Urdu Localization of Open Source Software

Huda Sarfraz, Sarmad Hussain, Mahwish Bano, Asad Mustafa and Rahila Parveen
Center for Language Engineering, Al-Khawarizmi Institute of Computer Science, University of Engineering and Technology, Lahore, Pakistan
firstname.lastname@kics.edu.pk

Abstract

This paper presents the process used to localize a set of open source software applications for Urdu speakers in Pakistan. The software applications were selected for use by rural area secondary school students and included OpenOffice.org (an office suite), SeaMonkey, (an Internet suite), and Psi (an instant messenger). This paper presents a survey of Urdu localization for open source software, describes the localization process used for the three software applications listed and discusses issues and challenges that came up during the localization process. The paper concludes with a note work to be done in the future in this area.

1. Introduction

Software localization is a process through which a software application is customized for a specific language-region pair, referred to as a locale [1]. This involves translation of the graphical user interface (GUI) text, adjustment of the GUI layout and customizing definitions of multiple elements, for example, date and time formats, spell checkers etc., such that it fulfills the needs and requirements of a particular language region pair, for example Urdu-Pakistan (ur-PK) or French-Canada (fr-CA) [2].

Software internationalization is a process that is complementary to localization. It is the process through which a software application is designed such that it can be conveniently customized for other languages [3].

Localized open source software has the potential to make a significant impact on the accessibility of information and communication technology for users who are not literate in English. Localization is becoming an increasingly important aspect of open source software for the global community. Many commonly used open source software applications are available for users in multiple locales. Mozilla Firefox for example, is available in over 60 locales.

This paper first presents a brief survey of currently available Urdu-Pakistan versions of open source software. After that, the complete process used to localize three open source software applications, 1) an office suite, 2) an Internet suite and 3) an instant messaging client, will be presented. Notable issues that were encountered during the process will be discussed. The paper will conclude with a note on future directions to be pursued in the context of Urdu localization of open source software.

2. Current status of Urdu localization in open source software

A brief survey of commonly used open source software shows that none have been localized for Urdu, with the exception of Ubuntu and OpenOffice.org, which has an unofficial release available through the work that is presented in this paper. A summary of the survey is shown in Table 1.

Unofficial ur-IN (Urdu for India region) versions of OpenOffice.org 2.0.3, Firefox 1.0.6 and Thunderbird 1.0.7 exist for the ur-IN locale, but these have not been noted in Table 1, which only accounts for ur-PK localizations.

Apart from popular open source software shown in Table 1, Urdu versions of SeaMonkey and Psi, two relatively low profile software applications, are available. The Urdu localization process for these two applications, along with OpenOffice.org is presented in this paper.

In addition, sometimes Urdu versions of specialized software are also available. For example, Poedit, a localization tool, has an Urdu version available for use.

Table 1. Open source software localization status for Urdu

Software	Description	No. of Locales	Urdu (ur-PK) Localization
Firefox 3.6.12	Web browser	66	Locale owner exists but no work done.
Thunderbird 3.1.6	Email client	49	Locale owner exists but no work done.
OpenOffice.org 3.1	Office suite	19	Unofficial release available.
Pidgin 2.7.4	Instant messenger	16	No work done
VLC Media Player 1.1.4	Media player	48	No work done
7zip 4.65	Archive manipulator	13	No work done
GIMP 2.6.11	Image Editor	13	No work done
Audacity 1.2	Audio editor	26	No work done
Ubuntu 10.10	Operating system	28	Active localization team, work in progress.

3. Internationalization and Localization Technology

Internationalized software applications, as mentioned earlier, allow for convenient localization into multiple locales. Internationalization implies that the portion of the software that needs to be adjusted for different locales is available separately for localizers, who can update this portion conveniently as per their requirements without having to get into the technicalities of the software itself. The bulk of this portion is made up of GUI and help content strings which are to be translated. Apart from the strings that are to be translated, other constituents of the software also need to be set as per the requirements of the locale being localized. One example is spellcheckers, which are inherently language specific, for software applications that involve some form of word processing.

Three major localization technologies are widely used within the open source software community currently. These are briefly described in the following subsections.

3.1. GNU gettext based internationalization

GNU gettext is the GNU internationalization and localization library used for developing multilingual software. It enables the production of a file that contains all translatable strings from the source code of a software application. These can then be translated for different locales and used to compile localized versions of the application.

