EMBEDDED QA IN SCRUM

Jitesh Anand
Senior Software Quality Engineer
EMC PS App Solutions & Systems Strategy

Chaithra Thimmappa
Senior Software Quality Engineer
EMC PS App Solutions & Systems Strategy

Vishwanath Channaveerappa
Software Quality Engineer
EMC PS App Solutions & Systems Strategy

Varun AdemaneMutugappe
Software Quality Engineer
EMC PS App Solutions & Systems Strategy

Neil O'Toole
Senior Manager
EMC PS App Solutions & Systems Strategy

EMC²

Table of Contents

Introduction

Releasing a product with bugs is potentially very expensive when costs of field upgrades, recalls, repairs, down time etc. are considered. Less quantifiable but equally important is damage to a company's reputation and loss of good will. Yet, many products which are developed using different Software Development Life Cycles—from Waterfall to Agile Model—are released without performing enough testing due to delivery deadlines and shortage of skillset necessary to minimize these problems.

For developers of such projects who choose to employ automated in-target software testing, the challenge is to find tools that are well integrated into their embedded development environment. Tools with this level of integration lend themselves to more efficient testing and rectification of discovered problems. Most existing software testing tools can only execute the tests. These tools will not be creating tests by themselves but will have logic driven by the developer only. This will have its own limitations when testing an embedded complex application because the actual system will have different hardware interfaces, timing issues, memory constraints, and other dependencies.

When utilizing automated tools it is prudent to have a skilled Quality Assurance Engineer who can perform validation as soon as code is complete and provide instant feedback on, for example, the look and feel of a product and provide validation at each exit point when a feature has been completed. Why not embed the QA person into the Scrum team? A strong bond between Development and Quality Assurance finally drives the construction of a quality product while both parties learn each other's practice and skills.

It has been observed that many people and organizations are confused about the differences between Quality Assurance (QA), Quality Control (QC), and Testing. These are closely related, but they are different concepts.

What is Quality Assurance?

QA starts at the Requirement Analysis stage, then propagates through test design/planning, test environment readiness, test execution, periodically assessing functional/performance and project risks, and providing defect/test metrics. This ensures that the right product is being built to meet defined quality standards and acceptance criteria.

"Quality is not something you add like salt at the end, it must be baked in".

QC deals with reporting bugs found in the software whereas QA is applied to every stage of SDLC to prevent bugs on outgoing software. In essence, QA is best carried out on process while QC is best carried out on product.

Parts of the process considered in QA are planning, design, development, production, and service through maintenance. QA activities ensure that the process is defined and appropriate. Two principles included in QA are:

- "**Fit for purpose**", the product should be suitable for the intended purpose;
- "**Right first time**", mistakes should be eliminated.

QA includes the activities of PDCA cycle[3].

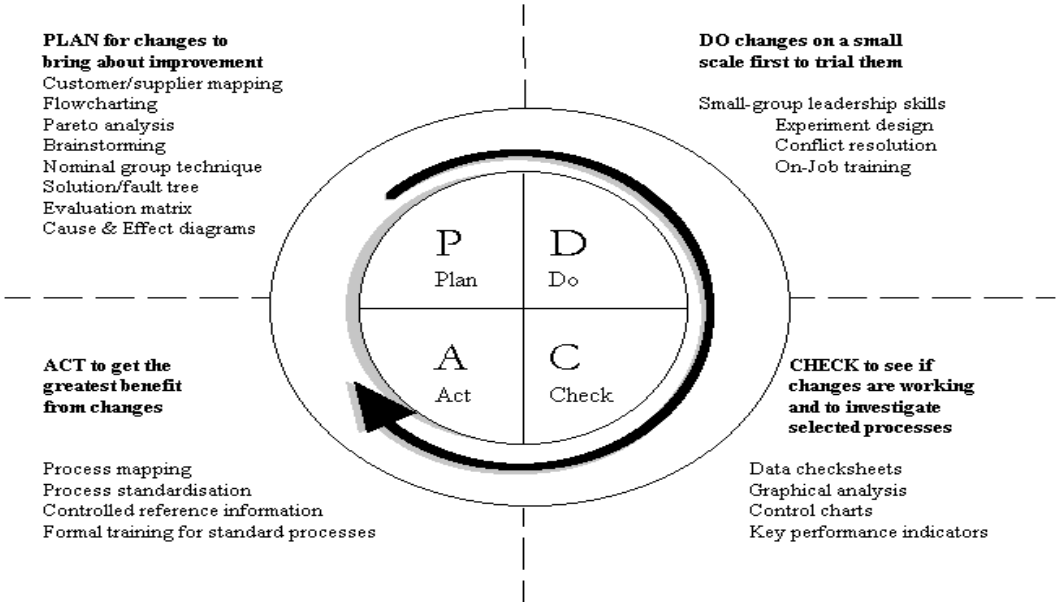

Diagram 1

Benefits of QA are:

- **Improved Client Satisfaction:** QA will perform testing before it is delivered to the end user. Hence, the end user will not face any blocking issues. This will lead to a profitable relationship which will result in more requests for new features as well as good feedback from end users.
- **Reduced Cost of Development:** The process will be streamlined and stages of Development will be simplified, enabling defects to be found at the planning or verification stage. This will help reduce software development cost. In certain situations, it is well known that fixing an issue at later stage of Development increases cost.
- **Reduced Cost of Maintenance:** Good Development is less troublesome to support. Support is not only costly but also impacts the work of end users.

