Going Global? Let's Measure Your Product For World-Readiness!

Kshitij Gupta
Adobe Systems Inc
ksgupta@adobe.com

Lily Wen
Adobe Systems Inc
lwen@adobe.com

Abstract

The paper introduces a framework to measure products for world-readiness when releasing into a new geography or to set measurable improvement goals. The framework classifies the subjective points of consideration into an objective set of questions grouped into logical verticals to help score a product efficiently.

1 Introduction

The process of Internationalization can be summarized as enabling a product for great experience to customers of different geographies. The experience might extend beyond the product, to other areas like consumer behaviour, purchasing habits, design tastes, response to marketing campaigns, service expectations and after-sales customer support etc. The culturization of these verticals to suit local tastes leads to great experience for local customers. This paper focuses on the product aspect for the world readiness of software applications.

Is your product ready for the global audience? Is string translation the only requirement you would need to cater to when you plan to go global? How would you efficiently measure your product's world-readiness? What all parameters should you measure your product on, before planning go-to market for global audience? This paper aims to enable one to answer these and other such subjective questions to measure their product for World-Readiness, by breaking them into simple objective questions classified into logical systematic verticals.

Web/mobile applications are more prevalent today than ever and their world-readiness goals need to evolve too. Such applications now have faster iterations and quick releases. Localization of a product in short cycles at times is a challenge in itself; in absence of an appropriate framework to measure, focusing on improvements is next to impossible.

"If one can't measure it, one can't improve it": One needs the ability to measure product's world-readiness to set improvement goals and to evaluate feasibility for go-to-market for a geography. We developed a framework to measure products on World-readiness score to help manage and plan global market feasibility better. The framework categorizes all globalization parameters, like calendar inputs, collation capabilities, resource externalization etc. into meaningful verticals, assigns relative weights and provide guidelines on expected behavior, testing procedures and implementation basics. On the basis of collective score for a product, its globalization goals for the next release can be set.

In this paper, we discuss the framework at length, for multiple platforms, the expected behaviors, relative importance of such parameters and the challenges encountered while resolving respective globalization issues and how the framework enables one to rightly grade their products on world-readiness and set improvement goals.

2 Grading Your Product For World-Readiness

As the product reaches the World-Readiness assessment stage, it requires certain choices to figure out the scoring criteria. On the basis of product and its platform, certain application

Proceedings of the 37th Conference Translating and the Computer, pages 120–126,
London, UK, November 26-27, 2015.

scoring features need to be weighed upon. The product helps decide the relative weightage of features that are to be graded. For example, the text input features are more prominent for the documentation product than are for a photo-editing tool. The platform, ranging from desktop, web, mobile to tablets and others, helps decide the set of features that the product needs to be graded on, along with their respective weightage. For example, the weightage for ease of selecting locales from within the product might be different for mobile, web and desktop in the increasing order of priority. The mobile app might not require such feature since the locale is decided by the OS.

On basis of initial questionnaire, with the weighted features finalized for the product, one can start grading the product using the framework. The framework asks objective questions, accompanied by a set of test cases, for grading internationalization features so the quality assurance team has an appropriate method of grading the product, leaving little scope of confusion and error. The criteria that the framework asks evaluator to list are as listed below.

1. **Region Selection:** Many of the application behaviors depend on the region it is being used in. Should the product refer the user with the first name or the last name? What format should the date/time be displayed in? What should be the measurement units to be used for display to user. How should currency numbers be formatted? All this and a lot of such information formatting depends on the regional culture of the user and hence it is important to provide the user with the ability to select region. The product should be penalized for not providing user an easy way to select a region and be awarded full score for providing such option in an easy way.

2. **Language Selection:** The user interface of the application is localized using the application language. How the product selects the locale in absence of locale selection by the user. Is the product enabled to pick the locale from settings of platform/system it runs on?; does it save user preference for locale and serves the application in user selected locale on multiple devices across multiple sessions? If the product does not maintain language consistency across multiple workflows, a product is penalized negatively and given full score for consistently managing application locale for user.

   A locale code represents both language and the region. 'en_US' and 'en_GB' while representing the same language English have a major impact on differentiating the region specific information. Thus the product should use only ISO locale codes to identify a language and the region. This brings consistency across multiple system apps and makes integration easy even in a complex system. In another case, if standard codes are not used, it adds an otherwise avoidable complexity to the system, making it really difficult to integrate with any other system or to use any other standard internationalization libraries. Inability to use the standard locale codes should invite heavy penalty to the product score. Below we mention the components which are formatted on the basis of application locale. There exist standard libraries (1; 2; 3; 4) for each programming platform to perform formatting of most of these components. The use of such standard library is highly recommended over writing own customized functions for each task as, along with taking care of internationalization formatting guidelines, this brings in formatting consistency across application.