3.2. XUL based internationalization

XUL (XML User Interface Language) is a technology developed by Mozilla. It provides support for localization, user interface layout and appearance customization. Like GNU gettext, it enables the isolation of translatable strings from source code.

3.3. Qt based internationalization

Qt is a cross-platform application and user interface framework which is well known for facilitating the development of applications across multiple platforms. It also enables convenient development of localized versions of applications as well, by isolating translatable strings from the source code.

The localization procedure for any software application is therefore dependent on the technology that has been used for developing the internationalized application.

4. Localization Process

The objective of the work presented in this paper was to develop Urdu versions of some common types of software to be used by rural area school students in Pakistan. In particular, Urdu-Pakistan versions of the following software applications were needed.

1. a web browser
2. an email client
3. an instant messaging client
4. a word processor
5. a graphics editor
6. a webpage development tool

The process used to develop localized versions of the required software types is summarized in Figure 1 and will be presented in this section.

It should be noted again at this point that localization is a process where translation of GUI strings and help content makes up the bulk of the work to be done. Due to this, any localization team should ideally include a balance of both technical and

language experts. The work presented here was completed by a team of three technical experts and three language experts.

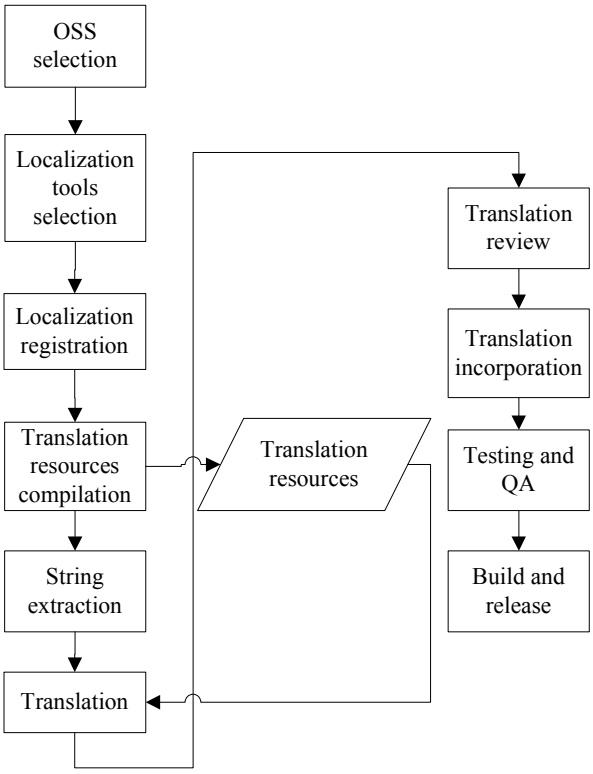

Figure 1: Localization process

4.1. Selection of software

The first step of the process was the selection of software to be localized. To select specific software applications, four points were taken into consideration.

4.1.1. Localization support. The first and foremost criterion for selection was that the application must be internationalized. As discussed earlier, internationalized development facilitates an efficient and convenient localization process by separating all the application elements that need to be customized for a locale.

4.1.2. Encoding support. The application selected had to support the character set encoding required by Urdu. It was also necessary for the application to provide proper bidirectional text support. This is because Urdu is a bidirectional language, written mainly from right-to-left, but also includes portions of text that are written from left-to-right, e.g., numbers.

4.1.3. Cross-platform support. Software that was supported across multiple platforms was preferred, because its localized version would then be available to a wider user base.

4.1.4. Active community. Software that had an associated active community was preferred. An active community ensures that technical assistance will be available when needed. It also a good indicator that development of the software will continue in the future, which in turn means greater potential of use and maintenance of the localized version that is developed.

Based on these criteria, and also taking the usability of the software applications into account, two software suites, OpenOffice.org (www.openoffice.org) and SeaMonkey (www.seamonkey-project.org), and a simple instant messenger, Psi (http://psi-im.org), were selected. OpenOffice.org contains a full suite of office applications including a word processor and a vector based graphics editor. SeaMonkey is a complete Internet suite available from the Mozilla Foundation. It includes a web browser, an email client, and a simple webpage development tool. This suite was given preference over popular individual applications like Firefox and Thunderbird because an integrated suite was considered more usable for the user base being targeted, and the localization effort was also considerably decreased for a single suite as opposed to multiple separate applications.

