Update

Agile Product & Project Management Advisory Service
Executive Update Vol. 11, No. 16

Starting Agile Adoption:

Part I — Quality Assurance

by Steve Berczuk

Agile software development — developing an application in small increments, where stakeholders can review the results and reevaluate goals after each time-boxed iteration — is simple and powerful. However, implementing the practices that enable agile software development can be difficult because adopting an agile approach requires change across the organization. This three-part *Executive Update* series will discuss how best to start the agile adoption process.

All stakeholders, from product management to engineering, need to work in a new, more collaborative, and iterative way. Cross-organizational change is often slow, risky, and uncertain, but you can start with your QA team. Here in Part I, we will describe how to leverage your QA team at various phases of agile adoption to help a transition to agile be more successful.

THE BIG CHALLENGE: DEFINING DONE

Agile software development can appeal to all project stakeholders in the following ways:

- Short, time-boxed iterations, with tasks tracked on a burndown chart, appeals to project managers who want to better understand how much work is left to do and what new work can be done.
- Lightweight specifications that can be revisited periodically appeals to product managers who need to deal with market uncertainties.
- Clear definitions of features on a product backlog appeals to developers frustrated with vague product requirements.

Despite the value that an agile approach provides each group of stakeholders, teams are often reluctant to embrace the changes necessary for successful agile adoption. For example, engineers need collaboration and technical skills to help them work effectively with each other and other stakeholders in an incremental fashion. Product owners need to be able to develop requirements appropriate for an incremental model. And everyone needs to be able to agree about what it means for a backlog item to be complete.

Agile specification processes are meant to be lightweight, starting with user stories, which lead to conversations about the details of a story. When first writing user stories, product owners often leave out the precision necessary for effectively defining when a story is complete. Understanding how much detail to put into a story to allow you to be agile, yet also not be vague, is tricky. While there are techniques to writing good user stories,[1,2] writing good agile requirements is hard and requires practice. Until a team and product owner get through a few sprints and understand the likely points of uncertainty, the possibility is great for stories that never finish or tasks that developers mark complete before they are really done.

WHY QA IS SUITED TO LEAD A TRANSITION TO AGILE

QA is the part of the organization traditionally responsible for testing and validating that the software the development team devises matches requirements. QA is well positioned to mediate between the sometimes conflicting needs of product management and engineering.

Traditionally, testing happens closer to when a product is "complete," with the majority spent on integration testing. Product development plans need to allow for time at the end of a release to identify and fix problems. The downside of this approach is that it's difficult to have shippable though feature-incomplete code at the scheduled release date, and the product owners lose flexibility in deciding what to fix.

Agile QA avoids some of these problems because testing happens throughout the development cycle. As the gatekeepers of quality, QA engineers are often the ones who struggle most with the complexity, writing tests based on ill-defined specifications. Because of this, they are well suited to help the team meet the goals of continually preparing code in a measurable state by:

- Working with both product owners and developers to understand how much detail a story needs to be testable
- Identifying components of the code base that are more critical or fragile
- Identifying barriers to more frequent automated and manual testing

On an agile project, developers take more responsibility for automated unit testing, and the QA team tests earlier, placing a heavier emphasis on automation as well as working collaboratively with other stakeholders. On an agile team, testing is a shared responsibility.

The QA team is well positioned to help all stakeholders understand the answer to the important question: "How do we know if we are done?"

ITERATE: DO, REFLECT, IMPROVE

While the hope is often that after a few training sessions, a team can change its work style to an agile method, a more practical approach to changing a process is to proceed in an incremental and iterative fashion. For any change to be successful, teams need a structure where they can attempt, reflect, adapt, and try again.[3] Some guidelines for successful initial iterations include:

- Picking an iteration length that provides a good balance between being able to complete useful functionality and not having to spend too much time if the team is struggling (two weeks).
- At the end of each iteration, reviewing the work you did — even if you missed your goals by a wide margin.
- Having short iteration retrospectives where you decide what areas to improve on in the next iteration. Either use a facilitator or follow the guidelines in a book such as *Agile Project Retrospectives*.[4] You can gather some useful data and decide what to do next.

At the retrospective, the team creates an action plan for addressing impediments that it discovered.

QA IN AGILE ADOPTION

On an agile project, QA performs more exploratory testing and writes more automation tools, while developers focus on making units of code robust. To leverage the skills of the QA team, plan to do development in vertical (feature-oriented) slices so that QA can test functionality during an iteration and provide rapid feedback to the developers. Rather than a downstream function, QA becomes part of the development team. Integrating QA into the team encourages collaboration and reinforces the idea that quality is an "everyone, all the time" responsibility — not just a handoff. This partnership between QA and development for testing is essential for iterative software development to work, as you want to maintain working software continuously and avoid a "toss it over the wall" attitude.

During agile adoption, the QA team can drive the team to be more effective in a number of ways:

- Initially, QA can help the team understand what's making it difficult to define completeness.
- Later on, the QA team can help the product management team develop user stories that are testable.
- As time goes on, the QA team can improve automation of integration tests, helping to ensure that the team makes continual forward progress.

