive API to integrate tools with Transifex.
- Creation of language teams.
- Built-in Translation Memory storing translations and offering suggestions whenever a sentence ”matches” prior translations.
- It is possible to label a project “Private” and hide it from public viewing.
- Built-in message system so translators and project managers may send each other notifications (received by email).
- View translation history, track changes and revert to older versions.
- Automatic daily pulls to public SCM repositories.
- Command-line client to manage large projects and automate your workflow.

Requirements

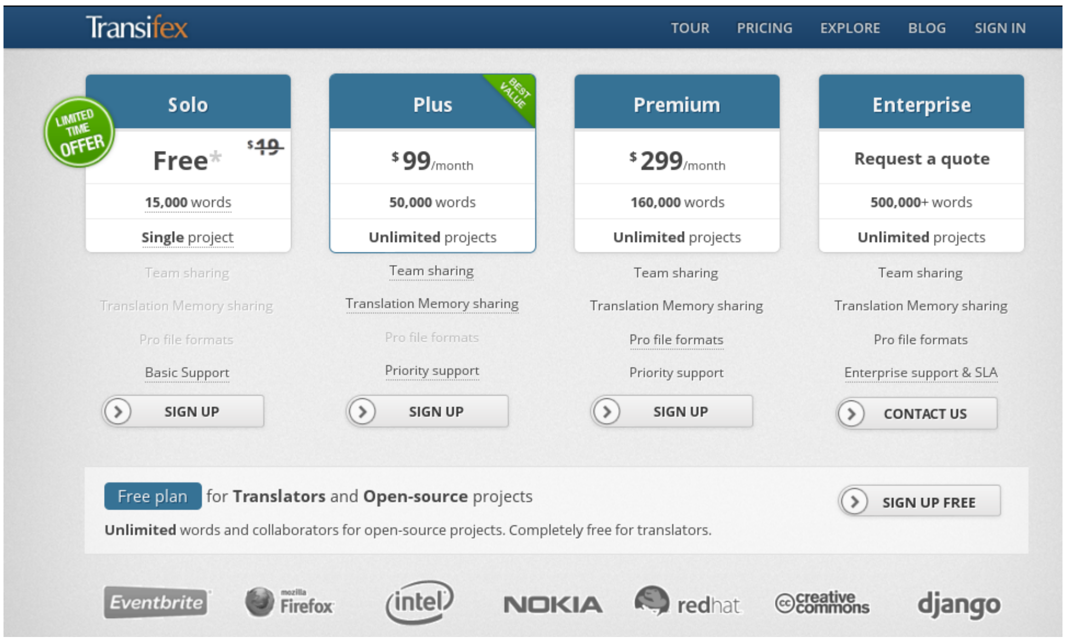
Transifex is written in Python, and needs a web server and a database server to run. Other dependencies are Django, Gettext, and intltool (for dynamic POT-file generation).

Documentation

The website `http://help.transifex.com` offers a wide documentation for translators and project managers, as well as detailed instructions on how to deploy your own Transifex server.

Usage

Transifex is a Google Summer of Code (GSoC) success story [End11]. Dimitris Glezos developed Transifex for the Fedora Project in 2007, and later he re-wrote it from scratch to make it more scalable and extensible. In 2008 he founded Indifex, to host Transifex SaaS platform which in 2011 is used by 2.000 open source projects and 10.000 users, and offer support services. In late 2011 they began to offer pricing plans for proprietary software projects, keeping the software itself free, and free of charge hosting and support for libre software projects (figure 6.7).



Solo	Plus	Premium	Enterprise
Free* ~~$19~~	$99/month	$299/month	Request a quote
15,000 words	50,000 words	160,000 words	500,000+ words
Single project	Unlimited projects	Unlimited projects	Unlimited projects
Team sharing	Team sharing	Team sharing	Team sharing
Translation Memory sharing	Translation Memory sharing	Translation Memory sharing	Translation Memory sharing
Pro file formats	Pro file formats	Pro file formats	Pro file formats
Basic Support	Priority support	Priority support	Enterprise support & SLA
SIGN UP	SIGN UP	SIGN UP	CONTACT US

Free plan for **Translators** and **Open-source** projects — SIGN UP FREE

Unlimited words and collaborators for open-source projects. Completely free for translators.

Figure 6.7: Pricing plans in Transifex

Some relevant projects using Transifex for their translations are: The FreedomBox Foundation (for their website), DoudouLinux (a Linux distribution for children), OpenTranslators (translations for all open source projects that are related to Joomla!), OwnCloud, Zotero citations manager, the Fedora Project (including also websites and documentation), Universal Subtitles (subtitling for online video), and many other.

6.3.4 Launchpad Translations (Rosetta)

Launchpad[11] is the software forge developed by Canonical, initially to host the development of Ubuntu Linux distributions, and later released as free software under the GNU Affero General Public License, version 3. Launchpad includes Bazaar source code management system, a bug tracking system, Ubuntu package building and hosting, and other kind of tools; among them we can find Launchpad Translations (which former name was Rosetta).

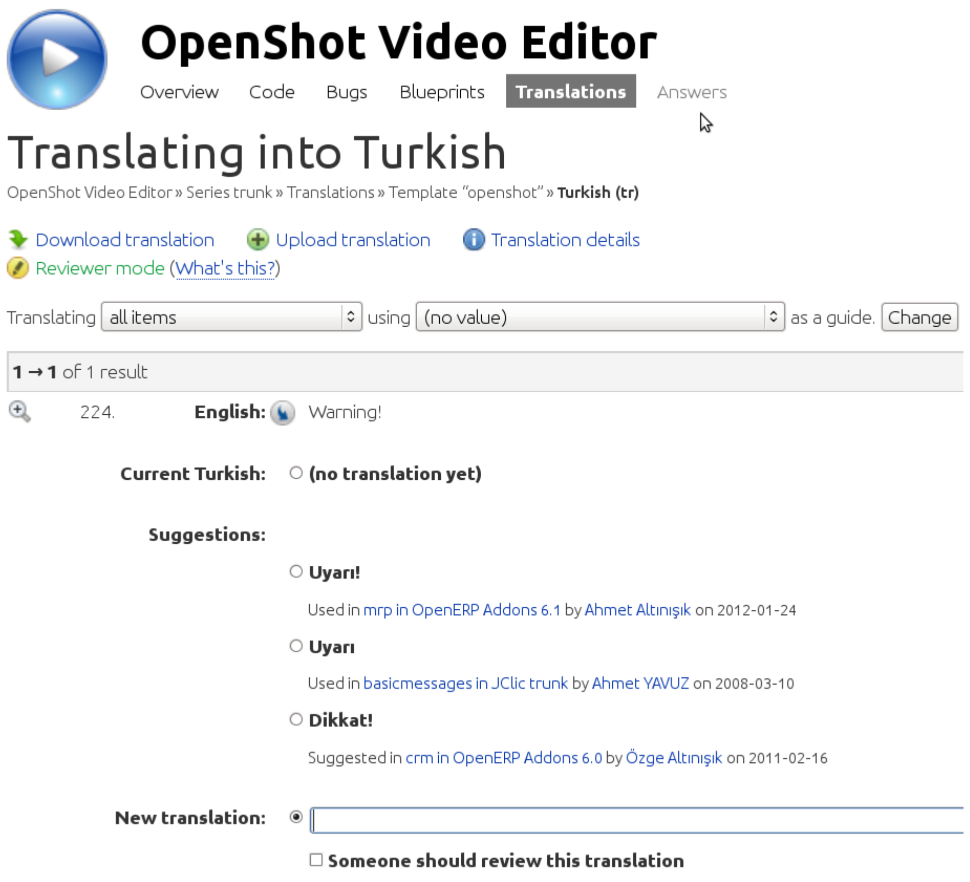
Features

Launchpad takes benefit from crowdsourced translation. It has a simple web-based translation and review, GNU GetText support and automatic suggestions from a library of sixteen million translated strings.

- It is possible to choose how open your project is to new translators: open (everyone can translate), partly restricted and structured (anyone can suggest translations, while trusted community members review and approve new work), or closed (only approved translators may make suggestions).
- The supported file formats are pot, po and mo (for offline as well as online translation), and it is possible to import Mozilla's XPI format.
- About licensing, it is required that all translations made in Launchpad are BSD licensed[12] (so they can be part of Launchpad's database and serve as suggestions to any kind of project, as shown in figure 6.8).
- It has a good integration with the Bazaar source code management system, in the sense that it allows automatic imports of templates from the Bazaar repository, and allows to make regular commits to a Bazaar branch you specify for updating the translations made via website (that commits overwrite the translation files present in the repository with the ones from the website). It is also possible to manually upload and download the files.

[11] https://translations.launchpad.net/

[12] As explained in section 1.1.2, it is a permissive libre software license, in which the four freedoms are stated and not additional conditions are imposed to the user


OpenShot Video Editor

Overview Code Bugs Blueprints Translations Answers

Translating into Turkish

OpenShot Video Editor » Series trunk » Translations » Template "openshot" » **Turkish (tr)**

Download translation Upload translation Translation details

Reviewer mode (What's this?)

Translating all items using (no value) as a guide. Change

1 → 1 of 1 result

224. **English:** Warning!

