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
• **Software and its engineering~Agile software development** • **Social and professional topics~Quality assurance** • *Software and its engineering~Acceptance testing* • Software and its engineering~Software testing and debugging

General Terms
Agile, Quality Assurance, Patterns, Testing

Keywords
Agile Quality, Quality Assurance, Software Quality, System Qualities, Testing, Patterns, Agile Software Development, Scrum, Quality Related Acceptance Criteria, Agile Quality Scenario, Whole Team

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. 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 can and should work more closely with the whole team on an ongoing and daily basis.

As organizations evolve to being more agile, it is important to not lose focus on Quality Assurance and the "ilities" of a system and how QA can contribute to quality while engaging more deeply with the agile team throughout the development process. Most agile transitions provide training for management, developers and product owners while QA is often left on their own. Central to successfully using these QA patterns is identifying how quality concerns and activities can better fit into your agile process. This includes ways for *Breaking Down Barriers* and *Integrating Quality into your Agile Process*.

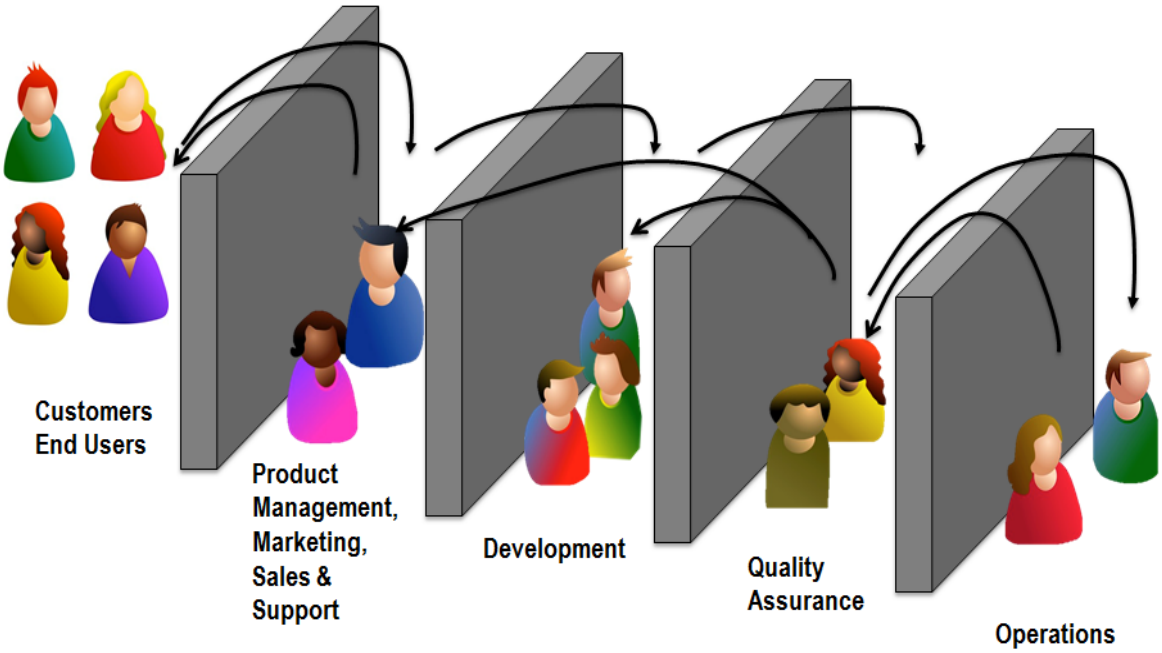
Our patterns are written in the spirit of Edward Deming's fourteen principles for business transformation and improvement [De]. Consequently, our patterns focus on actions for improving software quality and integrating QA concerns and roles into the whole team. We recognize that programming and development practices significantly contribute to or detract from software quality. Since others have written much about programming, design and architectural practices [Ast, Bec02, Bec04, CB, Mar, Sho, WY] we turn our attention to those often overlooked organizational and QA related actions that also improve software quality.

Previously in [YWA & YW] we presented an initial set of patterns on how to be agile at quality, classifying them into these categories: Fitting Quality into your Agile Process, Identifying Qualities, Making Qualities Visible, and Being Agile at Quality. There are often many barriers between Quality Assurance and other parts of the organization. This paper extends our previous work by writing the patterns "*Break Down Barriers*" and "*Pair with a Quality Advocate*."

