A Empirical Study on the Status of Software Localization in Open Source Projects

Zeyad Alshaikh[1] and Shaikh Mostafa[1] and Xiaoyin Wang[1] and Sen He[2]
[1] Department of Computer Science, University of Texas at San Antonio, San Antonio, USA
[2] Department of Computer Science, Harbin Polytechnic University, Harbin, China
Email:{zeyad.alshaikh, shaikh.mostafa xiaoyin.wang}@utsa.edu, he.sen@hpu.edu.cn

Abstract

In modern software development, software localization is a key process to support distribution of software products to the global market. During software localization, developers typically convert all user-visible strings, resource files, and other culture-related elements to the local versions that are well accepted by local users. Despite the popularity of software localization, there have been few studies on the its current status in software practice, such as the proportion of localized projects, the most popular locales, and more importantly, the quality of software localization. In this paper, we present an empirical study on the status of software localization in open source projects. We find from that, popularity of software localization varies a lot in different User Interface (UI) frameworks and domains. Furthermore, we surprisingly find that only about 60% of string keys are actually translated on average in localized top software projects and software localization often span a long period of time in the software development history.

1 Introduction

In this era of globalization, most software applications have potential users from different regions of the world. A typical process to develop multiple local versions of a software application includes two steps: an internationalization step in which developers externalize all region-specific code elements (e.g., user-visible strings, measures, date formats, writing-direction-specific inputs, and sometimes also laws and policies) to resource files, and a localization step in which software localizers transform the original resource files to local resource files for certain locales [5].

Despite of the popularity of software localization, there has been few research efforts to study the current status of software localization in practice, and the major challenges faced in software localization. In this paper, we present an empirical study on open source Java software projects in SourceForge [1]. Specifically, we studied the factors that affect where a project is localized, the popular languages that projects are localized to, and the quality of software localization in the localized projects. Our major findings are as follows.

- Software Localization is widely used in open source projects, but the popularity of software localization varies a lot for different software domains and UI frameworks.
- The quality of software localization in top open source projects are relatively low.
- The localization of a software project often spans a long period of time during software evolution.

2 Study Design

In our study, we plan to answer the following research questions.

- **RQ1:** How popular is software localization in open source software projects?
- **RQ2:** What is the quality of software localization in open source software projects?
- **RQ3:** How long does it take for a software project to finish its software localization?

In our empirical study, we used two subject sets, both are downloaded from Sourceforge. We choose software projects from Sourceforge as subjects for the following two reasons. First, Sourceforge is a popular software repository with a large number of open source

(DOI Reference Number: 10.18293/SEKE2015-018)

software projects and long history. Second, Sourceforge provides the statistics of software downloads during different time period and from different countries.

Specifically, we built a *random subject set* including 2,500 randomly selected software projects which are active in 6 months and whose weekly downloads is larger than 10. Also, we built a *top subject set* which includes 10 software projects among the top 100 projects, which are from different software domains, and whose property files are in standard “key=value” format (so that it is possible to partially automate our data extraction in our study). To answer **RQ1**, we studied the random subject set due to its large size and representativeness. To answer **RQ2** and **RQ3**, we studied only the top subject set, because some manual in-depth inspection of resource files and version history are required.

To perform our empirical study, we extracted the following information from each subject software project. First of all, from the web site of each software project in our random subject set, we extracted the relevant meta data including supported locales, weekly downloads, domains, and used UI frameworks. Second, to answer **RQ2**, and **RQ3**, we manually identified the local resource files for each project in the top subject set, and extracted the file content and version history.

3 Study Results

In this section, we present and discuss the results of our empirical study.

3.1 Popularity of Software Localization

In our study, we find that, the proportion of localized software projects in the random subject set is 38.0%. To understand the factors that may affect software localization, We further studied our random subject set to find out how the proportion localized software varies with different number of weekly downloads, UI frameworks, and domains. The results are shown in Figure 1 through Figure 3.1. Since there are too many different UI frameworks and domains, we present only the results for the 5 UI frameworks / domains that are most popular in our subjects (the 5 labels on the left), with highest localization popularity (the 5 labels in the middle), and lowest localization popularity (the 5 labels on the right). It should be noted that a UI framework / domain may belong to two categories of above, so they may appear twice as a label in a figure. Also, for representativeness, we consider only UI frameworks / domains with more than 10 projects using them.