   (a) **Date-Time:** The date is formatted on basis of application locale. To grade the product for date/time formatting, one needs to evaluate multiple parameters: Does the product take date inputs from users? If calendar/date input exists, are they formatted as per user locale? Is the product impacted by the public holiday schedules of the region?

Does it display the calendar to users? If it does, does it take into account to format the starting day of the week as per the region standards? Is the UI date-time components formatted as per locale using standard libraries? Does your product display the time zone to users while taking time inputs? Is the system able to adjust for daylight saving while dealing with time values?

(b) **Collation Support:** Does the product has a list of options to display to the users? Are the options of such a list sorted as per the product locale. Does the product support sorting the user generated lists, if any? Verify the sorting for the RTL languages too; are they rightly supported? International Component for Unicode portal (5) can be used to test the rightful collation of any sample list.

(c) **Numbers:** Does your product display numbers to users in any workflows? Does your product take number inputs from users? Are number elements on user interface as well as in user input properly formatted as per the language part of the locale? This is a subjective choice for the product owner to make. One can either use the language or the region of the locale for formatting. There occurs confusing scenarios at times when a user based in Germany decides to browse your web application in English. Since in such a scenario, user is making a willful decision to browse your app in English, our recommendation is to always use language part to format the numbers to maintain consistency across multiple language/region combinations. One needs to ensure the symbols accompanying numbers, like degree, percent etc, are also displayed as per the language rules. Since such signs are not part of number formatting which can be achieved dynamically using standard libraries, formatting of signs in the application should be validated once the translation is complete for the workflow.

(d) **Currency:** Currency is a special case of number formatting. If your product displays, takes input in currency values, you need to ensure that the currency is formatted as per the region code of the locale your app is using. Since currency is a special case and is specific to a physical region, its recommended to format currency values as per the region code, instead of the language code.

(e) **Units of Measurement:** Different regions use different metric systems to measure units of distance/quantity. Does your product provide proper formatting of such values on the basis of the region?

(f) **User Profile Information:** Does your product store user information in such a way that it can be formatted as per the user selected locale? A very common example of user information formatting is the name of the user. Many regions consider referring to user with the first name while many others consider the last name as a better way of referring their users. Such choices are primarily made on basis of the region user creates her account from. In addition to using the region, many advanced web applications use the language auto-detect of user name to decide on the name formatting. For instance, if the system detects users name is in Japanese language but the region of registration is United States, it will still go ahead and format the name as per the Japanese way of referring the second name first. Besides names, the product should be able to store other user information(like phone number, address etc.) in region agnostic manner and display the formatted output on user interface for the selected region.

3. **Localizability of the Product:** If a product can be easily localized, it gets a high score on localizability. A fully localizable product does not require any additional code change to

localize the product in an additional locale. The localizability of the product is primarily measured on the following parameters.

(a) **Build Process:** Does the product has a single build process to generate multi lingual builds? Does it support installation of language packs separately? If the answer to any of these questions is No, the product is penalized on the score since localizing it in a new locale would incur additional efforts on the core code to generate a build in additional locale.

(b) **Directory Structure:** Are the locale specific assets placed parallel to the core location using proper codes for each locale? Are such codes that are part of folder locations kept intact even when the folder locations are translated to another locale? If the asset placement is inappropriate, adding and assessing an additional locale asset becomes a pain since there is no standard way to do that. And if the folder locations are translated for localized builds, there is a great chance of runtime errors due to resource not found errors. The core team developer needs to take care of such scenarios.

(c) **Auto Layout Components for User Interface:** Does your product use the auto-layout technology for UI components? This helps save the effort of fixing truncation bugs for adding any new locale in the product. This makes a huge improvement over the standard native RC/MAC components which requires truncation bug fixing for each locale separately. The internationalization team should work with the core teams to maximize the use of auto layout technology for ui components.

(d) **Ease of Language Selection:** Does your product offer an option for user to select the locale from the list generated for the product locales? This is important since adding a new locale would otherwise require an additional effort to select that locale. If such list is available for user to select and save the default locale, selecting a newly added locale is pretty simple and straight-forward for the user.

(e) **Mirroring of User Interface for RTL Languages** Does your product appropriately mirror the user interface components while displaying the ui in right to left(RTL) locales? The multiple components that one should evaluate to verify the mirroring include but are not limited to:

- Minimize, Maximize and Close buttons on application bar.
- Scroll bars
- Resizing controls
- Menus
- Commands
- Directional arrows
- Directory breadcrumbs
- Search boxes
- Icons
- Buttons
- Dropdown menus
- Directory navigation boxes
- Radio boxes
- Checkboxes
- Text input boxes

The product should be able to mirror all such ui components appropriately to cater to RTL locales.If not, the product grades are penalized on prorated basis.