Waterfall vs. Scrum

Waterfall Method

The Waterfall model is a progressive design process which in the software development industry goes through stages such as Requirement collection, Feasibility study, Analysis, Design, Implementation, Testing, and Maintenance. It steadily moves downward completing one state after the other.

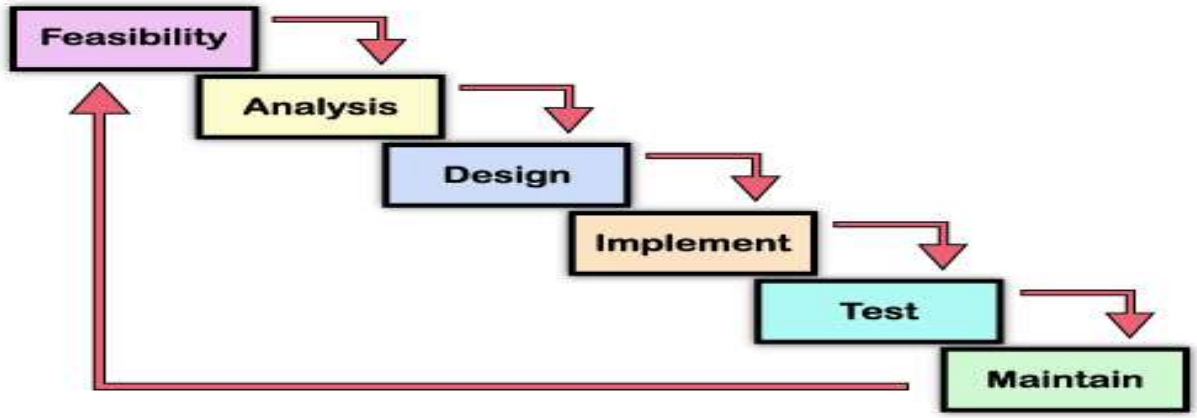

Diagram 2

Challenges of QA in the Waterfall Model are:

- A completely separate team from development comes into the picture only during testing activities.
- The main function is to certify the quality of the product.

- Mistakes in requirements lead to significant wasted effort.
- Not much interaction with the customer. Interaction contributes to ensuring that the application meets what is specified in the requirements document. With little to no interaction, QA does not feel involved in the Requirements and Development Stage.
- Depending on timescales and working software Automated Test Coverage maybe limited. As QA involvement comes late in the Waterfall cycle, Automated Suites lack quality as well as time.

Scrum Method

On the other hand, the Scrum method is an incremental approach. The team works on small modules and then responds to user's altered requirements rather than following a pre-determined plan. The design is simple and changes can be made as work progresses.

When compared to the Waterfall method, with Scrum, testing and responding to user change requirements can happen at the same time in the course of the project. Interactions among stakeholders are prioritized as compared to tools and processes.

This method became popular in the 1990s after many found the drawbacks of traditional Waterfall methods.

