QA to AQ Part Three

Shifting from Quality Assurance to Agile Quality

"Tearing Down the Walls"

Joseph W. Yoder [1], Rebecca Wirfs-Brock[2], Hironori Washizaki[3]

[1] The Refactory, Inc.,

[2]Wirfs-Brock Associates, Inc.

[3]Waseda University

joe@refactory.com, rebecca@wirfs-brock.com, washizaki@waseda.jp

Abstract. *As organizations transition to agile processes, Quality Assurance (QA) activities and roles need to evolve. Traditionally, QA activities occur late in the process, after the software is fully functioning. As a consequence, QA departments have been "quality gatekeepers" rather than actively engaged in the ongoing development and delivery of quality software. Agile teams incrementally deliver working software. Incremental delivery provides an opportunity to engage in QA activities much earlier, ensuring that both functionality and important system qualities are addressed just in time, rather than too late. Agile teams embrace a "whole team" approach. Even though special skills may be required to perform certain development and Quality Assurance tasks, everyone on the team is focused on the delivery of quality software. The patterns in this paper are focused on "breaking down the walls" or removing barriers between people and traditional roles as this is key for any change within an organization that is transitioning to being more Agile at Quality.*

Categories and Subject Descriptors
D.1.5 [**Programming Techniques**]: NEED TO ADD HERE

General Terms
Quality Assurance (QA), Software Developer in Test (SDET), Test Driven Development (TDD), Extreme Programming (XP), Pairing, Agile, Waterfall Methodology, Software Development Lifecycle (SDLC), Continuous Integration (CI), Extract-Transform-Load (ETL) Agile, Patterns, Testing

Keywords
Agile Quality, Quality Assurance, Testing

Introduction

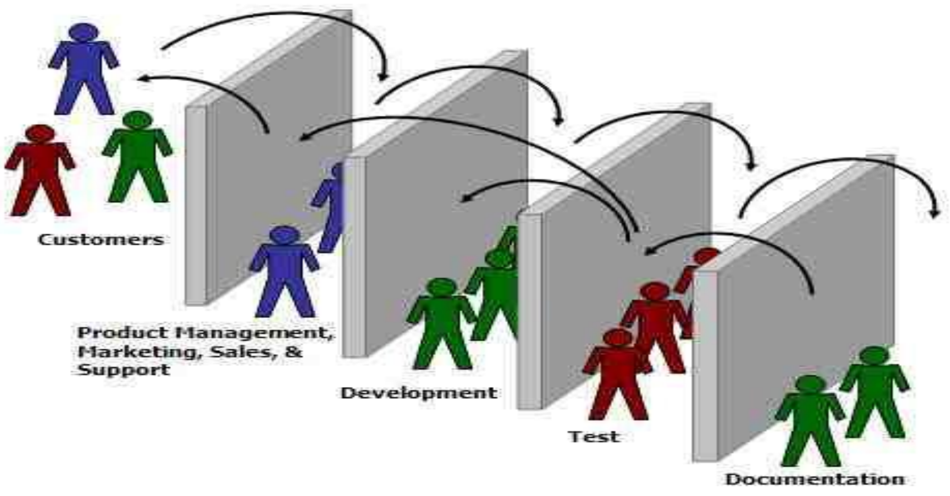
As organizations move to being more agile, it is important that this transition also includes Quality Assurance (QA). Nothing prevents QA from being involved throughout the development process, but generally this has not been the case. Unfortunately for many software projects, QA only becomes involved late in the development process, just before it is necessary to test and release the final product. This is partly because of a different mindset between in traditional software quality assurance processes over time. One important responsibility of QA is to certify the functionality of the application based upon the contract and requirements; usually with black-box tests. Typically, QA groups have worked independently from the software team. However, in agile teams, QA should work more closely with the whole team on an ongoing and daily basis.

Previously in [YWA & YW] we presented an overview of patterns on ways to become more agile at quality. This paper extends that work by writing the patterns "Breaking Down Barriers" and "Pairing with a Quality Advocate".

We have written a group of patlets listed in the appendix. A patlet is a brief description of a pattern, usually 1-2 sentences outlining the problem and solution. We are working on writing these as full-fledged patterns that can ultimately help guide organizations as they become more agile at quality. These patterns are intended for any agile team that wants to focus on important qualities for their systems and better integrating QA into their agile process. These patterns are for anyone who wants to instill a quality focus and introduce quality practices earlier into their process, too. These patterns need not just be for agile teams.

