NOT GREEK AND LATIN – LOCALIZATION TESTING IS NOW AT YOUR DISPOSAL

PREPARED BY : NISHA SANGAMITHRA

TABLE OF CONTENT

1 What is localization?

Have you ever tried to decipher those lovely Japanese characters that just seem like embroidery patterns to you? Imagine a large audience that sees English as Japanese. According to study only 8% of the world's population speak English. How do we reach out to the rest? Localization is the only solution.

Localization (in short L10N) in a broad sense is considering the fact that not everyone can read English, translating your product in other languages so it becomes readable to a wider audience.

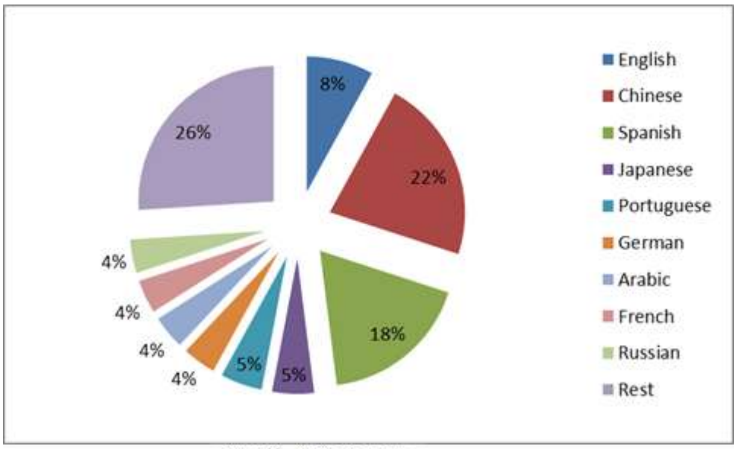

Figure (i) – Native Speakers

In spite of Ethnologue's statistics of 6,909 languages some products do take up the challenge of going global. Is a product global when whole world can read it? Yes. But it does not stop there. It also means, does the world read it with clarity and quality. Testing is very important here. Remember language issues are not cosmetic issues most of the time.

Localization does not only mean translated text. It also includes the way this text looks - formatting and alignment. Every product in today's world is being localized – from Operating Systems to Web Sites to Applications.

2 Is localization a hassle?

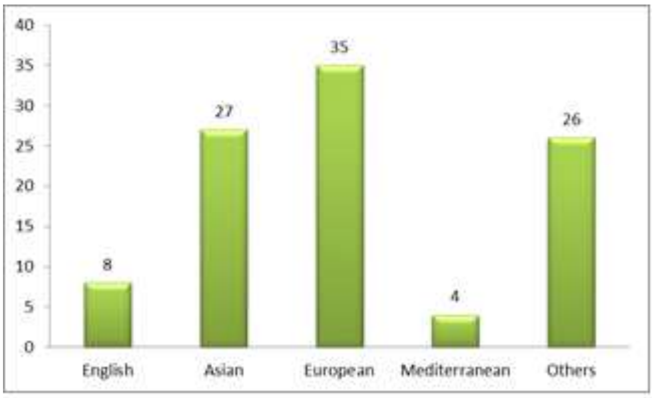

Figure (ii) - % of Revenue

Hassle may be too heavy a word but let us accept the fact that localization does have its own difficulties. Making the product global sounds exciting to business (the chart shows the % of revenue brought in by each language based on the number of users it caters to) but it is not that easy for the development team. A lot of planning and thought should go into the design, so it can be successful and most importantly scalable. Scalable does not mean providing provision to adding more languages to your list but also how well does the design accommodate new characters and symbols introduced in every language so often. For example, the symbol for Rupees (₹) that came into existence recently. When it comes to testing, the problem is most linguists are not computer / technology savvy.

