Volume 4, No. 8, August 2013
Journal of Global Research in Computer Science
ISSN-2229-371X
REVIEW ARTICLE
Available Online at www.jgrcs.info

A SURVEY OF SOFTWARE LOCALIZATION WORK

Manisha Bhatia*[1], Varsha Tomar[2] and Aparna Sharma[3]

*[1]Computer Science, Banasthali University, Jaipur, Rajasthan, India
manisha_bhatia23@yahoo.com[1]
[2]Information Technology, Banasthali University, Jaipur, Rajasthan, India
varsha1616tomar@gmail.com[2]
[3]Computer Science, Banasthali University, Jaipur, Rajasthan, India
aparna.home@gmail.com[3]

Abstract: Localization concerns the translation of digital content and software, and their appropriate presentation to end users in different locales. Localization is important because having software, a website or other content in several languages, and meeting several sets of cultural expectations is an important international marketing advantage. This paper presents the difference between software localization, globalization and internationalization and compares the traditional document translation with respect to software localization. The paper further includes the survey of various localization software and software localization services, methods and tools available worldwide. As part of conclusion the steps followed for localizing a software project is included. We intend this paper to be useful to researchers and practitioners interested in software localization.

***Keywords*:** Localization (L10n), Internationalization (I18n), Globalization (G11n), Document Translation, Indic Computing.

INTRODUCTION

Localization is the process of adapting a product or service to a particular language, culture, and desired local "look-and-feel (Rouse, 2005). Software Localization is more than the translation of a product's User Interface. Companies localize software in order to overcome cultural barriers for their products to reach a much larger target audience [1].

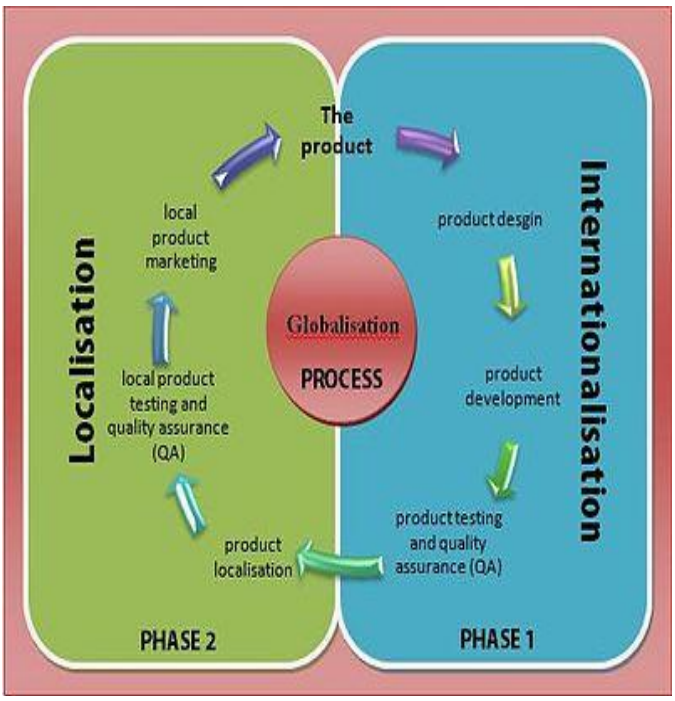


Fig.1: The internationalization and localization process (based on a chart from the LISA website)

In computing, internationalization and localization are means of adapting computer software to different languages, regional differences and technical requirements of a target market. Internationalization (i18n) is the process of designing a software application so that it can be adapted to various languages and regions without engineering changes. Localization (L10n) is the process of adapting internationalized software for a specific region or language by adding locale-specific components and translating text [2].

Some companies, like IBM and Sun Microsystems, use the term "globalization" for the combination of internationalization and localization. For example, by creating technical illustrations for manuals in which the text can easily be changed to another language and allowing some expansion room for this purpose. This enabling process is termed internationalization. An internationalized product or service is therefore easier to localize. The process of first enabling a product to be localized and then localizing it for different national audiences is sometimes known as globalization [3].

