A Preliminary Analysis of Localization in Free Software: How Translations Are Performed

Laura Arjona Reina, Gregorio Robles, Jesús González-Barahona

A preliminary analysis of localization in free software: how translations are performed

Laura Arjona Reina[1], Gregorio Robles[2], and Jesús M. González-Barahona[2]

[1] Technical University of Madrid (UPM), Madrid (Spain)
laura.arjona@upm.es
[2] GSyC/Libresoft, Universidad Rey Juan Carlos, Madrid (Spain)
{grex,jgb}@gsyc.urjc.es

Abstract. Software is more than just source code. There is a myriad of elements that compose a software project, among others documentation, translations, multimedia, artwork, marketing. In this paper, we focus on the translation efforts that free, libre, open source software (FLOSS) projects undergo to provide their software in multiple languages. We have therefore analyzed a large amount of projects for their support and procedures regarding translations, if they exist. Our results show that many, but not all, projects offer some type of support and specify some ways to those wanting to contribute. Usually, projects from a more traditional libre software domain are more prone to ease such tasks. However, there is no general way to contribute, as formats and procedures are often project-specific. We have identified as well a high number of translation-supporting tools, with many projects having their own one. All in all, information about how to contribute is the main factor for having a very internationalized application. Projects accepting and giving credit to contributing translators have high levels of internationalization, even if the process is rudimentary.

Keywords: free software; open source; libre software; translations; internationalization; localization; collaborative development; crowdsourcing; open innovation;

1 Introduction

It is common that free, libre, open source software (FLOSS) projects follow an open development model, accepting contributions from external developers and easing (at least in theory) the entrance of new members. The availability of source code, the use of version control systems, software forges and public mailing lists to manage the project are technical means to put in practice the freedoms that are guaranteed by the license of the software. However, many of these tools, documentation and support are focused on (source code) developers, who have the technical skills to use and understand them. Beyond source code, there is a myriad of elements that compose a software project, such as documentation, translations, multimedia, artwork, marketing, among others.

In this paper we put the focus on translations. We analyze how they are managed in a variety of FLOSS projects, if they have a defined process, and the tools and support that are provided. It should be noted that the use of libre software licenses creates a scenario in which any modification to the software is allowed, along with redistribution of those modified versions. This means for example that end-users or any third party may translate the software to their desired language or dialect, without the need of permission. This legal viability is usually viewed as an advantage of libre software versus proprietary alternatives, both by users and by developers. For users, it is a way to guarantee the availability of the software in their language, despite of the interests of the developer group or company. For developers, it is a way to increase the dissemination of the program, attract new users and new contributors to improve the project, specially in the case that the localization processes happens 'inside' the project. In order to ease this, libre software developers have created several means to materialize and take profit of the possibility to internationalize and localize a libre software project, in a similar way to what they do with coding tasks: developing helpful tools, collaborative platforms, giving credit to each contribution and creating a translator community around the project.

The main contributions of this paper are following:

- It provides a complete perspective of translations in FLOSS. To the knowledge of the authors there are few in-depth analyses of the field of translations in libre software.
- It looks if FLOSS projects are open to external contributions in the case of translations, and if they ease this task by providing tools, support and guidance.
- It studies if FLOSS projects have standard procedures and common tools that allow that the knowledge acquired from translating in one project to be re-used in other projects.

The structure of this document is as follows: next, we will introduce some definitions and concepts related to translation tasks. In Section 3 we introduce the research questions that we address in this paper. The next section contains the related research on localization in libre software projects. Then, we describe the sources of information and methodology used to answer the research questions, and show our results in Section 6. Finally, conclusions are drawn and further research possibilities are discussed.

2 Definitions

Software internationalization (often represented by the numeronym i18n) refers to the process to prepare a program to be adapted to different languages, regional or cultural differences without engineering changes [4]. It involves tasks as separating the text strings (those shown to the user in the different interfaces) to be translated, supporting several character sets, using certain libraries to manage dates, currencies or weights.

