Agile QA Process

Anand Bagmar
Anand.Bagmar@thoughtworks.com
abagmar@gmail.com
http://www.essenceoftesting.blogspot.com

Version 1.1

1. Objective

QA is NOT the gatekeeper of the quality of the product. This is a TEAM responsibility. However, we, as QAs, testers, have an additional responsibility – that of being watchful and proactive to ensure the quality of the product remains a TEAM responsibility.

To make this effective, more so in a distributed development and QA team working on an Agile Software project, the QA team should adhere to the guidelines and protocols recommended in this document to help achieve the following:

- Enable the QA team better test the product.
- Provide correct set of visibility in terms of the quality of the product.
- Will be simple, intuitive, easy to follow and achieve.
- Test early and provide quick feedback on the quality of the product.

2. Process

The QA process for the team fits into 3 broad categories:

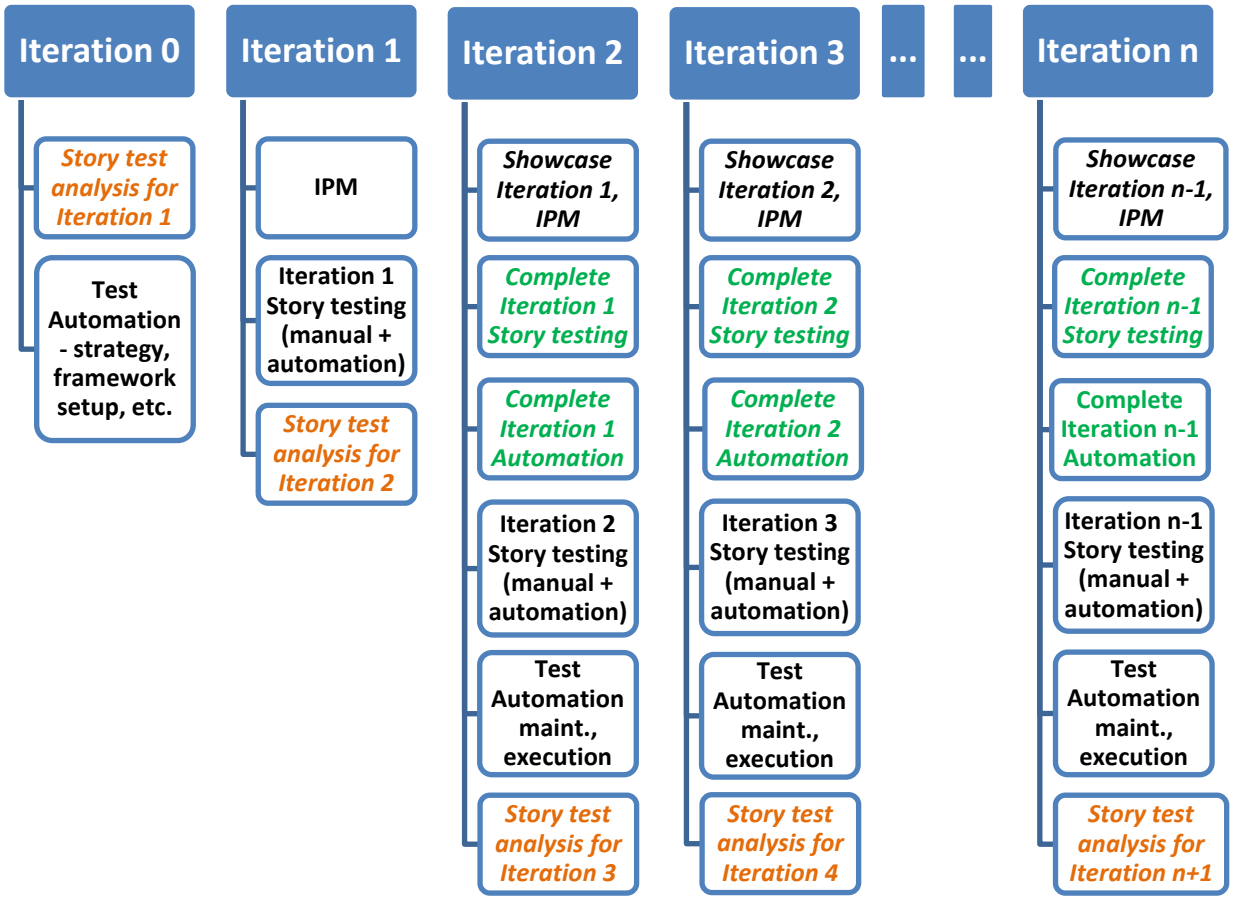
1. Agile Testing - from Iteration 0 to Release
2. Testing within each Iteration
3. Pre-Production / Release testing