DIFFERENCE BETWEEN SOFTWARE GLOBALIZATION, INTERNATIONALIZATION AND LOCALIZATION

Attributes	*Globalization*	*Internationalization*	*Localization*

Abbreviations	G11n (Globalization is sometimes written as g11n, where 11 is the number of letters between l and n.)	I18n (Internationalization is sometimes written as i18n, where 18 is the number of letters between l and n.)	L10n (Localization is sometimes written as l10n, where 10 is the number of letters between l and n.)
Include	Globalization= n (Internationalization)	Internationalization= n (localization)	localization
Definition	Refers to a broad range of processes necessary to prepare and launch products and company activities internationally. In other words *Globalization (G11N)* is the process of developing a program whose features and code design are not solely based on a single language or locale.	Developing content that is easily localizable doesn't include local/regional references. In other words *internationalization (I18N)* is the process of developing code in such a way that it is completely independent of any culture-specific information.	The process of adapting a product or service to a particular language and culture. [4] *Localization (L10N)* is the process of adapting an application, *ideally one which has already been properly internationalized*, to run in a particular locale, country, or linguistic market.
Description	Globalization application involves two independent steps: *Internationalization* and *localization*. Globalization aims to address all of the unique challenges an enterprise faces as it expands to the global market	Fully internationalized applications can be localized to a particular country or region without any changes in the program code.	Localization includes displaying all text in the native language, and using native conventions for formatting and sorting. It takes into account details as national regulations and holidays, cultural sensitivities and gender roles, and local currency [5].

Table 1: Comparison between Globalization, Internationalization, and Localization

Localization primarily includes:

1. Translating text content, software source code, web sites, or database content; machine translation may be used in early stages.
2. Adjusting graphic and visual elements and examples to make them culturally appropriate.
3. Post-production quality control of content, systems and the integrated product.

Internationalization may include:

1. Creating illustrations for documents in which the text can easily be changed to another language and allowing expansion room for this purpose.
2. Allowing space in user interfaces (for example, hardware labels, help pages, and online menus) for translation into languages that require more space.
3. Creating print or web site graphic images so that their text labels can be translated inexpensively.
4. Leaving enough space in a brochure to drop in different length languages.
5. Separating the language elements from the graphic elements, or abstracting content from markup in a web application and software.
6. Using written examples that have global meaning.
7. Insuring that the tools and product can support international character sets.
8. For software, ensuring data space so that messages can be translated from languages with single-byte character codes (such as English) into languages requiring multiple-byte character codes (such as Japanese Kanji).

Globalization describes a comprehensive process that incorporates, augments, and extends:

1. Research on and identification of global markets.
2. Market validation and selection.
3. Identification and formalization of global business requirements.
4. Language translation and cultural integration (a.k.a. Internationalization and Localization).
5. Identification of technology standards and solutions (format and structure).
6. Identification of cross-market affinities (to enable marketing and technology asset reuse).
7. Alignment with and support for Internationalization (i18n) and Localization (l10n) processes [6].

HOW DOES SOFTWARE LOCALIZATION DIFFER FROM TRADITIONAL DOCUMENT TRANSLATION?

Software localization is the translation and adaptation of a software or web product, including the software itself and all related product documentation. Traditional translation is typically an activity performed after the source document has been finalized. Software localization projects, on the other hand, often run in parallel with the development of the source product to enable simultaneous shipment of all language versions. For example, the translation of software strings may often start while the software product is still in the beta phase.

Fig. 2: Interrelation between globalization, localization and translation

Translation is only one of the activities in a localization project – there are other tasks involved such as project

management, software engineering, testing and desktop publishing [7].

Comparison between translation and localization, may be done using the following categories–

1. Activities
2. Complexity
3. Adaptation level
4. Technology used

Activities

Traditionally, translation is only one of the activities in projects where material is transferred from one language into another. Other activities that can be distinguished in traditional translation projects include terminology research, editing, proofreading, and page layout. In localization, many more activities have been added to this list. Examples of activities in localization which are not necessarily part of traditional translation are multilingual project management, software and online help engineering and testing, conversion of translated documentation to other formats, translation memory alignment and management, multilingual product support, and translation strategy consultancy.

Complexity

Compared to traditional translation projects, managing software or web localization projects can be very complex. First of all, localization projects contain a large number of components, such as software, sample files, online help, online and printed documentation, collateral materials such as product boxes and disk labels, and multimedia demos. In most cases translation starts before the source material is final, so in most localization projects the source files are updated several times during translation. Because volumes are usually very large and all components contain dependencies, managing localization projects is a tricky task. Large volumes and tight deadlines require teams of translators, who all need to be reviewed carefully to maintain consistency. For example, when translator A translates the software and translator B the online help files, all references to the running software translated by translator B in the online help must exactly match the software translations that translator A has chosen. Also planning localization projects is a complicated task, because many tasks depend on completion of previous tasks. For example, screen captures of localized software to be included in the online help or documentation cannot be created until the localized software has been engineered and tested.

