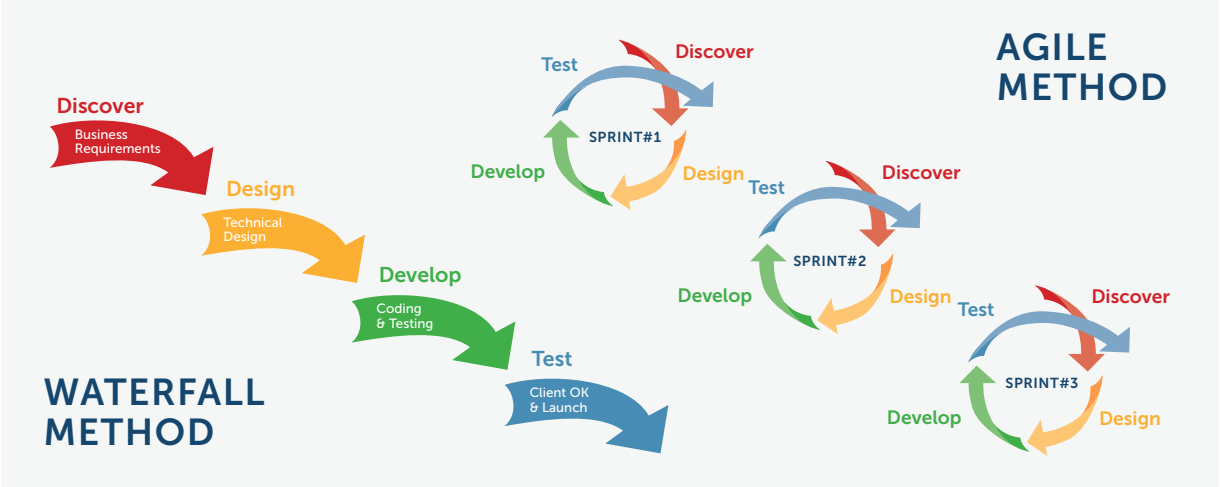
SOFTWARE LOCALIZATION FOR AGILE, WATERFALL, AND HYBRID DEVELOPMENT



Irrespective of the type of software being created, whether it is a thick or thin desktop client, a cloud service, a mobile application, or a website, the potential need for its localization should be assessed at the very beginning of any specification or development. There are certain principles that must be adhered to during the design process, as well as development patterns and best practices that need to be applied for successful globalization of a software product.

INTERNATIONALIZATION IN A STANDARD 3-LAYER SOFTWARE STRUCTURE

Software localization begins with internationalization across each layer of the software structure: Data, business logic, and presentation layer. This includes engineering the source code, e.g. with locale-specific application of text/dates/currencies, or preparing the database to handle multilingual data and offer dynamic support for foreign language markets. Additional tasks may arise within other components of the software system, and this entire internationalization process requires specialist engineering skill set as well as a thorough knowledge of the software.

The internationalization stage is followed by actual localization, i.e. the addition of culture-specific resources such as user interface strings or multimedia resources, via terminology management, translations, and in-country reviews.

It is important to note that thorough software internationalization is a fundamental in-house task for ensuring successful localization. Inadequately internationalized software can impede the localization process as it impacts on localization costs more than any other factor combined.

PREPARING FOR SOFTWARE LOCALIZATION

You can support the software localization process by providing reference material in the form of user guides or help files, screenshots, or other guidelines.

It is also advisable to work closely with your language service provider (LSP), internal staff, and other stakeholders to ensure efficient flow of information with regard to the handoff and delivery of translatable material or financial and administrative matters.

Define tasks internally and externally by preparing your stakeholders for the exact timeline of events, e.g. when your LSP can expect to receive material for translation (even if the volume of content is still unknown), and streamline the process as much as possible. Establishing best practices and increasing collaboration as a basis for your workflow from the outset will go a long way towards effective localization management and its successful implementation.

Such general preparation for 'localizability' is only one aspect of software development. In the modern IT world, all software development follows standard or slightly customized methodologies. These are mostly openly available to the public and have been refined through decades of software development from the IT industry.