Breaking Down Barriers

"You can focus on things that are barriers or you can focus on scaling the wall or redefining the problem." —Tim Cook



Most agile processes do a good job of focusing on functional requirements, how to prioritize them, and on the process for doing the development (product owners, scrum masters, TDD, etc.). As organizations evolve to being more agile, it is important to not lose focus on the "ilities" of a system and on Quality Assurance. Most agile transitions provide training for management, developers and product owners while QA is often left to their own. There are often many barriers between Quality Assurance and other parts of the organization.

How can agile teams remove the barriers and become more agile at quality?

❖ ❖ ❖

Often there are physical barriers where the QA team is located in different rooms or buildings.

Even if QA is located in the same building, possibly in the same physical space, there can be other barriers such as cultural differences, language differences, backgrounds and expertise.

Often because of barriers and differences, QA can been seen as the obstacle to getting the product out (sometime the enemy) which can often lead to an "us and them" mentality between QA and the development team.

QA is often slammed by the forces upstream from them and they are constantly in a response mode. Although they'd like to help more there just isn't enough time or people.

Many times QA is *only* seen as the final gatekeeper. When issues arise they are seen as the problem makers blocking the release, because as testers they are not perceived as contributing to the development process and not understanding how the application is supposed to work. They may find problems that aren't deemed important because they aren't using the software correctly.

Product owners and development teams like to focus on visible items that show progress, for example the core functional requirements. This may cause them to slight important system qualities.

Developers who are writing production code and unit tests may sometimes select an approach that makes their work go fast. Consequently, they may only care about their velocity and may be aware of how their design choices may adversely impact others including those responsible for assuring quality.

QA and/or product owners can often keep the real requirements from view of the software engineers, admins, and even the Business Analyst (BA). It may be the case that a specific “contract” or piece of legislation and government regulations contains the real requirements, but the product owner or lead QA person creates their own interpretation of how to achieve those requirements. They can get something critical wrong, leading to late disclosure of the critical pieces. Development scrambles, and quality control cannot make the deadlines.

❖ ❖ ❖

Therefore, tear down the barriers or walls through various actions such as including QA early on; make them part of the sprints, embed them in the teams. Also, include time for training and reward the whole team for quality.

An important principle in most agile practices is the “Whole Team” concept, where people work together to produce a high quality product. It isn’t just testers who need to care about quality. Everyone on the team needs to care about quality, even though they bring different strengths and experiences to their work. Having QA as part of team from the start can help build quality into the system and make quality an integral part of a more streamlined process. This helps others on the team to know what system qualities are important and how they fit into the process (when to do what for different qualities). Another benefit of including QA is that they can help others understand and validate requirements.

There are many ways to break down the walls. Have QA fully participate in the team’s estimation sessions. If they are located in another area, have QA specialists move to the same space and participate as part of the same team. Have the Product Owner (PO), development team, and QA sit in the same room and be part of planning prior to the upcoming sprint. As items are assessed, QA can use this opportunity and their experience to point out “ilities” that may be overlooked and need to be addressed. They can point out risks and help create high level tests and integration points across teams.

QA in agile groups can be more proactive, working to ensure quality across all levels of the development process. They can work closely and coordinate between business, management and developers. Additionally, during sprints developers can “Pair with a Quality Advocate”.

Trying to use summer interns and hiring someone remotely has shown to not work so well in the long term. A much better approach is to grow the QA expertise and make it part of your team from the start. It is a long-term commitment to quality throughout the whole process.

If you do not have enough QA people to put them on all your development teams, start out by having them rotate between teams, pairing on some of the daily tasks. You can then grow your quality expertise. Some QA testing so highly specialized that you can use this same approach to get functional testers to become more skilled at load testing and other types of testing by pairing them with performance QA experts.

Many of the *Fearless Change Patterns* [MR] can help you overcome the barriers and get buy in from the teams and high-level management. You may need to Ask for Help, locate a Corporate Angel, address Corridor Politics, Build Bridges and Keep Things Visible. It is important to retrospect and take Time for Reflection [MR], as you evolve teams to ensure quality and safety as you grow and adapt your ways of working.

