From Dinosaur to Cutting Edge: 5 Practical Keys to Avoiding Extinction in An Agile QA World

Robert Gormley

hotspur99@hotmail.com

Abstract

The advent of the mobile world has placed product quality and customer satisfaction squarely on the shoulders of the company's IT department because the speed and transparency of customer feedback in the world of social media means poorly designed or low quality software goes "viral" for all the wrong reasons. The response of the IT community to this challenge has been to rid itself of long-winded iterative software development methodologies like the Rational Unified Process (RUP) or Waterfall. In their place, dynamic, ever-evolving methodologies like Agile and Scrum have begun to dominate the landscape like a ferocious T-Rex because they offer more flexibility in the development process and expedite the time to market. While the adoption of a new development methodology can be difficult on all departments, the transition of the QA Department to Agile is laced with the harsh realities of having to overcome a change to their philosophic core and meeting the challenge of aggressive skill adaptions required for Agile.

Based on my experiences (from skeptical beginner to implementation leader) and supported by research, this paper points out 5 practical keys for making Agile QA more successful for those QA departments already engaged in Agile. These keys may not provide a Rosetta stone for perfect Agile QA but the unique perspectives derived from them may help other practitioners of Agile QA in their march to avoid extinction.

Biography

Robert Gormley is currently a Principal Consultant at SWAT Solutions where he focuses on executive level TQM strategy and developing bleeding edge test automation frameworks. He has worked for 12 years in Quality Assurance and has worked his way up through the ranks in industries like finance, healthcare, retail, security software, and device QA. From his early days to present day, Robert's focus has always been on creating lean, flexible processes that allow organizations to produce quality software that meets a consumer's needs. Leveraging his Master's in Education, Robert has passionately trained and mentored all types of resources (regardless of title or function) in the ways of Total Quality.

Robert holds a Master's in Education from Loyola College in Baltimore as well as BAs in English and History from the University of Maryland at College Park.

1. Introduction

About 6 years ago, I was asked by my wife (who also happens to be in software QA) what I thought about Agile. Her company had just made the announcement that they were going to transition to Agile. Based on some quick research and a lot of skepticism, I answered her in the following way:

> Well, you better start looking for a new job. Agile isn't a methodology, it's an excuse for people not to write down requirements and a way for business to take control of the development process. People like your product manager who knows nothing about IT are going to be running your life. Good luck!

As I review my response today, I note the general sense of fear I had about Agile and what it would mean for QA. One of the most difficult things to imagine for me at the time was how anyone in QA would be able to test without a comprehensive requirements document. The principles of RUP and Waterfall were so ingrained in my way of doing QA that the idea of changing my thought process was both scary and anger inducing. I'm not alone either as a study in 2012 shows:

Out of over 200 participants**,** 64 percent said that switching to Agile Development was harder than it initially seemed. Forty percent of respondents did not identify an obvious benefit to the practice. Out of those who did, 14 percent thought it resulted in faster releases, and 13 percent – that it created more feedback. Seven percent of participants noted that Agile developers were happier due to reduced future planning and documentation.[1]

The study's conclusions show how much psychology plays into the successful transition to Agile. At the time, I wouldn't have been part of a successful team because I was defiant on what Agile was and what the benefits were.

I now realize how naïve I was and how much of what I knew about Agile was based on fear of the unknown and how immensely dependent I was on highly inflexible processes. I had spent 6 years of my professional career dedicated to learning and mastering RUP and Waterfall then suddenly the world I knew was evolving into something I didn't recognize. The other fear I had was based on my lack of technical ability. I wasn't an automation guy; I was a thinker, a planner, a strategist. The skill set for an Agile world seemed to favor someone who could do development work along with QA. How on earth was I going to fit in if I wasn't willing to adapt my skills to the new world order?

Fast forward to today and the experiences I've had with Agile for the past 6 years have made me a better QA professional. But for every 1 person like me, I have encountered what seems like 5 others who have had a terrible experience with Agile and who continue to struggle with the transition. Being in leadership positions, I've had the unique privilege of being able to help implement Agile and have been able to watch with both joy and terror as companies I've worked for have tried to transition to Agile development. To this point, I've been able to identify 5 practical keys to being successful in the Agile transition when it comes to QA teams.

