

**Agile Teams Require Agile QA:
How to make it work, an experience report**

Kealy Opelt & Tracy Beeson
Menlo Innovations LLC

*Presented at the International Conference on
Practical Software Quality and Testing
(PSQT 2008 West)*

Agile Teams Require Agile QA: How to make it work.

Introduction

It is challenging to successfully integrate standard quality assurance (QA) practices within an agile process such as extreme programming (XP). Not only is expanding an agile process to include other disciplines difficult, but XP teams typically reject traditional QA practices. Integrating QA into an XP team does not have to be painful. Integration is achieved by tackling small goals using the XP principles of communication, simplicity, feedback, and courage in order to build an agile QA process. The following paper describes our experiences towards successful integration.

Getting Started

An important first step for QA integration is for both the business and the team to recognize that a quality assurance team is beneficial to the project. Benefits include but are not limited to: finding bugs before they go to production, adding a different perspective during development, and being advocates for critical bug fixes. However we have discovered that most project sponsors do not see these benefits. In our experience we have heard two common reasons why a business might claim the quality assurance practice is not necessary: A) QA slows down development. B) A good set of unit tests should suffice.

In respect to QA slowing down development, we respond by pointing out that speed and quality are often competing forces on a project. The key is to strike a balance that is right for the business needs. Certainly quality would be critical if you are working on a medical device that, should it fail, could place a life at risk. In this case, it would be reasonable to expect that product development would take longer. On the other hand, if you were working on a demo for a trade show that will never be released to production but will be used to attract investors, it would be appropriate to forego quality in turn for speed of development.

It is important to have an open conversation with key business stakeholders to discuss if an investment in QA is important to the business needs and the degree of that investment. Void of this conversation, it would be easy for the work of the QA team to be at odds with the current business goals, thus seen as merely a hindrance to the process. In our experience, QA is best integrated when the need for a QA team is recognized but the amount of resources devoted to quality is balanced with the needs of the business.

In respect to unit testing, a common debate among those who practice agile methodologies is whether a robust set of developer written unit tests replaces the need for quality assurance. Consider the following analogy: When constructing a bridge, each component must pass a rigorous set of tests before it can be used. For example, concrete must pass a strength test, each piece of steel must be of a certain grade and standard, and so on, such that every piece of the bridge has passed its respective test. However, if this was the only set of tests run on the bridge, would you want to drive your car over it? No. In fact, after completion the bridge must pass another set of tests in order to prove it will function under pressure from cars, trucks, wind, etc. before it is opened to the public. Just because each component to the bridge has passed its test, this does not mean all the components will work together as expected. The considerations are similar when writing software and discussing the difference between unit tests and functional tests. Unit tests verify that each component of an application works as designed. The functional tests, as designed by the quality assurance practice, focus on the user perspective to verify that all components work together.

The Role of QA

The second step to successfully integrating QA into an agile team is to understand the role of QA. The quality of a product should not be judged solely by the number of tests written against it or the number of bugs it contains. It is our belief that the role of QA is not to write as many tests as possible or to find as many bugs as possible, but rather to help the business understand and measure risk. To do so they must understand the user needs, the underlying architecture of the product, and the implications of the known bugs. The following simple steps can go a long way towards helping QA fulfill its role:

- 1) Listen, learn and ask lots of questions
- 2) Prioritize QA tasks

Listen, Learn and Ask lots of questions

In order to learn the business needs of the client, QA should participate in the process of feature prioritization and planning. Listening to the business analysts or the users themselves will enable QA to learn about user's needs. Also, QA should listen to the developers as they discuss architecture. All of the knowledge gathered during these discussions will not only help to focus the testing efforts on the most critical areas of the program, but will also help QA determine the areas of greatest risk to the business.

Prioritize QA Tasks

In our world developers work from estimated story cards that define system requirements, QA should too. Figure 1, is an example of a QA story card for updating the bug list. The number “2” in the top right hand corner indicates the card has been estimated at two hours.