From the figures, we have the following observations.

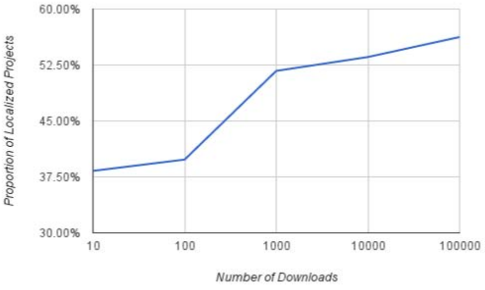

Figure 1. Localization Popularity among Projects with Different Downloads

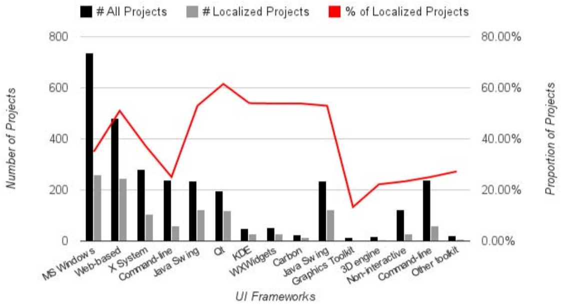

Figure 2. Localization Popularity among Projects with Different UIs

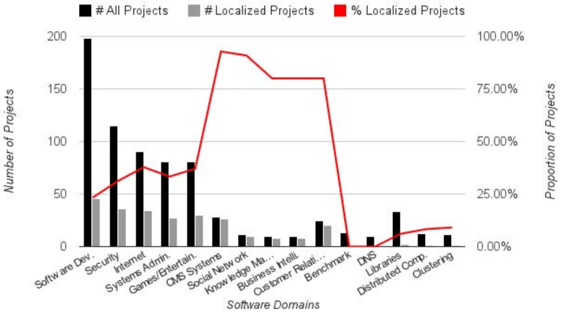

Figure 3. Localization Popularity among Software Projects in Different Domains

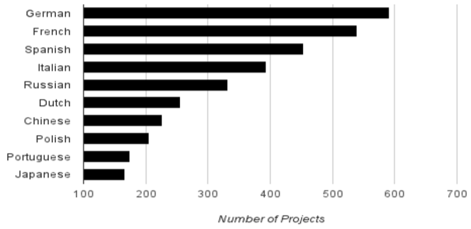

Figure 4. Top Locales in Software Localization

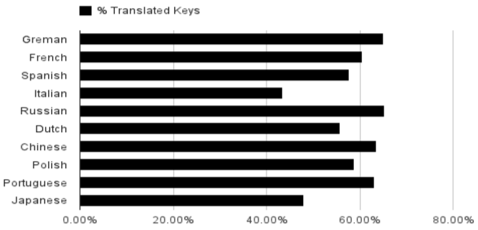

Figure 5. Localization Quality for Different Locales

First of all, among the software projects with higher weekly downloads, the proportion of localized software projects is higher.

Second, among the most popular UI frameworks (left 5 labels in Figure 2), projects that are web-based or Java-Swing-based are more likely to be localized (with a localization proportion higher than 50 %), while command-line-based projects are less likely to be localized. The results are reasonable, because web-based applications are typically accessible from users all over the world, and Java-Swing-based applications are light-weight and easy to be distributed.

Fourth, among the most popular 5 software domains, Internet applications, System administration applications, and game applications are relatively more likely to be localized, perhaps because they are designed for a larger variety of end users.

Finally, we also studied the popularity of locales, and the top 10 most popular locales are presented in Figure 4. In the rest of our study, we will focus on these locales.

3.2 Software Localization Quality

We studied the quality of software of localization by checking the proportion of key-value pairs that are actually translated in the localization. Our study reveals that software developers often perform partial localizations (maybe due to the lack of time / effort or proper translator). In such cases, an untranslated string either appears in their default language version in the locale property file, or the pairs of keys and translated strings are simply omitted from the local property file (the software will refer to the default locale property file when it cannot find a key).

We present the results of our study in Figure 5. Figure 5 shows the average percentage of translated keys for a certain locale among all projects in the top subject set.