All selected software was internationalized and had Unicode (UTF-8) and bidirectional language support which was required for Urdu. All three were available for multiple platforms. Finally, all three also had active communities, which ensured that the localization effort would be supported for some time.

4.2. Selection of localization tools

Localization tool selection was done on two levels. Firstly, tools were selected for each application being localized, in order to manage its localization file formats and to create localized builds.

Secondly, in order to keep translations across applications consistent and to keep the translation interface uniform for translators, a tool was selected purely to aid linguists in translation. These are described in the following subsections.

4.2.1. Qt Linguist. Psi is a Qt based application and Qt Linguist was used to obtain the strings which had to be

translated for it and create its installable Urdu language pack.

4.2.2. Mozilla Translator. Similarly, for SeaMonkey, Mozilla Translator was used to obtain the strings which had to be translated for it and create its installable Urdu language pack.

4.2.3. OmegaT. OmegaT is an open source, cross-platform computer aided translation (CAT) tool. It facilitates the translation process by maintaining a translation memory of previous translations. Translation memory can be defined as source and target language pair obtained from a previously completed translation. This is made available to translators to aid in future similar translations.

OmegaT is a versatile tool and one of its key features is that it can handle the translation of multiple file formats including plain text, HTML and OpenDocument formats. Due to this feature, it played a role at both levels in the localization process.

Firstly, it provided a uniform translation interface for translators. Files from both Mozilla Translator and Qt Linguist could be transformed and handled in it.

Secondly, it could handle OpenOffice.org files (PO format) natively, without any transformation. So these were translated directly in OmegaT, and then used for building the Urdu installer for OpenOffice.org

Another key feature of OmegaT is the support of terminology glossaries, which also aid in keeping translations consistent. A core terminology glossary was used during the localization process through OmegaT.

OmegaT maintains translation memories in TMX (Translation Memory eXchange) format which is an XML standard for the exchange of translation memory between different CAT applications. OmegaT is a single user application but allows for manual sharing of translation memory between multiple projects. So, during the localization process, translators had access to translation memories of each others' projects, which were updated manually, at least on a daily basis. As a result, all translators had access to all the translation memory that was developed over the course of time. This helped especially in keeping the translations consistent across the application set, which would not have been easily possible if an individual tool had been used for each application.

OmegaT also provides Unicode (UTF-8) support and bidirectional support for right-to-left languages so it was very convenient to use for English to Urdu translations.

Figure 2 shows a sample OmegaT project for English to Urdu translation. One source file from the project has been opened for translation, and a string "Minimum font size" has been selected (in the main window on the left). As soon as a string is selected, matches from the translation memories and glossaries are displayed in the windows on the right.

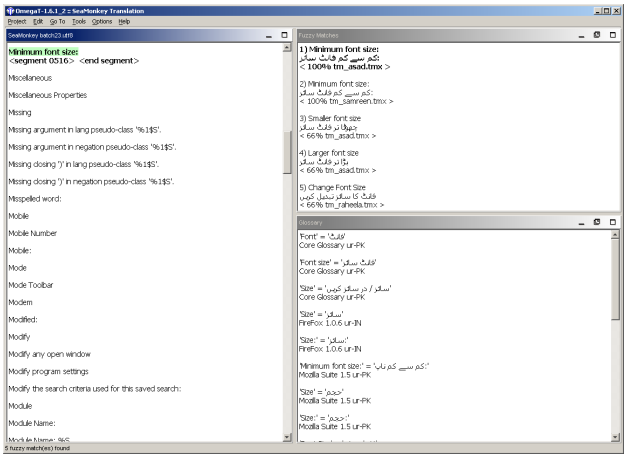
Minimum font size:
<segment 0516> <end segment>
Miscellaneous
Miscellaneous Properties
Missing
Missing argument in lang pseudo-class '%1$S'.
Missing argument in negation pseudo-class '%1$S'.
Missing closing ')' in lang pseudo-class '%1$S'.
Missing closing ')' in negation pseudo-class '%1$S'.
Misspelled word:
Mobile
Mobile Number
Mobile:
Mode
Mode Toolbar
Modem
Modified:
Modify
Modify any open window
Modify program settings
Modify the search criteria used for this saved search:
Module
Module Name:
1) Minimum font size:
< 100% tm_asad.tmx >
2) Minimum font size:
< 100% tm_samreen.tmx >
3) Smaller font size
< 66% tm_asad.tmx >
4) Larger font size
< 66% tm_asad.tmx >
5) Change Font Size
< 66% tm_raheela.tmx >
Core Glossary ur-PK
Core Glossary ur-PK
Core Glossary ur-PK
FireFox 1.0.6 ur-IN
FireFox 1.0.6 ur-IN
Mozilla Suite 1.5 ur-PK
Mozilla Suite 1.5 ur-PK
5 fuzzy match(es) found