We have written a group of patlets outlining all of our patterns in the appendix. A patlet is a brief description of a pattern. We are working on writing all of these as full-fledged patterns that can ultimately guide organizations as they become more agile at quality. These patterns are intended for any agile team that wants to focus on important qualities of their systems and better integration of 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.

Break 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 a collaborative environment that fully engages product owners, scrum masters and the development team. However, there are other roles and tasks that are important to the success of any software project. QA needs to verify correct functioning of the system as well as ensure that it meets its quality requirements. Architects may design infrastructure and implement architectural features. Groups or individuals that contribute to an agile project need be engaged and feel like valued contributors to the software's success. This is especially important for contributors who may not be engaged full time on the project.

Barriers can exist between people who are not "inside" the development team or feel tangential to it. This can lead to people focusing only on their job and not the overall progress of the team and the quality of the software. This can lead developers to view quality concerns as being "someone else's" problem and viewing QA as the obstacle to getting the product out. And it can lead to QA feeling frustrated that they aren't being listened to or understood.

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

❖ ❖ ❖

Many times QA is *only* seen by others on the team as the final gatekeeper. If QA is only verifying the software at the end of the release cycle they can be slammed by the forces upstream from them and constantly be in a response mode. Although they'd like to help more there just isn't enough time or people. When quality issues arise late, QA is then perceived as the problem makers blocking the release. Sometimes, as testers they are perceived as not actually contributing to the development process nor really understanding how the application is supposed to work.

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 or miss 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 not be aware of how their design choices can adversely impact others including those responsible for assuring quality.

QA and/or product owners may inadvertently 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 or government regulations contains the actual requirements, but the product owner or lead QA person creates their own interpretation of how to achieve those requirements. They can inadvertently get something critically wrong, leading to late disclosure of the critical requirements. Development scrambles, and quality control cannot make the deadlines.

There are different barriers across an organization. These barriers can be physical, experiential, or cultural. Not all these barriers are bad. Sometimes barriers can protect teams from outside forces which distract them from completing their tasks. However, some barriers can cause problems. When there are physical barriers such as the QA team being located in separate rooms or buildings, the distance between groups can magnify their differences. Those that are outside the core development team can feel isolated, which can result in an "us" versus "them" mentality. Even if QA is located in the same building, possibly in the same physical space, there can be cultural and language differences, as well as differences in backgrounds and expertise.

Therefore, tear down those barriers or walls that impede communication through various actions such as including QA early on; make them part of the sprints, embed them in the teams, 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 helps instill a quality mindset into the team and makes quality concerns an integral part of a more streamlined process. QA can help everyone on the team to know what system qualities are important and how they fit into the process (when to do what for different qualities). QA still focuses on system quality and their role in assuring quality, but at the same time it is important for others in the team to feel responsible for the overall quality goals.

There are many ways to break down barriers between QA and the rest of the team. 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 team bringing their expertise to the group. 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 become a *QA Product Champion* who points out risks and helps create high level tests and integration points across teams.

A benefit of including QA early and often is that they can help others understand and validate requirements. Early involvement also helps QA to understand the overall project vision and to find better ways to see that quality goals are met. QA in agile groups can be more proactive, working to ensure quality across all aspects 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*.

One of the best ways to break down barriers is to try and locate people physically closer to each other and make sure they have regular contact. For example, it might make sense to fly people in if they are located in different parts of the country or the world. This is especially useful early in the project to build closeness within the team and mutual understanding. Also, if the teams are not co-located, then having regular virtual online meetings can help keep different concerns in perspective and help the *Whole Team* feel connected. If QA is located in the same area as the development teams, then the QA team can collocate with the development teams several times during the week. If there are not enough QA people, they can rotate around sharing experience and getting insights into the project. Developers can also contribute to QA by *Spreading the Quality Workload*.

If you do not have enough QA people for all of your development teams, start out by having them rotate between teams, pairing on some daily tasks. Focus on what adds value. Forcing QA to write or review production code or participate in every standup may not be productive. Some QA testing is so highly specialized that you can *Shadow the Quality Expert* to help functional testers to become more skilled at load testing and other types of testing by pairing them with *Agile Quality Specialists* who are skilled at system performance, scalability, or other system qualities.

Trying to use summer interns or 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 agile team from the start. This can be done through training and mentoring. It is a long-term commitment to quality throughout the whole process.

