The Path to Agile Localization

Lingoport Resource Manager

A Lingoport White Paper
Authored by Richard Sikes
Localization Flow Technologies
www.lingoport.com

Table of Contents

Does Agile Development = Fragile Localization?

The growth in popularity of Agile development has caused a great deal of consternation among localization project managers on the client side of the industry. It is frequently interpreted by localization personnel to be a justification for software developers to twist and turn in whatever direction they wish, and to disregard the planning and budgeting needs of dependent processes. Indeed, it is a paradigm that has caused dependent process managers to rethink how they work because, without adaptation, the more waterfall-like orientation of documentation and localization cannot exist in a fragile state of continual and costly disruption.

But there is a viable path out of this dilemma: the ***Lingoport Resource Manager*** provides automation that resolves typical friction points that exist between Agile development and localization management.

A Difficult Decision, Now an Ongoing Dilemma

One of the most difficult decisions that a software localization project manager has to make is when to begin the translation process. If started too early, then the downstream processes will be doomed to inefficiency due to coping with change; if it started too late, then the delivery to market might have missed an important deadline such as a trade fair, fiscal year end, or important sales season.

The user interface ("UI") is a key trigger in the decision when to begin. Once the UI is completed, other product components produced by downstream processes such as online help, documentation, marketing material, and other peripheral content, can rely on it as a terminological heart and baseline reference. These form the basis of the localization effort.

A look behind the scenes at the reality of software development reveals that the UI is one of the most fluid parts of the entire project, often not being "frozen" – that is, left in a state that no longer changes – until very late in the overall development life cycle. In the past, a disciplined waterfall approach ensured that changes to the GUI were minimal after the "freeze" threshold had been passed. This supported efficient localization. The localization manager needed only to monitor the development process and judge when rate of UI change had slowed sufficiently to justify initiating the localization process.

Agile development, in which the UI may be altered throughout numerous iterations and throughout the development lifecycle, has thrown a monkey wrench into the classical localization paradigm. This has focused the spotlight on the disruptive nature of Agile development as it has gained popularity among development shops, especially as two of the four primary anchor concepts of the Agile Manifesto[1] are in direct conflict with prevailing localization wisdom. These are: favoring ***individuals and interactions*** *over processes and tools, and* ***responding to change*** *over following a plan.* In localization, by contrast, automated processes and technologies are needed, in fact are vital, to achieve cost-effectiveness and scalability. Furthermore, response to change on the development side introduces disruption in localization that can be extremely costly in terms of time and expense. The adverse effects on localization are clearly undesirable.

But Agile is here to stay, so smart companies are looking for a solution that benefits all stakeholders in the product delivery chain.

1 Beck, Kent, Beedle, Mike, et al. *Agile Manifesto*. 2001. www.agilemanifesto.org Web. 25 Nov 2012

Why Localization Processes Retain Waterfall Structure

Localization is typically performed in a sequence that is reminiscent of the waterfall method for many good reasons. For example, certain process steps should logically precede others to optimize efficiency for the overall localization effort. Some of these are:

- **Testing Resource Files for localization-readiness prior to translation**

Sometimes developers inadvertently introduce errors in resource files. Assuming that the file in question then goes to localization without testing, and then subsequently comes back with a missing string, someone has to research whether the translator made a mistake or the source string was missing in the original file.

The same can be said of other typical errors that occur in resource files, and the time spent tracking and rectifying such errors becomes inflated in proportion to the number of target languages. For that reason, the greatest efficiency can be achieved by testing resource files as they leave the build environment on their way to localization, and not allowing deficient files to enter the downstream process.

- **Isolating, packaging and transmitting only the resource files that are localization-ready**

Outbound resource files that are deemed localization-ready must be packaged in a localization kit that is then sent to language service providers for translation.

Not all resource files need to be processed at all times, especially when some may be subject to change while others are not. It is much more efficient to target the resource files with surgical precision when only working with the minimum level of relevant files that need changing. This, in turn, leaves far less work for service providers instead of some indefinite mix.