Each of these processes is detailed below.

2.1 Agile Testing lifecycle – from Iteration 0 to Release

To be effective in testing on an Agile project, the QA needs to keep track of the past, the present and the future! Also, the role of a QA is very dynamic as the project moves from Iteration 0 till the release date.

The pictorial representation shown below depicts the typical activities the QA is involved in, in the lifecycle of the project.



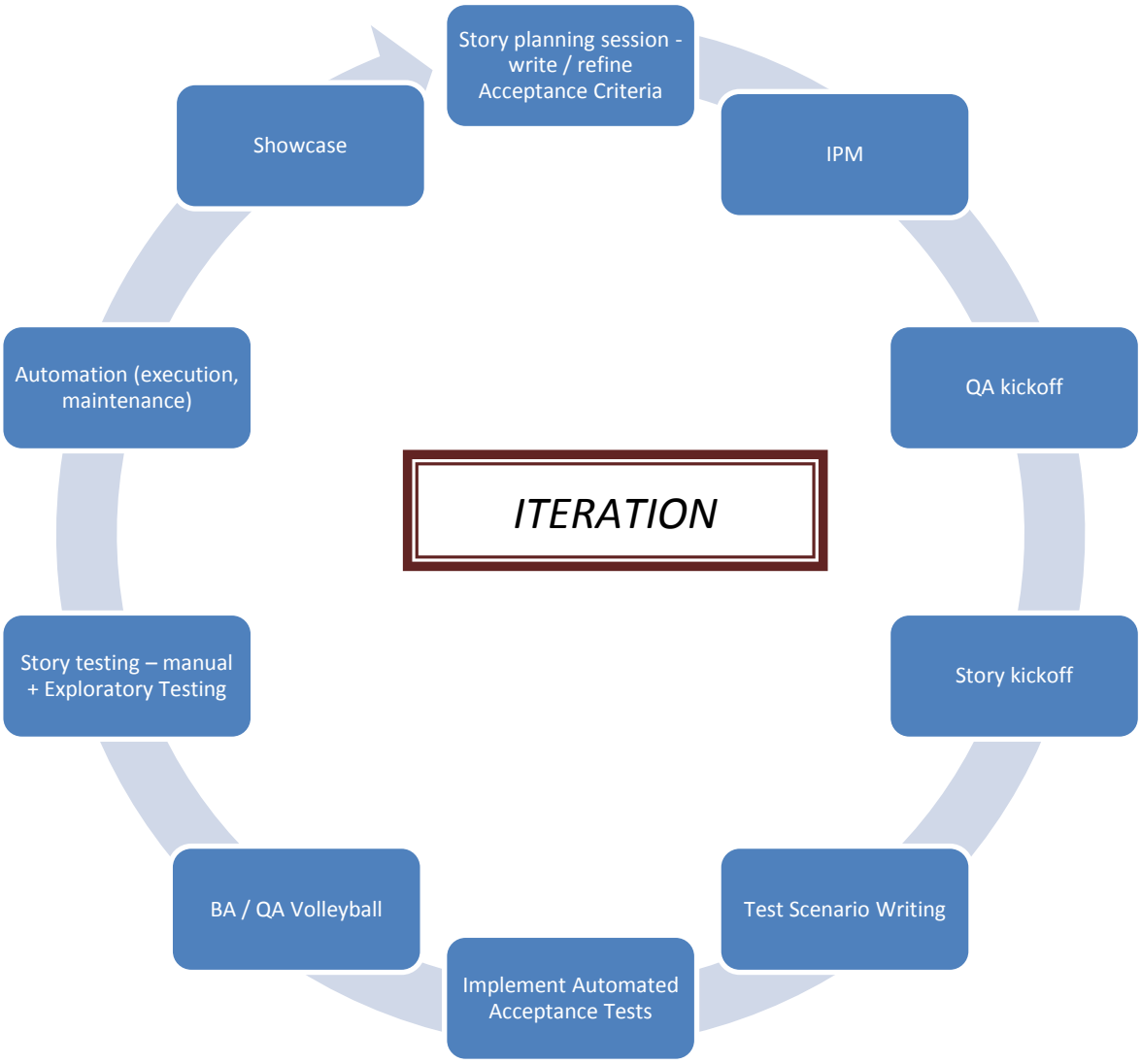
A QA does three types of activities in each Iteration:

