Introduction

Recently the mid-sized, regional health care company where I work revamped its localization tools and processes. The story of how it accomplished this and how the improvements that it made have resulted in greater productivity is an interesting one. The way that it achieved this may be of interest to other American health care organizations that are expanding overseas.

The products that benefited from this project differed somewhat from each other – one is a client-server Disease Management (DM) telemedicine platform and the other a web-based behavior modification application. As you might expect, the localization challenges were different for each product. As it happened, however, a careful analysis of the existing architectures and prior localization efforts helped an in-house team to assemble requirements, obtain approval from upper management, implement a solution, and then use it successfully.

In tandem with this project, the company set about defining how any localization project should get done. The translation process that resulted defines exactly what different team members and customers must do in order to prepare, implement and conclude any localization project. It also established guidelines for partner companies such as translation agencies and software development firms.

Some Background

The company began selling its products several years ago. Its first overseas customer was a German medical insurance company. Due to the fact that this client did not need multilingual functionality – its staff and all of its insured were assumed to speak and read German – the decision was made to translate the products without first internationalizing them. Since healthcare systems are usually country-specific, it was thought that every future customer would be based in one single country with one dominant language.

The development team created a utility that reflected the unilingual nature of the products. It gathered the English language strings from sources such as database tables, RESX files – in the case of the web application - and DLLs – in the case of the Disease Management application - and stored them in its own database. This tool then exported to different Excel files the English language strings requiring translation. Typically, these files were sent to an agency for translation. On their return, the files – now with translated strings - would be uploaded into the translation utility and new language packs would be created.

This rather simple localization process worked fairly well for a while. Since the telemedicine product sold in Germany was seen by only its own employees and not by the public at large, review of the translated UI could take place at a relaxed pace. Most of the strings requiring translation consisted of two or three words, so the structure of the Excel files did not pose any big problems for the translators. The web application at that time was not very complex and was considered as an add-on to the Disease Management product that was used by nurses to call patients and monitor their health. Any changes to the translated strings that were suggested by the customer could be made in the Excel files that were subsequently uploaded. Language packs containing the updated translations were deployed as part of service packs.

As the company's products evolved, however, the limitations of this process and of the utility at its center began to be noticed. 60000 words of HTML formatted text was added to the database in support of the behavior change web application. Translators on one project complained about having to insert HTML tags in Excel cells that sometimes contained three or four paragraphs. In addition, several newer database tables had three or more columns with text requiring translations and the resulting Excel files were not easy to translate. The import-export tool itself, it was discovered, did not always import HTML formatted text translated database strings successfully and I often had to load the translated database strings via SQL scripts, bypassing the tool altogether. Sometimes it was difficult to learn which Excel file contained the string requiring modification. In addition, the tool had a complicated UI that required extensive training and was accessible only by the release engineer who was busy with other tasks.

A Turning Point

As the company expanded its sales efforts, the problems with the translation process became known beyond the localization team. The head of development, for example, was concerned that the release engineer was spending too much time on tasks related to translation. Project managers wondered why it took two people to fix a language bug – one to find and modify the string in the Excel file and another to load it into the translation utility. But it was a translation vendor's inability to supply a quote that marked the start of a translation initiative.

Shortly thereafter the head of development asked for user requirements for a new translation utility. In the meetings that followed, one of the main requirements to emerge was the export and import of strings in XML and not in Excel. In part this was due to comments made by the translation agency, but there were other reasons as well. XML offered several advantages. First, it was much easier to load into SDL Trados. Second, the customer could review and modify the translated text with an XML editor and not with Excel. Third, text that was spread across several columns of an Excel file would be displayed in linear fashion in XML, making reading and translating easier.

The user requirements that I collected included other items as well, some practical, others less so. Making the translation utility accessible to more users was one suggestion that was retained. The tool would also support modularization i.e. searching for the files and database tables associated with a particular functionality, to make it easier to find exactly the right string to be changed. In addition, the utility had to support the changes to the database schema that a future multilingual capability would require.

A request for the real-time 'live' editing of translated UI strings was also made but ultimately rejected. Such a feature would allow the customer to modify translations rapidly, but would complicate the creation of service packs and other releases. There was no easy way to update development source code in response to this kind of editing and so any RESX files modified by the customer through 'live' editing could be overwritten by a service pack. In addition, even minor changes to the screens of a customer installation could lead to problems in testing and bug tracking.

