INFORMATICA, 2006, Vol. 17, No. 1, 3–12
© 2006 *Institute of Mathematics and Informatics, Vilnius*

Quantitative Evaluation of the Process of Open Source Software Localization

Valentina DAGIENĖ, Gintautas GRIGAS
Institute of Mathematics and Informatics
Akademijos 4, 08663 Vilnius, Lithuania
e-mail: dagiene@ktl.mii.lt, grigas@ktl.mii.lt

Received: June 2005

Abstract. Localization is a complex process based on translation and adaptation of software features. Usually localization progress is identified with the number of translated resource strings. The paper investigates the dependency of number of translated strings to amount of human resources used. It is shown that the number of translated strings increases much slower at the end of the work than at beginning. The last strings are especially difficult to translate. Quantitative evaluation of dependency between number of strings in progress and human resources is presented.

Key words: open source, localization, resource string, translation, process of evaluation.

1. Introduction

Countries and different speaking peoples are using increasingly diverse software. One of the main problems in software adaptation for local users is localization. Localization can not be interpreted as action of translation. Despite the fact that localization of software is estimated by translated resource strings, translation makes up only a small part of software localization (Esselink, 2000; O'Sullivan, 2001).

Software localization may be divided into several levels: 1) adaptation of software to locale of the language to which it is being localized, 2) translation of the dialogs, 3) translation of software's help and other documentation, 4) exhaustive final testing of the localization. In this paper we consider the translation of the dialogs, in particular, the evaluation of process of localization.

Usually localization begins with translation of the dialogs. The labour expenditure of translators' consumed or remaining work is identified with the number of the translated resource strings and is expressed using absolute units (turned into number of the translated strings) or relative units (turned into percentage of translated strings). However, those who are concerned with localization clearly know that at the end of the work number of translated strings increases much slower than at the start (Esselink, 2000). At the beginning translator translates what he or she knows very well. Thus, further work involves more and more difficult texts to translate. The last strings are especially difficult to translate since translator has to solve previously collected obscurities, to find and investigate more unusual and difficult situations the application work (Grigas, 2003).

The number of translated strings is easily measured parameter. However, in order to use it for the evaluation of the amount of the performed work and for the realistic planning of completion of the translation, the dependence between the number of translated strings and the expenditure of the work, which is not linear, should be established. The obvious way to determine such dependence is to register the translation progress data during the whole process of localization. Such data from the sufficient number of the projects may be used to obtain statistically reasoned result. It is, however, long and much effort requiring work.

To get dependence here we present another method (indirect one), which uses the one or more "snapshots" of the statistics of different software translations into many different languages instead of progress data from long lasted time period from one or more localization projects. In order to ensure the adequacy of such change, certain conditions concerning the stability of the whole localization process should be fulfilled, and sufficient number of data sources (localization projects) fulfilling such conditions must be available. Fortunately, the current state of localization process of a number of open source localization projects is such that they fulfill those conditions, and we use their data for our analysis.

2. Theoretical Model of Localization Process

Let's say we have a large localization project that involves the translation of n software components. Denote by:

- the number of untranslated components – u (untranslated),
- the number of translated components – t (translated),
- the number of components which are being translated – p (in process).

Then we have $n = u + t + p$.

The work is concerned to be started when the translation of at least one component is initiated (Fig. 1, the 1st point of u). Let's say that all processes pass equally and the resources of translators are limited. The number of components being translated increases until the process saturates (Fig. 1, the 2nd point of t), i.e., all translators are working and undertake the new component just when the translation of the previous one is completed.

After some time the moment comes when the translation of the first component is completed (Fig. 1, the 1st point of t). Since then the number of the translated components increases until all the components are translated (Fig. 1, the 2nd point of t).

The last graphics (p) presents the number of the components under the translation (i.e., in process). It remains stable for some time since the translation of the new components is being undertaken when the translation of the previous ones is completed. However, after certain time the moment comes (Fig. 1, the 6th point of p), when there are no new components left, the translation of all components is being completed and finally the process comes to the end (the 7th point).

