GUIDELINES

FOR

CONTEXTUALIZATION

Shankar Prasad Venkateshbhat
Ani Peter
Nilamdyuti Goswami

CONTENTS

PREFACE

"Hang not, leave him" - The judge pronounced to free a man off to life. But the verdict was written to the jailer as "Hang, not leave him" misplacing the coma which ended up in killing the innocent man. Thus its called "A coma killed a man". We can see how important it is to place punctuation in the correct position to communicate correctly and clearly, so that the recipient gets the exact meaning of what is conveyed. Similarly, describing and detailing on a subject without understanding the exact context may end up in confusions and not communicating the exact and expected meaning. Translating an English sentence into a native language, without the translator knowing the correct context can result in the end user interpreting incorrect or wrong meaning of the sentence.

Here, we discuss about contexts related to software applications. Each software application is intended to serve a particular purpose. Content writers understand the applications and provide the required instructions, manuals and GUI dialogues. Next phase translators translate these text to native languages for the end user. In the current scenario, where translatable strings are generated using computer program into .po format, translators work on one sentence/string at a time. Many a times, developers fail to provide correct context for each translatable string. In some cases, strings can be ambiguous, split up into two conveying partial meaning, can have application functions based new words etc, without correct context being provided. Such strings will be translated incorrectly, conveying the wrong meaning to the end users affecting the usage of application. Contextualization in the only solution to overcome this deficiency.

NEED OF CONTEXTUALIZATION

Contextualization is the process of providing relevant context for source strings. The objective is to provide meaningful descriptions of the source strings for translators to ensure the correctness and quality of the translations. Thus facilitating good end user experience of the localized applications.

Contextualization :-

- ensures that the correct meaning of source strings are conveyed to the translators, thus the quality of translation is not compromised.
- improves the user experience of the localized applications.
- increase the popularity of the localized Fedora desktop.

Contextualization not only help translators, this process extends its benefits to both users and developers too.

For TRANSLATORS:
Providing correct (relevant) context/meaning/reference helps the translators to create good translations ensuring better quality.

For USERS:
Good quality translations help the user to understand and use an application making best benefit of it. This aids to the betterment of the application.

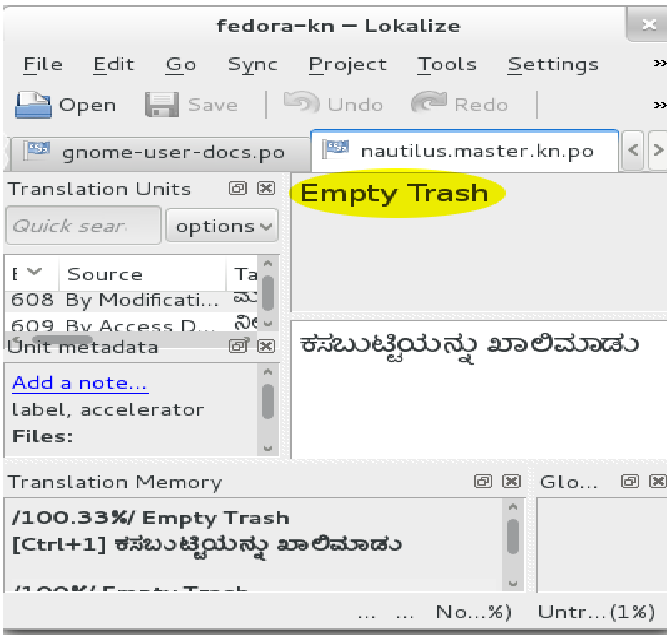
For DEVELOPERS:
Good translations increase number of users which in turn increases the number of application users and benefiting the developer's hard work. Users will be able to provide good feedback on the application enabling the developer to improve his application.

SCENARIOS REQUIRING CONTEXTUALIZATION

- ## AMBIGUOUS WORDS:

Some of the source strings have ambiguous words whose context needs to be mentioned.
Line: Here the word **Line** can be a line of text, a thin continuous mark i.e, a graphical representation, line number etc. All these different scenarios of word line can be found in a word processing application like LibreOffice Writer.
Empty Trash: Here word **Empty** can be a verb or an adjective i.e, whether to empty the trash or the trash is empty.