Each approach links project phases differently. Currently, the most popular development models are:

- Agile
- Waterfall
- and so-called Hybrid models, the latter of which form a compromise between Agile and Waterfall approaches adapted to the project requirements.

Agile

Agile development encompasses repeated cycles of development phases for each feature or software component, while maintaining a general, somewhat flexible structure for the entire project. This approach emphasizes the release of new features and functionalities at the end of each iteration (typically one to four weeks long). Within Agile development , there is a great deal of focus placed on each team member's ability to self-organize, communicate effectively, and adapt to changing requirements in order to consistently deliver a valuable product.

In this model, localization is performed alongside the iterations, meaning the necessary language teams must be highly flexible and at your constant disposal.

Waterfall

In the more hierarchical Waterfall approach, the project phases are sequential, so each phase is only launched when the previous one has been completed for the entire project. This requires thorough planning and comprehension of the product and what is needed for its development, combined with equally thorough and well-documented design, coding, and testing phases.

The fact that each phase is completed before the next one begins means that localization will not start until the software product is ready for release. Therefore, the localization project will consist of a large volume of content for translation that may take several weeks or months to complete.

Issues with internationalization are not uncommon with the Waterfall model. Imagine your application has a database system that supports English characters, but not Chinese.

When looking to localize into Chinese, you would need to change this database and rewrite, then test, a large amount of code that interacts with it. This can be a costly move, and a problem that may not be apparent until later phases in the development.

Hybrid

In some cases, specific business needs will require a "Hybrid" approach in order to encompass any appropriate advantages of the Agile and Waterfall models for different aspects of a project.

Localization needs should be included when reaching a decision about the development model best suited to a specific project by prioritizing the triangle of scope, cost, and time-to-market. The various software development methodologies weigh these elements differently and it is practically impossible to favor all three of them at the same time.

You must decide whether quality or timing will be more important to your business outcome. For safety-critical software, such as medical applications, translation quality is of paramount importance. In such cases, localization must involve subject matter experts (SMEs) and include several rounds of reviewing and testing by in-country reviewers.

Whenever quality is crucial, time-to-market must become less of a priority otherwise costs will increase. Should you wish to release worldwide in all target languages on the same day, you will require a much larger budget. In any case, scheduling localization tasks and synchronizing them with software releases is a complex organizational feat worth thorough consideration.

AGILE SOFTWARE LOCALIZATION

Agile software development as a concept focuses on the application of iterative and incremental development methodologies to the planning, designing, coding, testing, and maintenance phases of each feature, customer request, or correction of a software product. A fundamental part of such iterative development is code branching, with separate teams working on different aspects of the project.

One of today's most popular approaches to this end is the Scrum model: a flexible Agile development strategy with very short iterations termed 'sprints', typically between 1 - 14 days in length. These short sprints mean quick time-to-market for given functionalities.

The fact that the Agile model is so flexible and more responsive to user feedback also makes it more challenging for localization.

Purely Agile development calls for localization at every "Sprint", meaning localization tasks are generated, for example, every two weeks, leading to localized product updates available in all target markets at the same time. While highly productive, this type of localization is time-intensive. In-country review can become very difficult under such time constraints.

You may have lower individual translation volumes, but their management requirements are much higher than they would be within the Waterfall model. More efficient collaboration between the developers and the LSP is needed here, with effec-tive workflows that automate as many process components, such as build creation, testing,or version control, as possible for a successful partnership.