Adaptation Level

Localization is derived from the word locale, which is defined in the Collins Cobuild Dictionary as "a small area, for example the place where something happens or where the action of a book or film is set". In a software localization context, a locale is a region which is defined by a number of characteristics, such as language, culture, and all types of regional standards such as character set, currency, default page sizes, address formats, custom calendars, date/time formats, and other things that give many American software developers headaches. For example, a language is French; a locale is the region in Canada where French is spoken. In software localization projects, all local characteristics need to be implemented in the final product. A truly localized product shouldn't only be in the target language but should also use default settings for the target locale. So, a product sold in Germany should automatically use A4 as default page size, support input and output of accented characters, and display currency amounts using Marks and Euros instead of dollars.

Technology Used

In software localization, the integration of translation technology is ahead of traditional translation. Because of the nature of software products and web sites, which are highly repetitive, and updated on a regular basis, smart re- use of existing translations has become a competitive advantage and the use of translation memory a must. Most software products are updated at least once a year, and web sites are often updated on a daily basis. As a result, translation memory tools have been used successfully for many years in the localization industry. Other examples of translation technology that is widely applied in the localization industry are software localization tools for software user interface translations, terminology extraction and management tools, and machine translation. To sum up, localization has never and will never replace translation. It's just a term used to cover all activities related to adapting a software product or web site to be used in a target locale. Translation will always remain one of the most important activities in any localization project. [8]

RELATED WORK ON SOFTWARE LOCALIZATION

Alchemy software development limited (Ireland) developed a software suite for internationalization and localization that is Alchemy CATALYST. Alchemy CATALYST is a software localization tool and was one of the first tools that contained integrated translation memory technology [9]. Alchemy CATALYST is an integrated translation environment with powerful project management features. CATALYST seamlessly integrates editors and tools for Translators, Localization Engineers, Quality Assurance (QA) Specialists, Project Managers and Software Developers. The software supports all Windows™ program (DLL, EXE, INI), resource files (RC) and help files (RTF, CNT, CPP, HPP), as well as XML/XHTML based files and Internet applications [10]. The latest version of Alchemy CATALYST version 9.0 SP2 was released on the June 6th 2012[11].

The company *PASS Engineering GmbH (Germany)* develops their own localization tool that is Passolo. Passolo became available as an independent localization tool in 1998. In June, 2007, SDL acquired PASS Engineering [12]. SDL Passolo allows the user to Use the visual localization environment to see which translation is best when no context is available and view the translation of dialog boxes and menus in real-time and adjust them accordingly. SDL Passolo 2011® is the latest version of the award-winning software localization tool. With one visual environment for software localization, translate graphical user interfaces (GUI) more quickly and easily than ever before [13].

Schaudin.com (Germany) [14] is a software designer company developing custom-made applications running on Microsoft Windows that is RC-Win Trans. RC-WinTrans supports the localization of the Graphic User Interface

(GUI) of Microsoft Windows software consisting of dialog boxes, menus, string tables, accelerator tables and bitmaps. RC-WinTrans conveniently translates Windows resource script files (.RC, .DLG, .RC2), Program files (.EXE, .DLL and .OCX) as well as Java .PROPERTIES files, simple C/C++ text definition files (.H), Windows INI files (:INI) and InstallShield string table files (.SHL) [15].

According to chine's company *EPRO* process of localization is:

1. Complete pre-analysis of software projects and solution of any possible problems prior to initializing localization.
2. Translation and adaptation of software user interface and messages.
3. Localization engineering and testing, also at the client's location, if requested.
4. Complete help localization, including compiling, functionality tests and screen shots.
5. Implementation of current commercial and client-specific localization applications, compilers and other tools [16].

According to *MORAVIA* software localization engineering is a critical complement to the software localization and software testing cycles. It covers all the steps crucial for successful localization and testing, including software analysis, build engineering and mastering, bug fixing, scripting and overall localization process automation.

They provide the following software localization engineering services:

Software Analysis

- Product analysis and localizability assessment reporting.
- List of files to be localized; highlight of files that cannot be localized with standard tools, methods; highlight of non-standard file formats.
- Initial project scope (word count) analysis.
- Localization process estimates.

Localization Process Development

- Detailed localization process design based on Software Analysis and client input.
- Preparation of localization project schedule, localization kit and instructions for translators.
- Creation of parsers, processes for specific file types.
- Development of building/mastering process and pseudo-translation build.

