international Software Products

iSP

a division of translations.com

Software localization testing at iSP

This document gives background information on iSP's testing services. It explains what exactly happens during *linguistic testing, (localization) UI testing* and *functional testing* and it explains some of the terminology relating to testing. There are also some recommendations in the document for approaching localization testing.

1. Testing services offered by iSP

We distinguish between three types of software localization testing in our service description depending on what is tested:

1. **Language**: Linguistic test
2. **Layout**: Localization UI test
3. **Functionality**: Functional testing

1.1. Linguistic test

Definition
The linguistic test (or language test) is characterized by the fact that it is performed by a native speaker. The intention of the linguistic test is to carry out a linguistic evaluation of the product from the perspective of the end user. All linguistic testing at iSP is performed by native speakers of the language who are trained as experts in the product.

Items to check during linguistic testing

Context of the text	Linguistic testing checks whether the text is in the right context and reveals any previously unseen problems. Note however, that the linguistic test is not a pure text review activity. Text review should happen much sooner as an integral part of translation. (The sooner a change is implemented, the less costly it is.)
Text concatenations	Concatenation problems sometimes only become clear during

	testing. For example two strings are copied together which the translator did not expect during translation.
Consistency	The linguistic test should apply a comprehensive perspective. Is there for example text in a dialog, which makes reference to another dialog ? Or is there a reference to an item in the localized operating system? It is important to check that these references are consistent.
Layout	The layout aspect overlaps with the Localization UI test done by technical resources, but trained linguistic testers still need to be familiar with UI standards and report related problems. A linguist can detect certain problems (e.g. truncations) better than a technical resource because they can read the foreign language text. Other aspects like consistent layout across languages are better checked by technical resources. As there is overlap between the linguistic and technical resources, it is important to make the proper agreements as otherwise two teams will report the same issues. If possible, plan linguistic and technical testing sequentially.
Code set issues	This overlaps also with the UI test because although Engineers and Testers ensure that the right code set is applied, this should also be confirmed from a linguistic perspective. The aim is to check how characters are displayed ("corrupt characters"), but also includes issues such as alphabetic sorting.
Untranslated text or text in a wrong language	Again, here is an overlap with the UI test because Engineers and Technical testers in the project should make sure no untranslated text or text from another language is used in the application, but this aspect is also an important part of the linguistic test.
Non text elements	These kind of problems are not so common. However, it is good to keep this in mind. Think about the "B" button for Bold for example.

1.2. Localization UI test

Overlap with linguistic testing
The localization UI test verifies the interface of the application, and because linguistic testers (who do the linguistic test) also need to be familiar with UI standards, there is a certain overlap. At iSP, we find it most efficient to plan testing in such a way that technical resources already perform sufficient testing before the linguistic resources start so that the linguists can focus on the truly language related aspects. Specifically for projects with several language localizations, this approach has its efficiency gains.

Scope of the test
The localization UI test checks the layout of the localized application whereby a general rule is that the localized application needs to look the same as the original application. Because of this, the typical test setup includes both versions of the application to test, the

original and the localized side-by-side. While the original application is the baseline for testing, the testers will of course also report clear mistakes in respect to UI guidelines which exist in both localized and original application.

Difference to testing in WYSIWYG editor

Application interfaces can often also be checked in a WYSIWYG editor such as Developer Studio. At first sight you might wonder why there should be yet another test in the live application. However, there may be differences between the appearance of the application UI as it appears in a WYSIWYG editor and the live application - this defines the localization UI test. Nonetheless, it is very efficient to make use of the WYSIWYG editor to already filter out the UI problems which can be seen there.

Items to check during localization UI testing

UI Standards	Generally speaking, the localization UI test makes sure that the localized product looks the same as the original product as baseline. Beyond that, a good general guideline is to refer to developer guidelines for the operating system. The Windows Vista User Experience Guidelines for example include a chapter "Aesthetics" which mentions guidelines like a standard button size of 50 x 14, right alignment of commit buttons and many more.
Code set issues	This overlaps with the linguistic test, so it is a good idea not to plan the linguistic test too soon to avoid duplicate efforts. Already try to filter out these kind of problems prior to compiling, and then make it a part of the localization UI test.
Untranslated text or text in a wrong language	This is another item which overlaps with the linguistic test, so again, it is good to avoid testing this twice. Also this aspect can be already be largely filtered out prior to compiling.
Truncations	This means checking that the text fits in the UI. Non native speakers are not always aware of text not fully displaying, so again, the linguist test serves as a second filter for this problem category. Note, that this test can be partially supported by tools.
Alignment of text	The alignment should theoretically follow the original UI specifications automatically, but in some cases it can differ in the localized version (e.g. text wraps to another line), and sometimes where UI coordinates are leveraged together with the text, an item may appear at an unexpected position.
Access keys and shortcut keys	Both access keys ("hotkeys") and shortcuts are used to execute commands from the keyboard, but they can be easily distinguished: **Access keys** are the underlined characters in a menu or dialog and they are only available in this specific context. They are activated using the Alt key. **Shortcut keys** are usually Ctrl/Shift/Function key combinations and are not context-specific. Whether shortcut keys are localized is not always clear, so it is important to verify. If shortcut keys are localized, it is important to check that references to them are consistent and that they can actually be reached on different types of keyboards.