Software localization (often represented by the numeronym l10n) refers to the process of adapting the software to a particular target market (a *locale*) [4]. It involves tasks as translating the text messages, documentation and published material to the desired language or dialect, adapting graphics, colors and multimedia to the local language or culture if needed, using proper forms for dates, currency and measures and other details.
Crowdsourcing represents the act of a company or institution taking a function once performed by members and outsourcing it to an undefined (and generally large) network of people in the form of an open call [15]. 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 (what sometimes is called **crowdtranslation**).

3 Research questions

In this work, we target following research questions:

3.1 How do FLOSS projects enable localization?

With this research question we aim to know if FLOSS projects consider localization a separate task of the rest of the source code management. While it is possible to handle localization as any other modification of the source code of a software project, in a well-internationalized software project, the task of translation can be made by conventional users or other people that may not have the programming skills needed for contributing using software development tools.

To answer this question we will look at the websites of several libre software projects and specially at the pages related to how to contribute, and see if there are any special guidelines on how to contribute with translations, comparing the process to follow with the one for contributing with source code. In addition, we will inspect if the objects to be localized are enumerated, and look for the existence of specialized internationalization or localization teams, guidelines and support platforms.

3.2 What tools and collaborative platforms are used by FLOSS projects to approach l10n tasks?

The goal of this research question is to gather information about the tools that libre software developers have created to manage the localization process. These tools may be pieces of software such as standalone editors specialized in handling localization files, web platforms, scripts to integrate changes in translations into the source code repository, among others.

To answer this question, in addition to review the related literature, we will investigate libre software catalogs in order to find localization tools and platforms, and will inspect the website of libre software projects to assess if they recommend any particular tool.

3.3 What are the results and consequences of these approaches to software localization?

We plan to count the number of languages to which each software project is translated, and to know if there are significant differences depending of the translation tool or platform used, or other aspects of the localization process. We will review literature in order to find other consequences (economic, social...) of the efforts in localization made by the libre software communities.

4 Related research

Compared with matters related to source code and even documentation, the number of research articles that address the issue of translations in software is very scarce.

There are some papers that provide some insight on the presence of free software for translations (not necessarily translating libre software) [16]. For instance, Cánovas [1] and Díaz-Fouces [11] show that there are mature, libre software tools aimed for the translation of software. And directories with tools have been published, such as [7, 5], including a study of applications for crowdsourcing in translations by the European Commission [3].

From the software engineering perspective, Robles et al. analyzed the KDE desktop environment looking for patterns in different kind of contributions by the type of file (localization files, multimedia, documentation, source code, and others) [21]. From the field of economics, Giuri et al. analyze the division of labor in free software projects and how it affects project survival and performance [14].

In the libre software communities, 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 [10].

Souphavanh and Karoonboonyanan [22] provide a broad perspective on the localization of Free/Open Source Software (FOSS) for the benefit of policy- and decision-makers in developing countries, highlighting the benefits and strategies of FLOSS localization.

5 Methodology

We have gathered the information presented in this work from several publicly available sources: websites of the projects, documentation, mailing lists for translators, and related literature showed in the bibliography.

For each libre software project analyzed, we have written in a research script[3] following aspects: if there is information about how to localize the project, the recommended tools and website of localization, if internationalization and localization teams exist, and other remarkable aspects.

[3]The research script is publicly available at `http://gsyc.urjc.es/~grex/repro/oss2013-translations` for reproducibility purposes and further research.

For each libre software tool or web platform analyzed, we have annotated our research script with the following aspects: the type of localization tools (standalone software, plugin, web platform...), the year of first release, the original project for which was developed (if any), the number of projects that use the tool (if it is stated in the main website of the tool), if well-known projects that use the tool, and other remarkable aspects.