The interval between the 2nd and the 6th points contains large and stabile number of the components under translation ($t \gg 0$), at least one component whose translation

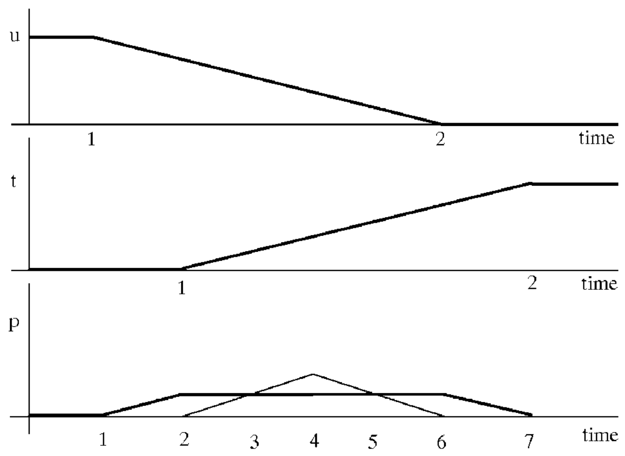

Fig. 1. Dependence of the components under translation (u), the translated components (t), and the components those translations has been initiated on the time.

is not started yet ($u > 0$), and at least one translated component. Therefore the translation process described in this interval may be considered as run up and stable. However this could be hold true just in the ideal case, if all the components were of the same size and complexity. It is not so in practice. When the components are unequal the situation is much more complex: for example, if there is just one, but large and complicated component left not translated, the beginning of its translation will be possible just when translator with satisfactory high qualifications finishes his/her previous work and during this period other translators will remain out of work. On the other hand, it is possible when just at the very beginning of the translation project, even the same day, translation of some very simple component (for example, which has just one source string) may be completed, while the translation of all other components will still require a lot of time to spend. Hereby another extremely strict rule of interval limitation comes: it should be taken just such part of the process, in which any of the numbers u and t is not less than p ($u \geqslant p$ & $t \geqslant p$, or $\min(u, t) \geqslant p$). It concerns the interval between the 3rd and the 5th points. We will name it by strict stability condition.

The circumstance that the translation process is stable in this interval may be judged by the fact that the number of options of the components to translate is not less than whose that are already being translated; thus each translator after the completion of his/her component may choose another untranslated component appropriate for his/her skills. Analogical rules could be applied for the translated components when the number of them is not less than the number of components that are being translated.

It is not difficult to ascertain that the stable interval depends on the project size. The bigger number of components that are being translated the longer time it takes to perform the translation and the interval is increasing. Meanwhile dealing with smaller applications the interval may be equal to zero. Therefore, if the stricter limitation of the interval is applied, always remains risk of not finding program applications proper to analysis. In such case the conditions of the limitations should be lightened ensuring at the same time that they provide proper stability of the process. It is not easy to do though. Nevertheless, as we will see further, it is possible to find applications that meet such requirements.

To evaluate the stability of the software translation process we will apply ratio r :

$$r = \frac{\min(u,t)}{p}.$$

The previously mentioned stability condition of the translation process is met when $r \geqslant 1$.

3. Data Source

To perform the analysis we have chosen versions and environments of operating system Linux for the following reasons: 1) production and localization of open source software is voluntary and therefore stochastic; 2) the process involves many people; 3) the data on the localization progress are announced regularly; 4) the work amount is large and takes time to perform; 4) the software involves large number of components. These facts let us set the assumption of the reliability of the statistical results.

The data concerning versions and environments of the operating system Linux are presented (Table 1).

The data is collected from the indicated sources on the 16th of April, 2005.

The strict stability condition ($r \geqslant 1$) was met by 15 Linux Debian localizations (Table 2).

The summary numbers of separate localization components slightly differ. This is because of the large size of the software and the different sets of the components to be localized estimated by localizers into different languages. Since these numbers have impact just on the localization selection, while the results of the analysis themselves are formed just from the completeness level (see further chapter) of the components under translation (p), such dispersal won't have any significant impact on the results.