This is how the string appears in a translation tool:



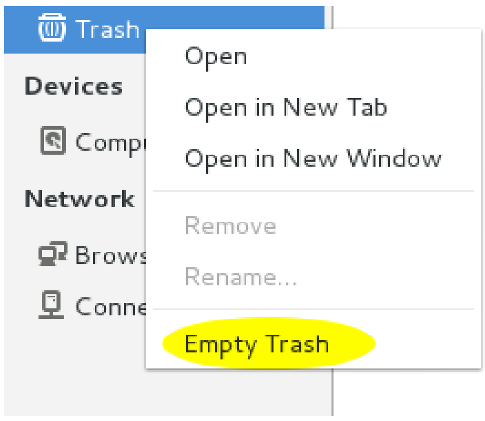
Empty Trash

ಕಸಬುಟ್ಟಿಯನ್ನು ಖಾಲಿಮಾಡು

label, accelerator

/100.33%/ Empty Trash
[Ctrl+1] ಕಸಬುಟ್ಟಿಯನ್ನು ಖಾಲಿಮಾಡು

This is how the string appears in application UI:

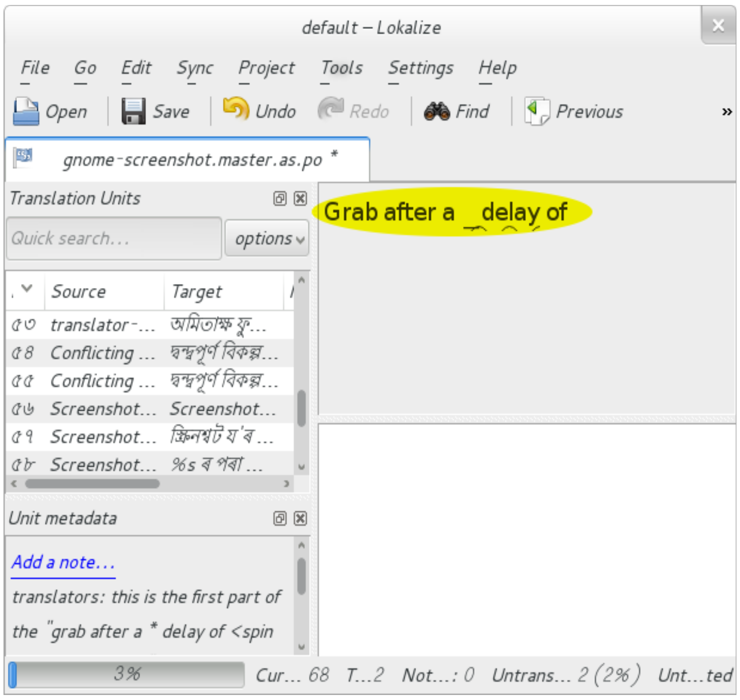


Here **Empty** is used as a verb.

- **SPLIT SENTENCES:**

Some of the source strings are split into two or more strings which need to be thought as a single sentence while translating to convey the correct message.
e.g. “Grab after a delay of”
“%s seconds”
Here these two sentences are actually a single sentence appearing in the application UI whereas in translation editor they are two separate sentences.

This is how the string appears in a translation tool:



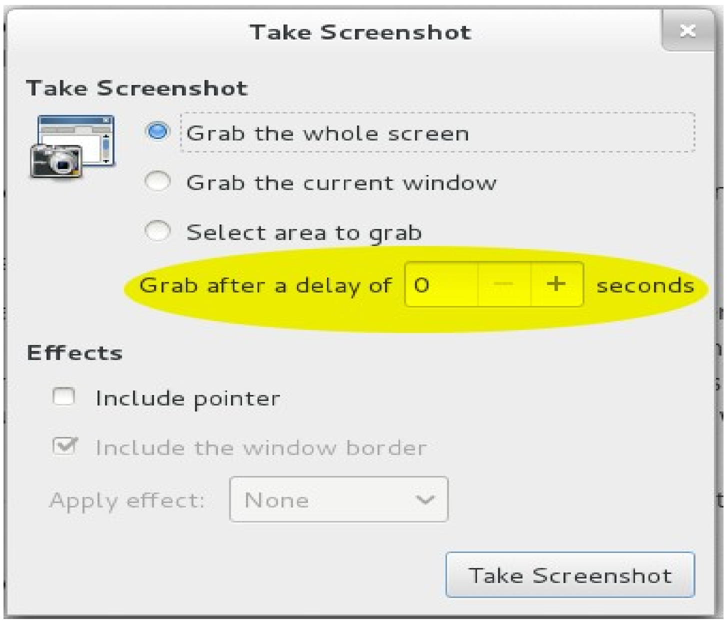
Grab after a _delay of

	Source	Target
৫৩	translator-...	অমিতাক্ষ ফু...
৫৪	Conflicting ...	দ্বন্দ্বপূৰ্ণ বিকল্প...
৫৫	Conflicting ...	দ্বন্দ্বপূৰ্ণ বিকল্প...
৫৬	Screenshot...	Screenshot...
৫৭	Screenshot...	স্ক্ৰিনশ্বট য'ৰ ...
৫৮	Screenshot...	%s ৰ পৰা ...