Building/Mastering

- Creation of localized versions of product using fully- or semi-automatic process.
- Post-processing of translated files before building/mastering.
- Build validation through automated smoke test.
- ISO image archive files creation.

Bug fixing

- Investigation of all reported bugs from testing stages.
- Repair of all defects introduced during localization process; reporting of any newly-found defect.
- Repair of all truncated, overlapped or misaligned strings; duplicated or non-working hotkeys; miss-translated, un-translated or over-translated strings; etc.
- Report of any broken functionality, hard-coded strings, etc.

Process Automation

- Optimization of localization processes through batch processing, automation of engineering tasks.
- Use of programming languages and scripts to optimize processes, primarily Perl, Python, VBA, VBScript, JavaScript, AppleScript and Windows Script Host.
- Automation of process steps such as mastering, smoke testing, downloading, leveraging.

Moravia has solutions for both possible *software localization approaches:*

- Source-based localization.
- Binary localization [17].

Approaches	***Description***
Source-based Localization	Source-based localization is a universal process that can be used with products designed for any platform. It is based on localization of text files which contain only textual representation of user interface elements, i.e. we not deal with the source code directly. In this model, source file compilation as well as build creation is typically performed by the customer's engineers.
Binary Localization	With binary localization, all activities are focused on the compiled modules or directly on the whole product as such. Our clients only need to provide us with the original product. Tools we use identify binary file structure and can make all application texts, dialog boxes, menus, etc. available for localization, while the application code itself is fully protected. In this approach, far fewer files are involved in the localization process, therefore file preparation and management time is reduced. Extra compilation is not necessary since translation occurs directly on the binary files and translated files can go into the client's mastering and QA process immediately. This approach is limited to the Microsoft Windows platform. At Moravia, we have the engineering experience to design and create full software localization process based on this model.

Table 2: Software Localization Approaches according to Moravia

To address the issues and to add simplicity to the process the software localization experts of *Shakti Enterprise (India)* work in coordination with their client right from the inception of the localization process so as to help them define and drive development and then perform engineering, translation and testing for more than hundreds of foreign languages.

The software localization process comprise of difficult technical task that is to be carried out by variety of specialists including graphic designers, programmers, project managers, testers, engineers and translators. To make the software localization process successful, integration of all these specialists is must. Shakti Enterprise has the expertise and in-depth knowledge about linguistic connotations, taboos, cultural nuances, regulatory requirements and languages that helps us to make the application as functional and useful for foreign users as it for currently for the domestic users. We specialize in providing up to date localization solutions to our clients for a wide variety of software platforms. We consider all the

crucial aspects of software localization and strive to deliver high standard localization.

Shakti Enterprise has expertise in the following areas and this greatly influences the successful accomplishment of any assigned software localization project [18]:

Area of Expertise	*Description*
Linguistic Consistency	They believe in performing the translation part of localization strictly and with utmost care and they employ only in-country linguistic teams and subject area experts for the same. There team of professional translators has the expertise to handle specific language and terminology complexities for all bi-directional (such as Arabic and Hebrew), European and double-byte (such as Chinese, Japanese and Korean) languages. They implement an effective approach towards terminology management to make sure that the localized products convey the message to target users in their own native languages and thus the consistency between software interface and any other user-facing components, like documentation and online help, is 100% guaranteed.
Shakti Localization Testing	They not just only localize the software, but also do testing of the localized products to ensure it is fully functional and linguistically perfect. There team of localization experts conducts the testing and make sure no issues is there in the localized products.
GUI Testing	Shakti Enterprise also has an in-house team to conduct GUI testing. The testing is conducting on the localized products so as to ensure that the user interface of the product is perfectly localized without any defects during localization like truncated text strings, overlapping controls, misaligned controls, duplicated hotkeys, etc.
Internationalization (I18N) Testing	They also conduct Internationalization (I18N) testing to find out whether or not the product has international functional issues before its global release. With this testing they find out whether or not the final product is ideal to perform under diverse regional computer settings and languages. This test help us to find out the competence of the product to display accented characters, to display correct numbering systems thousands, to run under non-English operating systems and decimal separators etc.
Validation	This is the testing to verify the language and context stability of the localized product user interface. They are equipped with specific-language linguistic experts, language-aware testers and universal testers to conduct this type of tests. For more complicated products They combine both types of resources to achieve effective results.
Graphics	To attain superior quality localized versions Shakti Enterprise has the expertise to manage all types of graphic scripts and formats. For example, Screen shooting, a process to take images of the localized user interface, is an area where automation is utmost important and in this area automated process tends to bring in considerable cost efficiencies.
Quality Assurance	They believe in providing quality assurance in each stage of localization process which is crucial to ensure quality releases. Shakti Enterprise is highly dedicated towards quality assurance and make sure that detailed quality assurance is provided in each stage of the localization process.