6 Results

As a preliminary analysis, we have analyzed 41 libre software projects, from small tools as Mustard -the microblogging client for Android-, to GNU/Linux distributions as Debian or Mandriva. We have analyzed groups of "similar" projects as the office suites OpenOffice.org and LibreOffice and the CMS Drupal and Wordpress.

The projects analyzed are listed and grouped as follows[4]:

- Operating Systems (10 projects): Debian, Ubuntu, Fedora, Mandriva, OpenSUSE, Android Open Source, CyanogenMod, Replicant OS, FreeBSD, OpenBSD
- GNU/Linux, cross-platform applications (16 projects): GNOME, KDE, Tux Paint, GNU MediaGoblin, Limesurvey, OpenStack, Wordpress, Drupal, Pootle, Scratch, BOINC, LibreOffice, OpenOffice.org, Mozilla apps, Calibre, VLC
- Windows projects (6 projects): Filezilla, Notepad++, 7zip, eMule, Azureus-Vuze, Ares
- Android projects (9 projects): F-Droid, Mustard, K9 Mail, AdAway, OpenStreetMap Android (OSMAnd), Barcode Scanner, aCal, Frozen Bubble, Cool Reader

We have analyzed 22 libre software tools or platforms for supporting the localization process. Among them we can find plugins for text editors, standalone programs for localization, and complete online frameworks. They are introduced in the following subsections. We also have compiled a list of 15 more translation tools for further research, which can be found in our research script, along with all the details about the selected projects and localization platforms.

6.1 How do FLOSS projects enable localization?

Information about how to contribute translations
Most of the projects have a dedicated page about how to contribute translations, and many include several objects to localize (in addition to the messages or strings presented to the user, for example the documentation and website). Some projects maintain translations for different releases (the case of Ubuntu) but

[4]More details about this sample (selection criteria and other) can be found in our research script.

most of them focus on the development release.
There are some projects with no information about how to contribute, neither code nor translations. This situation is more frequent in the case of libre software created for the Android or Windows platforms than in the case of cross-platform or GNU/Linux tools, we assume that because of *cultural* reasons. In Figure 1 we summarize our findings about this topic.

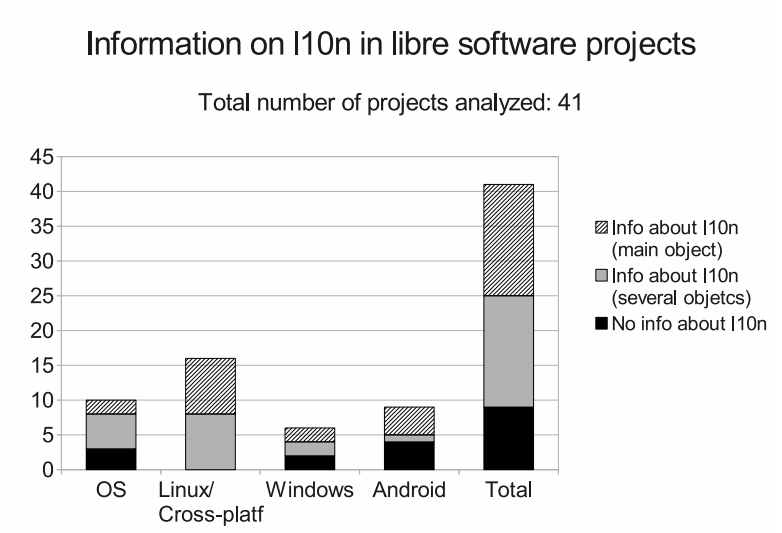

Fig. 1. Information about how to contribute translations

A remarkable exception is the Android Open Source Project, as it does not offer information about the localization of the software. There is a *Localization document*[5] explaining the process that Android developers should follow in order to make their applications localizable; the project has also published the *Hello, L10N* tutorial[6], providing an example of how to build a simple localized application that uses locale-specific resources [20]. However, there is no information about how to contribute translations to Android itself. In some forum threads Jean-Baptiste Queru (Technical Leader of Android OSP at Google) explains that Google translates Android internally, not accepting community contribu-

[5] `http://developer.android.com/guide/topics/resources/localization.html`
[6] `http://developer.android.com/resources/tutorials/localization/index.html`

tions[7]. Surprisingly, the libre software forks, CyanogenMod and Replicant OS, do not show either public information about localization. In CyanogenMod the topic has been discussed in several forum threads. However, at this time it still does not offer a specific protocol to contribute translations, managing them as any other modification to the source code. It does not have language teams or internationalization delegates either.