Figure 2: Sample OmegaT project

The bottom window on the right shows matches from the glossary, along with the name of the glossary where the match was found.

The top window titled "Fuzzy Matches" shows similar translations from translation memories. The "Fuzzy Matches" window shows five matches. The translation memory files in this case have been named after the translators they were obtained from, and this name can be seen at the end of each match along with the match percentage.

4.3. Localization registration

When starting an open source software localization, it is best to contact the software community and coordinate with them, so that localization efforts aren't duplicated and so it can be released through the community as an official build. This is usually done through a registration procedure, which varies for different software.

Urdu-Pakistan (ur-PK) localization teams were officially registered for SeaMonkey and Psi. The ur-PK locale for OpenOffice.org was already registered to a community member, so an effort was made to collaborate with the existing team.

4.4. Compilation of translation resources

A survey was conducted to collect resources that would help in the translation process. These included dictionaries, terminology glossaries and previous localization work done for Urdu. Some of the major resources used during the localization process are presented in detail next.

4.4.1. NLA Glossary. This is a computer terminology glossary based on the "Electronic Dictionary of Localization of Computer Applications (English - Urdu)", by the National Language Authority Islamabad, Pakistan. This is the main glossary that was referred to during the translation process as it represents the recommended standard for Pakistan. It has also been used by Microsoft for Urdu localization of its software products, so using it also ensured a uniform terminology for users across applications. Additional entries were made to this glossary during the translation process, as described ahead.

4.4.2. Localized software for Urdu. The following Urdu language versions of software were found during the survey.

1. Mozilla Suite 1.5 ur-PK
2. Firefox 1.0.6 ur-IN
3. Thunderbird 1.0.7 ur-IN
4. OpenOffice.org 2.0.3 ur-IN

Translations from these were extracted and used as reference glossaries during the translation process. The ur-PK translation was more useful as compared to the ur-IN translations because the ur-IN locale used translations of a slightly different style than the one adopted for the ur-PK localization. One example was the level of respect used when referring to the user.

4.4.3. Online technical terminology translations (English to Urdu): Two significant English to Urdu technical technology translations were available online. The first was the Urdu Word Bank (http://l10n.urduweb.org/dictionary/), which has user generated translations of technical terms. Users can look up translations for technical terminology, edit existing translations, add new translations, or put up requests for translations. The second was an Urdu technical terms glossary (http://www.qern.org/it/dict/urdu/dict_main.cgi) which also allows users to enter their own translations, but it is not as active as the first one.

4.4.4. Dictionaries: All major English to Urdu translation dictionaries were also been consulted in the process, e.g. Qaumi English-Urdu Dictionary published by the National Language Authority of Pakistan.

4.4.5. Miscellaneous: Other than the resources listed above, frequently consulted resources included: 1) WordNet (http://wordnet.princeton.edu/), an English lexical database; this is helpful when there is confusion about the sense or part-of-speech of a word being translated, 2) specialized terminology translations compiled by the National Language Authority Pakistan (e.g., mathematical terms, scientific terms etc.), and 3) various other online dictionaries and online documentation for the applications being localized.

4.5. String extraction

The next step of the process was the extraction of strings to be translated from each application such that they could be translated using OmegaT. Strings were extracted and divided into batches for management purposes. Each batch contained about 600-700 words. The number of strings in each batch varied according to the number of words per string. One translator completed the translation of four batches in about a week on average. Strings to be translated come from three sources in the application: 1) the GUI, 2) the application help, and 3) any other application documentation.

4.6. Translation

Each translator had an OmegaT project for translation and each subsequent file to be translated was added to the project. Each project contained a core glossary, reference glossaries and also the translation memory of all the linguists in the team (updated on a daily basis or as required).