Table 1

Localizable resources of Linux system

System	Number of strings	Number of strings (% over max)	Number of components	Number of localized languages	Number of analyzed languages	Source
Linux Debian	323000	100	991	191	15	(Central, 2005)
Linux Mandriva (previously – Linux Mandrake)	12782	4	37	121	0	(Mandriva, 2005)
Linux KDE	113304	35	856	80	4	(KDE, 2005)
Linux Gnome	32684	10	73	92	0	(The Gnome, 2005)

Table 2

Analyzed localizations of operating system Debian

Language code	Language	Untranslated, u	In process, p	Translated, t	Ratio, r	Components in total $u+p+t$
fr	French	335	260	334	1.28	929
de	German	313	265	398	1.18	976
es	Spanish	446	218	257	1.18	921
it	Italian	521	190	196	1.03	907
nl	Dutch	525	160	215	1.34	900
sv	Swedish	560	132	213	1.61	905
pt-BR	Portuguese (Brasil)	549	162	162	1.00	874
pl	Polish	565	164	169	1.03	898
cs	Czech	601	120	165	1.38	886
da	Danish	605	115	160	1.39	880
hu	Hungarian	632	113	121	1.07	895
ca	Catalan	642	98	148	1.51	888
pt	Portuguese	657	81	122	1.51	860
zh-CN	Chinese	643	117	142	1.21	902
sk	Slovak	658	89	142	1.60	889

Table 3

Analyzed localizations of operating system KDE

Language code	Language	Untranslated, u	In process, p	Translated, t	Ratio, r	Components in total $u+p+t$
bg	Bulgarian	205	78	576	2.6	859
nn	Norvegian Nynorsk	276	223	344	1.24	843
is	Icelandish	261	168	414	1.55	843
mk	Macedonian	249	107	198	1.85	654

The data concerning KDE localizations are given in Table 3.

Only four KDE localizations met the strict stability condition. This is because the system is smaller and its recourses contain three times less localizable strings than the operating system Linux Debian (the number of components is approximately the same, but nevertheless they are smaller). Therefore the localization process is much faster.

At the beginning ratio r is small because of the small number of completed components; when the number of translated components reaches the number of ones under translation, the number of components that are still not undertaken to translate remains too small. Besides we would like to point out the fact that languages that meet the strict stability condition are used by minor number of population and probably therefore the translations into them are performed slower: thus the process lengthens and hereby the assumptions to meet the stability condition emerge.

The last two systems (Mandriva and Gnome) do not contain any language that would meet the strict stability condition. It is natural, since the number of their localizable components is even lower.

4. Dependence Between the Localized Number of Strings and the Total Number of all Strings in Current Localization State

The dependence may be illustrated by the example of localization of Debian system to French (Fig. 2). We chose that language since its localization process during the experiment has reach most closely to the middle, i.e., the number of strings of untranslated components and the number of strings of translated components were almost equal (335 and 334, see Table 2).

The horizontal axis represents the declared number of localized strings, which is expressed in percentage, while vertical one shows a whole number of strings in the components present in the current state of localization. For example, when components contain up to 60% of translated strings, the total number of strings is approx. 7%; when components contain up to 99% of translated strings, the total number of strings is approx. 84%. If the localization process would be performed gradually the diagram would be linear. The curved form of the diagram indicates that at the beginning the number of translated strings increases very rapidly and becomes slow approaching the end.

Why the process of translation does become slower when it approaches the end? What is preventing the translation of the software to be completely performed?

