

# Testing for Globalization – Experience from IBM

Jerrold Landau, M. A. Sc. – Function Verification Test and Globalization Leader –  
IBM® WebSphere® Business Integration Tooling of the IBM Toronto Lab

This paper is the summary of a presentation prepared for the PSQT & PSTT East Conference, held at Washington Dulles Airport on March 23-24, 2004.

In the current era of the global marketplace, it is increasingly important to ensure that software products are sensitive to the linguistic and cultural expectations of users around the world. Non-English-speaking customers expect that computer programs will not fail on computers that are equipped with an operating system in their native language, and they often prefer to have the program present itself in their own native language, to the extent that this is feasible. Those software companies that focus on ensuring the acceptability of their products in the global marketplace will surely gain more market share than those that do not do so.

Globalization is the term used to describe the process of producing software that can be run independent of its geographical and cultural environment. Localization is the term used to describe the process of customizing the globalized software for a specific environment. For simplicity, the term ‘globalization’ will be used to describe both concepts, for in the broadest sense of the term, software is not truly globalized unless it is localized as well.

Globalization is, of course, a very broad topic, and a short presentation of this nature can merely scratch the surface. This presentation will provide a high-level overview, introducing some concepts, with a focus on software globalization as it relates to test.

Many aspects must be considered when producing globalized software. These include, but are not limited to, the following issues:

- Cultural sensitivity icons and display: For example, the rural mailbox is not known outside of North America. Various hand gestures may mean one thing in one country, and something entirely different, perhaps even obscene, in another country. Therefore, icons that utilize such hand symbols must be carefully analyzed for appropriateness. Various colors signify different moods and feelings in different cultures. One interesting example is that on an Arabic keyboard, the asterisk has five points rather than six, to avoid any similarity with the Star of David (a symbol of Judaism).

- Sensitivity to the English vocabulary: For the most part, the text that accompanies software (either documentation or internal to the product – more on that later) will be first written in English. It may later be translated into other languages. Many words have ambiguous meanings, which will confuse translators if and when the time comes to translate the product. Some words may be objectionable in certain languages and contexts – for example, even in English, words such as *execute* and *abort* should be avoided for general audiences. One interesting example is that calling Taiwan a country is politically objectionable in The People’s Republic of China, which considers that a region instead. It is, in fact, a stop ship issue.
- Date and time formatting: Different countries have different standards for representing dates and times: For example, does 03/04/02 refer to March 4 of 2002, April 3 of 2002, or perhaps even April 2 of 2003? The internal date and time handling in a properly globalized computer program must be independent of any specific convention.
- Currency handling: For example, handling of multiple currencies in an online shop or catalog.
- Paper sizes for printing: Standard paper sizes can vary from country to country.
- Address and telephone number formatting: For example, some countries do not have states, so making ‘State’ a mandatory field in an address can cause problems.
- Some languages tend to be more verbose than others. For example, a French sentence is, on average, almost 1.5 times as long as the equivalent English sentence. When designing GUIs that will be translated, this type of discrepancy must be taken into account; otherwise, text might overflow its intended location. Internal buffers must be long enough to accommodate translated versions of their text.

One of the most significant features affecting the globalization of a software product is the management of text strings. Computer programs accept input of text, produce output of text, manipulate text, use text to draw GUIs, and transmit text to and from other programs (such as a database program upon which a higher-level program rests). These text strings might be in the form of non-English languages, and some of these languages might consist partly or fully of non-English characters. A globalized computer program must be able to accommodate all character types and alphabet that might be presented to it.

Two specialized text types that must be considered in a globalization effort are DBCS and BiDi . DBCS (Double Byte Character Set) is used for languages such a Chinese, Korean, and Japanese, where a single byte is not sufficient to represent all characters in the alphabet. BiDi (Bi-Directional) languages include Hebrew and Arabic, where text is

written from right to left rather than from left to right. In both DBCS and BiDi language programs, there must be the capability to handle mixed text strings, with snippets of normal English style single-byte, right-to-left text, interspersed between snippets of specialized text.