1. Ideally, all testing (manual + automation) would be completed in the same iteration. However, in quite a few cases, I have seen that testing lags behind by some duration. Hence, in each iteration, there "*may-be*" some amount of work to be completed from the earlier iteration work.
   This is applicable mainly from Iteration 2 onwards. (See the boxes with the **"*green text*"** in the picture.)
2. Story testing for the current iteration. This testing includes manual and automation work. (See the boxes with the "***black text***" in the picture.)
3. Work with the Business Analysts to get visibility into the next iteration stories, and also help iron out those stories. (See the boxes with the **"*orange text*"** in the picture.)

2.2 Testing within each Iteration

The QA has various responsibilities in the iteration as depicted in the picture below.

NOTE: *Though these activities typically occur in the sequence shown in the image, many a times, based on the context, the sequence may be different, and also some of the activities may happen in parallel.*



Let us try to understand the semantics of these tasks.

2.2.1 Story planning session – write / refine Acceptance Criteria

The planning for the iteration starts at least an iteration before the current one.

Example, if we are currently in Iteration 2, the planning for this iteration has happened at the latest in Iteration 1.
In this step, the QA (at least one person from the QA team) should work with the Business Analyst to understand the story, and help write the acceptance criteria.
This helps in various ways:

- There is better visibility of the business requirements.
- Better and more complete acceptance criteria can be written in the story as early as possible which will then feed into the development team.
- Since the QA team now knows what stories are coming up, they can plan their testing strategy more appropriately.

2.2.2 Iteration Planning Meeting (IPM)

The IPM is a forum for the QAs to understand the business priorities and which stories are going to be played in the Iteration from the business stakeholders. The QA team should make full use of this opportunity and ensure the following:

- Make it effective for onsite and offshore QA team
- Raise risks / issues as soon as possible
- Ensure QA capacity / estimates are accounted for
- Identify / highlight dependencies

2.2.3 QA kickoff for the Iteration

After the IPM, the QA team should have a QA kickoff for the iteration. In most cases, the Business Analyst should be invited to ensure correct understanding by the team.
Items to cover in this step include the following:

- Overall understanding of the stories
- Any potential issues to watch out for
- Identifying and setting up test data for all the stories coming up
- Identifying automation maintenance tasks based on upcoming stories
- Creating a Status Matrix to identify priorities. (See below for more details on this.)
- Each QA will then signup for testing specific stories.
  In case of distributed team, a good guideline to follow is that the ***QA team should sign up for stories that are developed in the same location***. This will ensure quick turnaround time for testing each story.

NOTE: In many cases the points to be included in the QA kickoff would actually be covered in the IPM itself, hence making this step redundant.

2.2.4 Individual Story testing

2.2.4.1 Story kickoff

Once the story signup is done, there is a Story kickoff with the Business Analyst, the developer(s) and the QA person. This is a very important discussion that gets all involved parties on the same page in understanding the requirements and the deliverables of the story.

This discussion should be as detailed as possible with the QA(s) trying to identify and share the test scenarios with the BA and the developers. Also, input from the BA and the developers should be taken on any scenarios that may have been missed, or based on the understanding of the code changes involved, if any other scenarios need to be included when testing the story.