For translation purposes each word in a string was first classified as either a functional or a content word. All nouns, verbs, adjectives and adverbs are content words; words that fall into any other category, e.g., prepositions, conjunctions etc. are functional words. For each string to be translated, the translation of functional words was left to the discretion of each individual linguist, but translations of content words were taken from the core glossary only (which was developed with the mutual consent of translators and developers).

For example, in the following strings, the content words are in bold: "**Failed** to **remove** this **account**."; "**Filters associated** with this **folder** will be **updated**."; "**Horizontal scrolling**"; "**New languages** can be **configured using** the **Languages Panel**."

Keeping the above rule in mind, the translators would proceed with the translation in four stages as

described in the following sections, and shown in Figure 3.

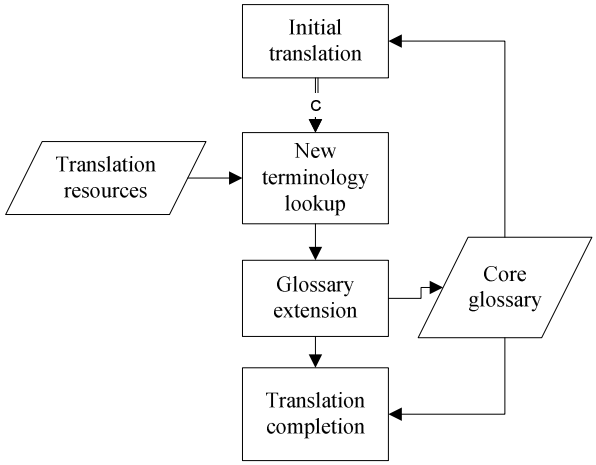

Figure 3: Translation process

4.6.1. Initial translation. At the beginning of each week, translators were given a set of four translation batches. Translators would initially go through these, translating those strings for which all content words have appropriate entries in the core glossary. The NLA glossary, described earlier was used as the core glossary, and was extended through the process being described here. Strings which had a content word which was not included in the core glossary were skipped and the missing word was entered into a list of new terms.

4.6.2. New terminology lookup. After the translation stage, translators looked up appropriate translations for the new terms. Translators had access to the translation resources described earlier during this step, and developers are also consulted when the context of a term could not be determined.

4.6.3. Glossary extension. After the compilation of new terminology lists, a team meeting was held including both translators and developers. During the meeting, new translations were finalized and added to the core glossary. Issues could be raised from both linguistic and technical perspectives. From the linguistic perspective, more appropriate translations were sometimes suggested, and from the technical perspective, incorrect senses and parts-of-speech for words used during translation were sometimes identified.

4.6.4. Translation completion. The translators would then use the updated glossary to complete the set of translations for the week.

This process was repeated on a weekly basis.

4.7. Translation review and incorporation

Translations were reviewed and finalized by developers and incorporated into the applications, using the application specific tools.

Control and accelerator keys were also assigned during this phase. Control and accelerator keys are shortcut keys for menus and menu items indicated to the user by underlining a character in a menu or menu item. For example the "File" menu in most applications has the "F" underlined, and it can be accessed by pressing Alt+F. In this case, "F" is the accelerator key. An example of a control key is Ctrl+S for the "Save" item (in the "File" menu), where the "S" is underlined. Control and accelerator keys both need to be set appropriately according to the translations.

Most translation errors detected during this phase were caused due to misinterpretation of the source string. This misinterpretation was usually caused by one of the following reasons. Firstly due to limited exposure to software in general, translators were not familiar with some types of sentence structures used in software GUIs. Secondly because the linguists had not used the software being localized, they could not understand concepts specific to the software (e.g., the notion of tabbed browsing), and might translate them inappropriately.

4.8. Quality assurance

A quality assurance process was used to ensure that the final localized product was free of errors. Some of the individual applications had their own quality assurance procedures as well which were followed where needed, but an overall quality assurance process was devised as well.