Current Turkish: ○ **(no translation yet)**

Suggestions:

○ **Uyarı!**
Used in mrp in OpenERP Addons 6.1 by Ahmet Altınışık on 2012-01-24

○ **Uyarı**
Used in basicmessages in JClic trunk by Ahmet YAVUZ on 2008-03-10

○ **Dikkat!**
Suggested in crm in OpenERP Addons 6.0 by Özge Altınışık on 2011-02-16

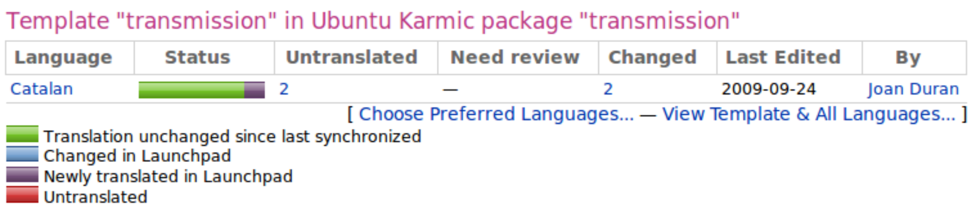
New translation: ●

☐ **Someone should review this translation**

Figure 6.8: Launchpad suggestions from other projects

- For translators, Launchpad acts as a central repository, in the sense that your Launchpad user account is associated with the languages that you choose to translate, and then, you choose the projects where to contribute (*upstream* projects, or the Ubuntu distributions).

- Launchpad shows statistics of translations, to give an overview of the status of each project, language or file (see figure 6.9).


Template "transmission" in Ubuntu Karmic package "transmission"

Language	Status	Untranslated	Need review	Changed	Last Edited	By
Catalan	🟩	2	—	2	2009-09-24	Joan Duran

[Choose Preferred Languages... — View Template & All Languages...]

🟩 Translation unchanged since last synchronized
🟦 Changed in Launchpad
🟪 Newly translated in Launchpad
🟥 Untranslated

Figure 6.9: Launchpad translation statistics

Documentation

The Documentation for users of Launchpad translations is in the website https://help.launchpad.net/Translations, integrated with the general Help of Launchpad.

Usage

As of June 13th, 2012, Launchpad database stores 1,923,617 strings in 33,465 translation templates, in 332 languages. There are 63,724 translators registered and 42 translation groups. It is possible to contribute to the translation of the Ubuntu Linux distributions version 6, 8, 9, 10, 11 and 12; and also to 1,470 projects, among them the OpenShot Video Editor, the OpenERP Server, Web and Add-Ons, Blender Animation Design software, the Linux Mint Distribution, the Astronomer's program Stellarium, and many others.

6.3.5 Drupal localization system

Drupal[13] is a Content Management System with a very modular infrastructure. It is written in PHP and both Drupal core and Drupal modules manage localization in the same way.

In table 6.1 some quick statistics can be found about localization in Drupal.

Name of the project:	Drupal
Main website:	http://drupal.org
I18n website:	https://localize.drupal.org
Number of languages:	103 translation groups
Translation team(s) and coordinator(s):	Each team has its own organization
L10n tools/platform:	localize.drupal.org + Localization Update Module
Contributors and modules	4,200 contributors, 8,247 projects

Table 6.1: Summary of Drupal localization

[13] http://drupal.org

Internationalization work

Drupal recommends to follow the internationalization guidelines to make easy the localization of any piece of software.

The objects to be localized in Drupal are the following:

- Text showed to the "Drupal Administrator" in the installation process and Administration Dashboard and tasks. This strings are part of the Drupal PHP code and usually can be found in the corresponding `.pot` templates. For new developments, it is recommended to abstract these kind of text from the rest of the code using the `t()` function.

- Text that are showed to the user or visitor of a website and messages. For example, labels showing "Site map", or "Home" in a frontpage, "Next" or "Read more" when showing a list of different pieces of content. These strings are "configurable" in the deployed Drupal Administration Site. For Drupal developers, it is recommended to offer a default English string for all these kind of fields, which should be also internationalized using the `t()` function, and then can be easily localized in order to provide "localized default strings" to the deployed Drupal websites.

About Drupal infrastructure for localization, Drupal was using CVS as control version system but in the beginning of 2011 they migrated to Git. Along with this change, other different (but related) decision was made: start to use their own web platform `http://localize.drupal.org` to manage translations [new10]. Although the translation was going to keep on using `.po` files, this approach separates the "code development" and the "translation development". The main positive aspect is that they can attract many new translators that are Drupal users but don't know anything about developing, git and `po` files. On the other hand, if anybody goes to the git repositories to get all the Drupal code, she will not find the translation files there.

Localization work

On translators side, Drupal offers the "Localization update" module (`http://drupal.org/project/l10n_update`) to deploy in the Drupal websites and make easy translation, the `localize.drupal.org` website where it is possible to join a team, online translate and review translations, and a synchronization system to retrieve translations from users Drupal websites, and send them the reviewed and accepted strings (using the Localization update module).

Maintenance and Quality Assurance

Suggestions are not be included in downloadable translations, and will need to be approved by an authorized user before becoming a translation.

6.4 Translation projects and teams

6.4.1 GNOME

GNOME Project[14] is a community that develops a GNU/Linux graphical desktop environment.

GNOME is one of the most used graphic desktops in GNU/Linux systems. The project includes not only window manager and system administration tools, but a complete set of applications for many different purposes as text editor, multimedia player, web browser and email client, CD recorder...

The main site of the project is `http://live.gnome.org`, and the information about localization can be accessed in "Damned Lies" web platform (`http://l10n.gnome.org/`).

GNOME software uses GTK+ development kit, which works with some tools for internationalization as **gettext**, and produces PO files.

The GNOME source code is managed with git repositories. Damned Lies has no git support so the changes have to be manually committed to the git repositories by the translators with git access.

Table 6.2 provides a summary of the localization of the project.

The localization workflow in GNOME

Gil Forcada, with the feedback from other community members, conducted a GNOME I18N Survey in August 2010, by sending a questionnaire to every GTP (GNOME Translation Project) language coordinator, and collecting answers. From that results[For10] and the information provided in the GNOME website, we have drawn the localization process:

[14]`http://gnome.org`

Name of the project:	GNOME
Main website:	http://gnome.org
I18n website:	https://live.gnome.org/TranslationProject
Number of languages:	178 registered; 50 released[Kov12]
Translation team(s) and coordinator(s):	Christian Rose, Mario Blättermann, Gil Forcada, Gabor Kelemen, Andre Klapper, Petr Kovar, Claude Paroz, Johannes Schmid, Simos Xenitellis, Wouter Bolsterlee, Og Maciel
L10n tools/platform:	Damned Lies http://l10n.gnome.org

Table 6.2: Summary of GNOME localization

1. **Objects to be localized**: modules, documentation. Both are in gettext portable object format (`.po` files).

2. **Internationalization framework**: mailing lists, language teams with coordinators, localization guide (general), GNOME Documentation Style Guide, Git access for manually committing changes to the repository. There is additional information provided for developers: Handling String-Freezes, Translation Coverage Statistics for your module

3. **Localization framework**: all teams use mailing list for coordination, and most of the teams have glossaries and translation guidelines. To keep track of translation issues most use bugzilla and/or Damned Lies.

4. **Quality and maintenance**: 19 out of 36 teams (52%) pass QA tools before committing. 19 out of 36 teams (52%) start working around string freeze (1 month before next GNOME release). Some scripts for helping QA and maintenance are developed and provided in https://live.gnome.org/TranslationProject/Scripts

Other remarkable aspects

- Teams range from 1 or 2 to +15 members (average to 3 members per team)

- Nearly all work is made by volunteers (only 2 or 3 teams have paid members)

- There's a good relation with downstream translators (Ubuntu, Fedora, and other GNU/Linux distros)

6.4.2 KDE

KDE[15] is other of the most used graphic desktops in GNU/Linux systems. As GNOME, the project includes not only window manager and system administration tools, but a complete set of applications for many different purposes as text editor, multimedia player, web browser and email client, CD recorder. . . The main site of the project is `http://www.kde.org`, and the information about localization can be accessed in the web `http://i18n.kde.org`.

KDE software uses Qt development kit, and PO files for localization. Parts of text are sometimes presented to the user in special format as bold or italic, title sized, etc with XML-like text markup.

The KDE source code is managed with Subversion repositories.

KDE uses custom tools to internationalize the GUI and documentation of programs. **gettext**, and produces PO files.

Table 6.3 provides a summary of the localization of the project.

Name of the project:	KDE
Main website:	`http://kde.org`
I18n website:	`http://l10n.kde.org`
Number of languages:	52 (released)[Ast12]
Translation team(s) and coordinator(s):	Albert Astals
L10n tools/platform:	Lokalize

Table 6.3: Summary of KDE localization

The localization workflow in KDE

Based on the available information in KDE website, and in particular in the Localization webpage [KDE12], we have drawn the localization process:

[15] `http://kde.org`

1. **Objects to be localized**: user interface, documentation. Both are in gettext portable object format (`.po` files) and sometimes have XML-like markup for text font and style. A guide for localization of non text resources is available at http://techbase.kde.org/Localization/Concepts/Non_Text_Resources

2. **Internationalization framework**: mailing lists, language teams with coordinators, Techbase Localization Tutorials, SVN access for manually committing changes to the repository.

3. **Localization framework**: all teams use mailing list for coordination. The main tool for localization is **Lokalize**, a computer-aided translation (CAT) tool, a full-featured GUI application, written from scratch using KDE4 framework. Aside from basic editing of PO files, it integrates support for glossary, translation memory, project managing, etc.

4. **Quality and maintenance**: Lokalize supports diff-modes for QA. Other tools for QA are recommended as Lbundle Checker for tracking changes on non-text localized resources, Translate Toolkit and Pology.KDE teams use "Ascription" system for reviewing translation work and include that information in the control version system. Details of the Ascription Workflow are given in http://techbase.kde.org/Localization/Workflows/PO_Ascription.

Other remarkable aspects

In section B.3 the reader will find an interview to the internationalization team coordinator in KDE, Albert Astals.

6.4.3 Debian

Debian[16] is "the universal operating system", a multi-architecture software distribution that ships Linux or Hurd kernel, the GNU Project software, different desktops and many software packages for multiple purposes.

Table 6.4 provides a summary of the localization of the project.