Table 3: Shakti Enterprise has expertise in the above areas

SOFTWARE LOCALIZATION SERVICES

Localization is the process of adapting products and services to a new location. This includes cultural, linguistic, legal and technical considerations and requires an in-depth knowledge of the local market. According to Integrated Language Solutions (India) the best possible localization service:

Analysis

They analyze the local market to identify any linguistic, cultural, environmental and business factors that could affect your product or service. They also analyze our product or service to ascertain which areas require modification and adaptation.

Modification

They extract any culturally sensitive or linguistically incompatible material from our documentation, translate text using an expert translator with in-depth local knowledge, and adapt all necessary aspects to the local market.

Testing and Review

These test the new product or service in the local market to ensure that it meets local requirements and exceeds our expectations [19].

LOCALIZATION METHODS

Localization has been tried by varying methods. However, every method has been developed for certain requirements. Some of the popular localization methods are:

1. Source code localization

The localization process started with localization of source code and it was carried out by the original developer. The steps involved in the localization on process with source code are:

i. *Extracting user interface strings*: In this task all the user interface strings like Menus, Dialogs, Messages, Status bar text; Tooltips etc are extracted from the source files.

ii. *Adapting language specific conventions:* The user interface strings are translated and number and Date & Time formats are applied. Appropriate character set, Code
Pages, Font names are also adapted and applied as per the requirement. String buffers are adjusted to store the translated strings. This also means altering and adjusting of dialogs to accommodate the translated strings. If enough care is not taken at this point truncation of strings would result in incomplete translated strings being displayed.

iii. *Compile and testing:* After adjusting the language specific interfaces, the source code is recompiled and is ready for testing. On successful testing, the product is ready for the market.

This method is still continuing in the open source community. The established software organizations like Microsoft also followed this method for the earlier products.

2. Executable Localization

As the runtime binaries are matured and PC penetration increased, the localization need for various applications have also increased. And also there is a compelling reason to localize the product and release in local market simultaneously. To meet these requirements, the software developed should take care of the international requirement at design and development of English version itself. Thus developed product will be sent to localization team, the localization team in turn extract and translate the user interfaces using some localization tools, normally with visual testing capabilities. The translation and testing goes on hand in hand. This localization process normally takes less time compare to earlier source code localization. The product is distributed with translated resources.

3. Runtime Localization

The recent development in localization is to localize the software during runtime. In this method also the localization was carried out with running process in the memory. But the resources are not embedded into binaries but distributed separately [20].

TREND IN LOCALIZATION

Indic Computing means "computing in Indic" i.e. Indian Scripts and Languages. It involves developing software in Indic Scripts/languages, Input methods, Localization of computer applications, web development, Database Management, OCR, Spell-checkers, Speech to Text and Text to Speech applications etc. in Indian languages.

Most of the Indic scripts nowadays use Unicode for working on Computers and Internet. In Unicode 5.0 following Indian Scripts have been encoded: Bengali script**,** Devanagari**,** Gujarati**,** Gurmukhi**,** Kannada**,** Limbu**,** Malayalam**,** Oriya**,** Sinhala**,** Syloti Nagri**,** Tamil**,** Telugu.

A lot of Indic Computing projects are going on. They involve some government sector companies, some volunteer groups and individual people [21].

PROBLEM RELATED TO LOCALIZATION

There can be many reasons why the localization of a string can cause a bug, be it user interface or functional. In the functional space bugs can be caused by [22]:

Problem	*Description*
Over-localization	The string should not have been translated.
Buffer limitation	The translation of the resource should not be more than a given amount of characters, generally referred to as a string length limitation.
Illegal characters	Certain characters may not be allowed in the string
Dependency	Two resources may have to be translated as one, in effect one resource is dependent on the other, references the other.
Backward compatibility	This is a special case of dependency, basically where changing a string from one version to another could cause a loss of backward compatibility.
Uniqueness	The string belongs to a group of strings that all have to have unique names (translations), for example, a list of commands.
Placeholder over-localized	Some localizable strings have placeholders in them. If the placeholder gets localized the program cannot drop the information into the placeholder and display it.
Required string decoration	Some strings may have control characters in the beginning or end of the string that should not be localized.