After translation incorporation, some preliminary tests were conducted by developers to identify commonly occurring errors, e.g., placeholders in strings not being displayed as expected. An example of this is shown in Figures 4, where the source string to be translated is "The web site %S does not support encryption for the page you are viewing.". Here "%S" is a placeholder, and may be misplaced during translation, as shown in Figure 4. The string inserted for the placeholder "www.google.com.pk" is appearing at an incorrect position. Errors of this type can occur

due to linguistic (lack of knowledge about the nature of the placeholder may cause incorrect placement) or technical reasons, specifically, due to insufficient bidirectional support – only in the case of left-to-right languages - the placeholder in the translated string may appear in a different position in the localization tools and in a different position within the application being localized.

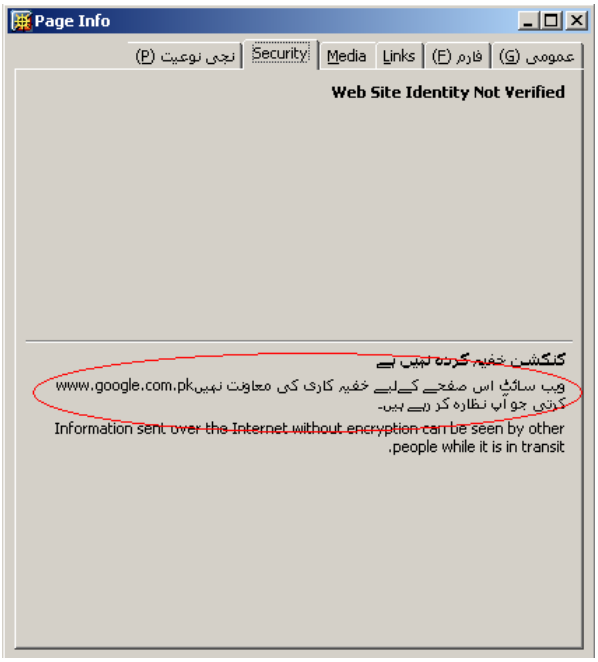

Figure 4: Misplacement of placeholder in translated string.

Another common error was the use of Urdu translation strings that were too long as compared to their English counterparts and did not fit in their designated position in the GUI. This would either cause some GUI components to expand and cause problems in the overall application, or it would cause the text to appear in truncated form. This had to be solved by developing an alternate, shorter translation.

Interim versions of the localized applications were also frequently deployed within the team for user testing.

4.9. Release

After translation and quality assurance was completed for Psi and SeaMonkey, Urdu language packs were released as per the process and release schedule for the software. An unofficial localized build was released for OpenOffice.org because the registered ur-PK localization team was inactive.

A total of about 26,000 strings were translated for OpenOffice.org. Figure 5 shows the Urdu version of OpenOffice.org Writer. The unofficial ur-PK installer, corresponding to OpenOffice.org 2.4.0 is available at http://panl10n.net/english/Outputs%20Phase%202/CCs/Pakistan/Software/2008/OpenOffice.org(unofficial).zip.

A total of around 2000 strings were translated for the instant messenger, Psi. The language pack for the current version, 0.14, released in collaboration with the Psi team is available at http://psi-im.org/download/lang/ur_PK.

The localized software was deployed in 10 rural area secondary schools as part of Project Dareecha, more details for which can be found at www.crulp.org/dareecha/.

5. Localized software

As a result of the process described in this paper, localized versions of the selected software applications were released.

A total of around 10,000 strings were translated for the SeaMonkey suite, and installable Urdu language packs were released in collaboration with the SeaMonkey team for versions 1.1.5 through 1.1.19. Release 1.1.19 is available at www.seamonkey-project.org/releases/1.1.19.

A total of about 26,000 strings were translated for OpenOffice.org. Figure 5 shows the Urdu version of OpenOffice.org Writer. The unofficial ur-PK installer, corresponding to OpenOffice.org 2.4.0 is available at http://panl10n.net/english/Outputs%20Phase%202/CCs/Pakistan/Software/2008/OpenOffice.org(unofficial).zip.