Observation #1: Many, but not all, projects offer information and support to contribute translations. Usually those projects from a more traditional libre software domain are more prone to ease such tasks to external contributors.

Existence of internationalization and localization guides and infrastructure

It is task of the internationalization team to study and decide how the localization of the project is going to be driven, setup the corresponding tools and write the proper documentation for translators and developers.

Most of the projects create language teams and use mailing lists to communicate (the use of the native language in those lists is common), and many software projects define the complete process of localization in a wiki page or website. For example, OpenStack discussed which platform to use: Pootle, Launchpad or Transifex, showing some comparative charts and assessment in their translations wiki page (old version[8]). The current version of the wiki[9] is a comprehensive guide of the localization project, both for translators and for developers. Another example of a complete guide for localization can be found in the Apache OpenOffice.org wiki[10].

Regarding the localization infrastructure, there are different approaches. Some projects use an external, web-based collaborative environment to coordinate the localization teams (for example, OpenStack uses Transifex, and Wordpress uses the official Pootle server). Others deploy an instance of those web-based software in a community server - the Fedora and Mandriva distributions have set up their own Transifex servers, and OpenOffice.org and LibreOffice offer their own, but independent, Pootle servers. Finally, other projects are developing their own localization tools: standalone programs -as Lokalize for the KDE project or the i18nAZ plugin for Azureus/Vuze-, and web based localization platforms, such as Damned Lies[11] -the translation platform developed and used by the GNOME project-, or the Drupal localization website[12] which is used in the Drupal CMS

[7] See for example: Proposal to add Basque translation to Android Source Code `http://tinyurl.com/b2fkpbn`; Hungarian localization – questions about legal issues `http://tinyurl.com/b9xqrek`

[8] `http://wiki.openstack.org/Translations?action=recall&rev=9`

[9] `http://wiki.openstack.org/Translations`

[10] `http://wiki.openoffice.org/wiki/Localization_AOO`

[11] `http://l10n.gnome.org/`

[12] `http://localize.drupal.org`

project. The Mozilla project uses (a) Verbatim (their own Pootle server) for the localization of some aspects, (b) an *in-house* developed wiki for the translation of their help documentation, and (c) Narro, an external web-based tool, developed by the Romanian translation coordinator.

There are as well efforts on providing tools for particular tasks of the translation process. One example is the KDE internationalization build system[13] which basically consists of a script running trough the KDE source code repository server, extracting the localizable strings in the KDE source code and generating the localization templates periodically. This way developers only need to care about writing code with localization in mind, and localizers will always find an updated template with the latest strings to be translated.

For 32 out of 41 projects that have been analyzed, we have obtained the way of submitting translations. Figure 2 shows the results.

