Agile Testing

- The Testing Manifesto
  - Testing throughout over testing at the end
  - Preventing bugs over finding bugs
  - Testing understanding over checking functionality
  - Building the best system over breaking the system
  - Team responsibility for quality over tester responsibility
- Agile Testing Quadrants
- References

With the adoption of Agile in software development, it's natural to extend it to testing related activities as they're also an intrinsic part of development.

There have been many contributions on this topic, including the Testing Manifesto and the Agile Testing Quadrants.

While the Testing Manifesto shows us how to look at testing in an Agile context and the mindset changes from "old ways" of looking at testing, the Agile Testing Quadrants provide a different perspective: what kinds of testing and tests are there, how can they be grouped, their purpose/fit and how to choose which testing to perform at any moment based on the quadrants and the risks we want to mitigate.

The Testing Manifesto

The Testing Manifesto, by Karen Greaves and Samantha Laing, provides some insights on how we should think about testing in an Agile context.

There are 5 key ideas that you need to pursue in order to have a more agile like mindset.

Testing throughout over testing at the end

Traditional waterfall-based teams tend to perform testing only at the end, after coding/implementation.

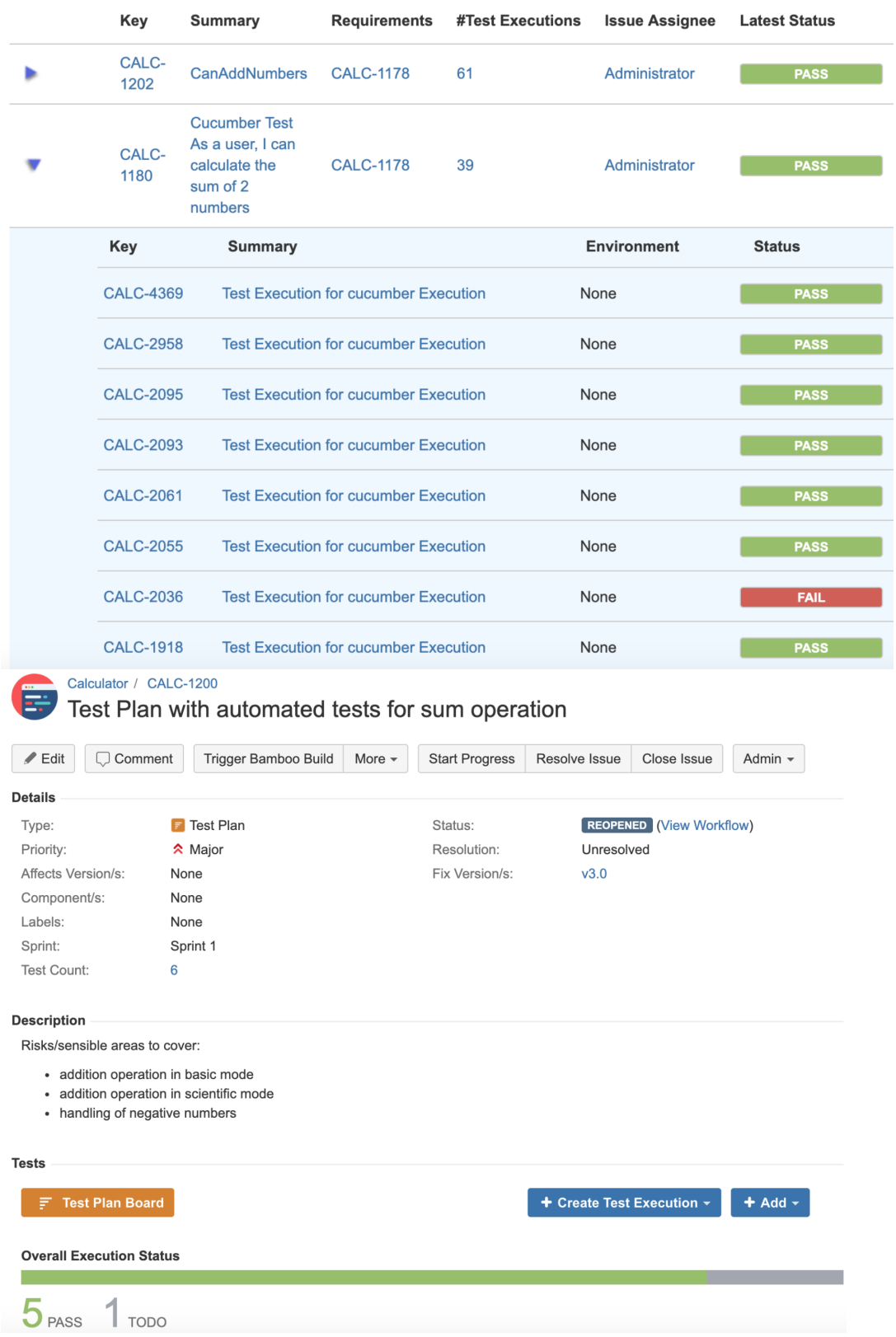
Nowadays we know that we should think about testing more deeply during the whole software development life cycle. Thus, testers are involved in discussing/reviewing the requirements (or stories, in Agile context) and other issues targeted for some release/sprint.