	Access keys are always localized because they make use of the strings shown in the menus or dialogs and therefore their uniqueness must be tested in the application. Fortunately, their uniqueness can already be tested in the UI WYSIWYG editor (a localization environment like Alchemy Catalyst can test this automatically), but the live test needs to consider where different UI components are combined at runtime in order to test for unforeseen results. Both shortcuts and access keys can be mnemonic, i.e. the letter which is used has an association with the command.

1.3. Functional test

Focus of the test
If an application is internationalized well, functional behavior is normally not at a too high risk for working in a localized product, i.e. the functional test on a localized product can focus on whether the localization process introduced any problems.

Original product is baseline
The functional test verifies if the application does what it is supposed to do. In order to simplify the testing process, functional testing on a localized product will typically take the original application as a baseline.

How to set up a testplan
The functional testing can usually be planned easily by using the original test plans as a basis. Sometimes however, there are cases where no original test plans are available or where a short and reduced functional test is required. In these cases, a good guideline can be a menu walkthrough or referring to the table of contents of the program help. It speaks for itself that the people who carry out the functional test should know the application well.

2. Test management

Testing is managed centrally
Next to performing the linguistic testing, localization UI testing and functional testing, iSP also does the test management. While different testing activities are carried out by different types of resources, test management is a centralized activity, usually carried out by the technical group.

Setting up test plans

Test management involves gathering or setting up test plans, grouping them and scheduling them on the timeline of the project. Sometimes test plans are available from the client and in other cases they are not available. In the latter case it is advisable to evaluate how big the need for documenting test cases is. Sometimes the knowledge of testers is underestimated and valuable time is spent on over-documenting test cases. This consideration should of course not lead to skipping the documentation altogether.

Document test case requirements
The test documentation should also describe test requirements such as hardware and operating systems, but also other programs, tools, settings and test files. Don't underestimate how much time you can save by providing the right test files, especially in multi language projects.

Setting up the test infrastructure
Setting up the actual infrastructure in the test lab is also part of the test management activity. This includes the computers with their operating systems and any other required software. Setting up the infrastructure needs to include a solution for starting test cases with a clean environment. Hard disk cloning software like Ghost or emulation software like VMWare can help achieving this.

The bugtracking system
Very often localization services providers test localized software for clients who test the original software themselves. As problems (and their solutions) are often repeated from the original version to the localized versions, it is advisable to share the same system. The system should enable the logging of localization specific information and have fields for entering the application and platform languages and other localization aspects.

Working with bugs
It is important to train the testing team on the requirements for logging bugs and on using the bug tracking system correctly. The typical scenario for using a bug tracking system is that a tester logs a bug, an engineer accesses the bug tracking system and fixes the bug, and the same tester accesses the system again and declares that the bug is closed (or returns the bug to the engineer if not). Finally, another activity of test management is the monitoring of bugs in order to oversee trends during the project (for example number of new bugs per day and number of bugs fixed on a day).

3. More terminology

The following is a list of terms which have not been explained yet in the previous sections.