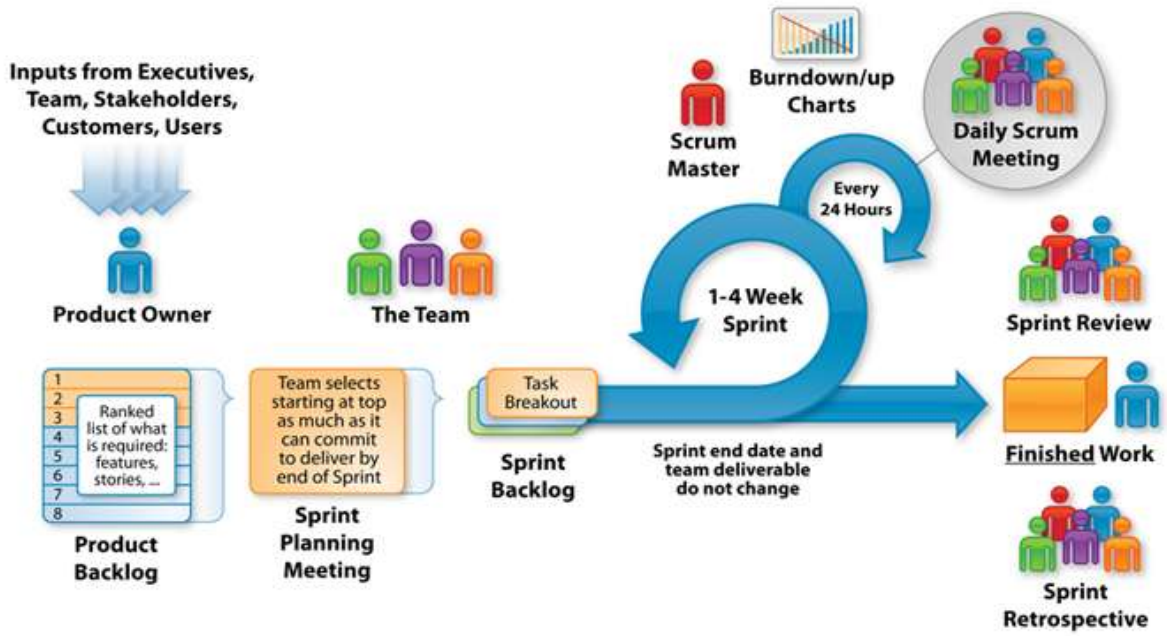

Diagram 3

Challenges of QA in the Scrum Model are:

- QA is not focused on Acceptance Testing which may lead to various loopholes in the product.
- As QA works in detached mode and QA lags behind Dev Sprint, QA is pressured to complete the testing within a few days and release the product.
- Some lower priority bugs found before release are moved to next release as Dev Scrum Team moves to next release/product.
- QA does not get enough time to perform Automation during the later stage of release as it is occupied with performing Functional and Non-Functional Testing.
- QA is not involved in the difficulties/changes in scope of work which is done by Dev Scrum Team during a sprint. QA is informed later after Demo and/or Retrospective meetings.
- Requirements for Non-Functional Testing are not captured at early stage of Development. QA performs Non-Functional Testing only after all features are integrated.

Evolution of QA as "Embedded QA"

Traditional QA started around the Alpha phase in a traditional Waterfall approach to vigorously test the software once all the features were implemented. Alpha phase had been the first real chance to test the software in some form near its final state. Often, discovering product value at the end of the project often raises the desire to change things even more.

Additionally, QA was underused. QA would be either unclear about insights into the software ("it's too late to change the software based on your suggestions") or their role would be so undervalued that, in some cases, people would be literally hired off the street to fill the role.

Scrum was about mixing things up. The word is taken from the sport of rugby where the team moves the ball as a group. So the question is why separate QA into a separate department that doesn't Scrum to produce a vertical and complete slice of the software?

"The Scrum and the tackle are the two really contentious areas of the game. If you get those aspects right, most rugby matches will work in your favor." – Allan Lewis

The best idea is to embed QA into the Scrum team. They will sit together, have open discussion together, analyze, design, and provide end user cases before coding starts. They will get their hands dirtier. Sometimes QA will be expected to do a bit of coding, design layout, and also take

on part-time associate producer roles. However, their main job will be to exercise functionality as well as drive and design Acceptance Tests.

Embedded QA will help in validating each story once it is resolved during a sprint, focusing on validating the acceptance criteria of the story. Also, Embedded QA will ensure that which was working previously is still working. Quality will now not be the sole responsibility of QA. Rather, it will be the responsibility of the entire Scrum team.

Better quality leads to higher customer satisfaction and increased sales. As QA ensures product quality, the role of QA is becoming more critical in modern SDLC like Scrum. QA will not only perform the role of Quality Assurance but also Quality Assistance with the implementation of Embedded QA.

Implementation of Embedded QA

Each team should be cross-functional and should contain DEV, QA, PM and, if possible, Architects. Each team member is wholly responsible for the quality of the features they produce. However, each one has different roles in those teams—it is not that a developer and QA are interchangeable. Oversimplified—the developer's role is to produce a high-quality feature; QA's is to enable them to do it and confirm this was achieved; PM's ensure that the team is building the right thing. Every member of the team needs to understand the actual purpose of the feature and the goals/skills of the people who will be using it.