(f) **Resource Externalization:** The product assets, both strings and content like images, video audio, should be appropriately externalized to relative locale specific locations to enable efficient localization of resources. If a part of content/strings remain hard coded, the localization of such resource is not possible and hence hinders the localizability of the application.

Corner Cases: While finalizing the product for locale specific geographies, the quality assurance team needs to pay specific attention to figure out corner cases for internationalization issues. One such live example that we found during our research is depicted in the images below.

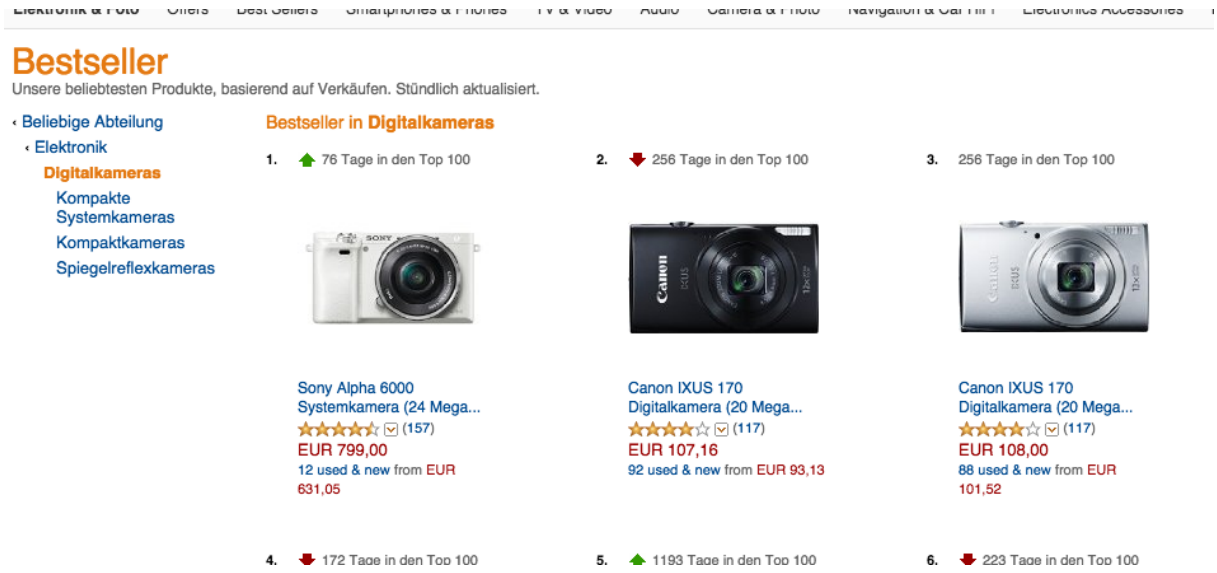

Figure 1: Product search with german formatted currency (price) values.