[16] http://www.debian.org

Name of the project:	Debian
Main website:	http://www.debian.org
I18n website:	http://www.debian.org/international
Number of languages:	70 (Debian-installer stable release)[Per12b]
Translation team(s) and coordinator(s):	Christian Perrier
L10n tools/platform:	No particular tool recommended

Table 6.4: Summary of Debian localization

The localization workflow in Debian

Based on the available information in the Debian website, we have drawn the localization process:

1. **Objects to be localized**: The Debian localization projects focus on translation of the installation system (the "debian-installer"), Debian documentation, Debian webpages and wiki pages, and Debian-related packages (specific Debian tools as dpkg package manager). They also translate the "debconf templates": messages presented to the user when a new package is installed to the system[17].

2. **Internationalization framework**: Each language has a localization team which uses a wiki and a mailing list to handle the work. There is a robot that listens to the l10n mailing lists, and understands pseudo-urls in the subject header. The pseudo-urls have a certain form, including tags for the coordination of translations: ITT (Intent To Translate), RFR (Request for Review), LCFC (Last Chance for Comments), BTS#bug_number or DONE#bug_number. Changes are uploaded to the repositories by package maintainers

[17] Although it is recommended that all the translation issues related to non-specific Debian packages should be submitted to upstream projects, we can consider several exceptions, for example packages that are substantially modified from the original source code prior to include them in the Debian distribution. Some of them are the Mozilla suite of applications (Firefox web browser which becomes "Iceweasel" in Debian, Thunderbird email client that ships as "Icedove", Lightning calendar as Iceowl, and others). Due to this modification, additional localization packages are created and maintained (or not) by the Debian community. This means that you can find the original program translated to certain languages, but not the related Debian package (an example of this is Iceowl, with no translation to Spanish, although the upstream Lightning has Spanish localization).

(translators send the files to upload to the Bug Tracking System with the category of 'wishlist'). All the localization teams are coordinated by the Christian Perrier, who has permissions to update the repositories if package maintainer is missing in action.

3. **Localization framework**: all teams use their mailing list for coordination. There is no particular recommended tool for translations.

4. **Quality and maintenance**: Quality of translations is ensured by the mailing list coordination and peer review.

Other remarkable aspects

- Christian Perrier, translation coordinator, maintains a blog [Per12a] where he periodically writes about the process of localization in Debian. In 2011 he was interviewed by Raphael Hertzog (another Debian Developer) [Her11].

- In section B.4 the reader will find an interview to Javier Taravilla, member of the Spanish localization team in Debian.

6.4.4 Tux Paint

Tux Paint[18] is a libre software drawing program for children ages 3 to 12. It is used in schools around the world as a computer literacy drawing activity. It combines an easy-to-use interface and fun sound effects.

In June 9, 2012, Tux Paint turns 10 years old.

The main site of the project is http://www.tuxpaint.org, and the information about localization can be accessed in the "Help Us Translate" web page (http://www.tuxpaint.org/help/po/).

Tux Paint is mostly written in C and uses **gettext** for internationalization, producing POT and PO files.

Table 6.5 provides a summary of the localization of the project.

[18]http://www.tuxpaint.org

Name of the project:	Tux Paint
Main website:	http://tuxpaint.org
I18n website:	http://www.tuxpaint.org/help/po/
Number of languages:	85
Translation team(s) and coordinator(s):	Bill Kendrick (Lead Developer)
L10n tools/platform:	None / Official Pootle Server (for some languages)

Table 6.5: Summary of Tux Paint localization

The localization workflow in Tux Paint

Based on the information provided in the Tux Paint website, we have drawn the localization process:

1. **Objects to be localized**: application, stamps add-on, configuration, website, documentation. All of them are in gettext portable object format (`.po` files), except documentation that it is in plain text or HTML format.
2. **Internationalization framework**: there is a mailing list for i18n. The website includes information about how to begin translation. Translator without commit access to the repository should email the translated files to Bill Kendrick (Lead Developer).
3. **Localization framework**: all teams use mailing list for coordination. Some teams use the Official Pootle Website for managing translations (http://pootle.locamotion.org/projects/tuxpaint/).
4. **Quality and maintenance**: There is no official information about this topic in Tux Paint website.

Other remarkable aspects

- Tux Paint users who use languages requiring their own fonts can download pre-packaged TrueType Fonts for Tux Paint.
- There is a webpage[19] dedicated to mention the most important contributors to Tux Paint,

[19] http://www.tuxpaint.org/developers

which includes a specific mention to Fabian Franz for his work internationalizing the code.

6.4.5 Pleft

Pleft[20] is a web platform for planning appointments in a collaborative way. Once you register an account in their website, you can create a meeting, defining the title, description, proposed date and hour, and list of participants. Each participant will receive an email with a link to the meeting, where she can state her availability for each proposed date, and suggest alternative dates.

Table 6.6 provides a summary of the localization of the project.

Name of the project:	Pleft
Main website:	http://www.pleft.com
I18n website:	https://www.transifex.net/projects/p/pleft/
Number of languages:	9 (7 in the stable release)
Translation team(s) and coordinator(s):	Nobody (Sander Dijkhuis, project leader)
L10n tools/platform:	Transifex

Table 6.6: Summary of Pleft localization

The localization workflow in Pleft

Based on my own experience contributing to Pleft translations in Appendix C, I have drawn the localization process:

1. **Objects to be localized**: Server strings, Website client strings, Android client strings, Email texts, Marketing material

2. **Internationalization framework**: Pleft is developed in Python, and uses Django templates [Fou12a] for internationalization of server and client text strings. Pleft developers

[20]http://www.pleft.com

have created a project in Transifex web localization platform to allow everybody to collaborate in translations (see figure 6.10). Pleft developers are the ones in charge of approve the creation of new language teams. There is no internationalization coordinator nor mailing list specifically created for internationalization or localization. However, the Transifex platforms includes a built-in message system that can be used for these kind of communication. Marketing material is not handled using Transifex. Project leader contacts by email or by the Transifex platform to the language teams and send the texts to be translated. There are no public internationalization guidelines specific por Pleft.

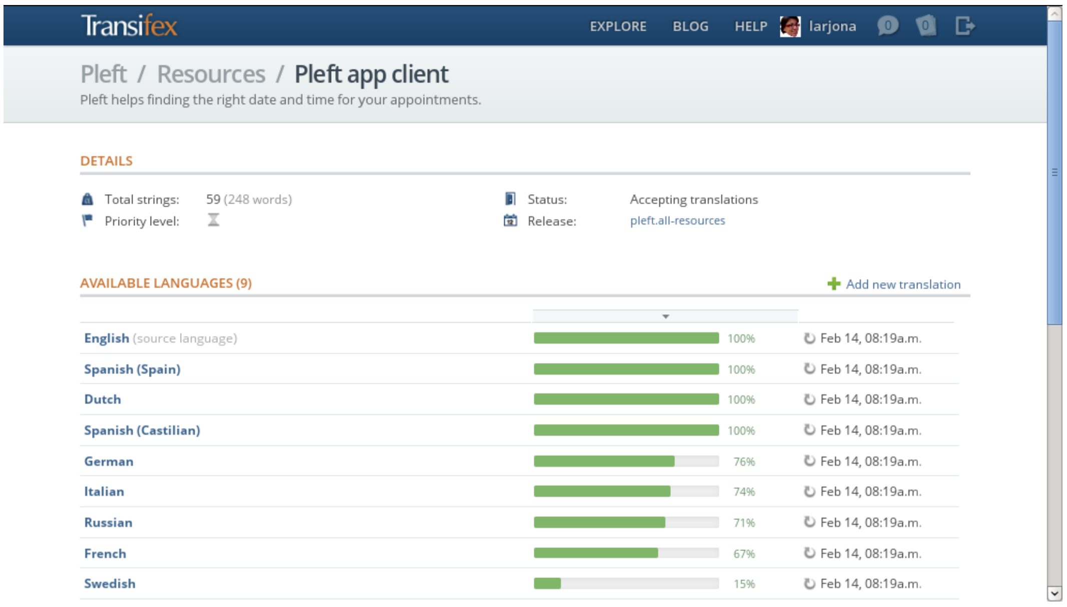
3. **Localization framework**:As of June 9th, 2012, Pleft is released in the following languages: English, French, German, Italian, Russian, Dutch, Spanish. In Transifex platform there are 2 more teams: Spanish (Castillian) and Swedish. There are no specific localization guidelines, the project trust on Transifex platform. If localization teams need to get their translations integrated in a live, test site, they can ask for it to the project leader.

4. **Maintenance and Quality Assurance**: Transifex allows to handle translation reviews and also has a built-in spell checker. In addition to this, it shows the context information of each translation string in a tab called "details": String ID, Description (but this field is not fulfilled), Comment (also usually empty), Occurrences (linking to the source code files). There is no specific treatment for bugs related to localization. General issue tracker may be used. To test or deploy translations, it is needed to install gettext and run `python pleft/manage.py mo`. New versions of internationalization files may appear and localization teams not being aware of them, since there is not an internationalization mailing list or communication channel to get subscribe yourself.

6.4.6 The power of open[21]: The Debian Administration Handbook

The Debian Administration handbook is the English translation of the *"Cahier de l'admin Debian"*, a book written by Raphael Hertzog and Roland Mas, two Debian Developers with a large career of contributing to the Debian project. This book was printed initially with Eyrolles, a French "traditional" editor, which holds the copyright of the work.

[21]Note: "The Power of Open" (http://thepowerofopen.org/) is the title of a book collecting successful stories of creators sharing knowledge, art, and data using Creative Commons licenses in their works.



Transifex EXPLORE BLOG HELP larjona

Pleft / Resources / Pleft app client

Pleft helps finding the right date and time for your appointments.

