Confessions of a Software Localizer

Opinion By
Yann Meersseman

My line of work is helping companies develop processes to improve the production and quality of their translated software. A development manager I recently interviewed presented the technical solution she and her team had imagined. They had developed a tool to scan resource files and extract strings into an Excel spreadsheet. After eliminating duplicates, the sheet was sent out for translation. "This is ideal," she said, "we keep the volumes down, the translators cannot mess up our source and as soon as we get the translations back we can rebuild the software." Proudly she added: "With the special identifiers we generate, we know exactly where each string belongs. so all the modules can be merged into a single spreadsheet and sorted alphabetically!"

"Breaking strings into words and eliminating duplicates," I noted, "would reduce the volume even more." Fearing that my remarks were taken seriously, I quickly added that applying the same technique at the character level would leave only 26 letters to translate.

Recipe for Trouble

The incident would be funny if the idea that software translation is all about replacing text wasn't as widespread among engineers. Needless to say, the process just described is a recipe for trouble. It passes over the importance of context, fails to recognize that a user interface includes more than words, and sets the stage for long and inefficient iterations between quality control and corrections. The first misunderstanding is that the sole knowledge of the source and target languages is sufficient to translate. This assumption completely ignores the predominance of subject matter and context. The meaning of the word *file* depends whether you are an office worker or a carpenter. You cannot select the right French word for *discount* (remise, rabais, escompte, to name but three possibilities) without the context in which the term is used. For those unfamiliar with your applications, the source language is not English but gibberish. Sending translators a list of labels, captions, messages, and loose strings is like cutting up a novel into individual sentences and translate them in random order. I'm sure you can imagine the results.

The second misunderstanding is that software translation is all about strings and words. What has to be translated is the user *interface*, whose elements have other attributes beside text, such as shape, size, and position. The proportions and spatial relations between these components are intrinsic qualities of the interface and constitute a *graphical language* intended to please the eye. Treating the written separate from the pictorial produces unsatisfactory results. The one-size-fits-all-languages approach is unrealistic, puts the burden on the developers, and leads to awkward interfaces in all languages, including the original. The stay-within-the-limits-of-English approach is equally unrealistic, puts the burden on the translators and leads to abbreviations and poor language in the translated versions. The linguistic must not be disassociated from the graphical: translators must process both as a unit.

Finally, this process violates the principle of immediate feedback. Translators are blind if they cannot observe the results of their work from an end-user's point-of-view. The spreadsheet or string-list method makes translators dependent on their customers for rebuilding the product in every validation cycle. In most situations, the high word volumes, the multiplication of languages, and the physical distance between the parties make this strategy highly inefficient and error-prone.

Nothing Basic About It...

An appalling fact is that today's supreme influencer of engineering thought largely contributes to the dissemination of the wrong ideas. If you browse through the Programmer's Guide shipped with Visual Basic 5.0, you will notice that chapter 16 is dedicated to international issues, which in itself is a laudable effort. The content however encourages exactly the text-isolation techniques that should be banned.

The actual recommendations presented in this guide are not the issue. The current Visual Basic environment is unfortunately not translation-friendly, and string isolation is indeed the prevailing solution to compensate for this deficiency. No, the damaging quality of the chapter's content lies in its pretension to introduce the key concepts for developing international software, thus sending out the wrong message to the planet's developers. It would require a humility difficult to imagine from a giant, but what the chapter should say is: "Sorry, we might be the world's biggest software company, but most of our managers and designers don't quite understand translation, and the ones who do are not getting much attention. For now, we are stuck with this surrogate solution, but maybe we'll do better next time." As insignificant as it sounds, I know at least one company tempted to switch to Visual Basic's new speed and comfort, but which is thinking twice about the step backward it would represent in its international-development processes.

Keeping Your Code Above Water

Note that the guidelines in chapter 16 mention that string growth should be taken into account during the design of an "international-aware" user interface. To this effect, the text includes a table showing the *average growth for translated strings*, ranging from 10 to 100 percent depending on the length of the English original. I invite the author of these lines to play a game of "don't get your books wet." The players have to carry printed copies of their publications and keep them dry while wading across a river that has an *average depth* of four feet.

I don't know if designers will ever bother to develop user interfaces that are truly disconnected from the underlying code. In such an environment, the user would have the ability to change on the fly any textual or cosmetic attribute without consequence for the application's functionality. I do know, however, that such a technology would be the welcome start of a new era for software translation.

If you browse through the Programmer's Guide shipped with Visual Basic 5.0, you will notice that chapter 16 is dedicated to international issues, which in itself is a laudable effort. The content however encourages exactly the text-isolation techniques that should be banned.

On the calculus of localization:

"Solving" the screen geometry problem by dictating average text expansion is like having to walk across a river with an average depth of four feet.

Yann Meersseman helps software publishers organize their resources to address international markets. email him at us038518@mindspring.com.