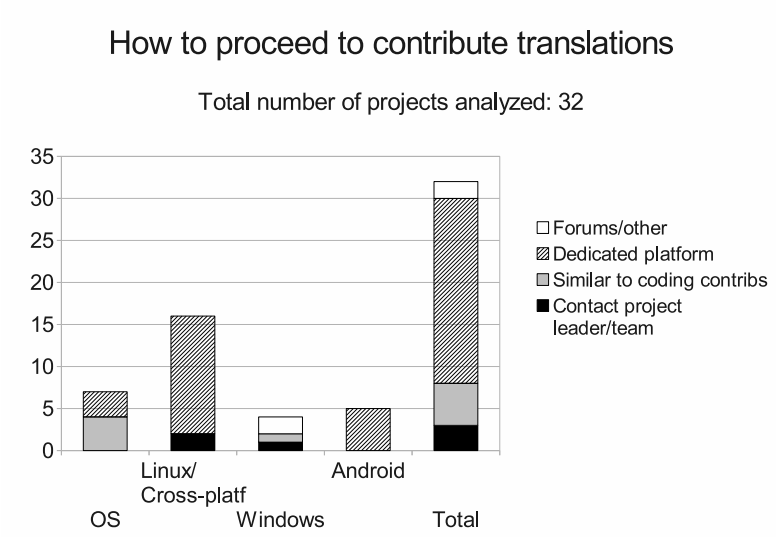

Fig. 2. Means to send translations in libre software projects

Some of the Operating Systems projects use a localization web platform, while others follow a traditional way, similar to code contributions (basically, sending translations via BTS or with mail systems). For GNU/Linux or cross-platform tools we can find that most of them clearly define a platform to manage l10n contributions, but there are some successful projects (in term of number of

[13] `http://techbase.kde.org/Development/Tutorials/Localization/i18n_Build_Systems`

translations) as Filezilla or Notepad++ that simply explain how to translate the corresponding files and ask to send the contributions to the project leader directly. The Windows specific projects analyzed do not use localization platform for managing translations; however, some of them offer templates in a forum thread and, if required, open another thread for translators to attach their contributions. Finally, for those Android projects that offer information about how to contribute translations, all of them use a dedicated platform, although not the same one (Transifex, Pootle, Weblate or Android-PHP-translator).

Observation #2: There is no standard way to contribute with translations. The tools and procedures, if they exist, are heterogeneous and sometimes even project-specific.

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. Hence, several tools can be used to perform the translation work.

According to the OAXAL framework [9], if we consider the complete process of localization (not only related to software localization), there are certain standard file formats to be used, as `tmx`, `tbx`, and `xliff`. XLIFF (XML Localization Interchange File Format) consists of an XML specification to hold translatable strings along with metadata [8]. This format is frequently used in professional translation, not only in software internationalization. Version 1.2 of XLIFF was approved by OASIS in February 2008.

However, these standards are recent. The libre software community has been concerned with localization from before these efforts, much before the OAXAL framework was designed. Therefore, other kind of file formats became *de facto* standards as the "PO" (Portable Object) file format for software translations.

The GNU Project [12] chose the Gettext [13] tool, implemented originally by Sun[14], to enable the internationalization of GNU software. Gettext explores the source code and extracts all the "printed" strings into a POT file, a template that contains a list of all the translatable strings extracted from the sources.

With time the use of PO files has become a *de facto* standard for libre software internationalization. However we can still find multiple file formats for internationalization of software, documentation and other elements in any libre software development project. For example, the Android Operating System uses "Android Resources" to make the localization of Android applications easy[15].

Observation #3: PO (Portable Object) files are still the *de facto* standard in use in libre software projects, although many projects use their own format. Newer, standardized formats such as XLIFF are seldom used.

[14] `http://compgroups.net/comp.unix.solaris/History-of-gettext-et-al`
[15] `http://developer.android.com/guide/topics/resources/localization.html`

6.2 What tools and collaborative platforms are used by FLOSS projects to approach l10n tasks?

In general, each language has an independent localization team that agrees on using locale-specific tools, formats, guidelines. Each team is, as well, responsible for the translation, review, and submission.

The tools used by these teams vary from traditional tools such as Poedit and plain text editors (with plugins for handling translation files), to sophisticated web platforms that integrate translation and revision processes, sometimes even automatically committing the changes to the versioning systems.

In our research script we provide a list with all the tools in use, which we have augmented with the most relevant tools referenced in the literature.