DETAILS

Total strings: 59 (248 words)

Priority level:

Status: Accepting translations

Release: pleft.all-resources

AVAILABLE LANGUAGES (9)

Add new translation

English (source language)	100%	Feb 14, 08:19a.m.
Spanish (Spain)	100%	Feb 14, 08:19a.m.
Dutch	100%	Feb 14, 08:19a.m.
Spanish (Castilian)	100%	Feb 14, 08:19a.m.
German	76%	Feb 14, 08:19a.m.
Italian	74%	Feb 14, 08:19a.m.
Russian	71%	Feb 14, 08:19a.m.
French	67%	Feb 14, 08:19a.m.
Swedish	15%	Feb 14, 08:19a.m.

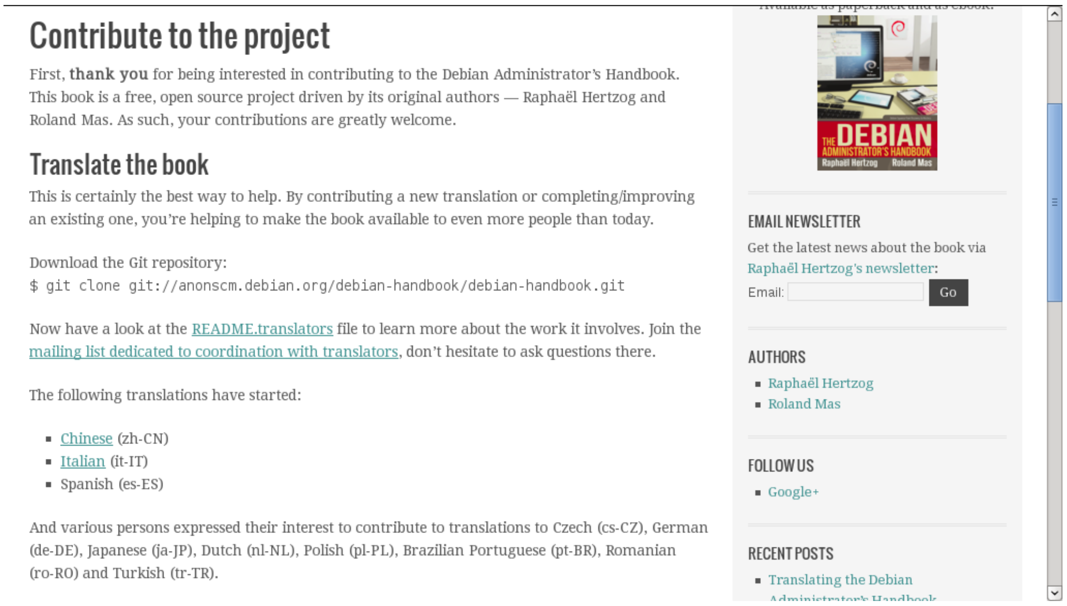
Figure 6.10: Overview of Pleft translations in Transifex

However, authors of the book wanted to translate the book into English and republish it under a free license, so all the Debian community would benefit. They set up a crowdfounding project to gather the required money to pay their translation work, so authors would be free to offer the new version of the book as free, redistributable, translatable and modifiable work.

Several months after they created the crowdfounding project, the “liberation fund” was reached. In May 10th the book was released under a CC-BY-SA licensed and freely (with no cost) available as electronic book in its website. In addition to this, sources were published in one of the Debian project git repositories, and a Debian package for installing the book (downloading it and creating a start menu shortcut) was uploaded to “Sid”, the Debian version created to receive new software from upstream and test before integrate it in the more stable releases.

After this work well done, authors of the book couldn’t event take two weeks vacation: Debian community gave a warm welcome to liberation of this book and even before than its website was updated including instructions for translation in the “Contribute” section, several localization teams had been already created to handle the localization to their respective language, setting up clone repositories, mailing list or other kind of communication channels, and committing the first changes with translation to them.

In May 23rd, Raphael Hertzog, acting as a kind of benevolent dictator, sent a mail to several people that contacted him about how to translate the book, and he explained how the localization should be done and created a mailing list in the Debian project to coordinate the work. He also updated the website including this information (figure 6.11).



Contribute to the project

First, **thank you** for being interested in contributing to the Debian Administrator's Handbook. This book is a free, open source project driven by its original authors — Raphaël Hertzog and Roland Mas. As such, your contributions are greatly welcome.

Translate the book

This is certainly the best way to help. By contributing a new translation or completing/improving an existing one, you're helping to make the book available to even more people than today.

Download the Git repository:

```
$ git clone git://anonscm.debian.org/debian-handbook/debian-handbook.git
```

Now have a look at the README.translators file to learn more about the work it involves. Join the mailing list dedicated to coordination with translators, don't hesitate to ask questions there.

The following translations have started:

- Chinese (zh-CN)
- Italian (it-IT)
- Spanish (es-ES)

And various persons expressed their interest to contribute to translations to Czech (cs-CZ), German (de-DE), Japanese (ja-JP), Dutch (nl-NL), Polish (pl-PL), Brazilian Portuguese (pt-BR), Romanian (ro-RO) and Turkish (tr-TR).

Available as paperback and as ebook.

EMAIL NEWSLETTER

Get the latest news about the book via Raphaël Hertzog's newsletter:

Email: Go

AUTHORS

- Raphaël Hertzog
- Roland Mas

FOLLOW US

- Google+

RECENT POSTS

- Translating the Debian Administrator's Handbook

THE DEBIAN ADMINISTRATOR'S HANDBOOK
Raphaël Hertzog Roland Mas

Figure 6.11: Information about Translations in the Debian Administration Handbook's website

That action has been vital in order to integrate the localization of the book with the general project and avoid unneeded forks that would become difficult to maintain and update when new versions of the book are released.

The localization workflow in "Debian Administration Handbook"

1. **Objects to be localized**: text, screenshots, diagrams.

2. **Internationalization framework**: mailing lists, coordinator for each language, git repository for changes, Publican for the generation of templates to translate, instructions on how to contribute to the project.

3. **Localization frameworks**: each team may organize themselves as they want. Spanish, Russian and German are some of the languages that set up a localization framework before the book's authors detailed the internationalization and localization framework.

4. **Quality and maintenance**: git branches, git rebase before pushing changes, participation of original authors, that are also translators since they did the French to English translation

before the release of the book as free culture. Several localization teams are thinking about using Weblate as web translation platform, which includes a set of tests for Translation Quality Assurance.

Other remarkable aspects

Although the infrastructure for internationalization and localization (Git repository, mailing lists) are deployed inside the Debian community (the Alioth forge), currently the project is treated as one more *upstream* project, in the sense that it has its own translation teams apart from the Debian i18n and l10n teams.

Chapter 7

Conclusions

7.1 Evaluation

In chapter 3, I have explained the different phases of localization process, describing tasks, tools and people involved. A minimum set of two examples for each phase has been given, and new trends have been covered, as the XLIFF standard format, the use of collaborative models and web based tools (some of them released recently as Weblate).

Origin and motivations for people getting involved in libre software translations have been explained in section 3.6.

Chapter 4 analyzed the benefits that caring about localization brings to the libre software community and society in general, and in chapter 5 some examples of possible internationalization and localization problems, to take into account, were explained.

A variety of software projects that carry out localization task and tools for supporting that processes have been provided, as well as personal experiences of different translators.

We can conclude that a panoramic view of the localization in libre software projects has been presented, and the reader may understand the complexity and diversity of the processes, but also feel there are many opportunities to get engaged and take benefit of libre software.

7.2 Lessons learned

7.2.1 Key aspects of localization in libre software projects

- Libre software communities have set up a wide range of tools to help the localization process.
- The communication channels and coordination between the different teams is crucial and can be boosted with automation, but "human touch" is still needed.
- Allowing external contributions, if specific guidelines are provided, may improve quantity and quality of translation, as well as produce other kind of benefits for the business environment and society in general.
- As conclusion, we can say that internationalization and localization in libre software projects are two of their most interesting advantages for dissemination, competition with privative alternatives, and penetration in new markets.

7.2.2 What did I learn

- Gathering information about how localization is performed in the different libre software projects helped my own evolution as translator and improved the quality of my translations, since I had more knowledge and resources to use.
- There are very new and powerful translation platforms (for example Transifex) that are taking advantage of the traditional and quality internationalization tools as Gettext. This made me value the internationalization effort developed by the libre software communities in the "early years".
- Libre software projects is not just about "coding". Many people are involved in localization of libre software, with very different ranges of commitment, but they are part of the community and their work is better understood and considered each day.

7.2.3 Knowledge and skills acquired in the M.Sc. studies that helped me on this work

- **Introduction to libre software** and **Legal aspects of libre software** provided the basic framework to understand how libre software works and its history and evolution along the years. Now I know that the software license may be applied for example to translation files, and to certain parts of documentation that are released altogether with the software, but translation memories or glossaries used for performing localization tasks may be released as free cultural works or not, since they are different works from the localization files.

- **Economic aspects of libre software** made me discover the different motivations of individuals, companies, public administrations, academic and research institutions, non profit organizations... to get involved in free software in general, and in translation in particular. Having a program localized to a certain language may open new markets or new business or may be used as a way of political or cultural activism.

- **Communities** subject gave me the tools and skills needed to perform data mining on software repositories. Although this kind of approach has not been used in this work, I know how this can be done so I have been able to draw a possible empirical approach in section 7.3.

- **Project evaluation**, **Project management** and **Developers and motivation** subjects made possible for me to assess software projects as a whole, taking into account not only the code or how the software works, but also how is the community, which roles and processes are involved and on which people they depend. I also learned that good documentation and communication channels are key aspects for reaching new contributors and sustainability of the projects.

- In **Case Studies I** and **Development and tools**, I learned how to use Git, the control version system that I used to keep track of the history of changes to this document, and helped me to contribute translations in Android software projects in the last months. **Advanced Development** helped me to better understand how Android applications are developed and encourages me keep being involved in F-Droid project, and try to help on the internationalization pending work.