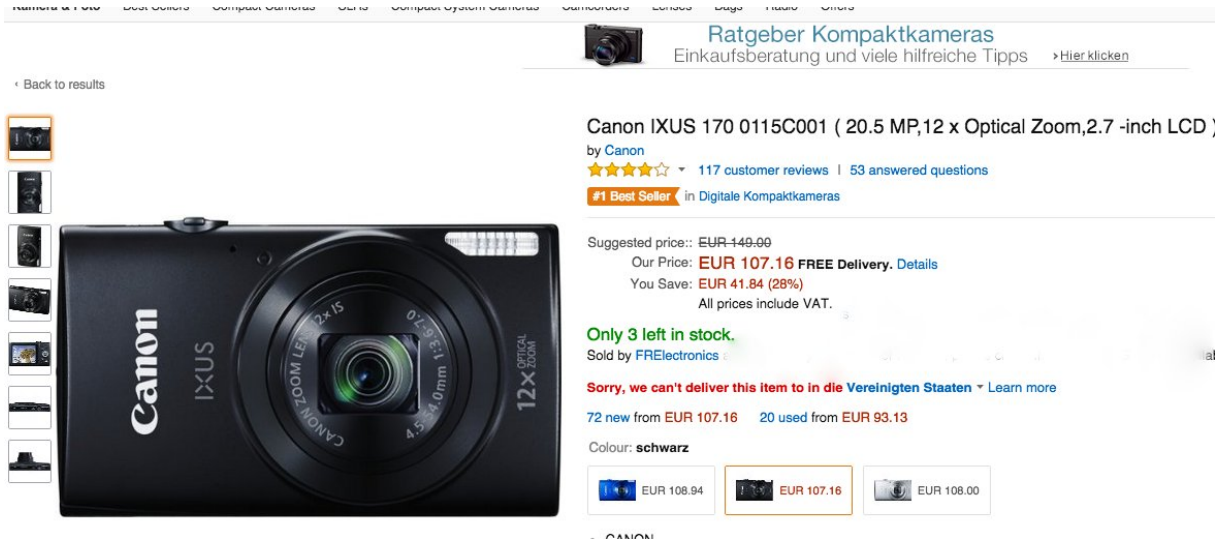

Figure 2: Product details page with same currency(price) values, formatted differently.

Above are the two screens of leading e-commerce company operating in Germany, with portal language set in English(beta). The figures have the currency value of the same

product formatted differently on two different pages, all other settings remaining same. Such cases can be avoided by using same apis with consistent setting parameters for formatting internationalization candidates throughout the application. The portal above is in beta and hence such issue might be justified but for the customers, such inconsistent behavior might be irritating and confusing leading to unpleasant experiences.

Another corner case for internationalization issues is when a specific string in a product workflow appears in core locale in localized version. In most cases this happens when the core developer mistakenly hard codes a string instead of adding it to externalized resources. Most such issues are identified during quality assurance for localized products, but for the rarely occurring workflow, there are chances that such issues might enter production environment missing qa team observations.

To get rid of such and other corner cases, it is necessary that core developers heed to international guidelines for product development.

4. **Unicode Architecture:** The application should store serialize the data for input/output streams into unicode encoded stream by default. If any data requires storing in another encoding, it should be stored with appropriate encoding tags to enable conversion back to/from unicode when required. Multilingual strings should be collated with the Unicode Collation Algorithm (6) or with ISO 14651 (7).

5. **Text Support:** The application should support multiple text operations in multiple scripts. The operations include, but are not limited to, Input, Editing, Printing and even File operations. To fulfill the requirements for Text Support, a product, irrespective of what locale its localized into, should be able to take input and process text content in any language. Since there are almost infinite number of locales available today, the team should test these requirements on a certain set of scripts that contain but are not limited to : Brahmic Scripts, Chinese Traditional and Chinese Simplified, Japanese, Korean, Latin, and other European scripts like Greek etc. among other scripts. Verifying the behavior in such scripts helps confirm the product assessment for multilingual text support.

6. **Experience Assessment:** When going global, it takes more than just product internationalization to give your local customers great experience. The experience starts right from the customer interaction during marketing campaigns. How the local consumer likes to engage in marketing pitches? What are their buying habits? Are they more comfortable buying online, or more comfortable buying offline through gift cards? Is there any specific kind of online payment methods that they prefer over others? Is the local customer more satisfied with the online download or they like to get their product delivered? Is the customer support provided for product satisfactory for them or a bit more of culturization would help? Of course, a lot of these parameters are business decisions, but keeping the local customers at the back of mind while making such decisions definitely helps.

3 Conclusion

When a product makes an entry into a new geography, it demands a lot more than just string translations. It is always better to understand the geography first before making a foray into it. Getting to know of the local market helps the management to take complex business decision for culturization of the product. We recommend using the suggested framework to measure the World-readiness of the product, so measurable improvement goals can be set for future releases in the local market.

4 Acknowledgement

We would like to thank Globalization team at Adobe, with special mention to Leandro Reis, Globalization Architect, for sharing his extensive knowledge on world-readiness and the related topics. His inputs really helped in coming up with the desired framework for World-Readiness.

References

(1) ICU is a mature, widely used set of C/C++ and Java libraries providing Unicode and Globalization support for software applications. - http://site.icu-project.org/

(2) Intl.js is a javascript library to fill in the void of ECMAScript Internationalization API specifications in Safari. - https://github.com/andyearnshaw/Intl.js/

(3) Globalize is a JavaScript library for internationalization and localization that leverages the official Unicode CLDR JSON data. The Globalize project is backed by jquery. - https://github.com/jquery/globalize

(4) Js18n.com is a portal devoted to comparison of latest developments around JS Internationalization libraries. - http://jsi18n.com/

(5) Demo site for International Components for Unicode localization data for over two hundred languages. - http://demo.icu-project.org/icu-bin/locexp?_=en_US&x=col

(6) Specifications for the Unicode Collation Algorithm (UCA). - http://unicode.org/reports/tr10/

(7) International string ordering and comparison: method to order text data independently of context - http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=57976