translators: this is the first part of the "grab after a * delay of <spin

3% Cur... 68 T...2 Not...: 0 Untrans... 2 (2%) Unt...ted

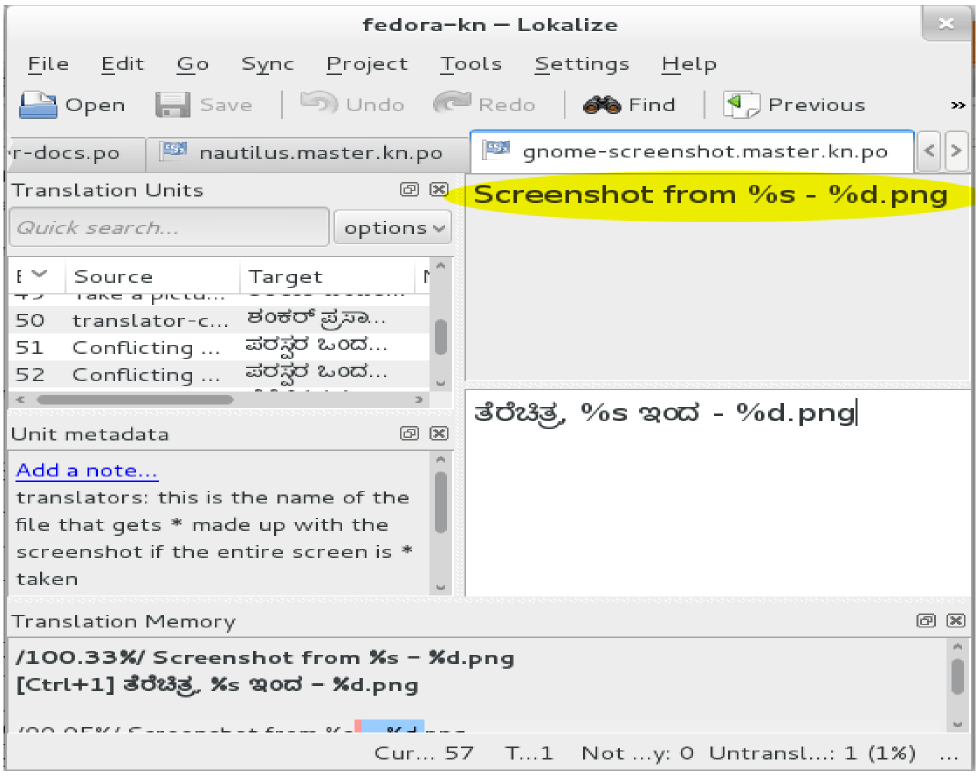
This is how the string appears in the application UI:



- **VARIABLES:**

  Strings that have variables should have their data type (character, integer, date time etc) mentioned to convey correct message for meaningful translation.
  e.g. "Screenshot from %s - %d.png"

  This is how the string appears in a translation tool:



Screenshot from %s - %d.png

ತೆರೆಚಿತ್ರ, %s ಇಂದ - %d.png

	Source	Target
50	translator-c...	ಶಂಕರ್ ಪ್ರಸಾ...
51	Conflicting ...	ಪರಸ್ಪರ ಒಂದ...
52	Conflicting ...	ಪರಸ್ಪರ ಒಂದ...