- Thanks to **Case Studies II** subject, I met some translators (Pedro García, Albert Astals) that kindly accepted to be interviewed for this work. I also met in depth the GNOME and KDE desktops which localization systems are analyzed in chapter6.
- **Systems Integration** subject in conjunction with the interview to Miguel Vidal as OpenBSD translation coordinator gave me the idea of keeping track of the history of the Debian Administration Handbook, and its predictable explosion of translation works after it was released with a free cultural work license.

7.3 Future work

7.3.1 More on localization results as free culture works

Some examples of standardized, quality translation results delivered to the public have been introduced in section 4.3.3; they are driven mostly by Public Administrations, since the professional translation world is usually reluctant to provide their translation memories to the community.

The community work on translations usually provide results with lack of standardization in translation formats: PO, PHP, CSV, XLIFF... This makes difficult the re-utilization of translations.

It could be interesting to continue the research in order to learn what kind of measures can be taken in order to overcome this limitation and increase the reusability of translations.

7.3.2 Apply libre software techniques to cultural works localization

There are other kind of projects, different than software, that are using similar processes and tools for localization, and taking also benefit of the increment of productivity. Among them, we have for example the collaborative project for subtitle translations of TED Talks[1], or the translations within Wikimedia projects[2].

[1] http://www.ted.com/pages/287

[2] http://meta.wikimedia.org/wiki/Translation

7.3.3 Mining for localization in libre software projects

Due to the availability of the source code of libre software projects, and the fact that many of them use an SCM (Source Code Management) system that keeps track of the history of changes to all the files, it is possible to inspect the source tree or mine the SCM logs to get information about the software. This would provide factual details about how localization is driven and may even throw hints about vulnerabilities in the process (as, for example, long-time untouched translation files with may mean that there are parts of the project that are outdated or unmaintained).

In Appendix D an initial analysis have been performed on Tux Paint project to show how this mining could be done. Some metrics and results are presented, although a much more in-depth research would be needed, specially if we want to compare the situation of two or more different projects.

Appendix A

Libre software tools for localization

Libre software communities use libre software tools to carry out their localization tasks.

Mata [Mat08] has researched about the presence of free software in the IT environments used by translators in general (not necessary translating libre software). Cánovas [CS08] and Díaz-Fouces [Fou08] show that there are mature, libre software tools that they can use. Some of them have been already introduced as case studies. Other applications can be found in directories in [FA11] and [Cor11]. In this appendix we provide a list contained all the tools mentioned in this document, and other among the most relevant tools, based on former referenced literature.

Damned Lies	`l10n.gnome.org`
Drupal localization	`localize.drupal.org`
Gettext	`www.gnu.org/software/gettext`
Gtranslator	`projects.gnome.org/gtranslator`
Launchpad	`www.launchpad.net`
Lokalize	`l10n.kde.org`
Poedit	`www.poedit.net`
Pootle	`translate.sourceforge.net/wiki/pootle`
The Translate Toolkit	`translate.sourceforge.net`
Transifex	`www.transifex.net`
Virtaal	`translate.sourceforge.net/wiki/virtaal`
Weblate	`weblate.org`

Table A.1: List of the localization tools analyzed in this work

Anaphraseus	anaphraseus.sourceforge.net
Apertium	www.apertium.org
bitext2tmx	bitext2tmx.sourceforge.net
Jubler	www.jubler.org
Mozilla Firefox useful plugins: WordReference, Answers.com (dictionary and thesaurus), QuickTranslation, Converter, FoxLingo	addons.mozilla.org
Okapi Framework: Rainbow, Olifant, Album, Tikal, Abacus	okapi.sourceforge.net
OmegaT	www.omegat.org
OmegaT+	omegatplus.sourceforge.net
OpenLanguage Tools	open-language-tools.java.net
OpenTrad	www.opentrad.com
Subtitle Edit	www.nikse.dk/SubtitleEdit
TranslateWiki	translatewiki.net
Transprocalc	code.google.com/p/transprocalc
Wordforge	sourceforge.net/projects/wordforge2

Table A.2: List of other localization tools

Appendix B

Interviews to translators

B.1 Miguel Vidal, OpenBSD Spanish Documentation

1.- Please introduce yourself and explain how and when did you start with libre software translations.

Since 1999, I have been involved in the Spanish-speaking free software community and other open source software projects related to Linux, OpenSolaris and (more recently) OpenBSD. I've been working as a Linux and Unix System Administrator since 2001. I'm focusing mainly on architecture design and server deployments with Unix and Linux systems.

My contributions to free software translations has been really humble and short. It all comes down to being in charge of the Spanish translation of the OpenBSD documentation[1].

Furthermore, I made a few FLOSS translations ten years ago. For example, I contributed to Catalan translation of GNU nano[2][3], a text editor designed as a free replacement for Pico. But these were only occasional contributions.

2.- Which tools are available/recommended in the project to help translators? Which ones do you use? (For example, there is a Pootle server but you usually download the PO file and translate it offline).

I'm translating web documentation, not source code. So, I don't work over .po files, but

[1] `http://www.openbsd.org/translation.html`

[2] `http://svn.savannah.gnu.org/viewvc/trunk/nano/THANKS?revision=693&root=nano&view=markup`

[3] `http://svn.savannah.gnu.org/viewvc/trunk/nano/po/ca.po?revision=547&root=nano&view=markup`

HTML files. My environment consists of the following tools:

- A text editor ('Vim')
- A 'diff' tools
- An HTML link checker ('linkchecker')
- An HTML validation program ('validate')
- A CVS client for commits (yep, OpenBSD project uses a CVS repository)

3.- Is there a translation team in the project? Which are the revision processes? Do you focus on revision or on translating the untranslated strings?

Yes, there is a translation team coordinated by Antoine Jacoutot and Benoit Lecocq. Furthermore, each project translation has a person responsible who facilitates sporadic contributions/contributors, checks quality translations and makes commits. So, as Spanish responsible, I have to face both issues: revision and translating/updating the untranslated things.

4.- Since you started to get involved in this project, did you increase your contributions to libre software? (For example, translating more languages, doing other kind of contributions to the project as bug reporting or programming code, or participating in other libre software projects as translator...)

Yes, maybe since then I have been more involved in the OpenBSD community, I've made some patch and bug report, etc.

5.- Which are the benefits you perceive for contributing in libre software translations? Any bad counterparts?

Benefits are to participate and contribute to free software community without being a developer/hacker. Also, in OpenBSD case, read and learn from their great documentation.

The counterpart is that it's a hard, steady work, lonely and often undervalued. Maintaining the translation up to date is critical: offering outdated information will just misguide people. So you must think if you really will be able to dedicate regularly to the translation work.

6.- Which are your future plans?

I would like to continue to keep up to date Spanish translation. When I took care of Spanish translation in 2010, it was orphaned since six years ago! So, there is much work to do. It's a long term job.

7.- Anything else you may say?

The OpenBSD project aims to maintain many high standards, one of them is to supply users with excellent documentation. A documentation error (or lack) is considered a serious bug and treated as harshly as any other serious bug. Thus, helping in such demanding project is a great challenge and a big responsibility. I'm very proud to do my bit.

B.2 Pedro García, Support Mozilla Messaging (SuMoMo)

1.- Please introduce yourself and explain how and when did you start with libre software translations.

Hi! My name is Pedro Garcia Rodriguez and I'm part of the Mozilla SuMoMo, SuMo, l10n and Mozilla-Hispano communities.

2.- Which tools are available/recommended in the project to help translators? Which ones do you use? (For example, there is a Pootle server but you usually download the PO file and translate it offline).

Almost everything I do translating software for Mozilla is done via a special wiki, called KitSune... developed by Mozilla to provide support to their users. It's easy to use so this is what makes it powerful but doesn't have the utilities that for example can have other translation utilities.

The part of my work that is not done via KitSune is done via Pootle... so here is where I stop translating... I hate Pootle and PO files...

3.- Is there a translation team in the project? Which are the revision processes? Do you focus on revision or on translating the untranslated strings?

Mostly all my work is done translating new articles or updating old ones... I find difficult done my on translations, revision them and accept them... I told you this because doing everything is not a hard work but after reading and writing, reading and writing, reading and writing, you start to fail seeing your own errors so I leave this part of the work to the "revision team" of Mozilla-Hispano.

Even if I am making a revision of an article and make changes I send it to a different person to not avoid my mistakes...

4.- Since you started to get involved in this project, did you increase your contributions

to libre software? (For example, translating more languages, doing other kind of contributions to the project as bug reporting or programming code, or participating in other libre software projects as translator...)

I started on this while I was bored in my room alone in Madrid... I'm from Pontevedra and I'm alone here in Madrid so I decided to do something useful and different...

I started the Galician translation of SuMoMo.

5.- Which are the benefits you perceive for contributing in libre software translations? Any bad counterparts?

The work that I am doing as translator and being an active member of the Mozilla Hispano community allow me travel the world... I have been in Berlin, Prague...

I missed the Wishtler Summit last year and in two months I will be in Buenos Aires... That's the reward for the work I've done all these years.

6.- Which are your future plans?

Quit translation!!!! Right now I have in mind contributing in a different way so when I complete the 100% of the Spanish translation of SuMo... I'm done!!!

So probably I will try to move to the IT part of Mozilla.

7.- Anything else you may say?

Nothing... Good luck!!! with your Master Thesis!!

B.3 Albert Astals, KDE

1.- Please introduce yourself and explain how and when did you start with libre software translations.

Hi, my name is Albert Astals Cid, I'm from L'Hospitalet de Llobregat in Spain. I have been collaborating with the KDE project since 2002 when I started improving Catalan translations for kdeutils, the KDE utilities package.

2.- Which tools are available/recommended in the project to help translators? Which ones do you use? (For example, there is a Pootle server but you usually download the PO file and translate it offline).

