



CLIENTSIDE

Intelligence for Global Business

NEWS

TECH SPOTLIGHT:

*Optimization
with
QA software*

**TECH WRITERS
CORNER:**

*GUI Strings...
"sounds
yummy"*

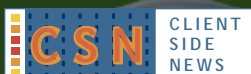
LPM CORNER:

*Translation
Project Success
Guidelines*

May 2008 FEATURE

***IP Intelligence with
geolocation...
accurate, powerful
and very cool.***

The Business Journal
for GILT Professionals
9.95 • US
12.95 • CAN
9.95 • EURO



MAKING MACHINE TRANSLATION EASIER

- USING LANGUAGE SOFTWARE TO IDENTIFY GUI STRINGS

by
Sabine Lehmann
Senior Computational Linguist,
acrolinx

In the last three decades, major companies have attempted to improve the comprehensibility of their technical publications by implementing various types of Controlled Language (CL). A CL places particular restraints on lexicon, grammar, and style to improve the clarity, consistency, and translatability of texts. More recently, software technologies have helped businesses make sure that their publications comply with CL standards. The best technologies have significantly reduced costs and time to market for businesses that publish information in multiple languages.

One of our customers has been using acrolinx CL software since 2005 to maximize the effectiveness of a third-party, rule-based Machine Translation (MT) system within its translation workflow. The acrolinx tool was fine-tuned to meet specific requirements with regard to style and terminology. Writers check their new XML topics using the acrolinx plug-in for the XMetal editor to make sure their documents comply with approved terminology, grammatical and stylistic rules. Before translating these topics, the localization department can examine reports to determine whether specific issues will affect the overall MT quality.

In this article, we look at one aspect of controlled authoring: the treatment of User Interface (UI) strings, which abound in the procedural sections of IT technical documentation.

UI strings encompass both Graphical User Interface (GUI) strings - which appear in Dialog boxes, Mouseovers, and so on - and text strings in non-graphical interfaces such as databases and older mainframe systems. "Strings" is a key word here because we are not merely dealing with sentences, or even words. They may include names

of software, field names, or messages from a program or operating system.

Here, we consider several challenges that UI strings pose for CL checking, focusing on GUI strings in particular. We first provide a broad overview of the issue and then explore two more specific strategies that can be used to detect GUI strings.

The Challenges of GUI Strings

Why do companies care about GUI strings? From an authoring perspective, GUI strings should match the strings that are present in the software itself. However, software is updated by developers on a regular basis, so it becomes difficult for technical writers to reflect the changes in their documentation. Typical discrepancies between software and documentation include capitalization, hyphenation, and spelling. And if writers do not extract GUI strings from their documentation to conduct regular cross-references with software strings, the discrepancies are bound to go uncorrected. This ultimately affects the end user's experience and subsequent translation process. From the end-user perspective, GUI strings are key conduits for communication, and users have to recognize GUI elements as GUI elements. And finally, CL checking needs to recognize GUI strings in order to analyze sentences correctly.

When our customer deployed their acrolinx software, their technical writers reported that grammar and style checking generated so-called "false alarms" - it mistakenly flagged items that were correct. They also reported translation problems because of the presence of GUI strings in source content.

For example, our customer uses a sentence-length rule that flags sentences of more than 25 words, since these sentences impede translatability. GUI strings presented a problem for this rule because long strings were handled as words:

On the Protection Manager Servers page, select the Main group in the View Servers pane and click Edit Group Properties to display the Settings dialog box.

If the controlled language checker does not deal specifically with the GUI strings, this sentence contains 27 separate words, or 2 words more than the rule limit. And it would be difficult to correct this sentence because the writer cannot merely shorten the GUI strings into one-word elements. The sentence is not too long. Rather, the GUI strings should each count as one word - for a total of 21 words in the sentence.

GUI strings also cause problems with grammar, style, and terminology checking. Sometimes these problems involve very basic issues. For example, GUI strings pose difficulties for identifying part of speech (POS), which is fundamental to the proper functioning of grammar rules. The grammar rule governing subject-verb agreement incorrectly triggered because of POS issues with GUI strings. As shown in the previous example, GUI strings can include verbs used in unusual positions - such as "View" following the article "the", or "Edit" following a verb. In both cases, the software may mistakenly identify the verbs as nouns. Ideally, GUI strings should be given only one POS assignment.

The acrolinx software also extracts term candidates from data. Here, GUI strings decreased the precision of the software significantly. The tool often suggested GUI strings - in a truncated form or with the following GUI object as part of the candidate term. However, GUI strings should be completely excluded from the term extraction.

From the translation or localization perspective, GUI strings have to be translated according to the translation found in a software glossary, as the following example illustrates:

To start the program, click Start.

The capitalized word "Start" should be translated with the phonetic Katakana characters "スタート" in Japanese, which are used to render foreign or technical words. Instead, it receives the more complicated Kanji translation "開始". On the other hand, if the MT system is not fine-tuned to handle GUI strings, the resulting translation output might be unusable, as shown in the following example in which a GUI string receives a literal translation in French:

Enter information in the Connect to a Media Server dialog box.