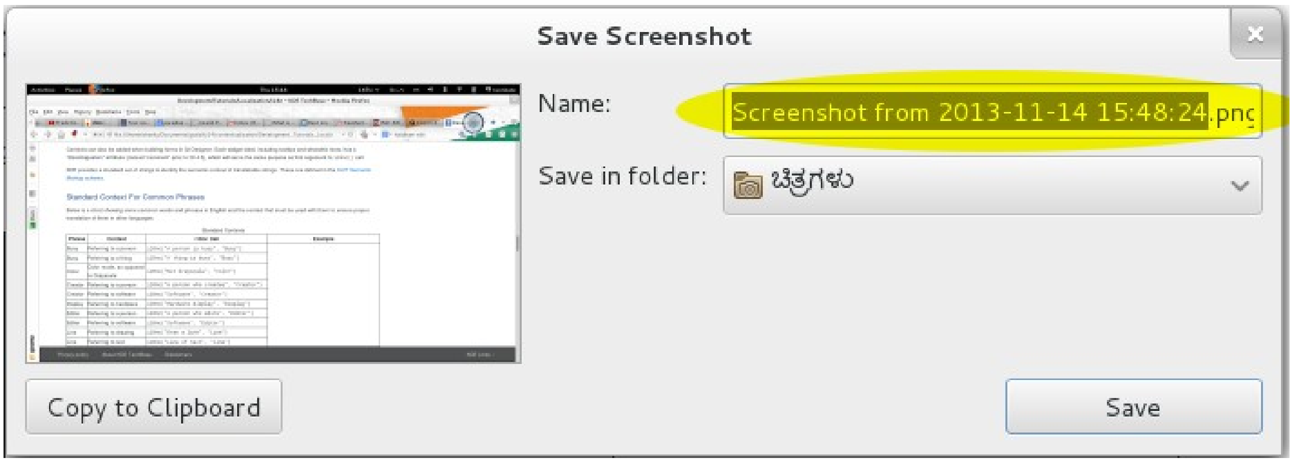
translators: this is the name of the file that gets * made up with the screenshot if the entire screen is * taken

/100.33%/ Screenshot from %s – %d.png
[Ctrl+1] ತೆರೆಚಿತ್ರ, %s ಇಂದ – %d.png

Cur... 57 T...1 Not ...y: 0 Untransl...: 1 (1%) ...

  Here %s has a date value and %d has a time value. So, the translation should be like:

  Screenshot from date – time.png



Name: Screenshot from 2013-11-14 15:48:24.png

Save in folder: ಚಿತ್ರಗಳು

- **WORD OR PHRASE MEANING:**

Sometimes the source string contains some words or phrases whose meaning is not clear or may be too technical.

e.g. : In the file firewalld**,** phrases like **“Shields Up” and “Shields Down”** don't have a clear meaning.

In Firefox, **Caret Browsing** doesn't have a clear meaning.

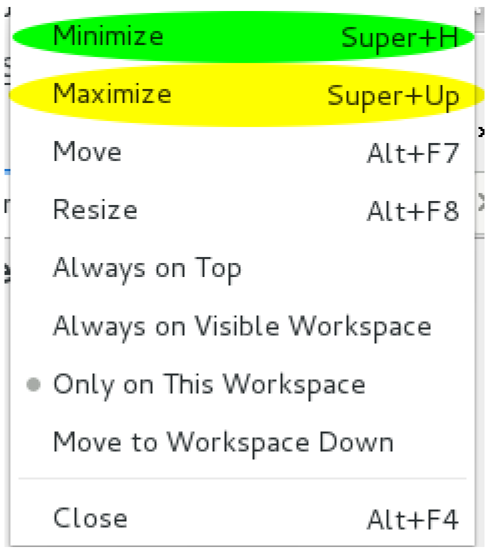
“**Load request canceled”:** Here meaning of the word **Load** needs to be given.

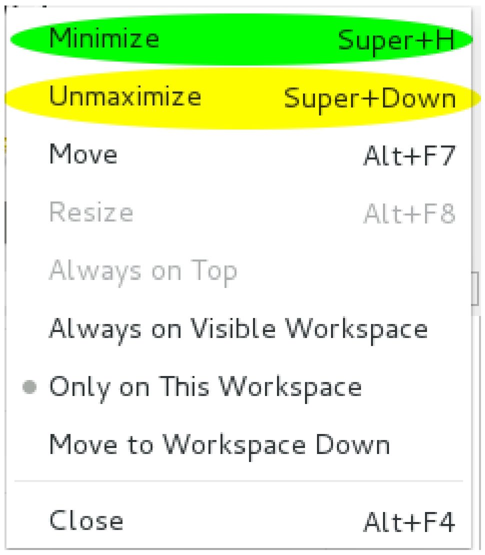
Words like **maximize**, **unmaximize** and **minimize** need to have a clear context as there use cases are different then their actual meaning. UI s having these words have the following meanings in terms of usage:

Maximize: To make the application window full screen.