- **Testing of resource files returned from translation prior to checking them into version control**

One common risk in any localization project is that, despite best efforts, errors may be introduced during the translation process. This is because at the far end of the translation supply chain, it is ultimately a human effort, and therefore prone to human error.

A set of tests that proactively exposes typical errors thus contributes greatly to reducing friction within the latter stages of the localization process, in particular at build time and during quality control.

- **Careful reintegration of the localized resource files prior to creating localized product builds**

Frequently, the source language resource files will have been altered by developers during the time in which their localized equivalents were out for translation. For this reason, the reintegration process must include an equivalency comparison among the build-readiness checks. Proactive determination of file discrepancies alerts build engineers that a problem exists before the software build begins. They can then take appropriate actions prior to initiating the build process instead of building, then researching, then rebuilding, etc.

- **Allowing for language independence for localized product builds**

Resource files often return from the service providers in a staggered fashion. Reintegrating the localized resource files back into the build environment as soon as possible is desirable because it allows subsequent processes that depend upon them to begin immediately. With more immediate throughput, early warnings signs about internationalization issues that have snuck through source language product quality control may be detected and dealt with on a timely basis.

Fitting a Square Peg into a Round Hole

The circular or spiraling nature of Agile development presents challenges for square-edged localization processes, and these challenges tend to show up as friction points between the two process types. Here are a few examples:

- **Having localization personnel at every scrum meeting may or may not be appropriate.**

Having localization personnel at the daily scrum may or may not be appropriate. From the viewpoint of the three classical scrum questions, Agile theorists may argue that information from daily meetings in dependent processes is superfluous to the core process. Changes that stem from within the core process may profoundly affect the dependent processes, so information flow, at least from the core outward, is desirable, and must be enabled in some fashion.

From the viewpoint of localization managers, attendance at scrum meetings might incur opportunity cost for localization personnel that cannot be afforded because their time and effort are simultaneously needed on other critical-path projects.

- **Sprints impose scalability limits, especially in more complex or larger projects.**

There are practical limits on the amount of software engineering time that can be devoted to carrying out tasks such as proactively detecting internationalization bugs that will later be exposed through the localization process. As projects grow in scale and complexity, the proportional amount of time that can be devoted to ensuring that new code is fully internationalized diminishes because developers need to concentrate on remaining productive under the time pressure that sprints impose. The necessity to deliver new functionality in the product's source language by the end of the current sprint can easily trump goals pertaining to international delivery. While this facilitates the primary goals of Agile development, it imposes friction on downstream processes such as translation and quality control. This is not necessarily a good trade-off within the context of overall corporate goals and strategy.

- **Changes need to be monitored and isolated with surgical precision**

Developers need to execute change, not monitor and document it. Downstream processes rely on information about change for resource and throughput planning and budgeting, not to mention setting of external expectations. Greater precision in monitoring and isolating changes in source language resource files translates into greater accuracy for managers of downstream processes and organizations.

This is not a primary goal of Agile development, though. Just consider the fourth point in the Agile Manifesto, "Responding to change over following a plan." Agile is meant to respond to changes by customers; there is no allowance for monitoring. Combine the foregoing point with the second one to get a picture of the reality for downstream resources, "Working software over comprehensive documentation." In other words: the software works; how it does so is secondary.

