

Agile Development Methodologies for Testers

A Practical Guide to Defining Testing Practices for Agile
By Peter Varhol, Seapine Solutions Evangelist

Copyright © 2010 Seapine Software, Inc.

This work is licensed under the Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 United States License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/3.0/us/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.



Seapine is proud to be a member of the Agile Alliance.

Table of Contents

Introduction	04
Traditional Testing and the Transition to Agile	05
The Traditional Starting Point	05
Developers and Agile	06
Agile and Testers: It Only Makes Sense	08
Steps to Quality Software With Agile Processes	11

Introduction

Enterprise application development teams are increasingly adopting Agile software development techniques as the principal method of building custom applications for business advantage. Agile methodologies, such as Scrum, Extreme Programming, Feature-Driven Development, and Agile Unified Process, offer the ability to iteratively develop applications with participation from the end user community.

Whatever the Agile process used, the role for testers is not always clear. Testers who seek to better understand how testing can impact Agile development methodologies, and their roles in Agile testing, must take the lead in defining just what those roles are. By understanding Agile development processes and how testing can improve quality in these processes, testers can define their role as essential to building quality software.

Here's how testers can redefine testing for Agile development, including a list of activities that can help turn testing into an essential part of any Agile process.

Traditional Testing and the Transition to Agile

Traditional testing practices have taken their lead from the Waterfall model of software development, where requirements gathering, software design, and implementation follow a deliberate and often lengthy path. While quality remains important with all Agile practices, it can be difficult to map existing quality strategies to a new and accelerated methodology.

The Traditional Starting Point

In traditional software development methodologies, testers meticulously examine requirements, determine the appropriate way of measuring quality, and devise a plan to assure those requirements have been implemented through functional and regression testing during the latter stages of code development. The prospective software user is removed

from the process. Instead, one or more business analysts interact with users to determine the business requirements, which are passed on to developers and testers. Testers focus on developing plans and individual tests to validate the software against the requirements. Testers not only look for obvious software bugs, but also for deviations from the business requirements. In this manner, they ensure both a defined level of software quality and a specific fitness for the original purpose.

Because software is complex, testers put in a large amount of work both before and during testing. While developers are translating requirements into specifications, testers are preparing test plans and test cases. This documentation can

easily run into the hundreds or even thousands of pages. Once the development team starts building the software, testers execute the test plan, running relevant test cases as the features gradually become available. Bugs and deviations from requirements are logged and described, and developers fix bugs as well as write new code.

Developers and Agile

Agile methodologies focus on immediate and rapid code development, and less on upfront documentation. They do this through involvement by product owners from the user community, who work closely with the development team to define and validate the software. Domain experts define the functional requirements of the software through user stories,

which are high-level scenarios of how the prospective software would be used as a part of a business process.

The team, including the product owner, developers, and testers, take these user stories, organize them, and use them as a basis for designing and building applications incrementally. Typically taking three to five user stories at a time, developers will reserve anywhere from a week to a month to implement that subset of stories, depending on the Agile approach used and the complexity of the stories. The features supporting those stories are coded, and at the end of that time those features are operational. The domain expert works with developers to answer questions and resolve ambiguities as the stories are translated from business processes to software features.