Quality assurance	*Quality Assurance* is a broader concept than testing. Testing the product is an analytical activity which verifies expected quality of the product. *Quality assurance* refers to the complete development process: "SQA [Software Quality Assurance] encompasses the entire software development process, which may include processes such as reviewing requirements documents, source code control, code reviews, change management, configuration management, release management and of course, software testing." (www.wikipedia.org)
Internationalization testing/localization readiness testing	The testing activities mentioned in this document are about testing a localized product. Internationalization testing happens sooner and is integrated into the development of the original product. It uncovers problems which can be expected in the localization process in order to eliminate them at the source.
Pseudo Localization testing	*Pseudo Localization Testing* resembles localization testing because it is carried out on a compiled product which uses dummy strings in place of the original strings (usually strings are made a certain percentage longer and extended characters are inserted), but in terms of the development cycle, pseudo localization is an activity which happens before localization because its idea is to filter out problems before starting localization. In this sense it can be seen as an activity of internationalization testing.
Online documentation testing	*Online documentation testing* refers to help and other types of online documentation, rather than software. There are special tools and techniques used, so online documentation testing is usually carried out by online documentation specialists, and you can typically do a good deal of testing outside the software, what is also easier for planning purposes. Some aspects like context-sensitivity or consistency between software and online documentation still need to be checked with the software.
Regression testing	*Regression testing* means to test a set of features and functionality which must stay the same from update to update. The challenge is to find efficient techniques for regression testing in order to focus on the new features. As regression testing often uses the same test cases from update to update it is a good candidate for automation. If there were problems or fixes to problems in one version, regression testing makes sure that these are re-checked in the next version.
Smoke test	*Smoke test* means to test the bare minimum in an application, only to find out whether the application is good enough for further testing or not.
Automated testing	*Automated testing* means using scripts in order to carry out testing activities. For testing (localized) interfaces, automated testing calls up certain parts of the UI by invoking internal program commands or simply by capture and replay.
Black box, white box testing	*Black box testing* means testing a program without insider knowledge about the code and internal working of a program. *White box testing* is the opposite and means testing a program by making use of the knowledge of its internal structure. All localization testing activities are usually black box tests.

Performance testing	*Performance testing* is a test type with the focus on how quickly an application does what it should do. It measures certain performance factors and compares this to the expectation. You can also put the application to a test under very heavy conditions (run many programs simultaneously, heavy network traffic etc.) and test if it still performs satisfactorily (stress testing).
Installation testing	During *installation testing* testers install and de-install the application. Installation testing is usually carried out on a clean system where only the OS is installed. This is to prevent that an application depends implicitly on another program which is present on many but not all systems (e.g. MS Office). Then during de-installation, testers verify that all components are removed and that installers of different versions work well together.
Integration testing	*Integration testing* means putting the individual modules of an application together and testing their interaction. In a bit broader sense, integration testing is also testing how an application interacts with other programs and its environment in general. From a UI layout perspective, it is important to test the integration of UI items into other UIs.
System testing	*System testing* focuses on how the application behaves in its operating system. Testers for example need to check for certain conditions such as a missing network connection.
Acceptance testing	*Acceptance testing* refers to the producer / developer - client relationship. It takes place after the developer releases the program, and and must then be approved by the client.

4. Recommendations

The following list contains some recommendations for a client/vendor scenario for outsourcing localization testing.

Establish the scope well
Make solid arrangements about what should be tested, how the scope is defined, and when testing is satisfactorily completed.

Share information between core test team and localization vendor
Share test plans, use cases, example scenarios, test files, settings files with the localization vendor. This can save a lot of time (and cost)!

Train testers on the product
Establish a good compromise between too little and too much training, but there should be a sound basis of familiarity with the product, its concepts and knowing how to use its basic features.

Warm up and allow time for setup
This is an extension of the previous item. Next to training, check that the infrastructure is well set up, that all required items exist, familiarization with known issues, problem areas etc.

Ensure scalability
Good planning and estimating the scope is important, but scalability is equally important. The vendor needs to have the capacity to scale the team up and down during the project. Discuss the bandwidth and lead times, and how expansion is realized!

Make sure the UI has been checked in the source files
It is good practice to use a tool set to verify the UI from a WYSIWYG editor. iSP recommends Alchemy Catalyst which has very elaborate QA features to check for truncations, overlapping items, conflicting access keys and more. Releasing the application to test without making use of this possibility would be very inefficient as it usually costs a lot more time to find and fix these problems later.

Establish responsibilities and the workflow well
This aspect is very closely related with the bugtracking system as a common platform for defining the workflow. Who has which access rights, who may close bugs, and what agreements are made about timing of fixes after bugs are found?

Determine the UI freeze milestone
The UI freeze milestone usually defines the point in the localization schedule where the UI is considered final and references in documentation may be finalized. It usually also

means that screenshots can be finalized. It relates to testing insofar as the UI-relevant testing needs to be closed by then.

Scheduling advice
A good way of scheduling is to start with UI test (technical resources), fix the UI issues, then perform the Linguistic Test, enter linguistic changes, and finally perform the Functional test.