

# THE ART OF MULTILINGUAL SOFTWARE TESTING

By Ian Henderson

What is software testing? What it is not is something that is bolted on at the end of a project. Testing of software should begin on Day One of development, and it should be viewed as an on-going process that does not end until release to all of one's global markets. There are many stages to testing, each complementing the others. Viewed as an on-going progression through the development and localization lifecycle, testing is critical to reliability, releasing quality product to market on-time and maximizing global revenue.

## Defining QA/Testing

QA and testing are vague terms. Unfortunately, a lack of understanding of proper QA and testing by a vendor can be hidden behind these terms without further clarification.

When companies hire a localization or testing company and request QA/testing of multilingual software, they are more likely than not receiving bids running the full spectrum of tasks, including preventative testing, test scripting, smoke testing, localization testing, regression testing, debugging and functional testing. Without documenting exactly what is meant by the terms "QA" and "Testing", one really cannot be confident of what services and scope of testing one will receive. Without knowing this, ironically, one really has no way to gauge the "quality" of the final output. Therefore, when requesting bids from localization companies, it would behoove the requester either to define the exact QA/testing tasks and scope or to take the time to identify the different levels of service actually being offered among the vendors.

## Test Scripts

Test scripts are essential to a testing project. Without these scripts, it would be impossible to define the quality that is being assured by the testing process.

The testing that takes place after localization is very different from that undertaken during the development of the core product. This means that each type of testing requires a different set of test scripts. Functional test scripts may include connecting to as many different third-party products as possible even though the UI screens may remain substantially the same, whereas localization testing concentrates on generating as many of the UI screens as possible across all languages.

## Testing During Development

Ideally, software developers have localization in mind during development. Internationalization is the process of incorporating support into a product for all the global market needs that a product will require. A properly internationalized product will work in any locale, regardless of language or data format. Internationalization should be integrated with development, so much so that in essence, internationalization cannot be viewed separately from development itself.

Correspondingly, it is just as important that internationalization testing be integrated with development, to verify that internationalization has been achieved and that an application is "world-ready". It is much faster, easier and cost-effective to correct an internationalization flaw during development than reactively during QA of a localized product. Internationalization testing is the process of verifying that an English language product (or other source language product, as the case may be) functions properly with data in other languages, and may include testing of the following:

- Installation
- Localizable strings
- Testing foreign characters
- Testing text input
- Multibyte support, for example Japanese or Korean
- BiDi support, for example Arabic
- Keyboards
- Sorting
- Searching
- Accelerator keys and mnemonics
- Databases
- Printing
- OS-driven regional settings
- Sound

## Localization-readiness Testing

Localization-readiness testing is the preventative process of verifying that global requirements have been met through design and development, and that an application is in a state to be localized efficiently. Preventative testing should occur prior to localization in order to identify any aspect of the source product that may cause problems in localization.

There are two general types of localization-readiness testing. These are source file testing and pseudo-translation testing.

**Source file testing** does not involve any testing of the target language software, but it does involve verification that the source language product can operate without any problems in the target environment. It is conducted after the product is believed to be internationalized. Source file testing includes verification of the source language product build on a clean machine, verification that the product runs properly on the target OS, verification that the product can accept data input without corruption, etc. This type of testing is critical in identifying problems in the source and distinguishing these from any localization-related problems. This distinction greatly expedites the debugging process.

**Pseudo-translation testing** is a remarkably time- and cost-effective method for verifying that an application is localization-ready. This type of testing involves first the pseudo-translation of the source language product, and then running tests against the pseudo-translated product to verify that the pseudo-translation does not break the product (the most serious kind of bug to address) and to uncover other internationalization issues such as hard coding, etc. Pseudo-translation testing can also be used to stress test internationalization issues such as projected text expansion resulting from translation, concatenation of strings, and concatenation of dynamic screens.

## Localization Testing

Localization testing generally follows the progression below once translation has taken place:

- Pre-testing verifications
- Engineering testing
- Engineering debugging
- Linguistic testing
- Linguistic debugging
- Regression testing

Pre-testing verifications are preventative, reducing bug volume, averting potential confusion and delay, and streamlining the entire testing process. The verifications enable one to identify, document and correct all core functional bugs and distinguish these from any actual localization bugs. For example, what level of internationalization testing has been conducted on the source product? What are the known bugs in the source language application that should be ignored (=not reported)? Are there any third-party product bugs that are already known?

Ideally, all generic issues should be identified and corrected prior to linguistic testing. This is an efficient approach to localization testing, where engineering testers eliminate core problems in the application prior to deploying multilingual testing across 5, 10 or 20 languages. By trying to skip the preliminary generic testing step, one runs the risk of encountering the same generic bugs across 20 languages, resulting in a massive duplication of effort in testing, reporting and reviewing bugs. As a general rule, the people who report bugs (testers) should be different from the people who fix bugs (engineers). It is also important that only a tester can close a bug, not an engineer.

Linguistic testing of the running application provides a linguistic tester the opportunity to view the translations fully within context as an end user would. Areas of concern would include:

- Linguistic clarity
- Linguistic consistency
- Compliance with glossaries
- Compliance with linguistic style guides

## Functional Testing

Functional testing is a complement to localization testing and may include the following types of testing:

- System testing
- Integration testing
- Performance testing
- Stress testing
- Security testing

Basically, functional testing of multilingual software is similar to that of the source language application. Test scripts need to be customized, but rather than testing for cosmetic errors in the GUI, one is testing for flaws in the underlying code and capabilities of the product. If third-party products are involved, it is critical to identify any known bugs in the third-party product support and to define the language versions of such products to be tested as well.

## Ingredients for Successful Software Testing

Two keys to successful multilingual software testing are scalability and superior project management. Testing global software means testing teams grow by the number of target language versions. It is not infrequent that teams of 50 testers or more are required, working across multiple combinations of platforms and browsers, with varying testing scopes depending on platforms and/or languages. Without a rapidly scalable model and rock solid project management, a global software testing effort could potentially result in a confusing mess.

## Conclusion

It is essential to view QA and testing as a continual process throughout the development and localization lifecycle. The process is fluid and involves a full spectrum of components and milestones. "QA and Testing" are generic terms and invite a great deal of risk to project planning unless defined clearly and accurately, meeting the specific needs of the software publisher. With a clearly documented QA and testing plan, though, one can increase quality, enhance the user's experience, reduce global technical support costs and strengthen global brand equity.

*Ian Henderson is director of engineering at Rubric, a leading localization and translation services company. Rubric localizes and tests software in all major languages, including multibyte and bi-directional languages. Ian can be reached at [ian.henderson@rubric.com](mailto:ian.henderson@rubric.com).*

Reprinted from November/December 2003 of Software Business Magazine, [www.SoftwareBusinessOnline.com](http://www.SoftwareBusinessOnline.com)  
©Webcom Communications Corp., 7355 E. Orchard Road, Suite 100, Greenwood Village, CO 80111, U.S.A, Phone 720-528-3770