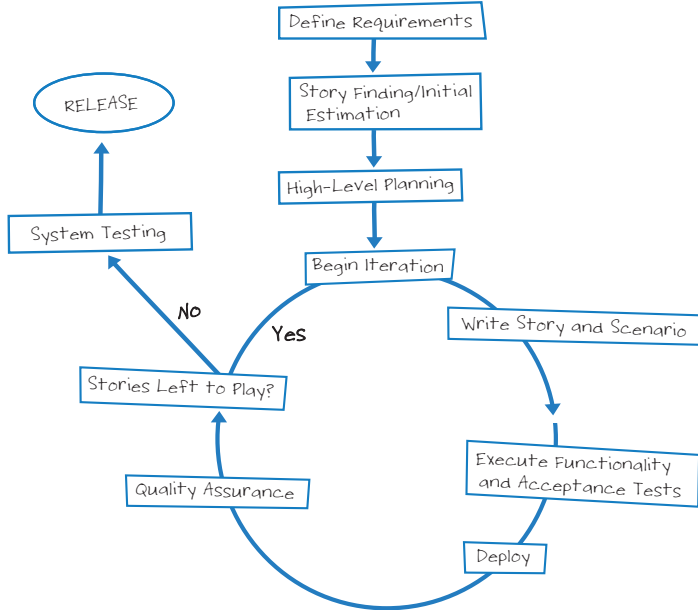


Figure 1. Agile development repeats a sequence of code, test, and deploy

After that set of stories has been implemented, the domain expert validates the software with respect to those stories, and the process begins again with the next set of user stories. This process is performed rapidly, and the application is compiled and built almost continually, so the coded and validated features can be deployed at any point in time.

Because the software can be built at the end of each sequence, and has been validated by domain experts, it can ideally be deployed to the business at any time. This means there doesn't have to be a year-long development cycle, during which the business needs have likely changed. Applications can be deployed early and often, making features available to users early, while adjusting later features in response to user feedback and changing business needs.

Agile and Testers: It Only Makes Sense

At first glance, an Agile process seems to leave little for application testers to accomplish. The connection between the end user and the developers appears to be direct, so a need to ensure the final product was what users wanted seems unwarranted. And, because the domain expert tries the software multiple times during the process, mid-course corrections seem seamless and straightforward. Developers can find and fix the obvious bugs, while domain experts ensure fitness for purpose.

However, independent testing is still vital. Quality is still one of the most important requirements in an enterprise application, and testing is the primary way of verifying quality and assuring that the process produces quality software. But testers have to substantially change their approach to doing their jobs in order to assume an essential role in Agile development.

Because of their unique position in the application development lifecycle, testers are best equipped to connect business needs with actual results. In particular, testers are in the best position to codify user stories, and to determine how to objectively

measure their completion. This is a business aspect to testing that is not typically the focus in traditional methodologies.

If the domain expert is the principal author of the user story, it's likely that he or she is making assumptions about the domain or the business process that is implicit in the story but probably not apparent to the development team. These assumptions can lead to misunderstandings with the developers, who may not fully comprehend the assumptions or point of view. Testers are in an ideal position to work with the domain expert to clarify assumptions and make them explicit, and to explain any ambiguities in those stories before developers start coding from them.

The business analyst has often filled this role. However, business analysts and testers can work together to accelerate this process. By combining responsibilities, testers and business analysts each do what they do best—turn user requirements into high-quality applications.

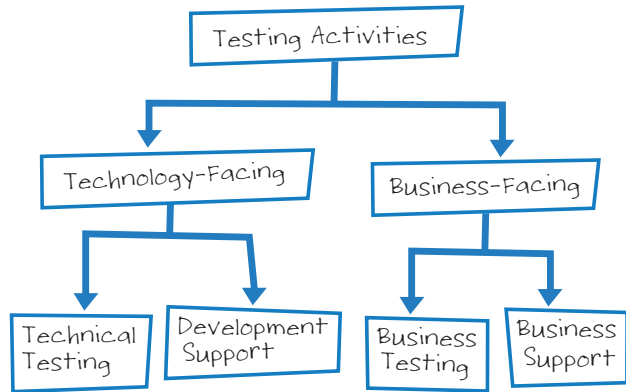


Figure 2. Testers have both technology-facing and business-facing responsibilities

Because testing is accelerated, it must be focused on the needs of the user community in addition to technical quality. All test cases should tie back to the user stories. By enabling testers and domain experts to work together to define the scope of the software, the testers are in the best position to determine how best to test the stories once they have been implemented.

Agile methodologies tend to leave acceptance criteria up to the domain expert, but that person is unlikely to be an expert in evaluating software, even against criteria he or she defined. Testers are especially qualified to fill that role because it is the essence of the profession. In traditional testing, this is referred to as acceptance testing—the practice of ensuring that every business requirement has been satisfied through a corresponding software feature. To facilitate acceptance testing, testers need to be intimately involved with translating user stories into application features. Doing so provides the ability for them to understand what the user community intended, and how it was translated into the application.

In Agile development, acceptance testing needs to be performed at the user story level. Making the crucial transitions from story to implemented feature and back to story requires participation in the story refinement, the breaking down of a story into features, and the development of cases to test those features.