Many of the *Fearless Change Patterns* [MR] can help you overcome the barriers and get buy in from team members 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. People with specialized skills, such as architecture, documentation, QA, or UX designers, who are not dedicated fulltime to an agile project, can be made to feel part of the team is by assigning them to support the team on an ongoing basis [RN]. They can continue meet and work with the team as needed with the agile team.

Pair with a Quality Advocate

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

"Look! It's paired programming!" by Ted & Ian is licensed under CC BY 2.0

There is a lot more to quality than just building and testing a system. There are many systems qualities and other considerations that must be undertook before the system is ready for production. Not focusing on important system 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 a Product Owner is responsible for prioritizing the work of the team, she may not be aware of how the system is tested and how testing for system qualities differs from the unit tests developers are writing. This can lead to ignoring quality-related tasks or misunderstanding their contribution to achieving overall product quality.

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

How can the agile team build quality into the system, especially when it comes to understanding and testing for system qualities?

❖ ❖ ❖

While agile developers are good at developing based upon the requirements from user stories, they are usually not as experienced on testing system qualities. QA has a lot of expertise understanding system qualities and how to validate them.

Focusing on non-functional requirements can sometimes distract from important functional requirements outlined by the product owner. Product owners, developers, and end-users like to see implemented functionality as it gives a feeling of progress. However, the system is not ready for release into production until non-functional requirements have also been addressed.

Developers working on the system implement and verify the functional requirements with tests they write. Often functionality impacts system qualities. Developers might not become aware of system quality limitations or requirements until testing and validation is done by QA. If there is not good communication between QA and the development team, it can be difficult to address quality concerns early.

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. Some developers may discount comments from QA, because they do not and can not 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 and other agile team members with quality assurance to complete quality-related tasks and to spread QA knowledge and quality perspectives.

QA is much more than just testing and validating the software. There are important quality considerations that if the team is made aware of can help with the success of the system. Involving QA throughout the process and pairing can help the team understand key qualities and how to validate and think about them. It is important to *Plan for Responsible Moments* [GWY] so that important system qualities are addressed at the most appropriate times.

QA can become a *QA Product Champion* by pairing with the Product Owner or Project Manager to inform them about how the software is expected to work and what kinds of testing is going on: "On more than one occasion, I've paired briefly with a project manager (PM) who wants to know more about testing, more about the product, or at least more about what remains to be done from the testing team. When I've tested with PMs, the time needed to test seems to become clearer to them, which in turn improves our communications and helps us better forecast time expectations [Joh]."

During programming tasks, QA members can pair with developers. When QA sits with the developers they can help them design better unit tests as well as those that focus on testing system qualities. Developers pairing with QA can also jointly create integration tests.

Pairing developers with testers is not new to the agile community [Joh, Koh, Mon]. Pairing can be achieved in many ways, starting with initial sprint planning and continuing through development, and even in production. Insightful experience reports and books highlight different ways QA can be strong advocates and collaborate with other team members throughout the development cycle [Bro, GC, Hil]. *Pairing with a Quality Advocate* supports the *Whole Team* mindset by *Spreading the Quality Workload* throughout the agile team.

The synergy achieved by pairing with a quality advocate has mutual benefits. Testers typically focus on testing from the users' perspective. As they pair with developers, testers gain a deeper understanding of how the software works behind the scenes. This helps QA to identify potential areas that might require more testing as well as help them better isolate defects. Developers learn much about testing boundary conditions, good interfaces, and input validation from QA, as well as what conditions might lead to potential failure. Pairing need not be constant. As [Joh] observes, "Once developers applied what they learned about testing to their development projects, they came up with great testing scenarios on their own. Best of all, it became difficult to find defects in the developer's code after pair testing."

One organization noted that they were able to greatly reduce duplication on test 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."

Pairing QA with a DBA is useful, especially when isolating data schema or data access issues. QA often has a deeper understand of how the software is expected to function than the DBA. Both come to better appreciate each other's concerns and get ideas for improved testing and validation of the system.

An architect can pair with QA to define the test architecture strategies [GWY] or to assess the impact of particular testing results on a specific system quality. When tests show a particular system quality attribute slipping below the minimum in an Agile Landing Zone [YW], QA works with the architect to address the issue. QA often pairs with the architect and development team to work on benchmarking the current software or extrapolating projected values for specific performance or scalability attributes when production system measurements aren't feasible.