Unmaximize: To make the application window small (similar to **Restore**)

Minimize: To hide the application window.





- **TRANSLATION GUIDELINES:**

  - Some strings have keywords, commands, command literals, keyboard keys, function names, service modes, values which should be retained in English else it will create functionality and usability issues.

    Function call: default:LTR

    Commands: subscription-manager, clear, exit etc

    Command literals: firstname, surname etc

    Keyboard keys: Shift, Enter etc

    Service modes: Selinux modes like enforcing, permissive...

    Values: TRUE, FALSE, NONE etc.

  - In another instance where words shown below can be translated or left untranslated but consistency need to be maintained:

    [COMMAND] , [PATH] , [FILENAME]

Hence, information is required on the above mentioned scenarios to avoid confusion.

WAYS OF WRITING CONTEXT RELATED MESSAGES

When writing the context related messages to the source strings, one has to study the application and its usage. Since this text will be inserted in the source code of the application, text gets multiplied with the number of languages to which it will get translated. This leads to considerable increase in size of the application. Hence one should write precise yet meaningful context messages. Providing proper reference urls and examples is also a good idea.

Standard Context For Common Phrases used in KDE

Below is a chart showing some common words and phrases in English and the context that must be used with them to ensure proper translation of them in other languages.

Standard Contexts

Phrase	Context	i18nc Call
Busy	Referring to a person	`i18nc("A person is busy", "Busy")`
Busy	Referring to a thing	`i18nc("A thing is busy", "Busy")`
Color	Color mode, as opposed to Grayscale	`i18nc("Not Grayscale", "Color")`
Creator	Referring to a person	`i18nc("A person who creates", "Creator")`
Creator	Referring to software	`i18nc("Software", "Creator")`
Display	Referring to hardware	`i18nc("Hardware display", "Display")`
Editor	Referring to a person	`i18nc("A person who edits", "Editor")`
Editor	Referring to software	`i18nc("Software", "Editor")`
Line	Referring to drawing	`i18nc("Draw a line", "Line")`
Line	Referring to text	`i18nc("Line of text", "Line")`
Name	Referring to a name of thing	`i18nc("A thing's name", "Name")`
Name	Referring to first name and last name of person	`i18nc("Person's first and last name", "Name")`
New	Create something	`i18nc("Action", "New")`
New	Status	`i18nc("New mail message", "New")`
No	Answer to a question	`i18nc("Answer to a question", "No")`
No	Availability of a thing	`i18nc("Availability", "No")`
(Re)load	(Re)load a	`i18nc("(Re)load a document", "(Re)load")`

	document, medium etc.	
(Re)load	(Re)start a program, daemon etc.	`i18nc("(Re)start a program", "(Re)load")`
Title	Referring to a person	`i18nc("A person's title", "Title")`
Title	Referring to a thing	`i18nc("A thing's title", "Title")`
Trash	Referring to the action of emptying	`i18nc("The trash is not empty. Empty it", "Empty")`
Trash	Referring to the state of being empty	`i18nc("The trash is empty. This is not an action, but a state", "Empty")`
Volume	Referring to sound	`i18nc("Sound volume", "Volume")`
Volume	Referring to a filesystem	`i18nc("Filesystem volume", "Volume")`
Volume	Referring to books	`i18nc("Book volume", "Volume")`
Yes	Answer to a question	`i18nc("Answer to a question", "Yes")`
Yes	Availability of a thing	`i18nc("Availability", "Yes")`

IMPLEMENTING CONTEXT IN SOURCE CODE:

- **C/C++ Files:**

```
/* Translators: this refers to an unknown language code
* (one which isn't in our built-in list).
*/
name = g_strdup_printf (C_("language", "Unknown (%s)"), code);
```

This will automatically turn into this in the pot and po files:

```
#. Translators: this refers to an unknown language code
#. * (one which isn't in our built-in list).
#.
#: ../plugins/spell/gedit-spell-checker-language.c:393
#, c-format
msgctxt "language"
msgid "Unknown (%s)"
msgstr ""
```

- **Qt Files:**

```
//: This name refers to a host name.
  hostNameLabel->setText(tr("Name:"));

  /*: This text refers to a C++ code example. */
  QString example = tr("Example");
```

This will automatically turn into this in the pot and po files:

```
#. This name refers to a host name.
      msgid "Name"
      msgstr ""
#. This text refers to a C++ code example.
      msgid "Example"
      msgstr ""
```