2.2.4.2 Test data preparation

Following the story kickoff, the QA should start preparing / identifying the test data that would be needed to effectively test the story – manually and via automation. A different set of data should be used for manual, automated and developer testing to prevent data-conflict related test failures.

This activity should be done keeping the other QA team members and development team members in the loop to ensure no overlap in the test data. Also, if any scripts / setup can be shared, that would help save time and effort and also duplication can be avoided.

2.2.4.3 Test scenario writing

The test data preparation and test scenario writing activities go hand-in-hand, and not necessarily sequentially. The (manual and to-be-automated) scenarios to be tested for this story should be identified and updated in the Test Case Management tool being used.

Once the test scenarios are written, these should be shared / reviewed (at least informally) with the Business Analysts and the developers developing this story, to ensure the scenarios are complete, and also the developers can probably identify missing pieces in the code at a much earlier stage.

2.2.4.4 Automate test scenarios

The QA team needs to identify test scenarios that make sense to be automated, and which can be automated. Then the script automation should be started.

Since the story is still not complete, there probably will be limitations how much can be done at this stage. However, in most cases, a lot of ground work and framework layers can be created without needing the completed story.

After the BA / QA Volleyball, the missing pieces in the automation scripts can be completed by pointing the tests to a local / developer environment.

QA's should be mindful that the Test Automation code should be of "production" quality. They should use proper OOPs (Object Oriented Programming) concepts while implementing the

automated tests and ensure that any new code that is written is NOT duplicated, and structured properly.

2.2.4.5 Business Analyst (BA) / QA Volleyball

Once the developers complete the story development work, there should be a BA / QA Volleyball. What this means is that the BA and QA will test the story's acceptance criteria on the developer's local environment. All the acceptance criteria failures should be fixed by the developer before the story can be said to be "development completed".

2.2.4.6 Test the Story

After the BA / QA volleyball is completed, a new build should be made available to the QA team to test the story based on the scripts identified and documented.

NOTE: The QA team should never be working against testing the developer code. Instead they should always insist on testing the compiled and deployed binaries in the proper test environment to simulate the end-user scenarios.

Once a build is available in the appropriate environment, the QA should execute the manual test scripts.

There should be at least one round of ***Exploratory Testing*** for the story, and the features / functionalities around the changes that have been developed as part of the story to ensure there are no regressions or other issues in the product. Based on the time available in the iteration, this can happen either on a developer sand-box, or deployed builds.

Exploratory Testing is very important and typically a lot of issues are found from this type of testing.

2.2.5 Automation

All the automated tests should be run at least a couple of times per iteration, preferably via some CI (Continuous Integration) mechanism. Else the automation is not of much value. The frequency of these test runs should be fine-tuned keeping in consideration the load on the system, availability of key resources needed by the system, etc. The test run reports should then be shared with the team.

Also, the automated tests should be grouped logically to facilitate running selected tests easily.

Note: The list of groups would probably be an evolving list.

2.2.6 Showcase

The functionality developed in the iteration is presented to the Business Stakeholders at the end of the iteration (, or, in some cases earlier too). It is important for a representative from the QA team to be part of this event in order to understand how the functionality was received by the various attendees, and also if any other new information (in terms of requirements, test scenarios) comes up in the session.

3. UAT / Pre-Production / Release Testing

This testing will be done using the guidelines published by the client organization.

4. Defects reporting and verification

Along with the standard / regular defect reporting and verification process, a few additional activities help a lot in reducing the turnaround for defects.

When reporting the defects, attach screen shots / videos (where applicable) to help the developers reproduce the problem easily. This will ensure in quicker turnaround on the defect.

Example: The tool "Jing" (http://www.techsmith.com/jing/) has a free tool version that can be used to take screenshots and video snippets from your computer.

5. Other

5.1.1 Code freeze for the iteration (example: 2 days before iteration end)

This will give the QA team a chance to do a quick pass of manual + exploratory testing of the stories in the same iteration. Without this there is always going to be a spillover of stories in the next iteration.