Pairing isn't always successful, and there are some risks to be aware of. According to Janet Gregory [Koh], "If one or the other goes in with the idea that it is a one-way learning experience, the experience will fail." Pairing is only effective when there is mutual respect and trust. If there are gaps in understanding, whoever is "driving" during pairing needs to ensure that the other party is actively participating and understands what is going on. The goal is to build on mutual respect and trust, working collaboratively. For example, the developer should not try to turn the QA person into a developer and vice-versa, rather they build upon each other's strengths. For example, the QA expert can pair with a developer to add specific details to *Agile Quality Scenarios* or determine the best testing strategy for them. Or QA can work with the developer to understand the implications of a User's Story's *Fold-out Qualities* and how best to test that they are met.

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; *Break down Barriers* and *Pair with a Quality Advocate*. 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 comments and feedback during the SugarLoaf PLoP 2014 shepherding process. We also thank our 2014 SugarLoaf PLoP Writers Workshop Group of Arisa Kamada, Fabio Kon, Jorge Melegati, Masafumi Nagai, and Takashi Iba for their valuable comments.

References

[Ast] Astels, D., Test-Driven Development: A Practical Guide. Second edition, Prentice Hall, 2003.

[Bec02] Beck, K., Test Driven Development: By Example. Addison-Wesley, 2002.

[Bec04] Beck, K., Extreme Programming Explained: Embrace Change (The XP Series), Addison-Wesley, 2004

[Bro] Brodwell, J., "Remote Pair Programming," www.slideshare.net/ jhannes/2013-0807-agile-2013-remote-pair-programming, August 2013.

[CB] Coplien, J. and Bjørnvig G. Lean Architecture: for Agile Software Development, John Wiley & Sons Ltd, 2010.

[De] Deming, W. Edwards, *Out of the Crisis*, MIT Press, 1986.

[GC] Gregory, Janet and Crispin, Lisa, More Agile Testing Learning: Journeys For the Whole Team, Addison-Wesley, 2015.

[GWY] Guerra, E., Wirfs-Brock, R., and Yoder J., "Patterns for Initial Architectural Design on Agile Projects," 4th Asian Conference on Patterns of Programming Languages (AsianPLoP 2015), Tokyo, Japan, 2015.

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

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

[Joh] Johnson K., Improve Your Testing and Your Testers with Paired Testing. Inform IT April 2010

[Koh] Kohl J. 2004. Pair Testing: How I Brought Developers into the Test Lab. of Better Software, Volume 6, Number 1, Pg. 14-16, JANUARY 2004.

[Mar] Martin, R., Agile Principles, Patterns, and Practices in C#. First edition, Prentice Hall, 2006.

[Mon] Montvelisky J., Pair-programming for testers. In On-Going Improvement, Test Process, October 2010

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

[RN] Raines, Brandon and Neher, Judy, "No Way! Agility in the Federal Government", Agile 2014 Conference, Orlando, Florida, USA.

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

[Sho] Shore, J. and Warden, S. The Art of Agile Development, O'Reilly Media, Inc. 2008.

[WY] Wirfs-Brock R. and Yoder J., "Patterns for Sustaining Architecture," 19th Patterns of Programming Language Conference (PLoP 2012), Tucson, Arizona, USA, 2012.

[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

Previous papers on this topic have been published which outlines some core patterns we found when evolving from typical quality assurance to being agile at quality [YWA, YW]. We outlined the patterns using patlets in the tables below. A patlet is a brief description of a pattern, usually one or two sentences. The patlets in **bold** have been written up as patterns in these previous papers. We break our software-related Agile Quality patterns into these areas: identifying system qualities, making qualities visible, fitting quality into your process, and being agile at quality assurance. Our ultimate goal is to turn all patlets into full-fledged patterns and make a pattern language for action and change useful to software teams who want to become more agile about system quality.

Core Patterns

Central to using these QA patterns is breaking down barriers and knowing where quality concerns fit into your agile process. The following patlets describes these considerations.

Patlet Name	Description
Break Down Barriers	Tear down the barriers between QA and the rest of the development team. Work towards engaging everyone in the quality process.
Integrate Quality	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
Find Essential Qualities	Brainstorm the important qualities that need to be considered and list them for inclusion on the roadmap.
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 Measurable 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.

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.
Monitor 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.