Table 4: Problem related to localization

INTERNATIONALIZATION DO'S AND DON'TS

Here are 11internationalization Do's and Don'ts that all software development companies can consider and implement at work.

1. *Do externalize messages in Message Catalogs, resource files, and configuration files:*

Messages are textual objects that are translatable components. These catalogs or files, such as Java resource bundle message files or Microsoft resource files, are installed in a locale-specific location or named with a locale-specific suffix. This i18n practice will facilitate the localization process, since localizers can work on these resource bundles without the need to modify source code. It will also permit the use of a single source code for all languages, where only the resource bundles will have different language flavors.

2. *Don't internationalize fixed textual objects:*

These are objects that should not be translated, such as comments, commands, and configuration settings. Only externalize the strings needing translation. If these objects appear in resource or configuration files, they should be marked "NOT_FOR_TRANSLATION."

Here are some examples of fixed textual objects not requiring i18n:

a) User names, group names, and passwords
b) System or host names
c) Names of terminals (/dev/tty*), printers, and special devices
d) Shell variables and environment variable names

e) Message queues, semaphores, and shared memory labels
f) UNIX commands and command line options (e.g., ls -l is still ls -l in all locales)
g) Commands such as /usr/bin/dos2unix and /usr/ccs/bin/gprof
h) Some GUI textual components, such as keyboard mnemonics and keyboard accelerators.

3. *Do allow for text expansion in messages (especially for GUI items):*

Here are some Microsoft translations into German:

- bullet –> Aufzählungszeichen
- bundle –> Einzelvorgangsbündel
- Link –> Verknüpfung
- Login –> Anmeldung
- Update –> Aktualisierung
- Undo –> Rückgängig (machen)

Apply the following expansion rules when possible during i18n. When the source text is:

a) 0 – 10 characters: The expansion required is from 101– 200%.
b) 11 – 20 characters: 81 – 100%
c) 21 – 30 characters: 61 – 80%
d) 31 – 50 characters: 41 – 60%
e) 50 – 70 characters: 31 – 40%
f) Over 70 characters: 30%

But keep the string length well below our limit (usually 254 characters) to account for the extra characters needed. Try to place the labels above the controls, not beside them. The expansion of a label can increase the width of the form more than the expected resolution, which will force horizontal scroll bars or cause truncation. This also simplifies localizing applications required into bidirectional languages (languages that are read from different directions [RTL or LTR], such as Arabic and Hebrew).

4. Don't use variables when we can avoid them:

Variables create questions in the translator's mind as to the gender of the term to substitute, making it difficult to correctly translate the sentences that incorporate it. If variables are to be used, offer a list of replacements. Also allow for gender and plurals variations in the translation of the sentences that incorporate the variable.

For example:
if err = 400
errtext = "server"
else
errtext = "connection"
end if
<P> The <%=errtext%> is currently unavailable </P>

While these displays grammatically correct sentences in English, the translation in French will be problematic. In French, the word "server" is masculine, while the word "connection" is feminine. The translator cannot use the correct translation for the article "the" based on the translation of the differing genders of server and connection.

The code should be instead:
if err = 400
<P> The server is currently unavailable </P>
else
<P> The connection is currently unavailable </P>
end if

At the same time and for similar reasons, we don't use composite strings. A composite string is an error message or other text that is dynamically generated from partial sentence segments and presented to the user in full sentence form. Use complete sentences instead, even at the expense of repeating segments. This will ensure the accuracy of the translation, regardless of gender, plurality, conjugation, or sentence structure.

Also, avoid using the same placeholders when using multiple variables in the same string, since the sentence structure does change in different languages. For example, <Total %s, %s of %s> (as in Total 5, 1 of 5) might read "5 of 1, Total 5? in the translated text. Instead, use numbered placeholders (e.g., "Total %1, %2 of %3?).

5. *Don't use IF Conditions or rely on a sort order in your code to evaluate a string value:*

For example, avoid (IF Gender = "Male" THEN). Always depend on enumeration or unique IDs.

6. Do use Unicode functions and methods to support all scripts:

Applications that store and retrieve text data need to accept and display the characters from any given language. Using Unicode encoding solves the problem of unsupported character sets and the display of junk characters.