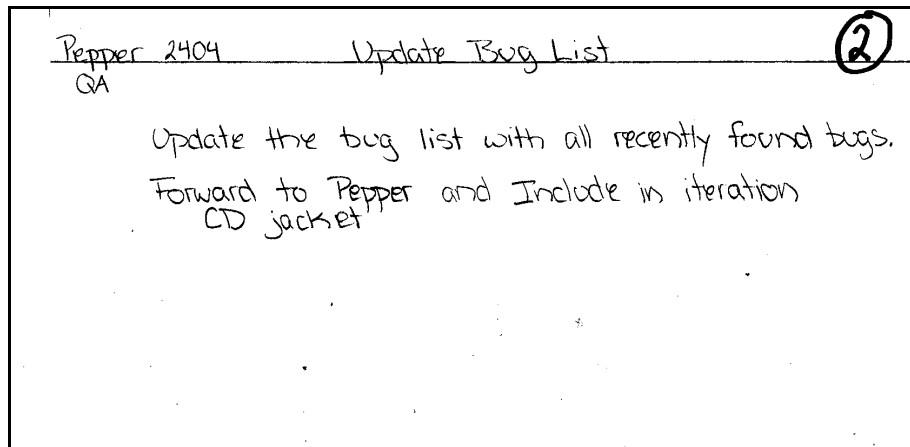


Figure 1. Example of QA Story Card

The priorities of these estimated story cards should be discussed with the customer at each iteration’s planning session. This creates visibility for potential limitations of resources, time, and scope of work facing the QA team. This also allows the business to reevaluate the level of quality they require.

True Integration into the Process

The third step is to integrate QA’s findings into the agile development process. In our experience, the earlier QA is involved, the more successful the integration.

As mentioned above, QA should listen to the development team’s discussion of architecture. A good forum for this is the iteration kickoff meetings. Iteration kickoff opens the door to collaboration of design and healthy debate. Each story card planned for the upcoming iteration is discussed and the team agrees on a plan of action. Participating in this activity creates opportunities to listen for the challenges the development team will face, for possible areas of confusion, or for forgotten use cases. Bugs can be caught even before code is written.

It is also important to be part of the developers' decision process of declaring a story card "done". In our world, the developers are not allowed to declare themselves done. Instead, they are given the chance to run through a mini "show and tell" with representatives from the High-Tech Anthropology™ (HTA) and QA teams. Our HTAs are the usability and design experts who advocate for the end user. In our experience, it is beneficial to have both HTA and QA declare a card done. In doing so, the HTAs will test the most likely scenario and QA will test any unlikely scenarios. This should be a fifteen minute first pass for broken functionality and usability issues. The purpose is to catch problems before development moves on to the next priority.

Perhaps the most traditional way in which we integrate QA is in verifying the build. We practice weekly iterations which result in a build for delivery to our customers. For this reason, we have devised a way to quickly and efficiently verify the build using formalized exploratory testing, allowing us to remain agile while creating consistency. We avoid building too much structure around this process, keeping in mind that by the time we get a build in hand for testing it will have already gone through several layers of rigorous tests. Testing the build should not be a valiant effort done into the early hours of the morning, but simply a final check. The goal is to gather a sense of the build's quality such that the QA team can help the business pick the best candidate(s) for release by measuring risk.

QA should also be involved with helping the business prioritize bugs quickly and efficiently. A prioritization system that relies on the physical placement of a bug can be effective toward helping the business sponsors pick out the most important bugs. We have dubbed this visual technique as our "Bug Board". As shown in Figure 2, the "Bug Board" is a graph on a piece of poster board. The x-axis represents impact to the user while the y-axis represents the bug's likelihood of occurrence. Each bug is then represented with a short description and identifying number on a removable sticky note. This allows the bugs to be easily placed, removed or repositioned.



Figure 2. Project Bug Board bugs placed.

During each iteration's planning session, have the business sponsor place the bugs on the bug board based on an informed quick response. As iterations continue, this process will create a powerful visual of just how "buggy" the application may be. If the majority of the bugs land in the highest likelihood of occurrence and the maximum impact to the user, clearly the application is in need of bug fixes. If the majority of the bugs land in the least likely to occur and minimal impact to the user then the application is in pretty good shape. After this exercise, it will become clear to the business sponsor which bugs are the highest priority, helping them make informed decisions within the planning process.

Automated Testing

As a fourth and final step to integrating QA, consider adding a suite of automated functional tests. Before doing so, it is important to recognize that automation can be time consuming and expensive. Avoid traveling down the overwhelming path of automating every functional test possible. If the need of the business permits, take a minimalist approach to automation, pinpointing key areas of the application that would most benefit by an automated test. Write and estimate story cards to tackle those first.

Choosing to tackle automated testing as a last step was at first just because it was a hard a thing to get started on, but as we say how much benefit simply being part of the team was it continued to

be less important. We choose to tackle it as we discovered features of the product that were extraordinary important and not easily testing by a unit test. For example: statistical calculations on data that could affect the users scientific studies and had infinite input data.

Conclusion

Following these four steps can reduce the challenges of integrating QA into an agile team. Implementing these practices should result in a QA process that is structured and consistent. Remember, it is important to consider the business needs when determining effort put towards quality. The role of QA is to help understand and measure risk. Listening, learning, participating, and prioritizing are all key aspects of successful integration. Integration within an agile team requires being agile: iterate the process. Revisit your QA priorities on a regular basis and constantly consider how you might improve the process.