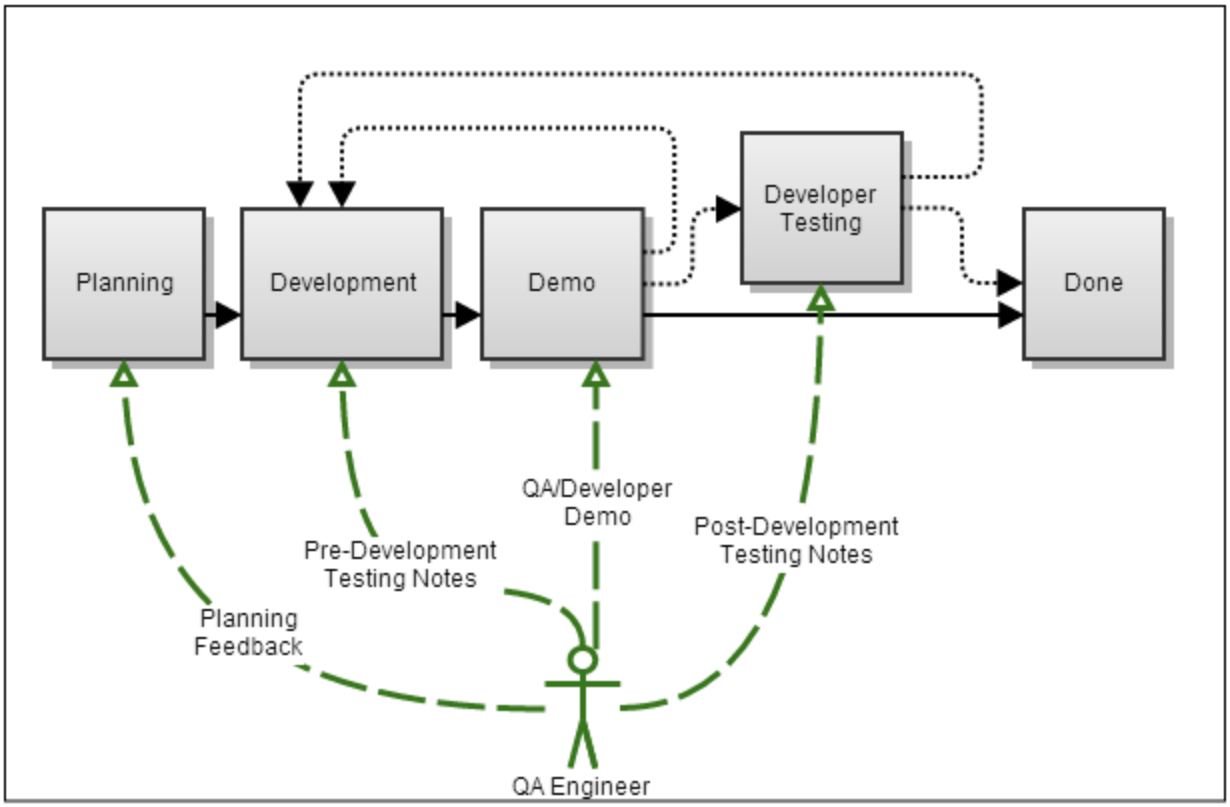

Diagram 4

A QA engineer will have multiple points at which to provide input on how the story is developed and tested, providing every form of quality improvement.

The opportunities for a QA engineer to influence the development of a story are:

Embedded QA in Planning Stage

Occasionally, a story will seem simple, but the QA engineer knows in advance that it will actually be very difficult. To get this feedback in as early as possible, the QA engineer needs to attend the iteration planning meeting.

For example, a story might say, “Create a Login Page.” The QA planning feedback for this story might be, “The sample Login Page makes assumptions about the available types of User (Employee, Vendor, Customer), and other configuration of the system. We can’t rely on the system being configured in this way when the instance isn’t brand new. Hence, this story will also need to cover rewriting the way that the sample data is injected.”

Based on this feedback, the team might choose to increase their estimates, change the scope, or move the story further down the backlog in favor of lower-hanging fruit.

Embedded QA in Pre-Development Stage

QA can often make accurate predictions about bugs that could be found in a completed story. Given that, why not warn the developer in advance and save everyone the bother of dealing with the bug?

To achieve this, QA can introduce pre-development testing notes. These are a set of hints, written before development starts, about the type of bugs that might be expected to be found in the story once it is complete. The goal is to enable the developer to avoid introducing the bugs in the first place.

Pre-development testing notes can vary significantly in size, depending on factors such as the scope and complexity of the story, the experience level of the developer, and the level of risk associated with the changes.

Below is a typical example of pre-development testing notes for a low-risk story.

Even for a simple story such as this, the QA engineer has found a little-used Login configuration option that was not known to the developer. By bringing this up before work starts, QA has saved the story from being rejected later on—saving time, effort, and irritation all around.

The expectation is that the developer will perform exploratory testing of the feature while they are developing it. After all, not all bugs can be predicted.

Embedded QA and Developer Demo

Once a developer is satisfied that they have completed a story, they will invite their QA engineer for a demo session. This has multiple purposes:

- To allow the QA engineer to understand the implementation details of the story, down to the code level. This informs their decisions on risks, what testing needs to be added, and what testing is unnecessary as it may have been already completed.
- To get a second opinion on the usability of the feature: the UI (if there is one), the API (if it’s a new public API), and any output to the logs.
- To discuss what testing the developer has done and what testing they still intend to carry out before they push the changes to the repository.