Text editor add-ons Emacs[16] and Vim have some tools for working with Gettext:

- **PoMode** for editing .po catalogs, **po-mode+.el** with extra features for PoMode, and
- **MoMode** for viewing .mo compiled catalogs.
- **po.vim** is the Vim plugin for working with PO files (`http://www.vim.org/scripts/script.php?script_id=695`).

Standalone tools

- **Poedit**[17] is a cross-platform editor for `.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.
- **Virtaal**[18] is a more modern translation tool, which allows working with XLIFF files [18].
- **The Translate Toolkit**[19] 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, as **poconflicts**, **pofilter** or **pogrep**.
- **Pology**[20] is a Python library and a collection of command-line tools for in-depth processing of PO files, including examination and in-place modification of collections of PO files (the **posieve** command), and custom translation validation with user-written rules. The Pology distribution contains internal validation rules for some languages and projects.

[16] `http://www.emacswiki.org/emacs/Gettext`
[17] `http://www.poedit.net/`
[18] `http://translate.sourceforge.net/wiki/virtaal`
[19] `translate.sourceforge.net`
[20] `http://pology.nedohodnik.net/`

Web platforms for supporting translation

In the last years, a number of web based translation platforms have been developed in order to benefit from the advantages of Internet-based collaboration, automate certain tasks and also lower barriers to external contributions. This new method has been called *crowdtranslation* [2].

In many cases, these platforms begin as a web-based infrastructure created for a particular project, and later they are used for other projects too.

- Probably the oldest example of a web based translation platform is **Pootle**[21], which is built on top of the Translate Toolkit. Pootle allows online translation, work assignment, provides statistics and includes some revision checks.
- **Launchpad**[22], which is a complete software forge including a powerful web based translation system, was developed by Canonical to handle the Ubuntu development, and initially developed under a proprietary license, but later released under the GNU Affero General Public License, version 3.
- **Transifex**[23] is a newer platform which enhances the translation memory support.
- **Weblate**[24] is a recent tool, developed by Michal Čihař (the phpMyAdmin project leader), which offers a better integration with the git versioning system.

Many different and popular projects decide to host their translations in an external service using one of the above platforms. The already mentioned `Damned Lies`[25] used in GNOME or the Drupal localization website[26] belong to this group. Wordpress uses the official Pootle translation server[27] (although some languages are handled in GlotPress, an online localization tool developed inside the project), and the FreedomBox Foundation and the Fedora community use Transifex[28] for the localization of their websites.

Observation #4: There is a large amount of translation-supporting tools.

6.3 What are the results and consequences of these approaches to software localization?

Results on the libre software projects

In our research script we have annotated the number of languages to which each

[21] `http://translate.sourceforge.net/wiki/pootle/`
[22] `http://launchpad.net`
[23] `http://transifex.com`
[24] `http://weblate.org`
[25] `http://l10n.gnome.org/`
[26] `http://localize.drupal.org`
[27] `http://pootle.locamotion.org/`
[28] `http://transifex.net`

project is translated. For some cases, the number of languages with more than 50% of the objects translated and the number of languages with more than 90% completion is provided. However, it is difficult to measure the success of the methods and tools used in these terms, since other causes may have to be taken into account.

We have found that the number of language teams or the amount of translations in a libre software projects does not depend on the translation tool(s) used. The most important factor is to have clear specifications on how to contribute with localization tasks. For example, the Notepad++ project recommends to download an XML file, translate it, test it and send it by email to the project leader. With this simple approach, the program has been translated to 66 languages (all the translators are credited in a dedicated page; this may contribute too to attract new translators).

However, we have identified some projects with users wanting to contribute translations, but without much success. These projects do not offer information on how to translate the software, or if this information is provided it is very sparse. In addition to Android (already mentioned in section 6.1), we have the case of the Ares/Galaxy project: its issue tracking system contains 18 tickets (out of a total of 33) with user-submitted translations to the program; many of them remain unconsidered.