3 How is localization applied technically?

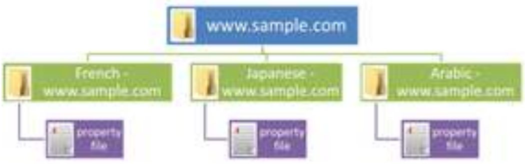

Figure (iii) – Case Study

To explain / understand the technical aspect of how localization is applied, let us take a small case study of a website (named www.sample.com) that is being translated into 3 different languages (French, Japanese and Arabic). Usually there are three property files maintained for each of these three languages. Each property file contains several key-value pairs. A key in simple words is a variable that represents the text to be translated – this text can either be a single word or an entire paragraph of text. Value is the corresponding translated text. When the user loads the site and choses to switch languages the corresponding property file is referred and the English display text is over ridden by the respective language's translated text. To understand this better, this is just like you do cobranding and want to just make slight changes to your customized site from your parent base site using jsps. Now what happens when some translation is missing? The default English text is displayed as the corresponding key-value pair is not found in the respective language's property file.

Usually chunks of text to be translated are sent to translation firms. These are sent in buckets to save cost as a package.

<u>On a technical note:</u> As we know characters must be represented as numbers so compliers can deal with them. Unicoding is one way of encoding.

Unicoding has:

- a built-in catalogue of million characters covering 90 scripts which includes charts for visual reference
- a coding methodology
- a bi-directional display of text

4 What does a typical localization testing checklist comprise of?

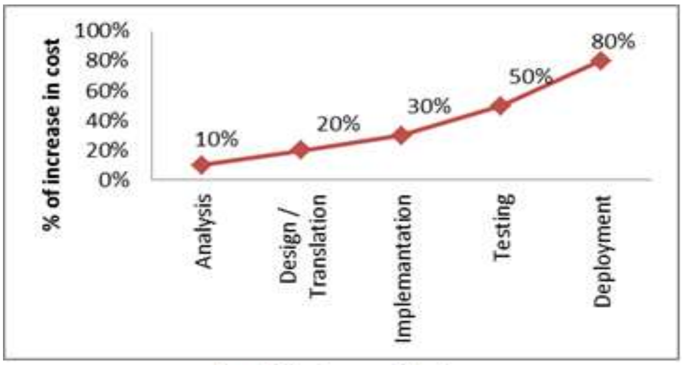

Figure (iv) – Increase in Cost

Just as in most cases, testing early is always cost efficient. As one moves towards production/release, the cost of translation and testing gets higher.

Localization cost gets higher because the translation firm go by packages not by number of words. A tester should be involved right from the design phase when the translation is happening in parallel. It is always good to work with the Business Analyst as the “Do Not Translate” list is being prepared.

Before testing commences, there are a few things to check:

- Input:
  - Hardware – Localized Keyboard. Going back to our case study how is Japanese input going to be accepted? Will the keyboard be marked with the Japanese character set? That may not be practical when the same application is going to be displayed in several languages. The keyboard will be cluttered with characters. So is a built-in virtual keyboard a solution? How is it going to be invoked?
  - Software - Editor, Copy / Paste, Character Map, etc. - Is there a built in editor? If www.sample.com is going to accept dynamic text, how is it going to be handled? It can be a simple input like a user name / password. How is a copy / paste going to be handled? Has the character mapping been established?
- Output:
  - Font determination
  - Dynamic text handling – Let us say our website is going to say "Welcome <Username>". Is the system is capable enough to carry the input user name and display it back?
- Content:
  - All translated text is available
  - Embedded texts in images are translated – Let us say there is a logo with the website's name, have we remembered to translate it?
  - Database fields for large text – concatenation – Let us say there is a long passage of 12000 English characters that the user is going to enter and it is in the background saved as 3 TEXT fields which is to be later concatenated for re-display. This means 6000 2-bytes Arabic characters and 3000 3-bytes Japanese characters.
  - Do not translate checklist - There are several items that cannot be translated. For example abbreviated text like GMAT, etc.
- Settings:
  - Font - Hard coded or externalized
  - Given font is installed