The demo is a discussion between equals, and not a test that the developer or story can “pass” or “fail.” However, sometimes concerns will be noticed during the demo and these are quickly noted as comments on the story. This is much more lightweight than raising issues at this stage—they don’t need to be separately tracked, assigned, triaged, resolved, or tested—because they will be fixed within a few hours, while the developer is still working in his or her original context.

At the end of the demo, the QA may do additional testing of this story. Also, QA can decide to do more testing when another story is completed as it may impact this story.

Post-Development Testing Notes

If either party feels that further testing is needed, the QA engineer will update the testing notes along with the notes of any testing that’s already been done by the developer, adding new scenarios and risks based on the implementation of the story. This will ensure efficient testing of the story.

Actual Testing

Once the testing notes have been updated, QA will pick up the story for testing. This testing is carried out using the testing notes as a guide, but doing so intelligently; not blindly following a checklist.

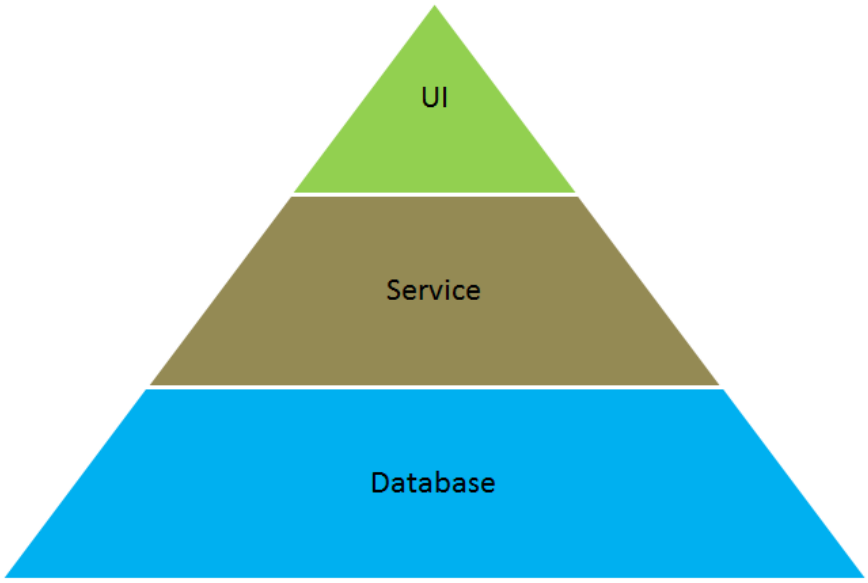

Diagram 5

Issues raised by a QA engineer that a developer may not consistently recognize include:

- Usability/design problems
- Unexpected configuration options that the developer didn't know regarding Security
- Performance
- Unexpected side-effects of changes
- 'What-if' scenarios

Perhaps more crucially, QA engineers also provide advice on what testing is not required. A quality-conscious but unskilled-in-testing developer often errs on the side of testing everything in every environment, which in most cases doesn't actually provide more quality. QA advice can enable pinpointing the minimum amount of testing required to get the maximum amount of quality.

Finally, only the QA team closes the feedback loop and looks at the big quality picture.

Of course, it helps that the product we're writing is also a product that we all use every day!

Roles and Responsibilities in Embedded QA

Craig Smith, an Agile Coach in Suncorp's Business Services division wrote…

"To create a high performing Agile team, a lot of collaboration is required. One of the challenges when it comes to collaboration on Agile teams is that testing is no longer the domain of a quality team. It is everybody's responsibility."

Key:

Symbol		Definition
R	Responsible	•Individual/s who perform a task/activity; the doer, responsible for action/implementation. •The degree of responsibility is defined by the Accountable person. •Responsibility can be shared •While Accountability can not be delegated, Responsibility can be delegated.
A	Accountable	•The individual who has ultimate accountability and authority. •There is only one accountable (A) to each task/activity. •Accountability can not be delegated.
C	Consulted	•The individuals to be consulted prior to a final decision or action is taken. •Two-way communication
I	Informed	•The individuals that need to be informed after a decision or action is taken.