7. Don't insert hard carriage returns in the middle of sentences:

Translation memory tools key off hard returns and assume that the sentence has ended. Inserting them in the middle of a sentence leads to incomplete sentences in the translation database and corrupts the sentence structure in the target language files. Instead, replace hard returns with soft returns (or better yet, use a break tag of some sort, such as
).Also be aware that sentence structures change in different languages, as well as the length of sentence parts. So, additional breaks may be needed in target languages.

8. Do choose your third-party software provider carefully:

Insist they support Unicode and comply with the above internationalization practices. Often problems are encountered with third-party software, and the fact that you don't have control over their code to fix the problems makes the localization tasks particularly difficult.

9. *Don't use text in icons and bitmaps*:

The translated text may be too long to fit. Also, avoid using symbols with cultural connotations and locale-specific idioms.

10. Do use long dates or month abbreviations instead of numbers when identifying dates:

Month vs. day orders in different parts of the world vary (e.g., mm/dd/yy in the US; dd/mm/yy in Europe).

11. ***Don't alphabetically sort strings in string tables and resource bundles:***

Try to offer as much context as you can with the externalized strings. This will help the translator better adapt the translation to that context. If context is non-existent, run-time QA will take much longer to correct the translation [23].

PHASES OF A PROJECT-

Phases in a localization project are project setup, translator training, terminology definition, user interface translation, test of user interface translation, documentation translation, review of documentation translation, finalization of documentation translation and lessons learned.

Phase 1: Project setup:

The basis for managing any project, including localization, is the project plan. Set up your project plan with milestones, time buffers and constraints.
To estimate the effort, count the words of the user interface and the documentation to be translated; add times for translation preparation and review.

Phase 2: Translator training:

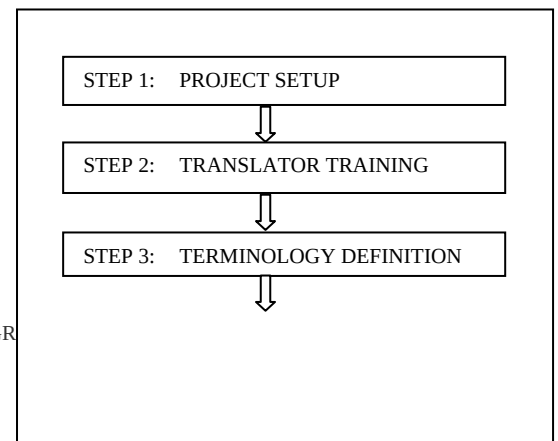
A translator's first contact with our software should not be the translation of a context-less list of words. Instead, train the translators in the software to be localized. Invite the translators to our site to participate in a standard user-training course or provide translators' training. Use this training to establish a relationship on a personal basis. Because the translators might reside in another time zone, country or even continent, the training could be a computer-based training, a simulation or a demonstrator version located on your website.

Phase 3: Terminology definition:

In project documents with yet undefined terminology, the use of a consistent terminology will only be achieved after various cost-intensive iterations. Define the basic terminology to be used for translation of the user interface and the documentation. The basic terminology includes button labels, menus, functions and concepts used in the software.

Phase 4: User interface translation:

After defining the basic terminology, start to localize the software. User interface localization can be performed with support of a localization tool or by translating string resource files. Software localization tools work the way translation tools do, with the segmentation rules applying to user interface strings rather than to sentences. A software localization tool supports the user in the definition of unique menu hotkeys and offers to display the form currently being localized.



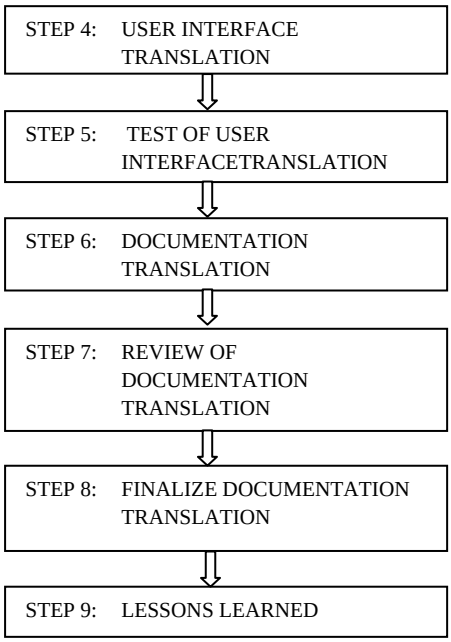


Fig. 3: Phases of a Project

Phase 5: Test of user interface translation:

To achieve our own product requirements, it is essential to thoroughly verify the localized version of our software. The user interface's translator should preferably perform the test of the localized user interface. The tester must turn his or her attention to a wide range of topics. A complete localization is only achieved if all controls (button labels, tooltips, menus, field labels and system messages) are translated.