Xray promotes this by fostering collaboration between all team members, which are all standard Jira users and thus may access all information, at any time.

A Test Plan can be built ASAP to define the testing scope and include, among other things, the automated tests results.

Automated testing plays a key role, starting with unit tests and then integration-level tests, followed by other tests, at a higher level of the Test Automation Pyramid.

Automated tests can run as many times as you wish during the whole development life cycle and their results can be tracked at the Test Plan level.



Key	Summary	Requirements	#Test Executions	Issue Assignee	Latest Status
CALC-1202	CanAddNumbers	CALC-1178	61	Administrator	PASS
CALC-1180	Cucumber Test As a user, I can calculate the sum of 2 numbers	CALC-1178	39	Administrator	PASS

Key	Summary	Environment	Status
CALC-4369	Test Execution for cucumber Execution	None	PASS
CALC-2958	Test Execution for cucumber Execution	None	PASS
CALC-2095	Test Execution for cucumber Execution	None	PASS
CALC-2093	Test Execution for cucumber Execution	None	PASS
CALC-2061	Test Execution for cucumber Execution	None	PASS
CALC-2055	Test Execution for cucumber Execution	None	PASS
CALC-2036	Test Execution for cucumber Execution	None	FAIL
CALC-1918	Test Execution for cucumber Execution	None	PASS

The code of automated tests can be linked to Test entities and reviewed, as you would do for user stories or bug fixes. Even during PRs, you can obtain a testable asset and execute tests against it (automated or not).

But, besides automated testing, manual and exploratory testing can be performed, as many times as you wish, in different Test Executions that can, once again, report back to the same Test Plan.