Pair with a Quality Advocate

"Unity is strength... when there is teamwork and collaboration, wonderful things can be achieved."—Mattie Stepanek

Agile developers write unit tests to exercise and validate system functionality. While unit tests are important, there is more to quality than simple unit testing. Good functional testing can be difficult at times let along trying to understand and test the important system qualities.

How can agile developers get the most out of validating the system, especially when it comes to being able to understand and test system qualities?

❖ ❖ ❖

Not focusing on important qualities early enough can cause significant problems, delays and rework. Remedying performance or scalability deficiencies can require significant changes and modifications to the system's architecture.

While agile developers are good at developing based upon the requirements from user stories, QA has a lot of expertise understanding system qualities and how to validate these.

Time-boxing lengths that are suitable for some team members may be inappropriate for another. Product Owners and developers may need a few days to address certain issues in the current functional spec, while the impact on design and QA could take many weeks.

Focusing on non-functional requirements can sometimes distract from important functional requirements outlined by the product owner.

Developers working on that validate the core functional requirements and some system qualities often overlap their work with testing and validation done by QA.

QA can be seen as trying to tell developers how to build and design the product without having sufficient background to articulate all the details. Software developers often discount comments from QA, because they are perceived to be inarticulate because they come from people who do not and cannot write production code and could not possibly understand all the issues. Developers grow impatient and want more details. For lack of specificity, developers start filling in details and implement what the product *probably* should do, possibly compromising testability.

❖ ❖ ❖

Therefore, pair developers with quality assurance to complete quality-related tasks that involve programming.

This paring can be achieved in many ways such as including QA through all phases of the sprint, including sprint planning, development, and closing out the sprint. A good experience report on different variations can be found [Hil].

During program tasks, pair QA members directly with developers. This includes QA sitting with the developers and helping them design the tests (both better unit tests as well as those that focus on system qualities). Developers pairing with QA can also create integration tests in addition to unit tests.

One organization noted that they were able to greatly reduce duplication on tests efforts [Sav]: “We found that we had a 50% duplication rate. Fifty percent of the automated tests that our SDET’s had written were also in the developer’s unit test suite. These tests, consisting of unit, happy path and some negative tests had already passed and did not need to be written and run again by a different team. This was waste. Waste of time and resources that could be reclaimed in our new methodology.”

Summary

This paper is a continuation of patterns for shifting from Quality Assurance (QA) to Agile Quality (AQ). The complete set includes ways of incorporating QA into the agile process as well as agile techniques for describing, measuring, adjusting, and validating important system qualities. This paper focuses on two core patterns for overcoming barriers to becoming more agile at quality. Ultimately it is the authors' plan to write all of the patlets as patterns and weave them into a 3.0 pattern language for evolving from Quality Assurance to an Agile Quality mindset.

Acknowledgements

We thank our shepherd Juan Reza for his valuable comments and feedback during the SugarLoaf PLoP 2014 shepherding process. We also thank our 2014 SugarLLoafPLoP Writers Workshop Group, xxx, yyy, and zzz, for their valuable comments.

References

[Hil] Hile E., "Head On Collision: Agile QA Driving In A Waterfall World," Agile 2014 Conference, Orlando, Florida, USA.

[Iba] Iba, T. 2011. "Pattern Language 3.0 Methodological Advances in Sharing Design Knowledge," International Conference on Collaborative Innovation Networks 2011 (COINs2011).

[MR] Manns, Mary Lynn and Rising, Linda, Fearless Change: Patterns for Introducing New Ideas, Addison-Wesley, 2005.

[Sav] Savoia S., "Tearing Down the Walls: Embedding QA in a TDD/Pairing and Agile Environment," Agile 2014 Conference, Orland, Florida, USA.

[YWA] Yoder J., Wirfs-Brock R., and Aguilar A., "QA to AQ: Patterns about transitioning from Quality Assurance to Agile Quality," 3rd Asian Conference on Patterns of Programming Languages (AsianPLoP 2014), Tokyo, Japan, 2014.

[YW] Yoder J. and Wirfs-Brock R., "QA to AQ Part Two: Shifting from Quality Assurance to Agile Quality," 21st Patterns of Programming Language Conference (PLoP 2014), Monticello, Illinois, USA, 2014.

Appendix

A previous paper on this topic outlines some core patterns when evolving from traditional quality assurance to being agile at quality [ref]. We outlined all that patterns using patlets. A patlet is a brief description of a pattern, usually one or two sentences. Following is an excerpt from that paper outlining the patlets.