The main reason is that translator at the beginning translates what he knows best. Latter he approaches the texts that are harder and harder to translate. It can be stated

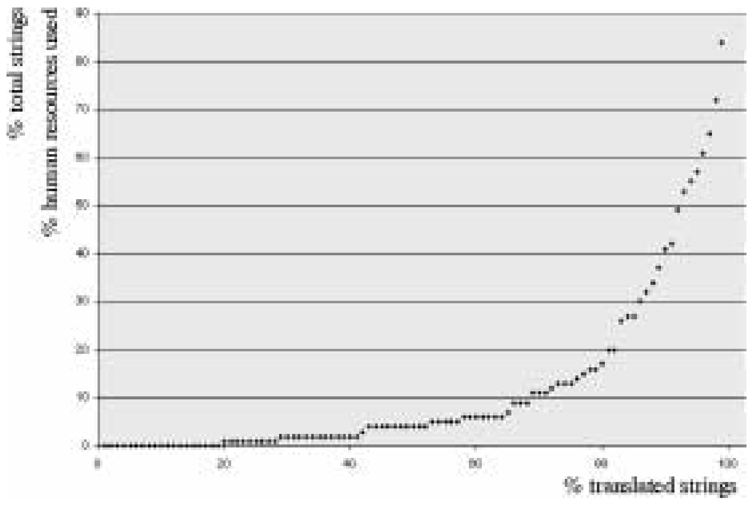

Fig. 2. Diagram of localization of Debian system to French.

that the last percent (the last strings) is the most difficult to complete. Therefore when the translation of 90% of the strings is completed it doesn't mean that 90% of work is already performed. As the diagram shows, it still remains approximately half of the strings in untranslated components of the software, i.e., it is performed approximately just half of the job.

The distribution of the software whose translation is not completed yet brings the serious difficulties for its user. If the meaning of the string was not clear enough to translator it will barely be clear to software user whose qualifications usually can't compare to the qualifications of the translator. Therefore only the complete translation has a real value. The given diagram shows exactly the dependence of the percentage quantity of already experienced expenditures (or expenditures that will be experienced in future) on the percentage of the translated strings.

The expenditures of calendar time on the increment of strings resource declared in statistics in comparison with the previous publication are equal to time period spend between the current publication and the previous one. The concordance is obvious.

However, the rate of work expenditure is not less important. Thus we presume that work expenditures are proportional to calendar time expenditures that are required to perform one unity of localization, i.e., to complete the translation of recourse string in this case. This presumption is true when work is being performed with homogeneous intensity. We know from the experience that work of particular person and translation of particular software very seldom happen to go evenly.

From the experience we also know that when the work approaches its end it do not come to a stand, although the amount of the results (the number of localized strings) decrease: remaining strings are being discussed and the situations of application performance associated with them are being analyzed. In other words: the intensity remains. Besides this it is worth to mention the psychological factor here: since the results are publicly announced approaching to the end translator wishes to see the 100% guide-mark in the summary of the results as soon as possible. It is somewhat similar to sport competition when sportsman is approaching to finish. It depends on each performer's individual characteristics. Therefore in case of different application or different localizer may appear certain deviations to one or another direction. However, if the sufficient number of analyzed data is provided the deviations should disappear. Therefore, we will use the data of both systems (Debian and KDE) and a number of languages for our analysis in order to examine the previously mentioned presumption or – to be more specific – to get the correlation between the calendar time and working expenditures as close as possible.

5. Summarized Results

We will provide more results of analysis to compare (Fig. 3 and Fig. 4)

We see that both dependences are similar. The points of the diagram on KDE system are more scattered. It is, however, natural since the system is localized to much lower number of languages that meet the strict stability condition; besides the number of localizable strings is approximately 3 times lower – the extend of works during the KDE localization to four languages is just 1.4 larger than Debian localization to one language.