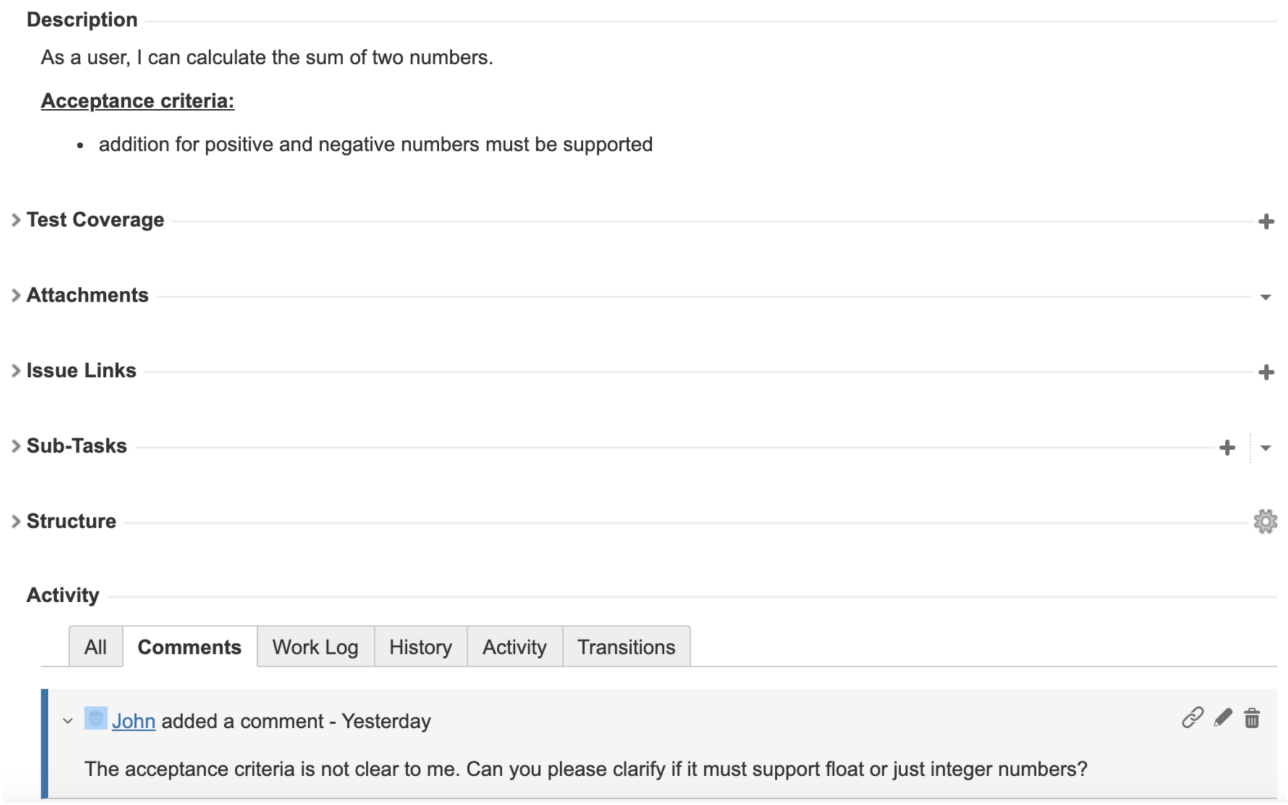
Thus, testing happens throughout, in different kinds of activities, to make sure your deliverables have the quality they need.

Preventing bugs over finding bugs

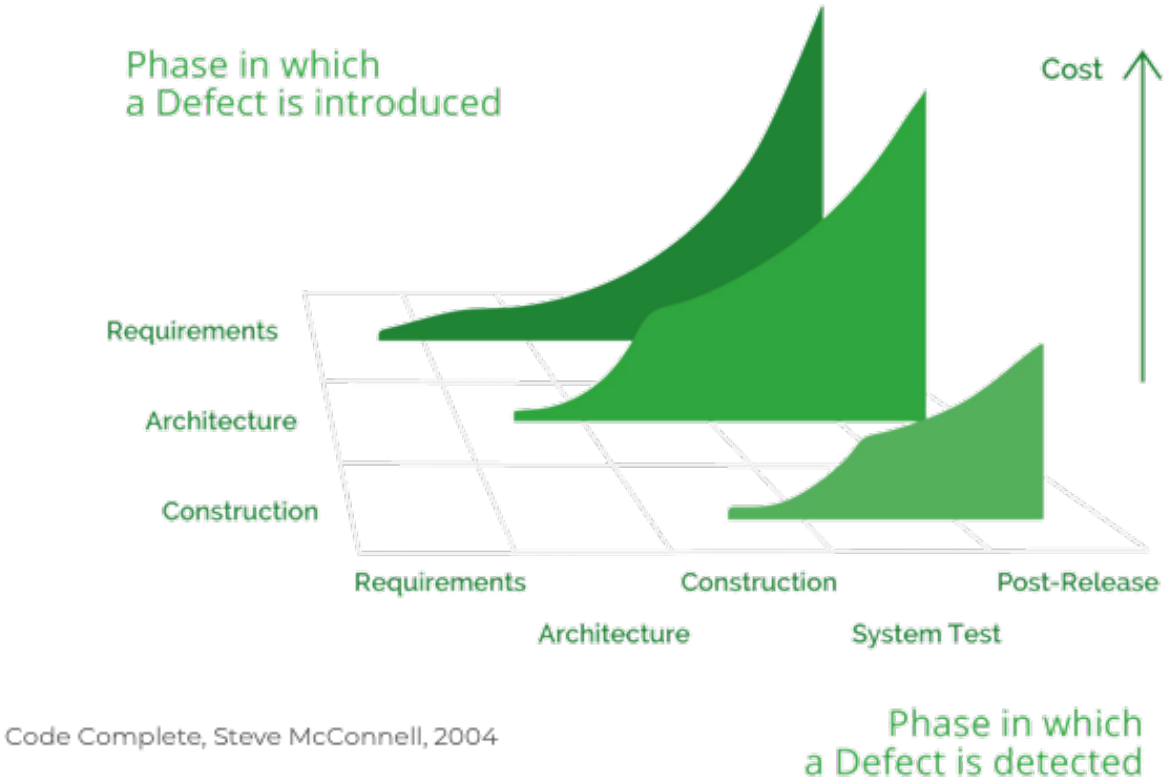
One of the several aims of testing is to prevent bugs (not exactly finding them) right from the start.