In KDE we recommend the usage of Lokalize[4] as a GUI editor for .po files but you can use

[4]http://userbase.kde.org/Lokalize

any editor you want, we are not strict about that. Besides that we have a script that each night processes all the code, creates/updates the .pot files, and merges the existing .po translations with the new .pot files. We also have a few additional scripts that run to ensure the validity of the translations like pology sieves[5]. We also have a webpage `http://l10n.kde.org` where you can check the statistics of the languages translates and serves as point of contact for the people that want to engage with the team.

3.- Is there a translation team in the project? Which are the revision processes? Do you focus on revision or on translating the untranslated strings?

KDE is a huge effort to translate so we actually have a "coordination" team and then one team per language. Each language team has different processes/policies depending on much available manpower they have.

4.- Since you started to get involved in this project, did you increase your contributions to libre software? (For example, translating more languages, doing other kind of contributions to the project as bug reporting or programming code, or participating in other libre software projects as translator...)

Yes, since 2002 I've been involved in lots of other projects inside KDE as a developer and also outside like poppler, to point in which I did not have time to translate anymore and stopped translating around 2008.

5.- Which are the benefits you perceive for contributing in libre software translations? Any bad counterparts?

As said, I'm not a translator anymore, but there was benefit huge benefit in increased knowledge of the target language (in my case Catalan) when i was.

6.- Which are your future plans?

Conquer the world! Sorry. I should have not said that. I don't really make plans, I just go along.

7.- Anything else you may say?

I encourage anyone to try to translate to their native tongue. It is a really gratifying experience and you will become much more proficient in it. And of course in KDE we are welcome to new translators so pay us a visit!

[5]`http://techbase.kde.org/Localization/Tools/Pology`

B.4 Javier Taravilla, debian-l10n-es team

1.- Please introduce yourself and explain how and when did you start with libre software translations.

My name is Javier Taravilla, and started my translations in Free Software projects in Debian, driven by the community or business practices of free software, necessary to complete the "Free Software Master" of Libresoft.

2.- Which tools are available/recommended in the project to help translators? Which ones do you use? (For example, there is a Pootle server but you usually download the PO file and translate it offline).

In Debian l10n team, I translated the web. So, the tools that I needed and used, were simpler than used people to translate .po files or deb-conf templates. The web sites I visited and used me for my work were:

- The robot coordinating translations: `l10nbot@debian-es.org`
- The interface translation statistics: `http://www.debian.org/devel/website/stats/es.html`
- Group's mailing list: debian-l10n-spanish

To translate I worked off-line and used gedit. But there are tools for translating strings like "gtranslator" of GNOME, "KBabel" or "Lokalize" of KDE, which people from the list of Debian used.

3.- Is there a translation team in the project? Which are the revision processes? Do you focus on revision or on translating the untranslated strings?

Yes there is a team of volunteers who translated, reviewed and uploaded to the repositories the translations approved.

And for the translation and revision in Debian, using a key system for items such as [ITT], [RFR], [ITR], [LCFC] or [DONE], corresponding to different phases in which a translation is from when you start up that is uploaded to the CVS.

1. So when you decide to translate a PO file, debconf template or WML file, you have to send an email to the list, with the label [ITT] equivalent to "Intent To Translate", marking

that a file has been booked by a translator to update. That is, if you send an email with the subject "[ITT] wml://devel-manuals.wml" means that you will translate the WML file "devel-manuals".

2. Then a few hours or days later when you have the translation, send an email with the subject [RFR], that means "Request For Review" which states that the translator has done his work with the file. In the mail you send the attachment, so that other translators can review and provide input. Normally, leave this file for 7 days, before moving to the next state. So would you send "[RFR] wml://devel-manuals.wml", with the translated file attachment.

   (a) Other translators or yourself, can use the tag [ITR] "Intent To Review" to indicate interest to broaden the review time of a file. This is so that a file does not go to the next phase when the interest of time, size, or interest of the people on the list, the review can not be done in the usual deadlines.

3. "Last Chance For Comments" [LCFC] is the key which indicates that the translation is complete and the changes or contributions have been incorporated. So, that the file is ready to be sent to the person who upload it to CVS. It is advisable to leave it for another 3 days in this state and if no further input, advise for members with permission to include in the repository.

4. After all you send an email with the subject starting with [DONE] to note that the work is completed and implemented, or that the translation is abandoned whatever reason.

Importantly, all these items and changes will be reflected automatically in the robot translation: http://www.debian-es.org/cgi-bin/l10n.cgi?

And all this is well detailed in: http://www.debian.org/international/spanish/robot

Finally, I focused mainly on the translation of WML files, which are the ones that translates the web. From time to time, or when requested by a translator, I also conducted reviews of wml and PO files.

4.- Since you started to get involved in this project, did you increase your contributions to libre software? (For example, translating more languages, doing other kind of contri-

butions to the project as bug reporting or programming code, or participating in other libre software projects as translator...)

No, as a "contributor", I have not participated in more free software projects. I promote it among friends and acquaintances, or at work, and I defend and use, but I have not participated in more projects, more than send some bug report.

5.- Which are the benefits you perceive for contributing in libre software translations? Any bad counterparts?

The main benefit was the experience of actually participating in a community and knowledge of other translators. The only counterpart might be the time, but it is not well understood, because it was an investment in learning.

6.- Which are your future plans?

Back to actively participate in the list.

7.- Anything else you may say?

I encourage anyone interested in contributing and creating or knows he can not do as a code developer, to participate in a community of translation.

Appendix C

Contributing to translations: a personal experience

C.1 Introduction

In this appendix I explain my experience about contributing translations to Spanish of several libre software projects. I list them in chronological order, covering a total of several years, different phases of involvement, different tools used and also different levels of 'performance', depending on each time acquired skills to act as part of a libre software community. I hope the reader will find this story interesting, specially if he or she is considering to get involved in any free software community as translator, for the first time.

C.2 PHPScheduleIt and Drupal 5

I was in charge of selecting and developing a web-based room reservation system, in order to handle my School's computer labs timetable. I found that the free software **PHPScheduleIt**[1], which consisted in a set of PHP scripts acting on a MySQL database, and it was as of version 1.2.8, was suitable with small adaptations.

In that time I knew that it was possible to modify the software by accessing the source code, and we needed to 'tweak' the system so the name for different things was different than the original one (for example use 'lab' instead of 'resource') and fix some typos. I just find the file

[1] `http://www.php.brickhost.com/`

I needed to modify by inspecting the folder structure of the PHP scripts of PHPScheduleit. I opened the required file (`es.help.php`, in a folder called `lang`) with a text editor and made my changes to the translation strings for Spanish, saved the file and the work was finished.

This was not exactly a translation work, but for example I fixed some typos in the translation file and it would be nice to have contributed back to the community that fixes. But in that time I did not know how to use control version systems, or submit a patch. I could have been subscribed to the main mailing list and send the file as attachment, but in that moment I did not know that the projects accepted easily external contributions without further involvement. We deployed the system with other adaptations to the code, but as the original project evolved, we did not upgrade to the new versions, because migrating our 'adaptations' required too much development and maintenance effort.

In the same times, I completed the translation of Scheduler module for Drupal 5, since the Spanish translation file was not completed. In this case the file was an `es.po` file but for me it made no different than former one: it was a text file so I use a text editor to open and modify it. I imported the new translation into my own Drupal site, but did not contributed back my translations to the community.

What did I learn?

- Thanks to the free software licenses and free access to the source code, if a program does not include a translation to your language or if it is not complete you can complete the translation yourself.

- If you contribute your changes *upstream*, you benefit from further updates with less maintenance effort required, and the community also receives your improvements, so probably more people will contribute too and enrich the project.

- However, in order to contribute your translations or patches, you need to know how the project works, who is in charge of i18n/l10n, and coordinate with other people, a task that involves time for learning and interacting with the community.

C.3 Drupal 7: contributing translations easily

Next time I deployed a Drupal site the stable version was Drupal 7. The deployed site was my School's website about libre software. That work was part of the Practicum of my M.Sc. on Libre Software, so I already knew that it was important to engage with the community of the software projects that you use or deploy in order to stay tunned about updates and also participate or contribute your improvements. On the other side, Drupal 7 includes a completely new localization platform, allowing to contribute translations in some different ways. I joined the Spanish team by clicking "Join" in the Spanish team group under the website http://localize.drupal.org, and used the web interface of that website to select the modules that I wanted to translate and complete some pending translations. In addition to this, I installed the localization modules in my Drupal site so I get a web interface for translating the strings that are pending from the modules that I have installed for my site. It is a very comfortable way of translating, since you already get your changes be integrated in your local site, and with just clicking a button ("Update translations") the strings that you translated are sent to the Drupal localization site (they will wait in the moderation queue to be approved), and you also download automatically the new approved translations that other people contributed to the system. With this both system I got approved 187 translations for Drupal[2].

What did I learn?

- Having a web platform to ease the translation is really a great improvement in order to lower barriers to new contributors.
- The Drupal system combines the needs of the end user to easily tweak the system and have their translations integrated in their production sites immediately (without waiting for approval and integration with *upstream*) and contribute your improvements to the whole community, also in an easy way.
- However, moderation queue is the bottleneck of this system. A single English string may have multiple suggestions sent from translation modules in different Drupal sites, whose administrators have good intentions but did not read the rules, conventions and glossaries

[2] http://localize.drupal.org/user/166874

for their language team in Drupal.

C.4 Pleft

In 2011 the M.Sc. on libre software students decided to use a web-based appointment planner in order to altogether decide a suitable date and time for a non-academic meeting called 'MSWL and Beers'[3]. The initial suggested tool was privative software so I tried to find a free software alternative, and I found **Pleft**[4] which was developed by two students from Amsterdam.