4.1 Jots for developers

- Start with the end in mind
  - Adaptable
  - Scalable – A few things to keep in mind:
    - Number of characters – Asian languages have thousands of characters more than European.
    - Usage of symbols – Like accents in French, etc.
- Keep it simple
  - Easy to follow and update
  - Easy to maintain

4.2 Jots for testers

- Ensure before, test after
  - Make sure the required mode of input / output is ready.
  - Ensure settings like font installation, etc...
  - Most importantly ensure the translated text is received on time to be placed in the property files.
  - Test once you are quite confident that the above is in place. Otherwise testing can get repetitive and tedious.
- Prevention is better than cure
  - Be careful while using gestures, emoticons, symbols, etc. For example: An innocent thumbs-up gesture is offensive in Australia.
- Let the expert review
  - This may not always be possible. But it would be good practice to run through the product with someone who knows the language. Many organizations are now appointing language teams.
  - But don’t worry if you do not have a language expert. Remember the translation firm has already done its review. Just have a concrete ‘Do not Translate’ list. A tester’s eye can be trusted to spot words that are in English in a long Japanese passage.

5 How is localization testing carried out?

Looking at it broadly what is it one would test in general in an application? Classifying any testing broadly:

Category	Description
Installation	• Standard • Compact • Custom - Full
UI	• Labels • Images / Bitmaps • Line breaks
Controls	• Text boxes, Text area • Check boxes, Radio / Option buttons • Drop-down Lists • Buttons
Functionality	• Application feature • Searches and Filters, etc.
Data	• Database • Field concatenation, etc.
Input / Output	• From the keyboard • Mouse selections – etc.

Not to beat around the bush, even in localization all of the above has to be tested – both happy and sad path. How do we ensure we cover all of the above effectively?

There are different approaches in testing a localized product / application. It can be done:

- Manually – A manual tester tests every single page / feature of the application in all languages.
- Semi-automated – The test is partly automated. Mostly used to identify GUI errors. Example: Let us say www.sample.com holds a competition and hands out certificates to all participants. Automation tools can be used to just test the size, font, log, colour etc. of the certificate but the manual tester is still responsible for the authenticity of the data.
- Fully-automated – Using automation tools like Silk Test/QTP for complete product testing.

6 What are the common defects found in localization?

Languages can be classified based on the bytes used for storage. (For example: One Japanese character requires 3 bytes whereas one Arabic character requires only 2 bytes.) So bugs found in one of the 3-bytes language are most likely to appear in all 3-bytes languages.

When a language requires more than 1 byte to store a character, there can arise issues of missing leading or trailing bytes. There can also be instances when large text is stored in several fields in the background, the first byte of a character is field 1 and the second byte of the character is in field 2. The way it is stored does not matter but on concatenation this should be handled to avoid data loss.

Again as mentioned earlier, language bugs are not all the time cosmetic. If it is important in English it is important in Arabic.

Here is a list of most commonly found bugs:

Defect	Possible Reason
Text appearing as question marks or boxes	Missing font
Missing translation	Missing key-value pair, hard coded text, etc.
Truncated string, overlapping/misalignment of GUI elements and controls, etc.	Text expansion
Data loss	The language is multi-byte and it is not accommodated in the database
Formatting issues like in Arabic where text goes from left to right	GUI alterations
Missing translation in embedded text (in images/bitmaps)	Oversight
Hot key issues like duplication	Automatic hot key assignment
Missing or broken functionality	GUI alterations

The above list can be used as a checklist for testing too.

With experience, translation bugs can be found in advance.

Testing Tips:

- Once you find a functional bug in a language, provided the code base is strong, you can almost be certain that you would find the same defect in all similar byte languages.
- It gets simple when you understand the structure behind the scene. This helps understand un-translated or missing buckets of text.
- Always know the territory you are translating for.