The best way to prevent bugs is by having a common and clear understanding of the issues (e.g. stories) you are going to address.

Xray provides the means to all team members, including anyone performing testing, to participate in the review of work items, discuss and clarify them before they're committed and their implementation starts.



Avoiding bugs right from the start is essential to avoid high bug fix costs.



Although we may not be focused on "finding bugs," if they arise we need to make sure they're found ASAP. Having automated testing, starting with unit tests and going the upper layers, is essential to find bugs and fix them right away.

Testing understanding over checking functionality

Checking functionality, i.e. checking that the system works as expected, can be done through traditional check based tests, either manually or through test automation, where the latter is the preferred approach. Xray supports both approaches, being a great supporter of test automation. Tester's knowledge is valuable here to make sure the right checks are made.

But besides checking something when or after it is built, we also need to make sure the right things will be addressed. Testers can demand a proper understanding of customer needs and iterate as much as needed until they have the necessary information, which in turn can be used to make better testing and also building features the right away.

BDD can be used to better clarify the behavior of features by describing the ultimate benefit behind it. Remember that features by themselves don't add value; it's the benefits they provide and the needs that they cover. Thus, all starts with proper user story description while always having the customer in mind.

From the clarified user story, proper testing scenarios can be depicted, using Cucumber, or check/step-based tests (manual or automated). Cucumber can have an important role, because its Gherkin based syntax (i.e. Given, When, Then) in natural language provides a clearer understanding of the test scenarios (what is their purpose and what they do exactly).

Both epics, user stories, and test cases can have a formal, lightweight, reviewing process; in all of them, all team members, including testers, can interact and leave comments.

Through better, clarified and reviewed user stories, clear test cases for validating customer usage scenarios can be specified. Test cases are created considering the risks identified by the team as a way to mitigate them. Tests will then be executed with the risk in mind, knowing that this risk is dynamic and also depends on the implementation.

Building the best system over breaking the system

We need to make sure we're building the best system, that addresses customer needs in an efficient, usable, performant and stable way. There are many quality attributes to have in mind though.

Focusing on "breaking the system" does not improve many of the quality attributes we aim to deliver.

As an example, BDD and Cucumber based Scenarios can help in building the right system, by involving testers, developers, and customers.

Test Details

Type:	**Cucumber**
Scenario Type:	**Scenario Outline**
Scenario:	

```
Given I have entered <input_1> into the calculator
And I have entered <input_2> into the calculator
When I press <button>
Then the result should be <output> on the screen

  Examples:
    | input_1 | input_2 | button | output |
    | 20      | 30      | add    | 50     |
    | 2       | 5       | add    | 7      |
    | 0       | 40      | add    | 40     |
    | 4       | 50      | add    | 54     |
    | 5       | 50      | add    | 55     |
```

Promoting collaboration between all "stakeholders" in any of the issues being worked on is essential to build the **best** system. Testers can provide valuable feedback besides authoring and executing tests and this feedback may be provided, or even given to them, on any of the work items.