	Who	Embedded QA	Developer	Scrum master	Dev OPs	Product Owner	Architect
1	Create QA sub-tasks that validate the stories Acceptance Criteria	R/A	I	C	I	I	I
2	Design automated test cases that validate the Acceptance Criteria	R/A	C	I	I	I	I
3	Maintenance of AT Scripts	R	R/A	R	I	I	I
4	Black box validation of Module delivered by Engineering team, execution of acceptance	R/A	I	I	I	I	I
5	Maintenance & Execution of Non-Functional	R	I	C	I	I	I
6	Maintenance & Execution of Non-Functional Scripts	C	I	C	I	I	I
7	Monitoring Build, AT Status	R	R	A	I	I	I
8	Update weekly QA metrics	R	I	I	I	I	I
9	Review of Resolved bugs/tasks/stories in JIRA and closing them out	R/A	I	I	I	R	I
10	Validate the Scripts Against Actual Arrays - Symmetrix/CLARiiON (if relevant for tool)	R/A	R	C	I	I	I
11	Review of Development Stories - Acceptance Criteria Completeness/Story Detail	R	R	R/A	I	R	R
12	Review of new Features and supporting documentation on Confluence	R	R	R	I	R/A	R
13	Participating sprint pre-planning, planning and Demo	R	R	R	I	A	A
14	Performing non-functional testing.	R	C	C	I	C	I
15	Verification of Architecture Stories	A	C	C	I	I	R
16	Responsible for attending Go/No Go meeting?	C/R	C/R	C/R	I	R/A	I
17	Closing Epics, Stories, Change Requests, Sub Tasks and Bugs	R/A	R	R	I	R	R
18	Documentation Review	R	R	R	R	R/A	R

Table is a draft version subject to change.

The scrum starts with the product backlog Product Manager collecting the stories for the scrum with the help of market survey, field engineers, customer feedback, etc.

These stories are given to the development team for estimation. Next the Scrum lead, Development team, Embedded QA, and Architect ensure that the story has clearly defined ‘Acceptance Criteria’. Stories will be broken down into sub-tasks; at this point a ‘QA Task’ is also added with Dev tasks to the story.

- QA Task is designed with inputs from Scrum lead and Development team and accepted by Product Manager. Previously, ‘QA Task’ was verified by the Product Manager.
- QA designs the automation tests which should be accepted by Scrum team (Product Manager can provide input too). Previously, QA was solely responsible for design of automation stories.
- QA can provide input to implement the automated test cases to validate the Acceptance Criteria. The team decides who is responsible for developing them, but QA can and do assist.
- Design of Non-Functional tests is primarily QA’s responsibility with inputs from Scrum lead and Development team.

The Story is deemed successful after it has passed a code review and all above mentioned (or required of above mentioned) sub-tasks are successful. Review of resolved bugs/tasks/stories is mostly the responsibility of QA with input of Scrum lead, Development team, and the Product Manager.

Previously, the QA sprint was separate to the development sprint. Now, under the embedded QA role, one sprint covers both roles. Hence, all parties must participate and estimate effort before the start of the sprint. QA is also part of the Scrum meeting and provides daily updates, concerns, issues, etc. to the Scrum lead.

Documentation review of the product is the responsibility of QA with input, or in agreement with Scrum lead and Development. Previously, QA was solely reviewing the documents developed by the Product Manager.

The other major task in Scrum is a regular backlog bug scrub, carried out by QA with input from the Product Manager, Architecture, and Scrum lead.

Along with the above, embedded QA has the following tasks:

- Monitor automation test status (Pass/Fail).
- Monitor build status, unit, integration tests.
- Present QA Dashboard with acceptance criteria for release in the program meetings.
- Participate in Go/No Go meetings.
- DevOps are responsible for pushing the release to production.

Benefits of Embedded QA

- First, in many places, QA represents the end users, acting as the voice of the user on the team. Marketing and Product Managers often try to articulate the voice of the customer, but often they don't actually see the product until one or more components are completed.
- Second, there may be little to no documentation and this could be their only chance to see the requirements. How can you test the software when you don't know what it is supposed to do? It is much easier to do your job if you were there when the requirements were set and the design decisions made.
- Third, you need to start writing test cases to know everything you are going to test ahead of time. It is easy to write test cases before the code is written; all you have to do is "translate user stories into acceptance tests."

Capabilities of Embedded QA

To be a high performing Embedded QA, a lot of skills are required, from understanding requirements to listing acceptance criteria. As well, good automated tests result from collaboration between developers and testers.

Embedded QA resembles a Robocop: part man, part machine, all testers! The new generation of Embedded QA needs a wider skillset than ever before. A few of them are:

- **Collaboration:** Working closely with analysts and subject matter experts to ensure that testable acceptance criteria are created for all stories.
- **Estimating:** Involved in developing estimates for projects at a story level, for tools and infrastructure and also for deployment/release activities.
- **Recruitment and Development:** Attracted to an organization because testing is considered a cool career path that offers a variety of opportunities and a way to continually develop skills.
- **Reporting:** Providing metrics that give insight to project health and system quality, not just number of defects.
- **Scrum:** Understanding of Scrum project delivery and the differences between testing in the different phases of a project.
- **Qualifications and Training:** Recognized qualifications in testing and continually updating and maintaining skills.
- **Architecture:** Understanding of the system architecture and able to create tests that verify individual components and the system as a whole.