Comparison of Advantages and Disadvantages in Localization for Agile, Waterfall, and Hybrid Software Development Models

MODEL	ADVANTAGES	DISADVANTAGES
AGILE	→ Highly productive so as not to impede launch of localized product versions → Small, individual translation volumes → Good risk management since issues are generally uncovered early in development → Can assist in bug exposure → Lower costs for bug fixes since issues are generally discovered early on in localization rounds → Quicker feedback from target markets	→ Requires extremely flexible and consistently available resources for each target market → The quicker the delivery rates, the more resources must be expended → High number of administrative tasks and handoffs → Requires internal localization engineers with specialized knowledge → Costly and time-intensive translation management requirements such as workflow automation → Any non-automated handling of translatables is inefficient and prone to error → In-country review difficult with potential for missed deadlines and challenging QA
WATERFALL	→ Predictable project scope and costs → Less resources required → No in-house translators needed → LSP has ample time to prepare and allocate resources → In-country review is uncomplicated to schedule and coordinate or outsource	→ Generally of large volume and therefore time-consuming → High costs for retroactive bug fixes or changes → Slower target market feedback → QA feedback from in-country review may be low with larger quantities of materials → May cause version management issues as slightly different language versions of a product can co-exist
HYBRID	→ Controllable project scope and costs with medium resource requirements → Product is localized within same time frame as source software is created → Sufficient time for preparation and review of target materials → Less frequent handoffs than with agile localization → Scheduling of in-country review is manageable	→ Relatively short time frames for meeting goals → Overhead for workflows and possible automations required → Dedicated in-house operator for workflows needed

Setting up communication channels and creating work-flows can be costly and somewhat time-intensive. It is therefore beneficial to choose a provider with a strong technical background in facilitating these channels between the in-house and LSP team. The advantages of localizing within a purely Agile development process are very similar to those it offers the software product itself. The short development cycles mean faster time-to-market for localized versions of the product and better risk management as issues can be discovered at the beginning of a feature's development rather than after its release.

Agile localization may even assist in bug exposure. By testing translated versions of an application, you may discover bugs that would have otherwise gone unnoticed and will be able to correct them before the software is released to market.

Furthermore, these shorter iterations allow feedback from your target market regions to be obtained much faster than with a Waterfall approach, where larger, more extensive features are released at a slower rate.

A major disadvantage to localizing within an Agile development model, however, is the number of handoffs involved. The average period of time for localization and release to customer may be one to two weeks, but in some modifications of this methodology, the sprint lasts only one day. This leads to frequent, small chunks of content being sent to the LSP, which in turn means that more localization resources are required. It is crucial to automate the frequently repeated process of sending translatable material back and forth as much as possible. Your LSP may provide you with a translation management system to automate this workflow. Otherwise, you may have to develop custom software tools or scripts that utilize CAT tool application programming interfaces (APIs).

These automation solutions are usually expensive and time-consuming to develop, but they are worth all efforts as they make the handling of translatable content highly efficient and less error prone.

In order to ensure that the same key linguists and/or SMEs are consistently available for the duration of your project, you will need to agree with your LSP that a team of translators and in-country reviewers are at your disposal awaiting each sprint. However, this approach will incur more administrative tasks and minimum charge rates as small volumes of content for localization are sent through after each iteration.

Internal review is also difficult to coordinate when using the Agile approach. At these speeds, quality assurance of in-country reviews can be rather challenging.

Since in-country review is also usually completed by native speaker SMEs who will already have other responsibilities and commitments in place, requesting additional tasks with little or no warning beforehand can engender resistance to those tasks. Ideally, the in-country review team should be comprised of dedicated in-house staff; however, additional expenses may be incurred through the hiring of standby team members for each target market.

Providing reference materials to your LSP, or perhaps even demonstrating your software's functionality, ensures that the translators are able to properly understand the meaning of the source text. Giving translators as much information and context as possible from the beginning of a project will avoid time-consuming clarification issues during later stages.

You may also be able to cut costs by limiting linguistic testing to higher revenue locales, and perhaps even consider excluding some markets from this stage altogether.

It also makes good economic sense to cross-examine all target languages for issues found during a particular translation.

Preventively addressing context questions that arise for Spanish in similar languages such as French or Portuguese, for example, will help to further improve quality. In summary, localizing software within an Agile development model works well if dedicated resources are available to your project.

It is also advisable to make existing in-house platforms for issue tracking and test case management available to external testers or to turn to a LSP that can provide similar solutions. A shared platform, such as Milengo's languagedesk.com, makes project and process stages visible to all testing participants and can prove invaluable during fast iterations.