2. Collective Commiseration Can Be Positive

Nothing seems to bring people together more than being able to commiserate about a shared experience. What I've learned in my Agile transitions is that collective sympathy can be a positive thing for your team and it can contribute to providing motivation[2]. It's hard to change the way someone approaches their profession, especially if they have spent years becoming an expert in it and they feel like they haven't had much say in the decision to change the way things are done. Let people "vent" while you empathize or you risk being one of the organizations that fails because Agile is "forced" on a team that is accustomed to other methods.[3] What I've found in my teams is 3 types of commiseration bubbles to the top: warm fuzzies, cautious optimism, and destructive blowhards. Managing and encouraging the first two kinds of commiseration while dealing swiftly with the third can make or break an Agile QA transition.

2.1 The Warm Fuzzy

Everyone hopes that the team will feel warm fuzzies when they transition to a new methodology which may be unfamiliar. I've always been surprised that the most unlikely of people will experience the warm fuzzy the quickest. These feelings have often been expressed by people saying things like "wow, I really feel more involved in this process" or "I never knew how our business worked but now I do having sat with the business owners". Another key indicator of warm fuzzies is the body language. Instead of shying away from having people share personal space, invitations for people to come over and talk increase. Instead of standing up with your arms folded, people relax and smile. If you see a warm fuzzy, encourage it. Also, don't be afraid to add a warm fuzzy by praising someone (especially someone who may have been skeptical to begin with) who has really embraced the Agile transition.

2.2 Cautiously Optimistic

Let's face it; there are 2 types of people in this world: people who see the glass half full and people who see the glass as half empty. Don't let the half empty people get you down, see it as an opportunity to step in and work to alleviate feelings of confusion, fear, or frustration. It is natural for people who are transitioning into something new to have these feelings; success in the transition can be determined by how you manage these feelings. Empathize with everyone and give them a forum to voice their opinions so they feel like they have a stake in the game. Work with your team to make sure they feel like they own the process, even if the process isn't fully developed. I recall a quote from Scott Barber that is apropos here: The reality is that Agile is far more a mindset and a culture than it is a collection of practices, processes and tools.[4] Focusing on making people feel comfortable is just as important as outlining new processes or pushing people to learn new skills.

2.3 Destructive

Regardless of how much you try, you'll always have one or two people who just won't gracefully accept change. They become the individuals that seek to make destructive comments to try and undermine both the process and the people. These individuals say things like "I hate this, it's never going to work" or "I'm not going to do it that way and no one can make me." It is best to identify people who may fall into this category as quickly as possible and move them. Find them a new role in the organization or move them to a role where they are more comfortable. Nothing destroys team chemistry and momentum more quickly than a squeaky wheel that you can't fix. There's a pretty good chance that they won't hold up their end of getting work done too. Don't let one or two people ruin the experience for the team.

3. Training and Communication Are Key

When faced with the task of transiting into a new development methodology, most IT resources are faced with some skills learning curve in order to become truly effective team members. For Agile QA teams, the skills adaptation process is aggressive as the time line for adoption is relatively short and the skills required fairly complex. From relearning to communicate to learning new tools, QA teams are asked to reshape their knowledge domain quickly. In order to help facilitate a speedy and successful transition, training is a must.

3.1 Training

One of the most obvious answers to the challenges associated with an Agile transition is training and the most common solution is to hire an Agile coach or a scrum master to the project team. While there may be great Agile coaches and scrum masters out there that can and do understand QA, I have found that few have the experience or the chops in QA to be able to help QA Teams with their transition. If an Agile coach or scrum master is not the answer, then where should a QA Team turn for help and training? In my experience, the best resources have been other QA managers who have already gone through the Agile transition, Agile QA specific forums/blogs, and QA conferences/classes. QA managers who have already gone through an Agile transition, whether successfully or not, can arm you with 2 powerful pieces of information:

- What worked well – tools, tips, processes that helped their team succeed in their transition
- What didn't work well – retrospective advice on what they wish they would have done differently so that their transition would have been smoother

One of the great things about talking to a QA manager who has gone through the Agile transition is that the information you receive from them will directly relate to what you ultimately have to accomplish. Find a resource or two you can tap into and do not worry if the resource you find is not involved in the same business field.