The tool was available in several languages, but not in Spanish. For MSWL students and teachers it was not a problem, since all of us know English, but I liked the software and thought that I could recommend it to other people or use it to arrange meetings with other friends that don't know English, so it could be a nice thing to have it in Spanish. The web based system is very visual and has few text strings presented to the user so the translation could be done in a very short time.

The documentation in the Pleft website was explaining that I needed to join **Transifex** web platform in order to contribute translations to the software. I created a Transifex account and joined the Pleft team, requested the Spanish language using the Transifex platform. The Transifex platform includes the possibility to send and receive messages to the projects maintainers and receive email notifications when a message has arrived to your Transifex inbox. I received a message welcoming me to the Pleft team and and approving the Spanish team. It was not an automatic message, it was written by one of the Pleft developers, so I felt really welcomed and go to translate the application to Spanish. I asked small questions about the translation (format of dates) by the Transifex platform, and always got a quick response. When the translation was finished I asked if there was a Pleft test site where I could see and review my translations "in their context". Pleft developers provided an URL for a test site so I reviewed my own work and corrected and improved some strings. The Pleft development team uploaded the translations to the production site some days later, and I was happy for having Pleft in Spanish with little effort involved, and in a relatively short time.

Some days later, the Pleft admins asked me to translate some text for the promotion of the

[3]http://docencia.etsit.urjc.es/moodle/mod/forum/discuss.php?d=12867
[4]http://www.pleft.com

tool in several websites. I didn't use any tool to translate that texts, since they were plain texts, and not part of the software.

I was happy that I could translate a complete web based application without need of downloading the source code, use a SCM or have to know how to send a patch, or other "development skills", thanks to Transifex.

Some months later, I visited again the Pleft website and it was changed in two ways: a new version was released, and the Spanish translation was gone. I read the Pleft blog and learned that the development was moved from Google Code (a software forge) to GitHub (a different software forge). I was not subscribed to the Pleft development mailing list (I thought it was not necessary for me, since Transifex platform was managing all the communication needed for translators) so I didn't get the news about changing the forge. I went to GitHub, browsed the new version source code, and saw that the localization files were three, while in Transifex there were only two files, and their names and contents were different. So it looked clear that the new version had a different architecture and the administrators had not updated the Transifex project.

There were two problems to solve. One of them was to translate the new version of the project to Spanish. The other one was to update the connection between the project and the Transifex platform, if the Pleft team was still using it to manage translations.

For the first problem, in that time I already knew how to use git and GitHub, so I forked the project to my GitHub account, cloned it to local, updated and reviewed the Spanish translation files. This time I used Poedit, because I wanted to try that program, and with it it was very easy to unset the "fuzzy" property to the machine-translated strings. I committed my changes to my local repository and pushed them to my GitHub account. Then I asked for a "pull request" to the original project. Since the "pull request" opens an issue in the issue tracker system, I commented about the outdated translation files in Transifex and asked if they were willing to keep using Transifex or not.

I received a quick answer, the Pleft admins were not aware of the problem with Transifex, and they did not know why it happened. I read the Transifex documentation and concluded that they (or somebody with privileges) had to update the URL to the source code repository, which probably was still pointing to Google code, and the project was not there anymore. I updated the issue with this suggestion and got an answer that as soon as the Pleft administration are available the issue was going to be fixed (both updating the translations and handle the problem with Transifex)[5].

What did I learn?

- Although you don't participate in the project as a developer, it is useful to be subscribed to the main mailing list or project forum, in order to get updates about the roadmap or changes that may affect the translations.
- If there is somebody in charge of internationalization, he or she could filter that information and coordinate with translators, so the translators don't need to receive all the information and events related to the project which maybe are not of their interest.
- Localize an application is more than translate the text strings presented to the user. Webpages with help, user manuals or marketing, advertisements are also needed to be translated.

C.5 F-Droid

The F-Droid Repository[6] is an easily-installable catalog of FOSS (free, open source software) applications for the Android platform. The server contains the details of multiple versions of each application, and the Android client makes it easy to browse, install them onto your device, and keep track of updates. I began to use this the F-Droid Android client in my smartphone and liked very much the project. But I saw that the Spanish translation was not completed (some messages were showed in English). In addition to this, the information of the Android applications listed in the repository was in English too. This was an important handicap to spread the word about F-Droid in my family and friend environments, since they use smartphones but they don't know English, and on the other side, the Google Play Store (the application repository that is installed by default in the Android smartphones) is completely translated to Spanish, and most of its applications too[7]

[5] `https://github.com/sander/pleft/pull/46`

[6] `http://fdroid.org`

[7] Many of this Google Play Store application descriptions are machine translation with a poor quality, but a Spanish speaking person is at least able to figure out what each application does.

I went to F-Droid website and looked for how to contribute with translations. They used a web-based translation platform called Pootle. I already knew Transifex so I thought it was more or less the same.

I created an account in the F-Droid Pootle server and joined the Spanish team, but I could not see the files that needed translation. Pootle has no built-in message system and the F-Droid project uses forums as communication channels, so I opened a thread in the development forum to ask about how to translate the pending strings [8]. The F-Droid administrator quickly answered my question updating the Pootle server so I could go and translate the pending strings. I also asked about the correct place to coordinate between translators in order to contact the previous translator to Spanish, and the F-Droid project leader created a forum for translation-specific issues[9]. I also asked about translating the application descriptions to Spanish, which it seemed not be possible in that moment, and got an answer explaining that the internationalization of that information was under development. I tried to contact with the Spanish previous translator in order to generate a more neutral-Spanish translation for F-Droid. He agreed with me so I updated some strings in the Pootle server, that will be shipped in the next update of the F-Droid client.

What did I learn?

- As with previous experiences, quick response of the project manager or person in charge for internationalization was key to get me involved in the project as translator.

- Having a web platform for translations is not enough. It is important to keep it up to date and ensure a communication channel between translators (specially when the project grows) and with the rest of the developers.

- When the program is developed without taking into account all the possible components to be translated (that is, define the internationalization objects), later it is more complicated to carry out this task.

[8]`http://f-droid.org/forums/topic/contribute-to-translations-in-pootle/`
[9]`http://f-droid.org/forums/topic/how-to-coordinate-between-translators/`

C.6 Android applications

Since I was an F-Droid user and had a free software repository at my hand, I decided to contribute to Android free software community translating in my free time some applications that I was already using in my smartphone.

C.6.1 Trolly and E-numbers

I am user of these two Android applications, and both of them are free software. I cloned the repositories and created the Spanish `strings.xml` files and translated the applications with a text editor. I was not sure about how to send a patch or a merge request to the original repositories (they were hosted in Google Code and BitBucket forges, which they don't have the "Pull request" easy system as GitHub). So I opened an issue for each of them, attaching the translated files[10]. Unfortunately, it seems that both applications are abandoned so I didn't get answer for my requests. How to proceed in this case? I can build my own binary with the Spanish translation and use myself and distribute it directly to my friends. I can fork the project and maintain that fork which includes a Spanish translation. I can try to contact the project leaders in order to know if they allow me to inherit the project and manage it from now on. I am not sure about willing to get involved further than as a translation to these applications, so I still did not make a decision about what to do.

C.6.2 Speech Trainer

I am using this application too, and found it very useful in order to train my public talks or improve my English pronunciation. The application was not in Spanish and it was hosted in GitHub. The first time that I made the Spanish translation I was not sure about how GitHub works so I decided to download the English template, modify it with a text editor so I got an Spanish localization file, and open an issue in the GitHub Speech Trainer project attaching the file. The project manager quickly updated the program and released a new version to the Android Market to let me review my work. I downloaded the new version and tried, it was working well. I asked about translating the information of the Android Market and we did it

[10] `https://bitbucket.org/uaraven/enumbers/issue/3/` for E-Numbers and `http://code.google.com/p/trolly/issues/detail?id=19` for Trolly

inside the issue tracker, since that information was plain text and the developer had to include it “by hand” into the system.

I asked about translating the “Help” and “About” HTML files that were also part of the application. That part was not internationalized but the Speech Trainer developer said that if I provide a `help_es.html` and an `about_es.html` he could find the way to integrate them in the program.

Some time passed and I learned how to use GitHub, so I forked the application, clone it to local, copy the `help.html` and `about.html` files and renamed them to `help_es.html` and `about_es.html`, translated the text with a text editor, and saved my work. I committed my changes to my local repository, pushed them to my GitHub fork, and issued a pull request to the original repository. I updated the former issue in order to explain what did I did and link both issues, and my changes were accepted soon.

The complete story about the Spanish translation of Speech Trainer can be read here: https://github.com/wrr/speech_trainer/issues/1.

What did I learn?

- GitHub makes easy to contribute to a project.
- Thanks to free software licenses, if a project is abandoned you still can make modifications or contributions, and give life to it again.
- As I said before, translating the strings presented to the user sometimes is not enough, there is additional information to be internationalized and localized.

C.7 Future plans

- Keep on maintaining the Spanish translations for the new versions of each program that I already translated.
- Take a decision about what to do with the abandoned Android applications.
- Contribute Spanish translations to other free software Android applications that I am using.

- Get more involved in the F-Droid project, contributing to the internationalization of the application descriptions.
- Contribute, if needed, to the Debian Administration Handbook translation (as I need to read the book anyway for improving my system administration skills at job).
- Join the FSF Spanish translation team and contribute translations to the web pages, if needed.
- Keep on blogging and studying about translations in free software.
- Suggestions are welcome!

Appendix D

Mining for localization in libre software projects

In this Appendix, I will explain how the SCM logs of libre software project could be inspected or mined in order to get information about the software. The libre software Tux Paint will be used as example; a similar process may be applied to other programs.

D.1 General overview of the process

The source code history of certain libre software projects should be studied to try to find information about internationalization and localization tasks. The process followed may have several steps:

1. Clone the source code repository to local, to get the history of changes (SCM log).
2. Convert that log in a SQL database, using CVSanaly[1] tool [Car12].
3. Query the database to get the information needed, and present it in a comprehensive way.