A Blessing in Disguise

Work on the new utility was delayed during the implementation of the web-based product at a new customer site. Certain events related to this effort, ironically enough, helped speed up development of the tool and the redefinition of the translation process.

Unlike earlier customer projects, this one had a much higher public profile. Parts of the behavior change web application would be visible to the public at large and all of the application would be accessible to tens of thousands of policy subscribers. Since the customer's image was very much on the line with what was an unknown technology in this market, an extensive review of the translated UI was a necessity.

Given this situation, as you might expect, the customer demanded substantial changes to the translated web application. Making changes to the static text took more time than planned. Some developers had neglected to put English languages strings in RESX files where the translation tool could capture them. This resulted in coding changes that required code builds before an actual translation could be uploaded via the translation tool.

As challenging as this was, entering changes to HTML text in database tables proved a bigger problem. The customer was ill-prepared to modify the contents of Excel files, and copied the translated text into Word, deleting all formatting in the process. This entailed the manual reformatting of HTML text on each occasion when the customer made changes to the translation. Since several different groups in the customer organization reviewed and modified this text, reformatting the translation into HTML became a fulltime job for one staff member for a time.

The labor-intensive character of this translation process, the related inadequacies of the existing tool and the lengthy of time required to change UI language strings did not go unnoticed by upper management. Once the customer approved the translation and the wellness web application went live, the push to complete the tool and update the localization process became a company-wide goal.

A New Way of Translating a UI

A thorough after action review took place following the go live of the customer web site. Representatives from development, project management, the in-country team and the translation coordinator, were all consulted on what worked well and what improvements were needed. The review confirmed, as expected, that the turnaround required to fix a translation change request was too long. The review also highlighted the need to 1.) have every engineer put strings in resource files religiously, 2.) hire and train in-country personnel prior to a localization project, and 3.) have a local employee review every version of the translated UI.

There were other lessons learned as well. Moving forward, translations would be performed only by professionals with domain knowledge. While most of the customer-facing text had been translated by an agency, some of the text had been translated by a native speaking employee who volunteered to help. (Fortunately, a local employee was able to review and modify this translation.) It was also learned that a local employee be in near constant engagement with the customer concerning translation review and have a backup plan ready when a customer wants to modify translations in their own way.

At the same time as this, with requirements in hand and a customer deployment behind it, the development team began work on the new translation utility. In the months that followed, the developers prototyped a tool that anyone could use to export both XML and Excel files. This new utility was shown to process strings from database tables, RESX files and DLL files effectively. In addition, a translation vendor confirmed that XML output files were significantly easier to process in SDL Trados than the Excels.

A new translation process was defined around the new tool, taking into account the lessons learned from the after action review. The release engineer was all but removed from the translation process and the translation coordinator assumed responsibility for translation exports and imports. In-country employees with knowledge of the local language were to review the translations before and after they were loaded into the product. Each version of the translated UI would have to pass a quick sanity check before a customer would be allowed to view it.

This new process was communicated to all participants in the translation process, including the translation vendors. One agency modified its workflow tool to incorporate the in-country employee review into its own translation process, using a web-based workflow management system. Language defect fixing was speeded up dramatically, in large part due to the new utility but also due to the way engineers routinely put translatable strings in resource files.

As for the review and modification of certain translated text, the new translation process admittedly left a gap. The HTML content - all 60000 words' worth – required special handling during review and modification if the formatting were to be preserved. After some investigation, a workaround was found. With minimal training, the customer could use the WYSIWYG XML Serna editor to modify the translated text while at the same time preserving the formatting. Preparing the modified HTML formatted translations for uploading now takes minutes and not hours and can be done whenever the customer needs to make changes.

Looking Forward

As the world's population ages, U.S. based wellness and health care companies will find themselves exporting their technology to all parts of the world. As they do this, they will need to adapt their products to local markets and cultures, as software, automotive and other companies have already done. For companies that are selling wellness and disease management programs, the translation challenges are likely to be greater than those facing other industries. Any company in this field should expect intense translation reviews from medical, usability and marketing experts who are affiliated with the foreign customer.

Having the tools and processes that will let a company rapidly integrate customer comments is therefore an essential part of a successful localization strategy. Through trial and error, one mid-sized American company was able to develop an effective localization process by determining what its needs were and by creating the tools and the system to meet them. Before venturing to sell its solution abroad, any other health care company should do the same.