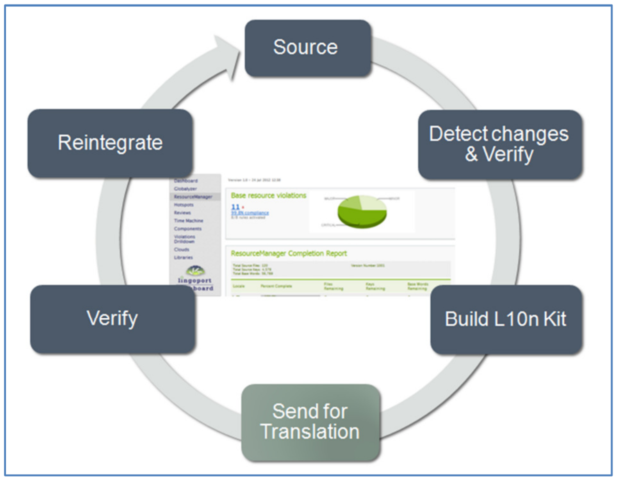
- **Software engineers should be adding product value, not doing accounting.**

Localization project managers spend a significant amount of their time tracking file location and changes, using tools to calculate word counts and turnaround timeframes. Additionally, a myriad of other "accounting" tasks intrinsic to localization but very much peripheral to core development activities falls within their responsibility. Since these tasks reside in the build environment, however, where localization personnel frequently do not have access, the question begs to ask: "Who should be responsible for providing the raw data that localization requires in a rapidly changing scenario?" This is certainly not the best usage of software developer time, yet it is a cornerstone requirement for efficient localization.

Lingoport Resource Manager: The Path to Agile Localization

Upon superficial consideration, one might be led to the conclusion that Agile development and localization are conceptually antagonistic, and therefore can never coexist in a mutually reinforcing way. This is not true. A much more accurate statement is that there are important friction points between the two methodologies. These friction points must be – and can be - bridged with smart technology that facilitates knowledge-based project component flow. With this friction gone, a state of Agile localization can be achieved.

Lingoport Resource Manager is a software solution that monitors the build environment for localization-relevant changes, flags and extracts changed resource files on behalf of localization project managers, automates the testing of both outbound and inbound, localized resource files, and automatically replaces, build-ready files back into the build environment in the proper location. This automation greatly reduces the amount of human effort required to test, transport, and track files, resulting in significant improvement in the ability of localization to keep up with the rate of change associated with Agile development practices.



More specifically, the Resource Manager automatically tests checked-in resource files that are in the build repository for localization-readiness. Only if the files pass the tests will they be copied to a "pick-up" point from where they can be retrieved by localization management and put into the localization process. Files that are returned from localization go through the reverse process; they will not be placed back into the build repository until they pass the automated testing process. This all takes place without hands-on human intervention. Stated another way, localization personnel do not have to learn how to interact with version control systems or to dig around by themselves in fragile, secure build environments, nor do they have to waste valuable time manually reviewing file content.

Automation is good, but knowledge is better! Over and above the automation it provides, Lingoport Resource Manager displays file status in an intuitive, dashboard portal. By doing so, it provides a "window" through which all stakeholders can clearly visualize project status. They can leverage the status information to make plans, manage by exception, catch and rectify systematic errors, and set external expectations. Subjective judgment about project efficiency can be replaced by objective data reporting, and the risks of human error in file management are mitigated.

Lingoport Resource Manager can be scripted so that its process steps execute automatically. For example, a reporting command set from the Resource Manager can be scheduled to execute in batch immediately after a product build. This updates the Dashboard data immediately after the regular nightly build is finished. When localization personnel come to work in the morning, they will immediately have access to the latest status reports.

With Lingoport Resource Manager, friction points between localization and Agile development teams are made frictionless through automation and reporting. Localization can become an Agile process, embracing constant change instead of resisting it.

About Lingoport

Lingoport is a trusted resource to the world's leading companies in modifying software and web applications to perform gracefully in any language or locale. Lingoport's product suite helps companies in developing software for the world by checking, measuring and fixing source code for internationalization (i18n) defects, then automating the flow of localization to keep up with ongoing development. Our extensive outsourced internationalization service offerings help companies meet challenging global release deadlines while augmenting development teams. For more information, please visit http://www.lingoport.com or contact Lingoport at sales@lingoport.com.

Visit Lingoport.com/ResourceManager for more information.