The cost of training QA resources can be a deterrent for many companies not only in dollars spent but also in productivity lost. Some of the best free resources available for QA teams in transition are the Agile QA specific forums and blogs. You'll find a lot of useful perspectives on the Agile QA transition from a large number of resources in all different business spheres and from people in different places in their own transition. It's easy to see the positives from what you read but being focused on the common mistakes can help you avoid the pitfalls others have already experienced. Encourage everyone on your team to research on their own and discuss findings as appropriate in meetings.

When cost is not an issue and you are able to attend or send resources to training, consider QA specific conferences/classes. At most QA conferences, you will find good Agile QA sessions which will give you valuable insight into your transition. Be mindful not to spend your entire training budget on sending everyone to the same conference or class. Instead, make the money go further and get more out of it by designating one resource that will go and report back to the team. If you use this approach, you will be able to have training all throughout your transition and will have an opportunity to address the issues you may be experiencing at the time of the conference/class.

3.2 Communication

Most organizations are either matrixed or siloed and rarely cut across functions. A byproduct of differentiated organizations is that resources within them rarely have the opportunity to communicate cross-functionally. Unless you happen to be in a leadership role, you may never actually have the opportunity to communicate outside your team. Being placed in a cross-functional Agile team where

precise communicate is important places most resources at a disadvantage. It's even more difficult when you have multiple Agile teams on a large scale project. To help your QA team “re-learn” to communicate in an Agile team, hold a couple QA only meetings that run like scrum meetings so your QA team has practice at communicating. Here are some meetings you can hold that will help your team communicate more effectively:

- Automation design meeting - allow your technical team members to geek out or use the meeting as an opportunity to address a particularly vexing technical problem.

- Team meeting - all QA team members should meet to discuss what's going on in their projects and how their particular tasks are going for the Sprint.

During these meetings, encourage team members to ask questions. Keep the meetings brief and to the point but share experiences (good and bad) especially solutions. By sharing both good and bad experiences while focusing on solutions, you will avoid the meeting becoming a “gripe” session. Don't be afraid to rotate the job of leadership in your QA Team meetings. Make sure everyone has a chance to lead a meeting so they gain the valuable experience of understanding what types of questions to ask, what information is needed, and how to help solve problems. These skills can and should be practically applied during everyday tasks and honing them is an important part of the Agile skills adaptation.

4. Paired Testing

As difficult as learning communication skills can be, the challenge of adopting technical skills required in Agile QA can be daunting for the majority of QA Teams. For most QA organizations, the team is comprised of two groups: the automation test team and the manual test team. Implicit in this team dynamic is the idea that automation resources are more technical then manual test resources because of their understanding and experience in 3 areas: development language(s), problem solving in technically complex automation frameworks, and integrating and building of automation components. What's more, automation resources tend to be more comfortable with Agile test tools which lean toward the development side of IT. Based on their understanding and experiences, automation test resources seem naturally fit for an Agile QA transition where whitebox testing, design and code reviews, and paired programming become central tasks for QA resources.

If a good percentage of your typical QA Team was comprised of automation test resources, the Agile QA transition would be fairly smooth. The reality that faces most QA Teams is that manual testers far out-number the automation resources giving rise to the challenge of how best to transition a mostly manual test team to a more technically demanding Agile QA.

One of the great successes I've had in the Agile space is the application of the paired-programming technique to the QA team. In paired testing, a manual QA analyst works closes with an automation QA analyst. Here's why I've had such great success using the paired-programming technique within QA: In addition to having a business and operational focus, manual testers can audit the automation to make sure the right things are being tested. Manual testers can also execute one-off, specialized tests much more easily and can provide the automation analyst with precise directions on how to reproduce the steps programmatically. Having concrete steps to automate the most difficult one-off tests can mean the difference between being "done done" at the end of the Sprint or carrying-over to the next Sprint.

If you use paired testing, make sure you rotate pairs regularly and make sure they work together often. Your pairs should work together at least an hour a day and should focus on doing peer reviews, technical reviews, or learning about each other's projects. Rotating pairs helps avoid the pesticide paradox[6] of testing. Adding a fresh set of eyes and perspective can help avoid missing bugs because testers can begin to have tunnel vision when executing the same tests over and over again. Having to explain the test to someone can also jog important discussions which can lead to evaluating whether or not your testing is as effective as it could be.