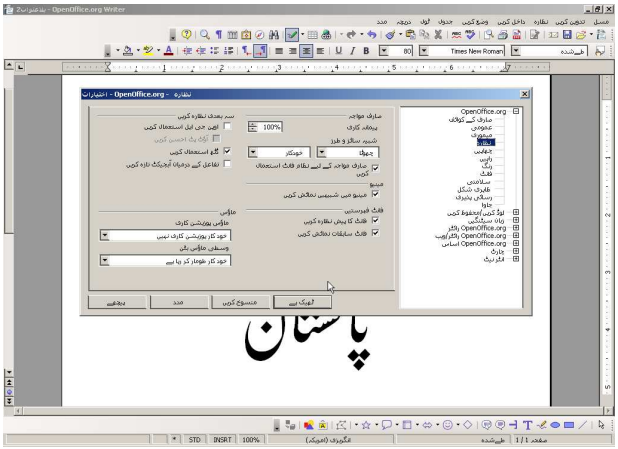

Figure 5: OpenOffice.org Writer in Urdu

A total of around 2000 strings were translated for the instant messenger, Psi. The language pack for the current version, 0.14, released in collaboration with the Psi team is available at http://psi-im.org/download/lang/ur_PK.

The localized software was deployed in 10 rural area secondary schools as part of Project Dareecha, more details for which can be found at www.crulp.org/dareecha/.

6. Translation issues

Translation was a critical part of the localization process. Inappropriate translations would have rendered the localized software unusable, so a meticulously planned translation process was used to ensure high quality translations, as described in earlier. This section covers some translation selection issues, and describes a problem specific to Urdu translation, in order to illustrate the types of problems that are encountered during localization.

6.1. Translation selection

When available, technical terms were translated as per the NLA glossary described in 5.3.1. This is the nationally recommended standard, also in use by Microsoft. The advantage of using it as the core reference was that users would be seeing the same, familiar, terminology if they switched from proprietary to open source software.

If a terminology translation could not be found within the core glossary, a translation was coined using the conventions followed by the NLA glossary. If there was a conflict, preference was given to the simplest option. Because all new terminology was coined through a collaborative process including both developers and translators, it was ensured that translations were both linguistically and technically appropriate.

There were a few cases where the NLA recommended terminology was inappropriate and therefore not followed. An example is the English word "Beep". The translation recommended by the NLA in this case is "پیں". There is no equivalent word for "Beep" in Urdu and it seems to be translated using the concept of onomatopoeia where a word itself suggests the sound that it describes [4]. During the localization of Psi, the following string had to be translated: "Beep twice". If the NLA recommendation had been followed, it would have had to be translated as either "دو دفعہ پیں کریں" or "پیں پیں کریں", both of which would have been equally awkward. A decision was therefore made to not use the NLA recommendation and simply transliterate the word in Urdu script instead.

6.2. No capitalization in Urdu

When a button is being referred to in an English string, the capitalization of the first letter and the syntax makes it clear that a button is being referred to. For example, in the text from SeaMonkey "Click Finish to create new profile," it is clear that "Finish" refers to a button due to capitalization. However, Urdu does not have capitalization so there is not easy way to identify the button in the translated text. The decision to make the translation unambiguous was to use single quotes to indicate a button name. So the sentence given above was translated as:

نئی پروفائل بنانے کے لیے 'تکمیل کریں' کلک کریں۔

7. Conclusion and future work

This paper presented the process used to localize three open source software applications for Urdu-Pakistan. These particular three applications were aimed for use by rural area school children, where they would aid in eliminating the language barrier in information and communication technology access. The survey presented at the start of the paper showed that there are still numerous software applications that can be localized to serve the same purpose. Therefore efforts like this must be extended and improved, as they play a crucial role in enabling information and communication technology access for the average citizen of Pakistan, who is not literate in English.

8. Acknowledgements

This work was carried out at the Center for Research in Urdu Language Processing (www.crulp.org), National University of Computer and Emerging Sciences, Lahore (www.nu.edu.pk), and was funded the PAN Localization Project (www.panl10n.net). Software specific technical support was provided by the user and development communities of OpenOffice.org (www.openoffice.org), SeaMonkey (www.seamonkey-project.org), Psi (http://psi-im.org/) and OmegaT (www.omegat.org) open source projects.

9. References

[1] A. Souphavanh and T. Karoonboonyanan, *Free/OpenS Source Software: Localisation*, Elsevier and UNDP, India, 2005.

[2] M.P. Pustakalaya, *Guide to Localization of Open Source Software*, Center for Research in Urdu Language Processing and the International Development Research Center.

[3] S. Hussain and R. Mohan, "Localization in Asia Pacific", in *Digital Review of Asia Pacific 2007-2008*, Orbicom and the International Development Research Center 2008.

[4] F. Katamba, *Morphology*, Palgrave Macmillan, 1993.