Observation #5: Information about how to translate a libre software is the main factor for having a very internationalized application. Project accepting and giving credit to contributing translators have high levels of internationalization, even if the process is rudimentary.

Other results or consequences

Business opportunities

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 [2], both for trainees and for the libre software community.

Companies specialized in translation and localization of software may cover several cases where the community or company developing a software cannot reach, such as the maintenance of several versions of the localized system, or localization of the software for a specific customer [19].

One example of this is **Acoveo**[29], a company offering a solution for software localization consisting in a Maven and a Jenkins plugin that are offered for free (under the AGPL license) to their customers. The Maven plugin takes care of extracting all the translatable strings of the client's software, and sends the

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

strings to `www.translate-software.com`, where professional translators will localize them at a certain price per word. The Jenkins plugin allows the customer to control the localization workflow and progress.

Another example is **ICanLocalize**[30] which offers website translations, software localization, Text translations and general translations. They have developed a Drupal module called "translation management"[31], released under the GPL license, and the Wordpress MultiLingual Plugin[32] (which is licensed as GPL and offered at a certain cost, including the fees for the website content translation).

Finally, there is **Transifex**, a libre software project that has become a company offering hosting and support on its platform.

Localization results as free culture works
In addition to the the localization files of the software, **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) may be released as free cultural works too, lowering the barrier for new translations of other projects to the target language. An example of this is the **Atshumato Project**[33]. The general aim of this project is the development of open source machine-aided translation tools and resources for South African languages [6]. Among them, Atshumato releases Translation memories in the Translation Memory eXchange format (TMX) with Creative Commons Attribution Non-Commercial ShareAlike 2.5 license[34] 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).

Observation #6: Translation of libre software projects offers new business and cultural opportunities.

7 Conclusions and further work

In this paper, we have investigated how FLOSS projects manage translations by studying a big number of them for the information and support they provide, and the tools that are being used.

[30] `http://www.icanlocalize.com`

[31] `http://drupal.org/project/translation_management`

[32] `http://wpml.org`

[33] `http://autshumato.sourceforge.net/`, initiated by the South African Department of Arts and Culture and developed by the Centre for Text Technology (CTexT) at the North-West University (Potchefstroom Campus), in collaboration with the University of Pretoria.

[34] The Non-Commercial clause makes them not a completely "free culture work" but allows quite more freedoms to the receiver than traditional copyright application.

Internationalization and localization in libre software projects are two of their most interesting advantages for dissemination, competition with proprietary alternatives, and penetration in new markets. Therefore, we have seen how most of the libre software projects offering information about how to contribute or join the community, offer information about how to contribute with translations as well. From the projects under study, we have found that the GNU/Linux or cross-platform projects handle translations in a more systematic way (much more standardized than Windows or Android projects), probably due to the tradition of the collaboration spirit of the GNU project and the dissemination of Gettext, and the *cultural* influence of those platforms where contributions have always been welcome.

However, there is big heterogeneity in how projects proceed with translations. Procedures vary from project to project, and no generalized way exists to contribute, as is normally the case for source code. The heterogeneity can be observed as well from the technological point of view. There are many tools that support translations, resulting in a productivity increase for the project, but not allowing translators to reuse their knowledge on other projects. If we compare the tools that exist to contribute source code with the ones that are used for translations, we have not found a concentration of localization tasks in few platforms in the same way as some software forges have concentrated a high number of projects [17]. However, web platforms for translations such as Launchpad (more than 1800 projects) or Transifex (more than 1700 projects) are gaining popularity in recent times and may change this in the near future.

From our analysis of the projects, we have seen that publishing information about how to translate a libre software is the key element to have a project with a large number of translations. This has been found to be true even for smaller projects that use simple ways to accept contributions, even per e-mail or in the e-mail forums. If this practice is augmented with giving credit to the contributing translators, the results have been found to be even better.

Regarding future lines of research, a more comprehensive analysis (involving a higher number of projects) should be performed in order to measure the effect of using different tools and procedures in the localization tasks and management.