One of the most important advantages of paired testing is the collective spread of knowledge amongst your QA team. With the help of the test automation analysts, your manual testers will be able to gain valuable technical expertise which will help them in the technical Agile world. What's more, your automation analysts will get to hone their qualitative skills by participating in tasks like reviews, audits, and traceability analysis. Honing these skills will allow them to develop more complete technical solutions as they will be more aware of all cases and decisions in their automation.

5. “Done Done” Means Test Automation is Complete

Test Automation is a must and “done done” should include having automated, integrated tests for each work task included in the Sprint. I’ve often observed in other organizations that people say they are Agile even though the test automation is 1 or 2 Sprints behind. I say they aren’t truly Agile and I often debate with my QA brethren this concept of “done done” being defined by having high quality automated QA tests which are part of the Continuous Integration (CI) process at the end of each Sprint. My logic is this: how are you Agile if you are behind? My defense of this idea always brings to bear the following questions.

5.1 How did I come to such a conclusion?

I arrived at this conclusion by thinking through the following:

- Exhaustive Testing is not possible - CTFL 7 QA principles[5] - Anyone who has ever been on a large or complex project knows you can’t manually exhaustively test. What’s more, you can’t expect to leave everything to the end and hope to catch up because you are then slowing down the development process which is contrary to the idea of Agile.

- Immediate feedback is imperative – If your automation is 1 or 2 sprints behind and your manual testing team can’t exhaustively test everything, then it stands to reason that there is a gap in the feedback loop for the areas that manual testing can’t get to in each sprint. Your team may recover the gap in the feedback loop in a later sprint but delaying feedback to your developers works contrary to the idea of Agile where an immediate feedback loop helps develop quality software faster.

- No business wants to spend days, weeks, or months doing Regression Testing at the end - if you have a certain number of requirements, the number of tests grow exponentially with each Sprint. Building a large automation “backlog” doesn’t work in favor of the QA team as they hold up the release and reverse all the delivery gains of the Agile team.

5.2 How do you get the time to build test automation infrastructure?

You can always make time for infrastructure by planning ahead. You should be applying the Agile development principles to the test automation framework development so change is inevitable. Don’t try and craft a solution from the get-go, try and slowly develop it over time. Here are some hints:

- Make resource (tools and personnel) decisions before Sprint 0.

- Make good use of Sprint 0: infrastructure, design, and architecture is important.

- Aggressively hire for automation but avoid the “developers can do QA Automation” last resort.

- Have coding standards and have a tool or a peer review process to make sure standards are being followed. Having standards will allow additional automation resources to slide in and help if needed in a “packed” sprint.

- Collaborate on enterprise framework design and architecture but have an ultimate authority so you don’t get bogged down by unnecessary “perfectionist” tendencies of group think.

- If your budget allows, have an extra automation resource that can float. You will always come across a situation where a sprint will be too heavy for the assigned QA resources so having a spare resource that is plugged into the environment and can lend a hand in a pinch could save the day.

Test automation is too important to leave to chance. Work diligently to plan for the long-term solution by building the foundational pieces one-by-one.

If you fail to automate all of the testing in a sprint, your team must fail the sprint. It sounds harsh to fail a sprint because the test automation isn't complete but holding your team accountable is important. If the developers on your team are required to deliver good, unit-tested code for all the stories included in a sprint, then the QA team should be held to the same standard for its test automation.

6. Customer-Driven

Regardless of which test methodology you use, being customer-driven can lead you to be more successful in identifying whether or not your software is fit for use. Borland recently conducted a webinar called Is Traditional QA Dead? In the webinar, they make the following point: QA is much more than just validating functionality – it's about customer satisfaction.[7] Borland adds to this point by stating that the raise in social media has created greater "transparency" and speed to customer feedback. These trends are putting tremendous pressure on QA teams to be able to predict how and what customers will do with their systems. If your system meets requirements but isn't fit for use, it won't take months, weeks, or even days for feedback to hit the airwaves; it will take a matter of seconds.