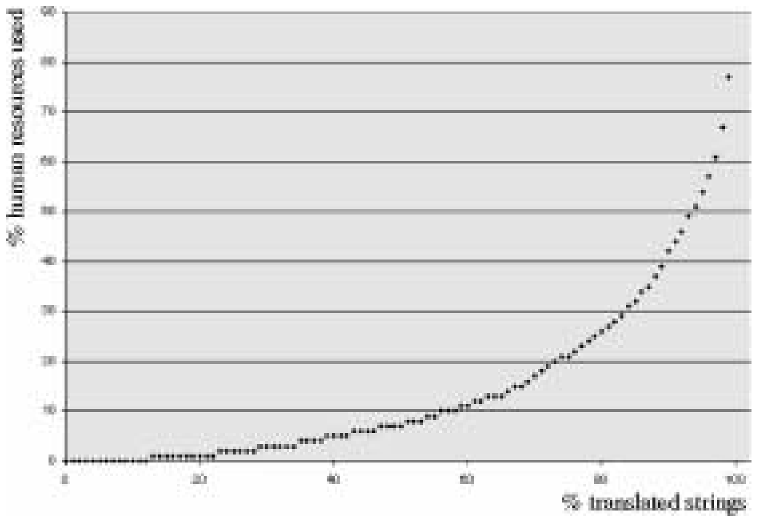

Fig. 3. Diagram of localization of Debian system to all 15 languages presented in Table 1.

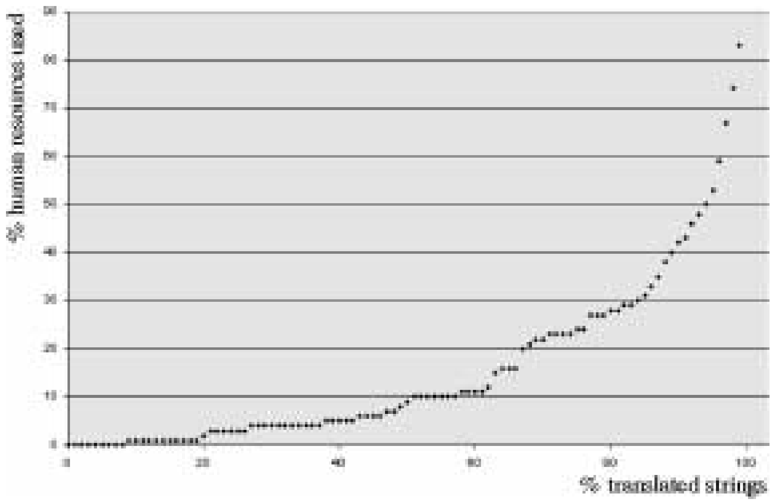

Fig. 4. Diagram of localizations of KDE system to all 4 languages presented in Table 2.

Due to the similarity of the diagrams we may conclude that the dependence between the declared number of translated strings and total number of all strings in the current localization state is similar in all analyzed cases. Therefore we'll present the common result of all languages mentioned in Table 1 and Table 2 as the generalization (Fig. 5).

The generalized results may be used as a tool for average instant calculation of work done according to number of translated strings. For convenience of use they are presented in table format (Table 4).

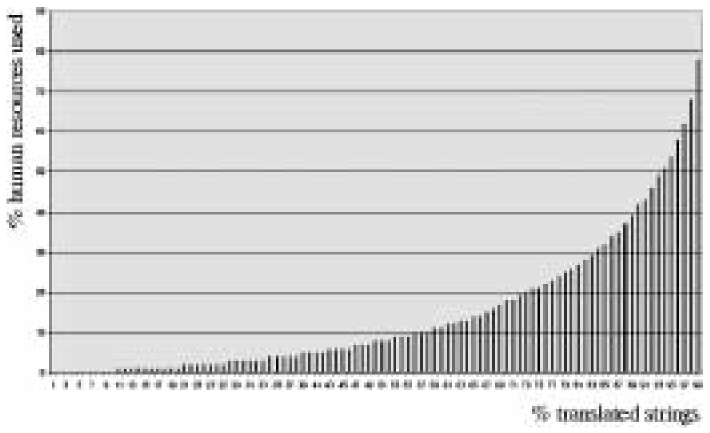

Fig. 5. The generalized diagram on the localization of Debian system to 15 languages and KDE system to 4 languages.