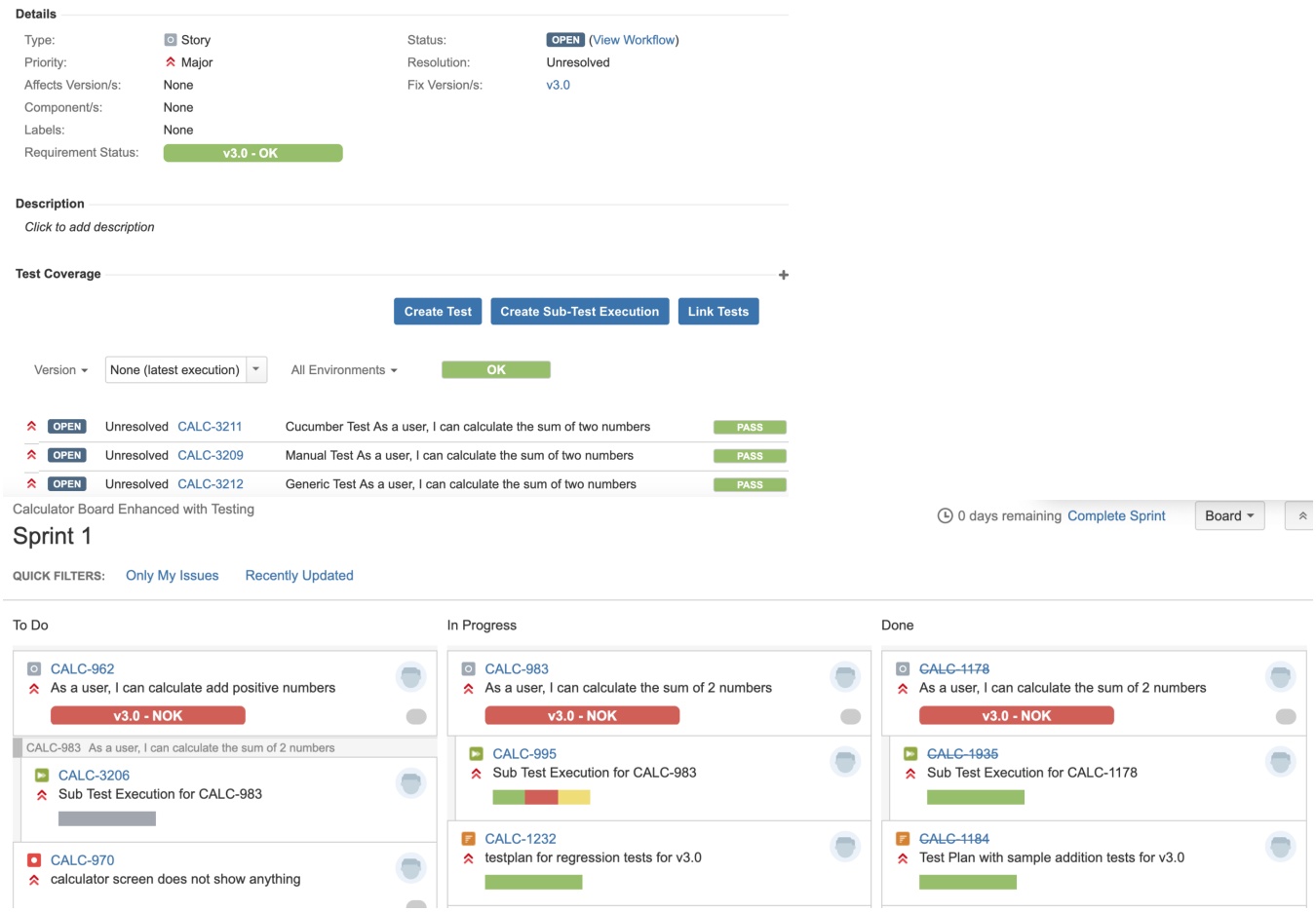
Xray also provides the ability to adopt exploratory testing as a way to explore the system in a critical way and provide precious feedback to the team, to make the feature and the overall system better.

Team responsibility for quality over tester responsibility

The best way to have team responsibility for quality is to close the gap between testing and programming. Team members work together in order to achieve common goals: having features/issues implemented the right way and working as expected.

To make quality as natural as possible, Xray provides coverage analysis capability, allowing anyone to assess real-time information on whether a user story, for example, is covered with Test cases and, if so, how the latest results of those Tests affect the (coverage) status of the story.

Thus, by having coverage information always present, on the issues being worked out and also on the Agile Boards where team members manage and track their progress, everyone can see their real status (e.g. OK, NOK), which also makes everyone responsible for taking actions in order to make sure they are OK (i.e. properly tested).