- **VALA Files:**

```
// Translators: The %s will be replaced with the name of the VM
toolbar.title = _("%s - Properties").printf (App.app.current_item.name);
```

This will automatically turn into this in the pot and po files:

```
#. Translators: The %s will be replaced with the name of the VM
#: ../src/properties.vala:19
#, c-format
msgid "%s - Properties"
msgstr ""
```

- **PYTHON Files:**

```
# global refresh command
# TRANSLATORS: 'r' to refresh
if self._screens and (key == C_('TUI|Spoke Navigation', 'r')):
        self._do_redraw()
        return True
```

This will automatically turn into this in the pot and po files:

```
#. TRANSLATORS: 'r' to refresh
#: pyanaconda/ui/tui/simpleline/base.py:451
msgctxt "TUI|Spoke Navigation"
msgid "r"
msgstr ""
```

- **JavaScript Files:**

```
// Translators: "Title" is the label next to the document title
                // in the properties dialog
                this._title = new Gtk.Label({ label: C_("Document Title",
"Title"),
                halign: Gtk.Align.END });
                this._title.get_style_context ().add_class('dim-label');
                grid.add(this._title);
```

This will automatically turn into this in the pot and po files:

```
#. Translators: "Title" is the label next to the document title
#. in the properties dialog
#: ../src/properties.js:82
msgctxt "Document Title"
msgid "Title"
msgstr ""
```

- **XML Files:**

```
<short><!-- Translators: Here “Preload” is a verb -->Preload
engines</short>
```

This will automatically turn into this in the pot and po files:

```
#. Translators: Here “Preload” is a verb
#: ../data/ibus.schemas.in.h:2
msgid "Preload engines"
msgstr ""
```

In Mozilla Firefox, context or translators comment are not added in the source code, they are directly given in the .dtd or .properties.

```
<!-- LOCALIZATION NOTE (pinAppTab.label, unpinAppTab.label):
"Pin" is being
used as a metaphor for expressing the fact that these tabs are
"pinned" to the
left edge of the tabstrip. Really we just want the string to express the
idea
that this is a lightweight and reversible action that keeps your tab
where you
can reach it easily. -->
<!ENTITY  pinAppTab.label                  "Pin as App Tab">
<!ENTITY  pinAppTab.accesskey                 "P">
<!ENTITY  unpinAppTab.label                  "Unpin Tab">
<!ENTITY  unpinAppTab.accesskey                "b">
```

So, as shown in the above examples, relevant context should be added to the source code according to the programming language used for the application.

e.g. In Gnome applications, where mostly C/C++ or Vala programming languages are used, contexts for the source strings are to be provided in the following way:

- **For C/C++ files:**

  The translators' comment should be written inside /* */ like:

  ```
  /*Translators: Here, 'Empty' is a verb*/
  g_printf (_("Empty Trash"));
  ```

  Here, '**Empty Trash**' is the source string in the .po file.

- **For Vala Files:**

  The translators' comment should be written after // like:

  ```
  // Translators: The %s will be replaced with the name of the VM
  toolbar.title = _("%s - Properties").printf (App.app.current_item.name);
  ```

  Here, '**%s - Properties**' is the source string in the .po file.

In Fedora applications, where mostly Python programming language is used, contexts for the source strings are to be provided in the following way:

- **For Python Files:**

  The translators' comment should be written after # like:

  ```
  # global refresh command
  # TRANSLATORS: 'r' to refresh
  if self._screens and (key == C_('TUI|Spoke Navigation', 'r')):
          self._do_redraw()
          return True
  ```

  Here, **'r'** is the source string in the .po file.

These days most of the devices are supporting locales other than en. A statistics says that 56.2 percent of consumers say that the ability to obtain information in their own language is more important than price. – (Common Sense Advisory, Can't Read, Won't Buy: Why Language Matters on Global Websites, 2006). This tells us the importance of localization of software applications. Providing proper context information in translatable files will benefit both the translators as well as the end user. This is still more important in the FOSS community to get more translators as the localization is done voluntarily. At the end a local language will be the winner.

REFERENCES:

- ***Context in Technical Translation Concept and Guidelines – Rajesh Ranjan***

  http://svn.fedorahosted.org/svn/fuel/qa/fuel-context-in-technical-translation-concept-and-guidelines.pdf

- ***Development Tutorials Localization i18n - KDE TechBase***

  http://techbase.kde.org/Development/Tutorials/Localization/i18n