Table 4

Quantity of translated strings (row and column captions) vs translation work done (%, rounded to integers)

	0	1	2	3	4	5	6	7	8	9
00	0	0	0	0	0	0	0	0	0	0
10	0	1	1	1	1	1	1	1	1	1
20	1	2	2	2	2	2	2	2	3	3
30	3	3	3	3	4	4	4	4	4	5
40	5	5	5	6	6	6	6	7	7	7
50	8	8	8	9	9	9	10	10	10	11
60	11	12	12	13	13	14	14	15	16	17
70	18	18	19	20	21	21	22	23	24	25
80	26	27	28	29	31	32	34	35	37	39
90	42	43	46	49	51	54	58	62	68	78

Conclusions

1. The method to determine the dependence between the translated resource strings of software which is being localized and the labour expenditures was presented; it applies the “snapshot” of the statistics on the translations of different software into a number of languages.
2. It was shown that long-term software translation may be switched with the “snapshot” of many applications that are in different state of completeness. in order to ensure the adequacy of such change certain stability conditions during the whole localization project should be met.
3. The results of experiment performed using described method were summarized and presented in the table, according to which it is possible to assess the quantity of already consumed and still required expenditure from the declared number of localized strings (both expressed in percentage).

References

Esselink, B. (2000). *A Practical Guide to Software Localization*. John Benjamins Pub., Amsterdam & Philadelphia.

Central Debian Translation Statistics (2005).
`http://www.debian.org/international/l10n/`

Grigas, G. (2003) Programinės įrangos vertimo į lietuvių kalbą dabartinė situacija, problemos ir jų sprendimai (in Lithuanian, Translation of software into Lithuanian: situation, problems and their solution). *Informacijos mokslai*, **26**, 251–256.

KDE Internationalization Statistics.
`http://i18n.kde.org/stats/`

Mandriva Linux Localisation (2005).
`http://www.mandrakelinux.com/l10n/teams.php3`

O'Sullivan, P. (2001). *A Paradigm for Creating Multilingual Interfaces*. PhD Thesis. University of Limerick.

The Gnome Translation Project (2005).
`http://developer.gnome.org/projects/gtp/`

V. Dagienė graduated from Vilnius University, Lithuania, in 1978, and received the PhD in physical science, informatics from the Vytautas Magnus University, Lithuania, in 1993. In 2005, she passed the Habilius Procedure in social sciences (educology) from the Vytautas Magnus University. She has been working as a head of Informatics Methodology Department at the Institute of Mathematics and Informatics and also as a professor at Vilnius University. V. Dagienė is an author of more than 50 textbooks in the field of informatics and programming for high schools. Her main research focus is teaching informatics and information technologies in high school. She is also engaged in programming languages, algorithms, learning Logo, problem solving, distance education, localization of software. She takes care of informatics teaching in primary, basic, and secondary schools, she has initiated and supervised the preparation of curricula of informatics and information technologies. Besides she works in various expert groups and work groups of the Ministry of Education and Science in Lithuania and abroad.

G. Grigas received the PhD degree from the Kaunas Polytechnic Institute (Kaunas, Lithuania). His research interest include abstract date types, programming methodology and teaching, software localization. Has puslished 35 books and about 200 scientific papers.

Atvirųjų programų lokalizavimo eigos kiekybinis įvertinimas

Valentina DAGIENĖ, Gintautas GRIGAS

Įprasta lokalizavimo eigą vertinti išverstų išteklių eilučių skaičiumi. Tačiau lokalizavimo praktika rodo, kad darbo sąnaudos paskutinių eilučių vertimui yra žymiai didesnės, negu pirmųjų. Tai yra dėl to, kad pabaigai lieka sunkiausiai verčiamos eilutės, kurių prasmę reikia analizuoti, diskutuoti. Šiame straipsnyje bandoma kiekybiškai įvertinti sanaudų kiekį, išreikštą procentais, priklausomai nuo išverstų eilučių skaičiaus procento. Naudojamasi didelių programų – *Linux* operacinių sistemų *Debian* ir *KDE* vertimų į daugelį kalbų eigos statistika. Analizei imamos tos kalbos, vertimų į kurias procesas yra įsibėgėjęs ir stabilus.