Playbook Methods Repository

# **Internationalization and Localization**

Evaluate and implement design practices and technologies to enable flexible localization of the product for audiences in target regions / markets. Define multi-regional specifications for end-to-end product testing.

### Remote Agility: **•** High

### Linked Tactic(s): Agile Development

## Why we do it:

Globalizing your application will require a certain amount of complexity. Many variables need to be considered, including languages, formats, legal and compliance requirements, APIs, operating systems, and devices. Any of these can cause a localization challenge.

Developing language-neutral architecture and functionality is part of internationalization. It allows the application to be used worldwide. Therefore, you will need to drill down to locale-specific versions. An internationalized software application allows it to function as effectively in any of its supported locales; it provides localization capabilities. Internationalization enables you to plan for future markets and languages while developing software or mobile applications. It is the process of neutralizing your product's code, content, and design in order to make it simpler to adapt it to different cultures in the future without having to fully re-engineer it.

Localization typically follows internationalization. However, in addition to tailoring your product to a particular location, the localization process might reveal any terms, phrases, or user interface (UI) components that have not been properly internationalized.

A localized application is one that has elements modified to fit into the requirements of its locale. You will have to make changes to the UI and content. A thorough test of localization should precede deployment. An entire market should not be disrupted by a cultural blunder. This is where localization testing fits in.

## When to apply it:

* Implementation

## Best Practices & Considerations:

* Feature-based: Depending on the region, some features may not be available. During testing, ensure the feature is hidden for the regional users to whom it is not applicable but visible and functional for the regional users who need it.
* Locale/ Culture awareness: Locale/cultural awareness includes understanding how dates and numbers are formatted differently in different regions. The differences include calendars, holidays, festivals, date formatting, time formatting, currency formatting, number formatting, addresses, phone numbers, zip codes, and measurement units.
* User Interface: All language content should be tested for adaptability to the user interface. To accommodate longer text without distorting alignment, it should be changed accordingly. Also consider languages that are read right-to-left; some elements and layouts may need to be mirrored.
* Rendering: The language-specific characters associated with all the supported scripts should be displayed correctly. The scripts on a page in a specific language should be perfectly executed, i.e., no script errors should be displayed, and all characters should be displayed in the particular language.
* File transfer: To determine whether a file transfer interface is language-specific, we need to test the application's file transfer operation. Transfer of the file was successful or not, and the file was not corrupted.
* Database: Unicode characters will be supported in the database during database testing for internationalized applications. For this purpose, special data types are available. SQL Server defines data types like nchar, nvarchar, and ntext that allow storing Unicode characters.

## Responsible roles:

* Software Engineer to implement internationalization.
* QA Engineer to test localization.
* Product Manager to give requirements and oversee processes, such as working with translators.
* Product Designer to create designs for localization. They will collaborate closely with Product Managers to understand requirements and with Engineers to understand technical support and constraints.

## Tools:

* Online tools/platforms/services
  + IBM Rational Functional Tester, eggPlant, .NET application localization tool, Applitools
* Websites
  + xx
* Databases
  + xx
* Other
  + xx

## 

## Thoughtworks Examples - Linked

* Client working docs, airtable, miro/mural boards
  + xx
* Client polished presentations/deliverables
  + xx
* Internal assets - clinic materials / guild docs
  + xx

## 

## Learn more: How we do this?

* Templates (docs, decks, sheets, miro, etc.)
  + <https://www.softwaretestinghelp.com/localization-and-internationalization-testing/>
  + <https://www.ibm.com/docs/en/spm/7.0.11?topic=widgets-internationalization-localization>
  + <https://blog.andovar.com/what-localization-testing-why-important>
  + <https://www.functionize.com/blog/what-is-localization-testing-and-why-is-it-important>
  + <https://www.mjt.me.uk/posts/falsehoods-programmers-believe-about-addresses/>
  + <https://www.kalzumeus.com/2010/06/17/falsehoods-programmers-believe-about-names/>
* How-To Resources (external or internal)
  + xx
* Outside References (articles, books, etc.)
  + Connectivity, Culture, and Credit - Nine ways designers can be more empathic and effective when creating products for emerging markets ([article from Google](https://design.google/library/connectivity-culture-and-credit/))
* Sub-set Activities
  + xx

## 

## 