Working with Automated Tests

Managing Automated Tests

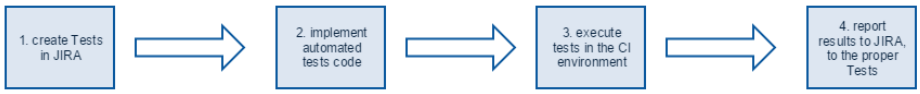
You will need to have visibility of the results of all your tests, regardless of how they are executed.

Your test results will affect the status of your requirements in the same way, no matter if they're manual or automated.

As a team member, you will need to evaluate your requirement coverage at any moment. For this, you will need to link your Tests with your requirements, and report your automated test results from your CI environment.

Xray is the ideal tool for this because you manage manual and automated tests essentially the same way.

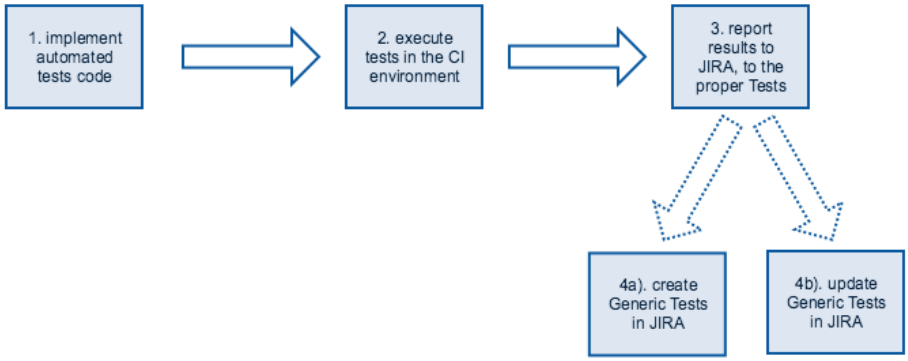
This is the standard workflow:



However, with automated tests and Agile teams, tests are normally written continuously during development. Keen developers will add as many tests as possible or, at least, the most relevant ones.

Thus, the most common workflow when dealing with automated tests would be to write the automated tests code, execute them and then import the results and the test themselves to Xray.

Note that in the final step, Tests are only created as entities in Jira only if they are "new" Tests. This means that the Tests must be uniquely identified.



Type of Tests used for automation

Xray provides two different type of tests that may be used to represent automated tests:

- **Cucumber**: a test specified in natural language (i.e., in Gherkin)
- **Generic**: all other automated tests (or any automated test in general)

> **Learn more**
>
> See Testing in BDD with Gherkin based frameworks (e.g. Cucumber) and Using Generic Tests for Automation to learn how to use the two different types of automated tests.

Understanding functionalities

It's important to understand how the different tools fit together and the functions of each one.

Jira	**Xray**	**SCVS** **(e.g., GIT, SVN, Mercurial)**	**CI tool** **(e.g., Bamboo, Jenkins, Team City)**
• manages requirements • manages releases • etc.	• manages Test-related entities • mapping between Tests and requirements • execution of Manual Tests • stores test results • provides info on requirement coverage and test progress	• stores product code and automated tests code	• executes builds (periodic or triggered by code commits) • executes automated tests (periodic or triggered by code commits) • stores detailed execution logs • stores build artifacts