Clearly, testing for all of the above globalization issues poses a significant testing challenge. There are two stages of globalization testing: testing for internationalization, and translation verification testing (TVT). Testing for internationalization is designed to prove that a product is capable of functioning in a globalized fashion, and is independent of any particular localization. TVT, as its name implies, verifies that a given translated version of the product works as expected.

Testing for internationalization does not form a test phase on its own, but rather should be part of both function test and system test. This test should attempt to run the product on several computers, each installed with an operating system of a different language. Most commonly, one DBCS and one BiDi operating systems are chosen. It is estimated that tests for globalization enablement should take up about 10% of a testing effort.

It is impossible to outline a procedure for globalization enablement testing that is applicable for every situation; however, some general guidelines can be presented. To test for internationalization, a set of test cases are run to demonstrate that the aforementioned globalization issues relevant to the specific application, such as date and time handling, currency handling, function correctly. These tests should also provide input to the program using some non-English characters, and solicit output that contains non-English characters. If text strings are passed from the program to other programs, the tests should be designed to ensure that such text manipulation occurs with foreign language text strings. If the program is designed to accept a 'locale' (set of localization standards for a particular localization instance) as a configuration point, tests should be done with the configured locale equivalent to the default locale of the machine, as well as with the configured locale different from the default locale of the machine. If the program will be run in a client-server mode, it is important to try a situation where the client and server are of different languages, to ensure that the appropriate language is used for each end of the application.

In addition to the above tests, if the program includes the mechanics of handling a multilanguage translation (over and above being able to handle input, processing and output of multilanguage data), the internal mechanics of translation selection must be tested. As the translation might not be available at this time, this can be accomplished by producing a 'mock' translation, and ensuring that the mock translation is used with the appropriate locale selected.

The above paragraphs represent a selection of issues that should be taken care of in a national language support internationalization test. A globalization test plan, designed for the particular needs of each product, will have to be set up to ensure that

issues appropriate to the specific product are handled. These tests will be divided between functional test and system test, as appropriate.

Translation Verification Test (TVT) involves testing that the software product works correctly with a specific language translation of the product. The translation will include program messages (for example, error and warning messages), as well as text to be displayed on a GUI. In a properly globalized product, this text will not be embedded in the program itself, but will rather be separated into separate files (in Java™, for example, these will be properties files). When the program is run, text from the appropriate language file will be selected by the internal mechanism of the program to supply the text used for program messages and GUIs. This selection will be determined by the current locale in force. When running a TVT, the product will be installed on a computer with an operating system of the specific language. The program will be run, and each GUI will be examined for the appropriateness of the text. All input and output fields will be exercised. Ideally, every program message should be driven out during the TVT test. Realistically, this may not be feasible, and some reasonable compromises will be made.

The TVT for a specific language must be run by a person who is familiar with the language in question. TVT for a complex product with many GUIs can be extremely time-consuming and expensive, as each GUI must be examined in context. The products that I am currently involved with are designed as components to be used by developers to create end-user products, rather than as end-user products themselves. As such, they tend to have very few if any GUIs. Thus, the translation for these products was restricted to the program messages. We have been able to simplify the TVT significantly by focusing only on the integrity of the message files. In our case, the need to issue the messages within the context of a GUI was obviated, since the product provides no GUI to instantiate the messages (this being the responsibility of the user of the components). This simplified the TVT effort significantly, since all that was needed was a verification that the translated error messages are valid and usable by the components.

This presentation provides a very general introduction to the concepts involved in testing a software product for globalization. As is evident, globalization is a very broad and interesting topic. It offers the software developer and tester a chance to explore issues that affect different cultures and languages, and perhaps even become familiar with the rudiments of different alphabets. Needless to say, software globalization is an up-and-coming field, as the software marketplace expands to encompass the entire world. Those interested in a full understanding of the topic should find opportunities to read further and take courses on this fascinating subject.

## **Copyright and Trademarks**

© Copyright International Business Machines Corporation, 2003. All rights reserved.

IBM and WebSphere are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based logos are trademarks of Sun Microsystems in the United States, other countries, or both.