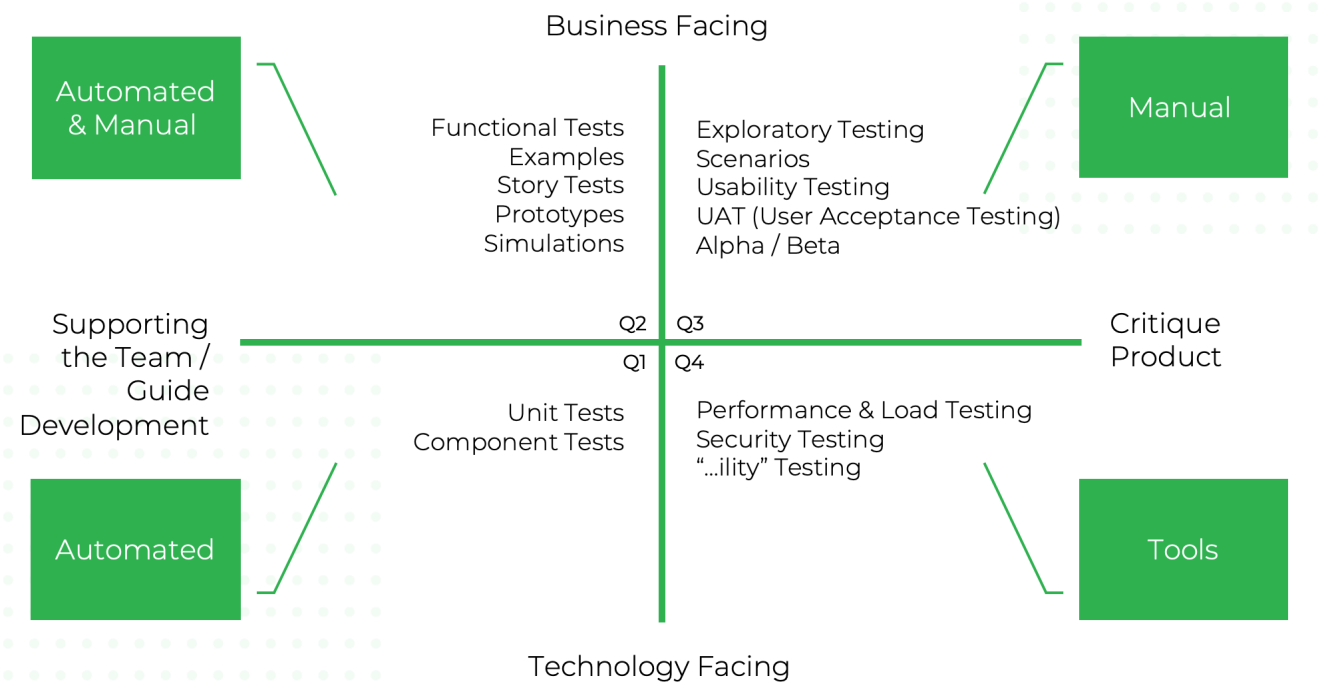
Agile Testing Quadrants

The Agile Testing Quadrants (by Brian Marick & Lisa Crispin) provide a way to group different kinds of testing in 4 quadrants. This approach considers whether they are business or technology facing, and whether they are mostly used to support the team and guide development or to critique the product.

Quadrants are numbered although their number is just a mere reference to be used between team members.

The idea is to continuously assess risks and to mitigate them focusing on the quadrant(s) that can better target those risks.

Quoting Lisa Crispin: "***Consider where the highest risk might be and where testing can add the most value.***".



What is clear from this diagram is that Tests can either be manual, automated or you can also use tools to help you perform certain types of tests (e.g. for load testing).

The value provided by each testing approach ("manual", exploratory, automated) and for each different type of test is different and complements each other.

Xray may be used to manage all sorts of Tests, have visibility of their results, and see how they impact the related stories/epics.

In order to perform testing from the fourth quadrant, you may use whatever tools you want. Likewise, for implementing automated tests you may use whatever automation framework best covers your needs, including Gherkin based ones (e.g. Cucumber, Behave, SpecFlow) among others. Xray does not enforce or add restrictions in using a specific tool whatsoever, thus your team can choose what they think is best and still track your testing progress in Xray.

References

- Agile Testing definition by Lisa Crispin & Janet Gregory
- Testing Manifesto