3.3 Software Localization Process

To further understand the software localization process in practice, we studied the version history of resource files to calculate the time span of the software localization process. In Figure 6, we present the break down of all local resource files on the time difference between its commit, and the commit of the default resource file.

From the figure, we can observe that, about 32% of all local resource files were committed more than 6 months after the default resource file was committed. This indicates that the software localization process often takes a long time. Actually, it is unlikely that the developers were working on the localization during this period of time. A more possible reason is that, the developers failed to find a proper translator / contributer to perform the localization for certain locales. Also, we observe that there are 4% of resource files were committed before their corresponding default resource files were committed. The reason is that, some modules of some projects were initially written with a locale different than English, and the developers later added an English resource file, and reset the English resource file as default.

4 Related Works

The most related work to our study is a recent study [2] on the Android Framework. This previous work studies the version history of the Android project, and reported some findings on the quality and code commit frequency on software localization. Compared to their study, we used a much larger set of software

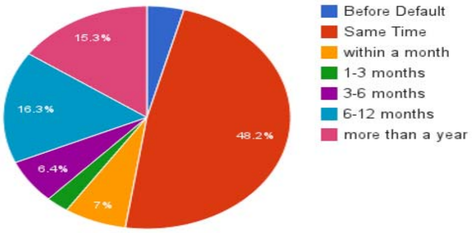

Figure 6. Break Down of Resource Files based on the Commit Delay

projects, and provide more detailed results on the popularity and quality of software localization.

In academia, researchers have summarized a number of important research issues in software internationalization and localization [6, 3], including architectural practice, extraction of region-specific code elements, management of strings in resource files, translation, and cultural adaptation. These issues have been studied by various research efforts [4, 7, 9, 8, 10].

5 Conclusions

In this paper, we present a study about the software localization status on open source software projects from SourceForge. In our study, we find that, the popularity of software localization is higher in software projects with more downloads, and varies for different UI frameworks / domains. Furthermore, we find that the quality of software localization in open source projects is relatively low and the localization process often span a long period of time.

Our findings mainly call for two future research directions. First of all, the current study shows low quality as well as high time cost (or the difficulty to find proper contributor) for software localization. So automatic support fo software localization is highly desired. Second, it is surprising that some top software projects have a low quality in software localization. So it would be interesting to perform a more in-depth research on the impact of software localization on the success of a software project on the global market.

Acknowledgment

The authors from University of Texas at San Antonio are supported in part by NSF grant CCF-1464425, and DHS grant DHS-14-ST-062-001.

References

[1] Sourceforge, http://sourceforge.net/.

[2] L. Arjona Reina and G. Robles. Mining for localization in android. In *Proceedings of Internation Working Conference on Mining Software Repositories*, pages 136–139, 2012.

[3] V. Dagiene and R. Laucius. Internationalization of open source software: framework and some issues. In *2nd International Conference on Information Technology: Research and Education*, pages 204–207, 2004.

[4] A. Danko. Formalization of functional aspects in business software globalization. In *14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW)*, pages 107–116, 2010.

[5] B. Esselink. *A Practical Guide to Software Localization: For Translators, Engineers and Project Managers.* John Benjamins Publishing Co, 2000.

[6] J. H. Hogan, C. Ho-Stuart, and B. Pham. Current issues in software internationalisation. In *Proc. Australian Computer Science Conference*, pages 1–10, 2003.

[7] X. Wang, L. Zhang, T. Xie, H. Mei, and J. Sun. Locating need-to-translate constant strings for software internationalization. In *International Conference on Software Engineering*, pages 353–363, 2009.

[8] X. Wang, L. Zhang, T. Xie, H. Mei, and J. Sun. Transtrl: An automatic need-to-translate string locator for software internationalization. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 555–558, 2009.

[9] X. Wang, L. Zhang, T. Xie, H. Mei, and J. Sun. Locating need-to-translate constant strings in web applications. In *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 87–96, 2010.

[10] X. Xia, D. Lo, F. Zhu, X. Wang, and B. Zhou. Software internationalization and localization: An industrial experience. In *18th International Conference on Engineering of Complex Computer Systems*, pages 222–231, 2013.