So, here are two things to consider as it relates to customer-driven Agile QA:

6.1 Fit for Use vs. Meets Requirements

At a conference for Healthcare software early on in my career, I heard a vendor say the following: if doctors don't use it, it's the same as being broken. I hadn't thought about software in that way before but I realized that designing software for customers who don't use it is just as bad as having buggy software. What I learned is that thinking like the customer drives out more bugs. In Agile, the beauty of acceptance criteria is being able to clearly define what a system should do; the short-coming is in defining what a system should NOT do. Thinking like a costumer often forces you to consider complex decisions and shake out all possible outcomes not just the happy path. A complex decision requires difficult code to be written which means more bugs will cluster in those areas. Also, when you focus on being customer driven, you help the software become more usable and improve the user experience.

6.2 Include Operations in the development process

In my experience, one of the major short-comings of Agile is in the negative impact it has on most organization's Operations teams. Volk's study on comparing Agile at over 200 companies states, "The Agile movement shifts the broad, inter-departmental process of software engineering to one that is focused on software development to the exclusion of QA and operations".[8]

When things go right for a company engaging in Agile, they put software into production quickly and continue to iterate against the product in production. This quick development cycle puts enormous strain on Operations teams from hardware/infrastructure teams, call center/customer support teams, deployment teams, and even supply chain teams. Think of the Operations teams as being consumers of the software as they are directly influenced by the software in one way or another. Engage them early and often in the process so you know how your software decisions are impacting their needs as well as the needs of the user. What's more, Operations teams like the customer support team may have valuable insights into how consumers are using your products. That feedback may help you avoid developing needless features and functionality allowing you to deliver faster.

7. Conclusion

Six years ago my understanding of Agile was based on fear, skepticism, and naiveté and I incorrectly believed it was the end of QA as I knew it. At the time, I believed that nothing anyone in the QA team did would help with an Agile transition. What I've learned in six year of Agile implementations since then is that there are practical keys that can guide QA teams in their journey to evolve into dynamic Agile QA teams. This paper examined some key pivots within an Agile QA transition and offered up advice on how to better manage the process. The following pivots were examined:

- How the psychology of a team's shared experience could help draw the team closer together. By pulling together, a team could face their fears, skepticism, and confusion and turn them into a positive, galvanizing force.
- How training and communication are vital for QA teams in transition. Learning to be an effective member of a cross-functional Agile team requires that QA teams get training and practice what they learn as often as possible.
- How applying the principal of paired-programming to the QA teams can not only help manual tester resources get technically up to speed faster but can also lead to more fully developed, complete automation tests.
- How "done done" in test automation means having fully integrated test automation running within each sprint and not delayed.
- How focusing on being customer-driven will allow QA teams to focus on what matters most: delivering high quality software that's fit for use. You don't want your product to go "viral" for all the wrong reasons so focusing on the user experience and integrating Operations into the testing process is important to quality product delivery.

By exploring the practical keys and understanding how they may be applied, QA teams in transition should be able to focus in on areas of weakness and take necessary steps to improve them.

References

1. Smolaks, Max. “Agile Development Comes With Hidden Costs Warns Report”. *The fashionable development practice may be bending IT out of shape.* TechWeek Europe, July 16, 2012. Web http://www.techweekeurope.co.uk/news/agile-development-report-86052

2. Rooney, Dave. “Agile Transitions and Human Nature”. *Practical Agility,* January 25, 2012. Blog http://practicalagility.blogspot.com/2012/01/agile-transitions-and-human-nature.html

3. Smolaks, Max. “Agile Development Comes With Hidden Costs Warns Report”. *The fashionable development practice may be bending IT out of shape.* TechWeek Europe, July 16, 2012. Web http://www.techweekeurope.co.uk/news/agile-development-report-86052

4. Johnston, Matt. “Testing the Limits with Scott Barber Part I”. *Testing the Limits.* UTest, April 26, 2010. Web http://blog.utest.com/testing-the-limits-with-scott-barber-part-i/2010/04/

5. International Software Testing Qualifications Board, 2011. ISTQB_CTFL_Syll_2011. September 2011

6. International Software Testing Qualifications Board, 2011. ISTQB_CTFL_Syll_2011. September 2011

7. Roboostoff, A & Rechberger, G. (2013, July 10). Is Traditional QA Dead? [Webinar]. Web Seminars, A TechWell Event.

8. Smolaks, Max. “Agile Development Comes With Hidden Costs Warns Report”. *The fashionable development practice may be bending IT out of shape.* TechWeek Europe, July 16, 2012. Web http://www.techweekeurope.co.uk/news/agile-development-report-86052