Écrivez l'information dans le connecter dans une boîte de dialogue de serveur multimedia.

Bottom line: All of these problems occur because the software tool does not identify the GUI strings or treat them as single entities. However, GUI strings - like UI strings more broadly - need to be handled as single units, whatever their length is. This approach presupposes that checking technologies can identify them as such.

TREATING GUI STRINGS AS SINGLE ENTITIES

At a gut level, it may seem strange to treat a GUI string like "Connect to a Media Server" as a single unit, since it clearly contains multiple words. But as we have seen, if we treat such strings as multi-word entities we encounter serious problems in the authoring, checking, and translating processes.

Two factors influence the ability of software to deal with GUI strings as single entities. First, **XML tagging** can help software identify GUI strings according to their contexts in XML documents. In the typical information development workflow described earlier, writers must ensure that their XML topics comply with a subset of the DocBook XML DTD. The full DocBook DTD contains various tags related to GUI strings: `guibutton`, `guiicon`, `guilabel`, `guimenu`, `guimenuitem`, and `guisubmenu`. Yet to date, the only tag in use has been `<guimenuitem>`, which sets off some of the GUI strings that end users interact with. The following example shows how a `guimenuitem` is rendered:

Click `<guimenuitem>` Yes `</guimenuitem>` to install the Enterprise edition.

■ Click Yes to install the Enterprise edition.

When this <guimenuitem> tag is present in the source text, both the acrolinx tool and an MT engine can be configured to handle it in a specific manner.

Second, **token classification** helps ensure that GUI strings are interpreted as single units. Linguistic software typically sorts text into so-called “tokens.” Usually tokens are words, but it can be useful to treat single units of meaning as single tokens, for example, in the case of “vice versa”, or for GUI strings. The software sorts tokens according to class, for example “Capital” belongs to the class “FirstCapitalWord” due to its initial capitalization, or “123” belongs to the token class “Number.” The software segments text into sentences and applies spelling, style, grammar, and terminology rules based on the token classes. It is therefore important to define GUI strings as one token so that they do not cause alarms for issues like sentence length or subject-verb agreement, which we discussed earlier.

GUI strings can be extracted from source documentation with a basic XPath expression (such as //guimenuitem or //guilabel if a DocBook DTD is used). However, untagged GUI items will not be extracted and, as a result, the CL checker will include them as sentence elements. A two-pronged strategy was pursued to deal with this issue.

A CUSTOMIZED SOLUTION BASED ON ACROLINX TECHNOLOGY

acrolinx software enables token classes to be defined based on what immediately precedes and follows a token. The goal was to create a special token class for GUI strings with an eye towards treating them as single units. This approach presupposed a relatively consistent use of GUI strings and a good knowledge of the syntactic context in which they might occur.

First, we decided to create a token class for GUI strings even where XML markup already existed, for the tag <guimenuitem> which discussed above. Doing so ensured that the tags received treatment at the core of the system. The tag <guimenuitem> is translated into the token “GuiMenuitem”, for example, which ensures that the GUI string is treated as a single entity.

The more significant challenge was to identify GUI strings where no markup existed. Here, an innovative approach was developed. To take one example: We created the token class name “GUIString” along with a rule that extracted GUI strings based on predictions about patterns in which they occur. In a first round of token classification, the sentence “Under Options, click OK” was given the following token classes:

Under	FirstCapitalWord
Options	FirstCapitalWord
,	Comma
click	LowerWord
OK	CapitalWord

But “Options” is actually a GUI string. The rule specified that “FirstCapitalWord” tokens preceded by “Under” and followed by a comma belong to the token class “GUIString.” As a result, “Options” is correctly categorized as a “GUIString” before CL checking and MT.

We have argued that companies, especially those operating on a global scale, need to deal with GUI strings because they profoundly affect the end-user experience. And, for a broad range of processes - authoring, CL checking, and translation - GUI strings present significant challenges. acrolinx centers its approach on a central insight: GUI strings need to be treated as single entities. Based on this insight, we developed two strategies for finding GUI strings and classifying them as units. The second strategy is particularly innovative because it attempts to deal with GUI strings that have no identifying markup. It offers the exciting possibility of enabling businesses to find their GUI strings even when working with less structured content.

ABOUT ACROLINX

acrolinx is market leader in quality assurance tools for professional information developers. These tools help companies worldwide to maintain their corporate image, address compliance issues, improve quality, and control document production and localization costs. Its flagship product, acrolinx IQ™, is used internationally by thousands of customers in a variety of industries, including software, automotive, life sciences, and aerospace. It has been deployed at global enterprises like SAP, Symantec, SAS, Philips, Adobe, Siemens, Motorola, and Bosch.

acrolinx maintains its headquarters in Berlin, Germany with a sales and support subsidiary in North America.

For more information, please visit www.acrolinx.com