A general report on severity of bugs:

- High – Crash/Incorrect/Loss of Data. Example: Copyright issues, broken links, etc.

- Medium – Functionality issues like missing hot keys, etc.
- Low – Cosmetic issues like missing punctuation, etc.

Graphs below document common defects:

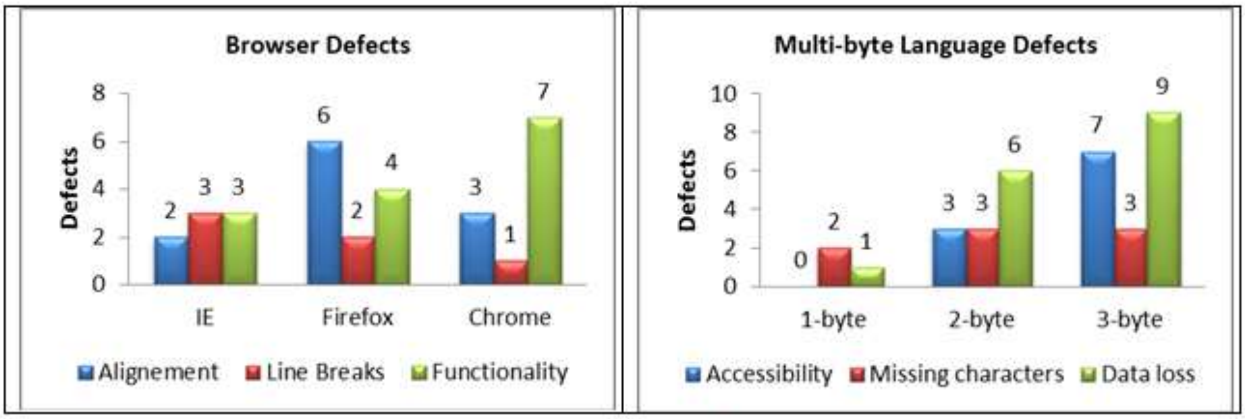

Figure (v) – Common Defects

Inference:

- Alignment issues are especially around titles and alerts. Unexpected line breaks and wraps are introduced by certain browsers. Hot spot issues are most common when it comes to functionality issues. Based on a general analysis of the most commonly used browsers (IE, Firefox and Chrome), if you look at the chart, IE seems to support languages better than other browsers.
- From the chart it is clear that more the number of bytes more the number of defects especially around data which in most cases would be high in criticality as it would have the most impact on users.

7 What can be practiced to overcome challenges?

Challenge: Data

Impact: Errors those are easily identifiable by the native user.

Practiced approach: Always a good test bed means well prepared for testing. Try to keep long passages of text for the different languages that are going to be tested. An easy way to do this is as always the internet. When there is long text saved split in the back ground and then combined for display there are always chances of data loss. Also a quick check would be, does a Ctrl+A in a text area (that stores in three fields in the back ground) select data in all three fields of data.

Challenge: Scalability

Impact: Growing in this competitive market is going to get difficult.

Practiced approach: As stated in earlier sections, scalability does not mean adding a few more languages to your product's already existing list. It also means accommodating new characters discovered every day. So is your variable mapping flexible enough to take more? Make sure it allows character mapping. This has to be ensured at the design stage. This is more a pre-testing activity belonging to the 'ensure' checklist.

Remember scalability also means not affecting what is already being used. Keep in mind the visually impaired that relies on short cuts rather than on a mouse.

Challenge: Browser

Impact: There are always chances that users are trying the latest browser.

Practiced approach: Be updated. Study the most used browser. How is information rendered in each browser? Study its UI, for example what are the browser's default features. How does it handle dialog and message boxes? Are these customizable in code? If so, has it been done to help languages which align differently? In many cases, in languages like Arabic, the title of an alert reads left to right (when it should read from right to left) because it comes with the browser.