Collaboration is a key requirement for successful integration of localization into an Agile software development project The Agile model will allow very fast time-to-market, albeit with higher associated costs and potential quality drawbacks when compared to Waterfall and Hybrid models.

WATERFALL SOFTWARE LOCALIZATION

The Waterfall model is a linear approach to the phases of a software development project. Here, a requirements analysis (planning) is followed by system and software design (designing), implementation (coding), verification and validation (testing), and, finally, release (maintenance).

With this more traditional approach, a company will tend to wait for its software product to be in final beta phase or ready for release in the source language before localizing it. This leads to a much larger project volume as the entire software needs to be localized at the same time. Predictable project scope and cost is the advantage to Waterfall localization. A complete file set is obtained, the entire product with all its elements is localized, and there are no unplanned updates or changes to the software. The required resources can therefore be more readily calculated and you will also require fewer of them. Your LSP of choice will have ample time to prepare and allocate its resources, and the presence of in-house translators is rendered unnecessary.

Furthermore, in-country review is easier to schedule when the project's time factors are clearly defined. SMEs, such as your company engineer in China, can be booked well in advance with minimum impact on that team member's other responsibilities. Review rounds can be planned and coordinated, and tasks can even be outsourced to minimize costs, e.g. to your LSP or to an additional company for linguistic testing. This is, in effect, a shared service approach.

Depending on project size, a disadvantage to localization within the Waterfall model may be the potential sizeable lag between the source software and localized releases. When all languages are not localized at once, one may end up with different language versions that further complicate software version management. Quality assurance feedback from in-country reviewers may also be low when an especially large quantity of material is being produced.

With Waterfall development, you will experience a slower time-to-market, albeit at a lower cost, compared to Agile or Hybrid approaches to localization, but higher quality can be achieved because the entire translation context and rich reference materials are available to the linguists, instead of the small text chunks generated in Agile sprints.

This model works well whenever limited resources are available and time-to-market is not crucial. For quality assurance, the time available to plan in-country review usually allows you to meet target market and customer expectations even if individual language versions of the product differ slightly.

HYBRID MODELS

Hybrid software development approaches aim to combine aspects of both Agile and Waterfall methodologies suit a project's specific needs at various phases of software production. Typically, the result is a more controlled process in terms of scope and costs. Handoffs are not as frequent as with Agile development, yet the product is still localized within the same time frame in which the source software is being created.

At four to eight week intervals between translation assignments, or whichever iterations are deemed appropriate for a particular project, you are left with enough time to plan, localize, test, and review target language materials. Within these more reasonable time frames, fewer localization resources are required than for the incredibly fast-paced sprints during Scrum development. At the same time, the lag between releases experienced with pure Waterfall development is reduced.

In-country review is easier to schedule at this more moderate pace, and your SMEs will not be overwhelmed with new materials as they would be in purely Agile models. However, scheduling must be handled efficiently and competently as the time frames for meeting localization goals are still relatively short.

Some additional overhead will be required here as a functioning workflow must be in place to extract, collect, and transfer localizable content to the LSP. There are automation tools available for these purposes, but such systems must still be operated and customized to specific needs by a dedicated team member.

In summary, Hybrid software development yields the best localization results for situations in which time-to-market and costs matter, but limited localization resources are available and quality assurance can potentially be compromised on to accommodate these if needed.

ABOUT MILENGO

Milengo delivers translation and related localization services to the world's most successful international businesses.

Our team of over 350 translators, project managers, engineers, and more work alongside clients from 19 offices located across the Americas, Europe, and Asia.

Milengo has won the business and praise of global leaders that include Walt Disney, DuPont, Plateau and Saba Software.

EUROPEAN HEADQUARTERS

Skalitzer Strasse 49
10997 Berlin
Germany
Tel: +49 (0)30 47 37 59 96
contact@milengo.com

US OFFICE

1065 Avenue of the Americas
New York, NY 10018
USA

Tel: +1 (888) 424-2325

If you would like more information please contact us at:

sales@milengo.com