This level of up-front involvement makes getting from user story to acceptance test significantly easier. Tracing the implementation of user stories has the potential to be much faster and simpler, and require far less overhead. Because developers aren't writing extensive specifications, and test plans aren't required to reflect those non-existent specifications, testers can go right to building test cases that reflect the particulars of one or more user stories. An acceptance test case should enable the product owner to verify and validate that the user story has been implemented correctly.

Virtually all of these activities can be improved and accelerated with automation. Tracking a story from inception to acceptance test can be done manually, but that requires testers to spend time examining documentation to perform that activity.

A requirements and test management package will enable testers to record user stories and resulting features, while tracking acceptance test results back to both features and user stories.

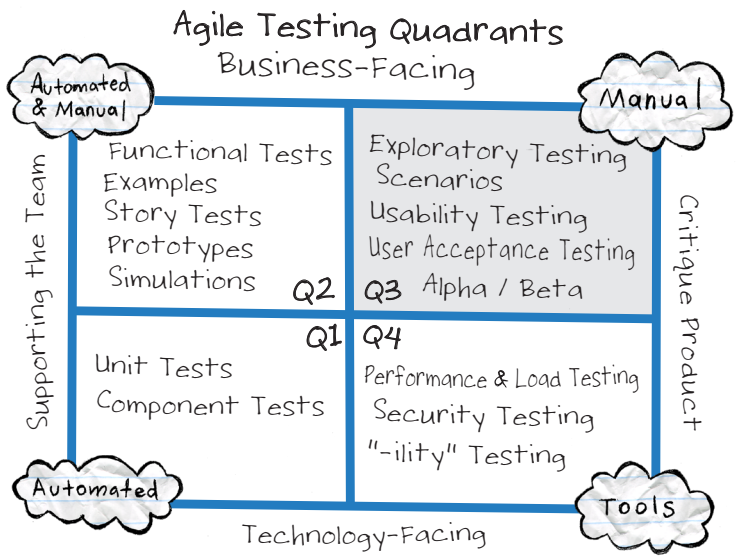
Likewise, functional, regression, and acceptance testing can be performed manually, at the cost of additional time and resources. After tests are validated, most can be executed automatically, even as part of the build process. In addition to accelerating testing and ultimately user acceptance of the application, automation ensures repeatability of an important part of the Agile development process.

Steps to Quality Software With Agile Processes

For testers to take a leadership role in assuring quality in Agile software development and delivery, they have to work with, and at the pace of, the project. Testers must work with both users and developers in defining the project and ensuring its quality and fitness for the user business needs.

Following are the key steps to preparing and executing testing on an Agile project:

1. Get to know the domain expert and user community. Understand fundamentally the business goals of the application. Your testing goals serve that group.
2. Break user stories into prioritized testing requirements and track those requirements to completion. Use automated systems to capture user stories, distill requirements, and trace requirements through to implementation and back to user stories.
3. Translate testing requirements into test cases as early as possible. Work with users and business analysts to ensure that the test cases reflect real business needs. Work with developers to devise technology-facing tests.
4. Automate test cases and test execution so that tests can be rerun automatically as part of the build process. Test automation ensures that any regression bug is caught and corrected quickly.
5. Track test case execution to ensure the fitness of the application. Be able to report on test execution at any time, so decisions can be made about application deployment.
6. Trace requirements from inception to delivery to ensure business needs have been adequately addressed. You may need to add requirements and tests later in the process, or delete obsolete requirements.



These are the major ways that testers can demonstrate value and make essential contributions to software development and delivery in an Agile process. An equal focus on both the business and the technology, with support for the Agile team plus the development project, is critical in ensuring testers make a complete contribution to project success. Automation of these activities can also ensure timeliness and repeatability of testing activities, not only in the current iteration, but future iterations as well.

Figure 3. Many testing activities can be automated, supporting project and team goals

About the Author

Peter Varhol is a well-known writer and speaker on software and technology topics, having authored dozens of articles and spoken at a number of industry conferences and webcasts. He has advanced degrees in computer science, applied mathematics, and psychology, and is currently Solutions Evangelist at Seapine Software. His past roles include technology journalist, software product manager, software developer, and university professor.