D.1.1 Metrics to extract

In a summarized analysis to learn how localization is done, for each project, we could query its database to obtain the following metrics:

[1]CVSAnalY cvsanaly is a tool that extracts information out of source code repository logs and stores it into a database.

- Total number of files.
- Total number of i18n files.
- Total number of l10n files.
- Total number of languages.
- Total number of contributors.
- For each contributor, total number of commits (changes), number of commits involving at least one localization file.

We need to take into account the type of SCM. For example Tux Paint uses CVS and that CMS stores only the identification of the contribution "actually" performing the commit for uploading changes to the repository. Other projects may use for example Git where information about both the committer and the real author may be stored.

D.2 SQL queries

Based on the table structure of a CVSAnalY database [Car12], and knowing that Tux Paint uses gettext and PO catalogs, we can write the following SQL queries to obtain our metrics:

Total number of files:

```
select count(*) from files;
```

Total number of i18n files.

```
select * from files where file_name like '%.pot';
```

Total number of l10n files.

```
select * from files where file_name like '%.po';
```

Total number of languages.

```
select distinct(file_name) from files
where file_name like '%.po';
```

Total number of contributors.

```
select count(*) from people;
```

For each contributor, total number of commits (changes):

```
select people.id, name, count(*) as total_commits
from people, scmlog
where people.id = scmlog.committer_id
group by scmlog.committer_id order by total_commits desc;
```

Contributor and number of commits involving at least one internationalization file:

```
select people.id, name, count(*) as total_commits
from people, scmlog
where people.id = scmlog.committer_id
and scmlog.id in
(Select distinct commit_id from actions_file_names
  where file_id in
    (select id from files where file_name like '%.pot'))
group by scmlog.committer_id order by total_commits desc;
```

Contributor and number of commits involving at least one localization file:

```
select people.id, name, count(*) as total_commits
from people, scmlog
where people.id = scmlog.committer_id
and scmlog.id in
(Select distinct commit_id from actions_file_names
  where file_id in
    (select id from files where file_name like '%.po'))
group by scmlog.committer_id order by total_commits desc;
```

D.3 Results

The CVS source code management system stores a separate log for each module of the project (each main folder of the tree structure). In Tux Paint we have several modules. In table D.1 there is a summary of the file structure for each module.

	Tux Paint	Stamps	Website	Config	Music Magic	Drawtext
Files	1817	12502	2727	182	43	506
i18n files	1	1	1	1	0	0
l10n files	158	77	85	46	0	0
Languages	103	70	85	44	0	0
Committers	32	19	11	15	1	2

Table D.1: Some metrics in Tux Paint modules

We can see that for each module there is only one internationalization file. This simple structure probably is one of the key factors for Tux Paint to be translated to so many languages, since new translators can easily learn what to do: touch only 4 files to get the whole application and the website translated to one language.

For the analysis of contributors and their commits on internationalization and localization files, we have mined only the main Tux Paint module (results in Table D.3; for the sake of brevity only results of contributors that changed localization files are presented). For a complete analysis of the whole project, it would be necessary to mine each module and merge the results in only one table which takes into account all the contributors.

We can see that 21 out of 32 committers did changes on localization files. Most of the changes have been performed by Bill Kendrick, the leader of the project, who is the person in charge of receiving the translation files from people with no access to the repository.

We can find that access privileges have been given to 9 contributors (greendeath, ahven, terjeb, cos, giasher, najmi_zabidi , smarquespt , xandruarmesto and friedelwolff) that did changes only on localization files; this may be a sign of the importance of localization on Tux Paint, and the natural division of work that localization brings to any software project.

It would be interesting for example to see the evolution of commits in time for each contributor, to see if the last committers that came to the system are only contributing translations or getting involved in development too. The evolution of commits of Bill Kendrick may also give us hints about if he is still "centralizing" the management of the system, or if he is delegating in other committers.

We could also compare metrics between the languages that are "managed" under the Official

Committer Name	Id	Total Commits	Commits on i18n	Commits on l10n
1	wkendrick	8928	60	4422
2	huftis	1265	13	944
3	vindaci	87		12
5	greendeath	4		4
6	songhuang	61		18
7	srtxg	139		129
12	jfriedl	4		3
13	kartik_m	18		15
14	ahven	4		4
15	terjeb	1		1
16	el_paso	8		5
18	dolphin6k	65		5
19	secretlondon	780	7	585
20	perepujal	356	1	121
24	jchion	17		15
25	cos	3		3
27	giasher	1		1
28	najmi_zabidi	6		6
29	smarquespt	2		2
31	xandruarmesto	6		6
32	friedelwolff	1		1

Table D.2: Commits from translators in Tux Paint (main module)

Pootle Website, and the languages that are translated “manually”, to see if there are significant differences among them (for example, more people involved in its management, or updated more frequently).

Bibliography

[Ast12] Albert Astals. Presentation about kde. mswl case studies ii. Website, 2012. http://docencia.etsit.urjc.es/moodle/mod/resource/view.php?id=7056.

[Car12] GSyC LibreSoft (Universidad Rey Juan Carlos). Data analysis: Cvsanaly. Website, 2012. http://projects.libresoft.es/projects/cvsanaly.

[CCfTTC12] South Africa; Department of Arts CTexT (Centre for Text Technology, North-West University) and South Africa Culture. Atshumato project. Website, 2012. http://autshumato.sourceforge.net/.

[Com12] Creative Commons. About creative commons. Website, 2012. http://creativecommons.org/about.

[Con10] World Wide Web Consortium. Questions and answers about internationalization. Website, 2010. http://www.w3.org/International/questions/qa-i18n.en.

[Cor11] Gonçalo Cordeiro. Open source software in translator's workbench. *Tradumàtica: tecnologies de la traducció*, 0(9), 2011.

[CS08] M. Cánovas and R. Samson. Herramientas libres para la traducción en entornos MS Windows. pages 33 – 55, 2008.

[CS12] Marcos Canovas and Richard Samson. Open source software in translator training. *Tradumàtica: tecnologies de la traducció*, 0(9), 2012.

[Def08] Freedom Defined. Freedom defined. Website, 2008. http://www.freedomdefined.org.

[End11] Rikki Endsley. Transifex: The gsoc project that translated itself in an international business. Linux.com Website, 2011. https://www.linux.com/news/enterprise/biz-enterprise/495159-transifex-the-gsoc-project-that-translated-itself-in-an-international-business.

[FA11] Sílvia Flórez and Amparo Alcina. Free translation software catalog. *Tradumàtica: tecnologies de la traducció*, 0(9), 2011.

[For10] Gil Forcada. Gnome localization update for q1 2012. Website, 2010. https://live.gnome.org/TranslationProject/Survey.

[Foua] Free Software Foundation. Gnu general public license. Website. http://www.gnu.org/licenses/gpl.html.

[Foub] Free Software Foundation. Gnu general public license, version 2. Website. http://www.gnu.org/licenses/old-licenses/gpl-2.0.html.

[Fouc] Free Software Foundation. Gnu project. Website. http://www.gnu.org.

[Fou08] O. Díaz Fouces. Ferramentas livres para traduzir com GNU/Linux e Mac OS X. pages 57 – 73, 2008.

[Fou10] Free Software Foundation. GNU Gettext. http://www.gnu.org/software/gettext/, 1998-2010. [Online; Updated: 2010/06/06; Accessed 2012/01/25].

[Fou12a] Django Software Foundation. Django documentation. internationalization and localization. Website, 2012. https://docs.djangoproject.com/en/1.4/topics/i18n/.

[Fou12b] The Free Software Foundation. The free software definition. gnu project. Web, 2012.

[ftAoSISO08] Organization for the Advancement of Structured Information Standards (OASIS). Xliff version 1.2. oasis standard. Website, 2008. http://docs.oasis-open.org/xliff/xliff-core/xliff-core.html.

[ftAoSISO09] Organization for the Advancement of Structured Information Standards (OASIS). Open architecture for xml authoring and localization reference model (oaxal) tc wiki. Website, 2009. https://wiki.oasis-open.org/oaxal/.

[Her11] Raphael Hertzog. People behind debian: Christian perrier, translation coordinator. Website, 2011. http://raphaelhertzog.com/2011/03/03/people-behind-debian-christian-perrier-translation-coordinator/.

[Ini12] Open Source Initiative. About the open source initiative. Website, 2012. http://opensource.org/about.

[KDE12] KDE. Localization. Website, 2012. http://techbase.kde.org/Localization.

[Kov12] Petr Kovar. Gnome localization update for q1 2012. Website, 2012. http://blogs.gnome.org/pmkovar/2012/05/22/gnome-localization-update-for-q1-2012/.

[Mat08] M. Mata. Formatos libres en traducción y localización. pages 75 – 122, 2008.

[MW11] Lucía Morado and Friedel Wolff. Bringing industry standards to open source localisers: a case study of virtaal. *Tradumàtica: tecnologies de la traducció*, 0(9), 2011.

[new10] Gábor Hojtsy Drupal Newsletters Maintainer news. Drupal 7 coming, project, cvs and translation changes. Website, 2010. http://drupal.org/node/991166.

[Par11] Cristina Gomis Parada. Free software localization within translation companies. *Tradumàtica: tecnologies de la traducció*, 0(9), 2011.

[Per12a] Christian Perrier. Bubulle's blog. Website, 2012. http://www.perrier.eu.org/weblog.

[Per12b] Christian Perrier. Debian-installer language support, statistics and translators. Website, 2012. http://d-i.alioth.debian.org/doc/i18n/languages.html.

[RR12] Laura Arjona Reina and Gregorio Robles. Mining for localization in android. In *MSR 2012: Proceedings of the 2012 workshop on Mining software repositories*, 2012.

[Tra12] Transifex. Transifex help pages. Website, 2012. http://help.transifex.com/index.html.