- **Development:** Taking an interest in development practices and able to monitor code quality metrics.
- **Community and Teams:** Part of a community of testers that are embedded within teams but share common values.
- **Tools and Techniques:** Apply both manual and automated techniques, using the preferred testing tools based on the situation.
- **Strategy and Planning:** Involvement in a project at all stages to ensure that testing provides the greatest value and ensures quality objectives are achieved.
- **Automation:** Work with developers to automate tests that provide business value and identify system risks while reducing regression cycles.
- **Able to spot problems or potential for problems:** Ability to see the issue during the design phase before code has been written.
- **Ability to question everything:** Not making assumptions but asking everything to be sure.
- **Ability to go beyond the models given:** Think beyond the limit and also about what-if scenarios.
- **Systems thinking (high-level):** Have a high level picture of the system and how it works.
- **Patience:** Can remain motivated even when not able to find bugs.
- **Speed to learn:** Ability to grasp thing quickly. In today's world, tools and technology change every day.

Criteria to Achieve Best of Embedded QA

Scrum Team, also referred to as the Development team with Embedded QA, is responsible for developing the product. Possessing the essential skills required to carry out the work of the project, they practice a high level of collaboration to maximize productivity so that minimal coordination is required to get things done. To minimize dependency, team members are experts in chosen domains but also possess basic knowledge and skills about other domains.

Establish a common understanding of the customer's requirements and the approach to develop the product

The Scrum Team consists of members with different levels of expertise, experiences, and viewpoints. Thus, all members should be aligned with the customer's requirements to successfully develop the product and meet (or exceed) their expectations.

Function as a single unit to achieve the goals of the project

A Scrum Team is a cross-functional unit that consists of members from diverse groups. This diversity might lead to friction within the team, especially in the formative stage. The team must strive to function as a single unit to avoid internal conflicts that can disrupt work.

Create an environment that fosters collaboration among the Scrum Team members

Collaboration refers to a team proactively sharing thoughts, ideas, and expertise to overcome challenges or to improve a product's quality. Collaborating can help a team deliver high quality products in less time. Knowledge sharing is an important part of collaboration.

Be prepared to address customer's change requests at any point during the product development lifecycle

Scrum projects are characterized by high rates of changes, depending on the customer's requirements. Change requests may be initiated due to fluctuating market conditions, change in the preferences of end users, financial parameters, etc. The Scrum Team members should be able to accommodate change requests as the objective of a Scrum project is to deliver functionality of the highest value to the customer.

Possess some business skills to ensure smooth communication with product owners and customers

Scrum Teams are often required to interact with product owners and sponsors. They might be required to negotiate with the product owner to decide which features can be delivered during a sprint or which features might contribute to the highest value. While the Scrum Team does possess technical skills, it is important that the team also possess adequate business knowledge to be able to better interact with the product owner.

Ensure team velocity is sustainable and that the team delivers the committed work

The Scrum Team should work at a pace that is sustainable. This means that the team should neither overestimate nor underestimate tasks. Estimating may be difficult initially. However, after a few sprints, teams should be able to estimate with more accuracy. Since a sprint is time-boxed, the team must find an optimal rhythm to ensure that it meets the objectives of a sprint in a time-bound manner.

Ensure continuous process improvement

The Scrum team is responsible for continual process improvement over the course of a project. Teams must proactively participate in Daily Standup Meetings, Retrospect Sprint Meetings, and Retrospect Project Meetings to share their learning and brainstorm for process improvement.

How Embedded QA works in a Sprint

The following example demonstrates how Embedded QA works in a full Sprint Cycle.

Sprint Planning

- Typically requires an entire day.
- Usually involves checking what stories are left in the backlog. Estimating the stories, the top level story does not have an estimate against it; only subtasks have the estimates.
- Dev tasks are suffixed with DEV, QA tasks with the suffix QA. Both should be marked as the correct type, i.e. DEV Task and QA Task.
- Convene a pre-planning meeting before the Planning meeting with team members to go over the estimates. Each story is thoroughly analyzed to ensure everyone agrees with the estimates for both Dev and QA. If concern is raised about a particular estimate, the estimate can be adjusted.
- Each sprint contains a set 8-hour code review.
- Following the review, analysis is done on bugs which should have been pre-estimated and take in as many of the highest priority bugs as possible to fill the remaining time.
- All stories, bugs, etc. should have the proper acceptance criteria.
- A subtask should not have more than a 12-hour estimate. If greater than that, it should be split it up as best as possible to ensure one person is not holding everything up and, if needed, another person can work on the other subtasks.
- Cases where it is uncertain if a story can be developed/QA'd should be rejected.
- The Planning meeting with the Product Manager and Architects should just involve going over the sprint, see that everything is OK, and getting approvals.

Sprint

- Each team member is aware of how this works and should be given authority as much as possible to make sure everyone is being productive.
- DEV/QA – These are one role. There is no reason why QA shouldn't take on a development task if it has a strong programming skillset. This is true for developers as well. Dev should actively encourage QA to try and break their work to reinforce the

mindset that there is no ‘us’ vs. ‘them’. It also makes the person working on a task consider whether this story will pass QA if they resolve it.