Phase 6: Documentation translation:

If we localize both the user interface and the documentation, we should finish and verify the user interface translation before starting translation of the documentation. Each modification of the user interface translation requires repetition of testing that translation with all its checks. So, provide enough time for the tests of the user interface translation.

Phase 7: Review of documentation translation:

Our documentation quality and usability requirements also apply to localized versions. If possible, a translator other than the documentation's translator should review the translated manuals. In addition to the linguistic check, the review must ensure the consistency between user interface and documentation. This consistency is important. It supports the user in locating the description of a user interface's form in the manual and vice versa.

Phase 8: Finalize documentation translation:

After completion and review of the documentation translation, we can finalize the documentation. Insert the graphics in the documents, perform the final formatting, update the table of contents and index and compile the online help.

Phase 9: Lessons learned

This phase applies to every project, not just localization projects. Review your lessons learned log [24].

C0NCLUSION

As part of conclusion we conclude the number of points that have to be considered, in order to effectively localize a software product or website.

The localization process includes analysis, assessment, creation and maintenance, translation, adaptation, media localization, testing, quality assurance and project delivery.

Steps	*Description*
Analysis	Analysis of the material received and evaluation of the tools and resources required for localization.
Assessment	Cultural, technical and linguistic assessment.
Creation and maintenance	Creation and maintenance of terminology glossaries.
Translation	Translation to the target language.
Adaptation	Adaptation of the user interface, including resizing of forms and dialogs, as required.
Media Localization	Localization of graphics, scripts or other media containing visible text, symbols, etc.
Testing	Compilation and build of the localized files for testing.
Quality assurance	Linguistic and functional quality assurance.
Project delivery	Deliver the final product to end user.

REFRENCES

[1]http://www.studymode.com/essays/Localization 1659610.html

[2]http://en.wikipedia.org/wiki/Internationalization_and_localization

[3] http://searchcio.techtarget.com/definition/localization

[4] http://blog.adaquest.com/2011/09/23/g11n-i18n-l10n-translation-%E2%80%93-sure-you-know-what-these-mean-or-feeling-gilty-because-you-need-clarification-2/

[5] http://www.compassrosetech.com/services/i18n.html

[6] http://philip.pristine.net/glit/en/

[7] http://www.sdl.com/technology/language-technology/what-is-software-localization.html

[8] http://www.translatortips.net/tranfreearchive/tf10-localization one.html

[9] http://en.wikipedia.org/wiki/Alchemy_Catalyst

[10]http://www.localizationworks.com/DRTOM/Catalyst/Alchem_Catalyst3.htm

[11] http://www.alchemysoftware.com/support/ep.html

[12]http://www.localizationworks.com/DRTOM/Passolo/Passolo.html

[13] http://www.passolo.com/

[14] http://www.schaudin.com/web/Home.aspx

[15]http://www.localizationworks.com/DRTOM/rcwintrans/rcwintrans.html

[16] http://www.epro.com.hk/outs-local.shtml

[17] http://www.moravia.com/en/services/engineering-testing/software-localization-engineering/

[18] http://www.shaktienterprise.com/software-localization-services-india.html

[19] http://www.indiamart.com/integratedlanguages/news.html

[20] N. Anbarasan "Software Localization Process and Issues".

[21] http://en.wikipedia.org/wiki/Indic_computing

[22] O. Martin, "Systematic validation of localization across all language", The International Journal of Localisation, *Vol.7 Issue 1.*

[23] http://www.globalvis.com/internationalization-i18n/

[24] http://www.ccaps.net/newsletter/05-07/art_2en.htm

Manisha Bhatia is an active researcher in the field of localization, currently working as Assistant Professor in Department of Computer Science at Banasthali University (Rajasthan), India. She has done M.Sc (CS) and undergoing PhD from Banasthali University (Rajasthan), India in the field of software localization.

Varsha Tomar is an active researcher in the field of software localization, currently studying in M.Tech (IT) from Banasthali University.

Aparna Sharma is an active researcher in the field of image processing, currently studying in M.Tech (IT) from Banasthali University.