Central to successfully using these QA patterns is knowing where quality concerns can fit into your agile process. The following patlet describes those considerations.

Patlet Name	**Description**
Breaking Down Barriers	Tear down the barriers between QA and the rest of the development team. Work towards engaging everyone in the quality process.
Integrating Quality into your Agile Process	Incorporate QA into your process including a lightweight means for describing and understanding system qualities.

Identifying Qualities

An important but difficult task for software development teams is to identify the important qualities (non-functional requirements) for a system. Quite often system qualities are overlooked or simplified until late in the development process, thus causing time delays due to extensive refactoring and rework of the software design required to correct quality flaws. It is important in agile teams to identify essential qualities and make those qualities visible to the team. The following patlets support identifying the qualities:

Patlet Name	**Description**
Finding the Qualities	Brainstorm the important qualities that need to be considered.
Agile Quality Scenarios	Create high-level quality scenarios to examine and understand the important qualities of the system.
Quality Stories	Create stories that specifically focus on some measurable quality of the system that must be achieved.
Specify Measureable Values or System Qualities	Specify scale, meter, and values for specific system qualities.
Fold-out Qualities	Define specific quality criteria and attach it to a user story when specific, measurable qualities are required for that specific functionality.
Agile Landing Zone	Define a “landing zone” that defines acceptance criteria values for important system qualities. Unlike traditional “landing zones”, an agile landing zone is expected to evolve during product development.
Recalibrate the Landing Zone	Readjust landing zone values based on ongoing measurements and benchmarks.
Agree on Quality Targets	Define landing zone criteria for quality attributes that specify a range of acceptable values: minimally acceptable, target and outstanding. This range allows developers to make tradeoffs to meet overall system quality goals.

Making Qualities Visible

It is important for team members to know important qualities and have them presented so that the team is aware of them. The following patlets outline ways to make qualities visible:

Patlet Name	**Description**
System Quality Dashboard	Define a dashboard that visually integrates and organizes information about the current state of the system's qualities that are being monitored.
System Quality Radiator	Post a display that people can see as they work or walk by that shows information about system qualities and their current status without having to ask anyone a question. This display might show current landing zone values, quality stories on the current sprint or quality measures that the team is focused on.
Qualify the Roadmap	Examine a product feature roadmap to plan for when system qualities should be delivered.
Qualify the Backlog	Create quality scenarios that can be prioritized on a backlog for possible inclusion during sprints.
Quality Chart	Create a chart or listing of the important qualities of the system and make them visible to the team; possibly on the agile board.

Being Agile at Quality

In any complex system, there are many different types of testing and monitoring, specifically when testing for system quality attributes. QA can play an important role in this effort. The role of QA in an Agile Quality team includes: 1) championing the product and the customer/user, 2) specializing in performance, load and other non-functional requirements, 3) focusing quality efforts (make them visible), and 4) assisting with testing and validation of quality attributes. The following patlets support “Becoming Agile at Quality”:

Patlet Name	**Description**
Whole Team	Involve QA early on and make QA part of the whole team.
Quality Focused Sprints	Focus on your software’s non-functional qualities by devoting a sprint to measuring and improving one or more of your system’s qualities.
QA Product Champion	QA works from the start understanding the customer requirements. A QA person will collaborate closely with the Product owner pointing out important Qualities that can be included in the product backlog and also work to make these qualities visible and explicit to team members.
Agile Quality Specialist	QA provides experience to agile teams by outlining and creating specific test strategies for validating and monitoring important system qualities.
Monitoring Qualities	QA specifies ways to monitor and validate system qualities.
Agile QA Tester	QA works closely with developers to define acceptance criteria and tests that validate these, including defining quality scenarios and tests for validating these scenarios.
Spread the Quality Workload	Rebalance quality efforts by involving more than just those who are in QA work on quality-related tasks. Another way to spread the work on quality is to include quality-related tasks throughout the project and not just at the end of the project.
Shadow the Quality Expert	Spread expertise about how to think about system qualities or implement quality-related tests and quality-conscious code by having another person spend time working with someone who is highly skilled and knowledgeable about quality assurance on key tasks.
Pair with a Quality Advocate	Have developers work directly with quality assurance to complete a quality related task that involves programming.