- **Reopening vs. logging bugs** – The story is always re-opened if the acceptance criteria have not been met or if what is being tested is implemented in such a way that it was concluded at the demo or during the sprint that it cannot be displayed to a user. If anything else is noticed, a bug is logged against it and a DEV subtask and QA subtask are created with the appropriate estimates.
- **Automation** – Everyone on our team is more than capable of creating automation tests, including selenium, unit tests, and integration tests. If a team member is weak at any of these (or any task), they will be bought up to speed by the more experienced members of the team. Automation is taken very seriously. Always try to find ways to automate and bring it into the build process.
- **Knowledge sharing** – If a task that someone takes on is of interest to other members, we usually create a confluence page or have a workshop for those interested. Similarly, if someone is testing something, they should know exactly what they are testing and figure out how to try to break it. If they need help from the person who worked on it, this is accommodated.
- **Scrum Meetings –** This is a standard Scrum meeting QA/DEV should attend. It gives everyone a chance to talk about what problems they are having and how they can be fixed. If someone has had enough of a certain task, someone could take the task on instead, or at the very least help out. The Product Manager and Architect should be present at these meetings daily.
- **Test Failures Responsibility** – If someone has checked in code that happens to break any test, we know immediately. We all have tray notifiers installed that plug into team city, enabling us to easily see if a test fails and who broke it. Our process is simple; those who break the test, fix it. The normal process is to run the tests before checking in.

Demo Day and Retrospective

- Demo only those stories that are fully tested and closed.
- Usually the person who worked on the task will demo the feature, i.e. if a member of QA worked on the task, they will demo.
- Bugs are only demoed at the request of the Product Manager and Architect or anyone else present.

- After the demo, we have a retrospective. The Product Manager is not present for this, only team members and Architects. This usually involves analyzing the sprint outlining what was done well or more importantly what was done poorly and how we can improve. QA should actively contribute to this.
- At the end of the sprint a merge is done back to the trunk only if all automated tests are passing (which is usually the case).

The above may not work for everyone; however it worked in our sprint. Every member of the team now has the necessary skills to contribute to both DEV and QA, making for very efficient team.

Conclusion

To sum up, it is important for all stakeholders to know about the quality of software produced. In this article, we have analyzed and shown how QA roles have changed in different Software Development Methodology. There is a complete switch of mentality for QA from Waterfall to Scrum and to Embedded QA in Scrum. In Scrum, QA is now expected to be a very proactive part of development. QA will no longer just certify the functionality of the application based on requirements but will be part of the day-to-day development. All members will ensure quality at all levels and act as a communication hub between the business and management.

Neil O'Toole Snr. Quality Assurance Manager for PS App Solutions & Systems Strategy and co-author mentions:

"Organizations want the most out of their quality assurance practices; consumers or sponsors of projects demand accelerated development/release cycles and better software. To achieve this, all barriers between the Development team and the QA team must be removed. Thus, embedding QA into the Development team/process has a number of benefits".

- Cross-pollination (QA learns Development traits/practices while Development gains exposure to the QA mindset and test techniques)
- Whole team is working right at the source
- Constant communication among all team members
- Team ownership through sprint cycle

A story is complete, i.e. done when the code is in place and has passed unit and all other tests "

QA needs to be in on things from the beginning. They should be there when requirements are set, for a number of reasons we discussed. We think the best way to achieve this is by Embedded QA. This would mean that members of the Scrum team need to be flexible and social to make it a success.

Embedded QA solves project management overhead, increases Development and QA interaction, and improves product quality. The success of QA plays a vital role in the success of Scrum.

Ultimately, if we want 'great software', we need to be more than just testers. Don’t separate—get EMBEDDED.

References

http://www.mosaicinc.com/mosaicinc/rmThisMonth.asp

http://www.wisegeek.com/what-is-quality-assurance.htm

[3]http://asq.org/learn-about-quality/project-planning-tools/overview/pdca-cycle.html

http://blogs.atlassian.com/2012/02/tester-skillset-speed-to-cool/

http://blog.agilegamedevelopment.com/2005/03/embedded-qa.html

When-Waterfall-and-Agile-Collide_tcm32-208653.pdf

CandoAgileScrum.pdf

ASQ Agile v Waterfall.pdf

7.2_dhondt_-_ensuring_qa_in_scrum_projects_update.pdf

http://blog.agilegamedevelopment.com/2005/03/embedded-qa.html

http://blogs.atlassian.com

http://agile.dzone.com/articles/waterfall-vs-agile-qa-management

http://www.bth.se/fou/cuppsats.nsf/all/ed80321892f638fdc12576550008771d/$file/qa_activities_in_agile.pdf