We think that due to the availability of the source code of libre software projects, and the fact that many of them use a versioning system that keeps track of the history of changes to all the files, it is possible to inspect the source tree or mine the versioning system logs to get information about the software. This could provide factual details about how localization is driven (for example, looking at changes in the localization files or the information about the translators present in their headers), give hints about vulnerabilities in the process (as, for example, translation files without any change for a long time, which may mean that there are parts of the project that are outdated or not maintained).

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, introducing additional effort for reviewers. It would be interesting to make a comparative analysis on software projects that

use the crowdsourcing models with the ones that maintain a stable translation team, in order to see if they evolve similarly o differently in number of languages, size of the translator community, and medium-long term involvement of the contributors).

References

1. M. Cánovas and R. Samson. Herramientas libres para la traducción en entornos MS Windows. In O. Díaz and M. García, editors, *Traducir (con) software libre*, pages 33 – 55. Editorial Comares, Granada, 2008.
2. Marcos Canovas and Richard Samson. Open source software in translator training. *Tradumàtica: tecnologies de la traducció*, 0(9), 2012.
3. European Commission. *Crowdsourcing translation*. Publications Office of the European Union, Luxembourg, 2012. `http://ec.europa.eu/dgs/translation/publications/studies/crowdsourcing_translation_en.pdf`.
4. World Wide Web Consortium. Questions and answers about internationalization. Website, 2010. `http://www.w3.org/International/questions/qa-i18n.en`.
5. Gonçalo Cordeiro. Open source software in translator's workbench. *Tradumàtica: tecnologies de la traducció*, 0(9), 2011.
6. 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/`.
7. Sílvia Flórez and Amparo Alcina. Free translation software catalog. *Tradumàtica: tecnologies de la traducció*, 0(9), 2011.
8. 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`.
9. 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/`.
10. Gil Forcada. Gnome localization update for q1 2012. Website, 2010. `https://live.gnome.org/TranslationProject/Survey`.
11. O. Díaz Fouces. Ferramentas livres para traduzir com GNU/Linux e Mac OS X. In O. Díaz and M. García, editors, *Traducir (con) software libre*, pages 57 – 73. Editorial Comares, Granada, 2008.
12. Free Software Foundation. Gnu project. Website. `http://www.gnu.org`.
13. Free Software Foundation. GNU Gettext. `http://www.gnu.org/software/gettext/`, 1998-2010. [Online; Updated: 2010/06/06; Accessed 2012/01/25].
14. Paola Giuri, Matteo Ploner, Francesco Rullani, and Salvatore Torrisi. Skills, division of labor and performance in collective inventions: Evidence from open source software. *International Journal of Industrial Organization*, 28(1):54–68, January 2010.
15. Jeff Howe. Crowdsourcing: A definition. Website, 2006. `http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html`.
16. M. Mata. Formatos libres en traducción y localización. In O. Díaz and M. García, editors, *Traducir (con) software libre*, pages 75 – 122. Editorial Comares, Granada, 2008.

17. A. Mockus. Amassing and indexing a large sample of version control systems: Towards the census of public source code history. In *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on*, pages 11–20. IEEE, 2009.
18. 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.
19. Cristina Gomis Parada. Free software localization within translation companies. *Tradumàtica: tecnologies de la traducció*, 0(9), 2011.
20. Laura Arjona Reina and Gregorio Robles. Mining for localization in android. In *MSR 2012: Proceedings of the 2012 workshop on Mining software repositories*, 2012.
21. Gregorio Robles, Jesús M. González-Barahona, and Juan Julián Merelo Guervós. Beyond source code: The importance of other artifacts in software development (a case study). *Journal of Systems and Software*, 79(9):1233–1248, 2006.
22. Anousak Souphavanh. *Free/open source software : localization.* United Nations Development Programme-Asia Pacific Development Information Programme, New Delhi, 2005.