Another difficult thing is to handle line breaks in browsers. This has to be watched closely.

Also when a browser is made to be accessibility friendly, ensure increasing font at the browser level does not cause chaos.

Challenge: Humungous test effort

Impact: Chances of missing the most important.

Practiced approach: One has to pick cleverly what combination to test. Imagine per our case study we have 3 languages in addition to English. That means even if it is a small application with 10 pages, you end up with 40 pages to test. Imagine all the validation and verification points to cover. Classifying languages based on bytes would help. Select a testing approach like All Pairs or OATs to handle the functionality.

Maintaining checklists: This could be as simple as ten items printed out and stricken through as they are tested in the different languages.

Areas	Impact	Takeaways
Handle over Testing	Increased by 80%	-Logical categorization has better control over testing -Ease in onboarding testers for localization
Test Effort	Reduced by 40%	-Standards, checklists, etc. eases testing
Defect Leakage	Reduced by 60%	-Better control over the application
Customer Calls to Customer Support	Reduced by 70%	-Ease of use -Better accessibility
Introducing Languages	Easier by 35%	-Increase in revenue -Wider audience

8 To conclude…

What does the future hold for localization? When it comes to localization we have already come a long way from just translation. We have online applications like the Google translator which not only translates text to the desired language's font phase but also is linked to a dictionary. When we think about advancement from a hardware perspective, we already have keyboard skins to help avoid a non-cluttered keyboard. But why not have infrared virtual keyboards with buttons that enables users to switch to different languages?

What localization means to us (the development team/business): When begun with the end in mind and the right approach is adopted it can be smooth sailing and a grand success reaching out to a whole new set of audience.

What localization means to the world: With localization nothing is going to seem Greek and Latin. And you can still be Japanese in Rome!

9 References

- http://www.wikipedia.org/
- http://blog.submittable.com/2012/10/23-countries-and-counting-what-language-is-your-submittable/

About Indium:

Indium Software is exclusively focused independent software testing services firm since 1999. Over the years, Indium mastered objective methods that minimize the risk of failure of applications and software products. With a global headcount of over 300 employees, Indium works for a mix of marquee Enterprise and ISV clients spread across the globe. Indium is aggressively pursuing the Social, Mobile & Cloud agenda to make these the core of our next wave of service specialization.

Contact us:

USA - Indium Software Inc.

Corporate Office
California
Suite 210, 1250 Oakmead Parkway
Sunnyvale, CA - 94085.
Phone: +1(408) 501-8844
Fax: +1(408) 501-8808

Atlanta
Crown Office Suites,
1870 The Exchange, Suite 100,
Atlanta, Georgia 30339. USA
Phone: +1(770) 989-7302,
+1(678) 608-0809

INDIA - Indium Software (India) Limited

Global Delivery Headquarters
Chennai
VDS House
41, 2nd Floor
Cathedral Road
Chennai - 600 086.
Phone: +91-44-2811 6330
Fax: +91-44-4210 4033

Prince Arcade Building
22-A, 3rd floor
Cathedral Road
Chennai – 600 086.
Phone: +91-44-4347 7200

Steeple Reach,
Old No. 25, New No. 39,
Cathedral Road, Chennai –
600 086.
Phone: +91-44-4347 7122

Bengaluru
Salarpuria Adonis, 2nd Floor
Binnamangala, 3/1, Old Madras Road
Kadiranapalya, Indira Nagar
Bengaluru – 560 038.
Phone: +91-80-6784 7500

New Delhi
F-1/5 Okhla Industrial Area,
Phase-I,
New Delhi – 110 020.
Phone:+91-11-6613 0400

Sales Inquiries
americas.sales@indiumsoft.com
apac.sales@indiumsoft.com
emea.sales@indiumsoft.com
india.sales@indiumsoft.com
sales@indiumsoft.com

General Enquiries
careers@indiumsoft.com
info@indiumsoft.com