I'll expand on each of these in the following sections. For more about QA and the agile team, see *Agile Testing*.[5]

Early Iterations: Finding Impediments

During the first stages of agile adoption, the QA team can help the project team identify when work is "done." The team can then decide that a story cannot be marked complete until the QA team validates it. And QA should start to test the build before the end of the iteration. Adopting these rules can drive the following:

- Development in vertical slices so that the QA team can test features independently
- Always working software, without which the QA team can't start testing

- Features that are well enough defined that the QA team can determine feature completeness in a way that everyone can agree on
- Automated integration testing environments and tools so that the team can perform regression tests in a timely fashion without using too many manual resources

In the early phases of agile adoption, the team aims to understand the impediments to rapid delivery of working code. The testing activities of the QA team also provide an added degree of confidence as the development team refactors the code to make it more unit-testable. Making the code available for QA testing throughout the iteration will motivate essential agile practices, such as:

- Continuous integration
- Unit testing, so that the code builds and runs
- Automated or simplified deployment and configuration

In this phase, QA becomes an integral part of the engineering team, and not just "the testers." To help foster this collaboration, you need to create an environment that encourages the QA team to communicate with the development team directly, and not just through problem reports.

Consider the following questions at the sprint review:

- Was it possible to deliver any stories before the end of the sprint, and if not, why?
- How difficult was it to identify when a story was complete?
- Did we complete all of the stories for the sprint? If not, was the problem inaccurate estimation, which is expected, or misunderstanding of what the story was.
- How much time are people spending checking out broken code?

Better Beginnings: Defining Doneness at the Start

As the team gets more skilled in unit testing and gets a better sense of how to devise development tasks to enable frequent feedback from QA, the QA team can start to work with the product management team to help develop better backlogs.

Use the brief time at the start of each development iteration before there are features to test to have the QA team work with product owners to evaluate whether features for the next iteration are defined in a way that they can be completed and tested by the end of an iteration.

The work of the QA team in this phase will encourage:

- User stories that can be more accurately estimated
- Automated integration tests
- Adding more metrics to the CI build, such as unit test coverage

Better stories and better traceability will help develop an environment where product owners have confidence that the team can deliver work in a predictable fashion, making them more comfortable for prioritizing bugs so that they do not interrupt current work.

At the sprint review, consider the following questions:

- Is the unit test coverage improving, and do unit tests seem to add value?
- Is the team getting better about meeting their sprint commitment?
- Are the stories you get from the product owners easier to estimate and test?
- Are you getting enough access to product owners to clarify questions midsprint?

Improvement: Adding Automation

As the features in a product grow, you'll soon find that that manual testing isn't an effective regression-testing strategy. At this stage, the QA team can help:

- Define processes that encourage developer testing (for example, any bug filed needs to have a unit test as part of the fix)
- Start sooner to automate tests from the last iteration
- Introduce integration test frameworks to the development team to test "service-layer" interfaces

Even as features are completed, interactions between features may make it difficult to write automated integration tests that cover the entire application stack, so some manual testing may still be necessary. Start manually testing all features from the current iteration once they are completed, and spend the first part of the current iteration automating tests from the last one. By doing creative manual testing, the QA team identifies where best to spend energy on automation and helps developers identify parts of the code base that can benefit from unit tests.

At the sprint review, consider the following questions:

- How many regressions have their been? Are we getting good automated integration test coverage?

- Is the test coverage improving?
- Is the rate of progress improving?
- How happy are all the stakeholders with the process and the progress?

While you will still have opportunities to improve, by this point you should have a good agile rhythm on your project.

CONCLUSION

Testing is a technical skill that is central to successful agile adoption, as it enables rapid change, and drives good user stories and development in vertical slices. Quality assurance work naturally lives at the boundary between product requirements and engineering. Because of these factors, your QA team can be a central force in enabling successful agile adoption.

To use your team effectively:

- Identify a QA lead or test engineer to be a dedicated point of contact for each agile project. This person should participate in all of the planning and day-to-day activities, including sprint planning and daily Scrums. This person should be available daily to do work on the project, and work closely with the rest of the engineering team.
- Encourage the QA team to build skills in automation tools.
- Empower the QA resource to identify tasks as complete or not based on their testability.
- Encourage the QA team to do exploratory testing of builds at various stages of the sprint, not just at the end.
- Allow time to do retrospectives at the end of each of the first few iterations, and at the end of each project. You need to have a mechanism for people to figure out how to work better.

Because many of the practices of agile software development are closely aligned with those of traditional QA practices, using QA to drive agile software adoption is a natural step. In Part II, we'll discuss how to use agile planning effectively.

ENDNOTES

[1]Cohn, Mike. *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional, 2004.

[2]Cohn, Mike. *Agile Estimating and Planning*. Prentice Hall Professional Technical Reference, 2005.

[3]For more, see Manns, Mary Lynn, and Linda Rising. *Fearless Change: Patterns for Introducing New Ideas*. Addison-Wesley Professional, 2005.

[4]Derby, Esther, and Diana Larsen. *Agile Retrospectives: Making Good Teams Great*. Pragmatic Bookshelf, 2006.

[5]Crispin, Lisa, and Janet Gregory. *Agile Testing: A Practical Guide for Testers and Agile Teams*. Addison-Wesley Professional, 2009.

ABOUT THE AUTHOR

Steve Berczuk is an engineer and ScrumMaster at Humedica, where he's helping to build next-generation clinical informatics applications based on software as a service (SaaS). The author of *Software Configuration Management Patterns: Effective Teamwork, Practical Integration*, he is a recognized expert in software configuration management and agile software development. Mr. Berczuk is passionate about helping teams work effectively to produce quality software. He has a master's degree in operations research from Stanford University, a bachelor's degree in electrical engineering from MIT, and is a Certified Practicing ScrumMaster. He can be reached at steve@berczuk.com.