5.1.2 QA estimates

A story estimate really means the total time it takes to develop and test the story. If this estimate changes to being a "development" estimate, then we need another approach.

A 1 point Developer story does not equate to a 1 point QA story. Hence, each story should have QA estimates which would allow better capacity and scope planning for the iteration and release.

This also facilitates data collection over a period of time w.r.t. the actual time needed by the QA team to do effective testing versus the currently allocated QA capacity, so that the management team can then take proper action to mitigate the issues.

5.1.3 Visibility

An appropriate dashboard should be created that provides all relevant details about the quality of the product under test.

Example list of items to include in the dashboard:

- Number of stories in the iteration in "Ready for QA" state
- Number of stories in the iteration in "In QA" state
- Number of stories in the iteration which are not yet in "Ready for QA" state

- Number of Defects open (including the iteration reported in), with current assignees, priority and severity
- Number of Defects pending verification
- Automation backlog (Stories for which Automation is not yet completed)

5.1.4 Communication

Effective communication and collaboration within all the team members (co-located, distributed) is critical to the success of an Agile software project.

A few tips mentioned below can help in this regard:

- Daily catch up calls / emails / IMs
- Sharing knowledge of story across QA team
- Keeping the QA status in the story updated
- In case of distributed teams, if a story changes location for any reason, the QA team needs to be made aware of the change, and they need to ensure testing is not going to be affected for it.
- Any change in story (addition, deletion, modification of narrative / acceptance criteria) should be communicated to at least one QA team member by the PM / BA / developer. This QA team member will then coordinate with the QA working on the story (if not the same person) and ensure all testing related to the story is adjusted appropriately.

5.1.5 Task list

There should be a common page / task list that the QA team across locations can keep adding / editing tasks they encounter along the way in the Iterations. Any QA team member having bandwidth then signup for it along the way and complete it.

The list can be of the following format:

Number	Task / Item	Estimated effort (in days)	Signup	Status	More notes

5.1.6 QA section in each story

Each story should have a QA specific section to help track the testing status on each story.

QA Assignee

QA Estimate - Text box

Defects reported - Text box

QA Status - each status should be separately tracked - either as a checkbox or radio button (except Automation), or as a drop down list.

Status	**Description**
Kickoff Done	for when the kickoff for the story was done
BAV Done	for when BA volleyball was done
Test Scenario Done	for when the test scenarios (manual and automation) were identified and documented in Quality Centre
Automation **> In Progress** **> Blocked** **> Done** **> N/A**	what is the status of test Automation for this story
QA Complete	when the story was done and tested

5.1.8 QA Status Matrix

This matrix can help the QA team come up with priorities of their tasks with respect to when the stories for the current iteration are going to be available.

The matrix looks like this:

Current Iteration # 2

Story \ Date	Sept 22	Sept 23	Sept 24	Sept 27	Sept 28	Sept 29	Sept 30	Oct 1	Oct 4	Oct 5
#10										
#11										

The 1st column lists the Story numbers. The 1st row lists the days of the iteration. Based on communication with the Business Analysts and developers working on each story, the QA team can use different status to track when the story will be available for them for testing. They can then use this information to prioritize their other tasks.

The following different status messages can be used on this chart:

Status	Acronym (Estimate)	Acronym (Actual / Completed)
Kickoff done	K	K
Development complete ETA	D	D
Scenario done	S	S
Automation	A	A
Complete	C	C

6. Disclaimer

"QA" Vs "Tester": There are various different terminologies and everyone probably has a different view about that. To me this is just a label.

In the past 11+ years I have carried the label of "Test Engineer", "Software Engineer", "Software Engineer in Test", "Software Development Engineer in Test", Product Quality Engineer, Quality Analyst, etc. Regardless of the label, I have pretty much been doing slightly different manifestations of the same thing – and that is doing what is required to ensure the quality of the product that is being shipped out is of top quality.

I use "QA" Vs "Tester" depending on who I am talking with, so that the set of people can relate better to what I am saying.