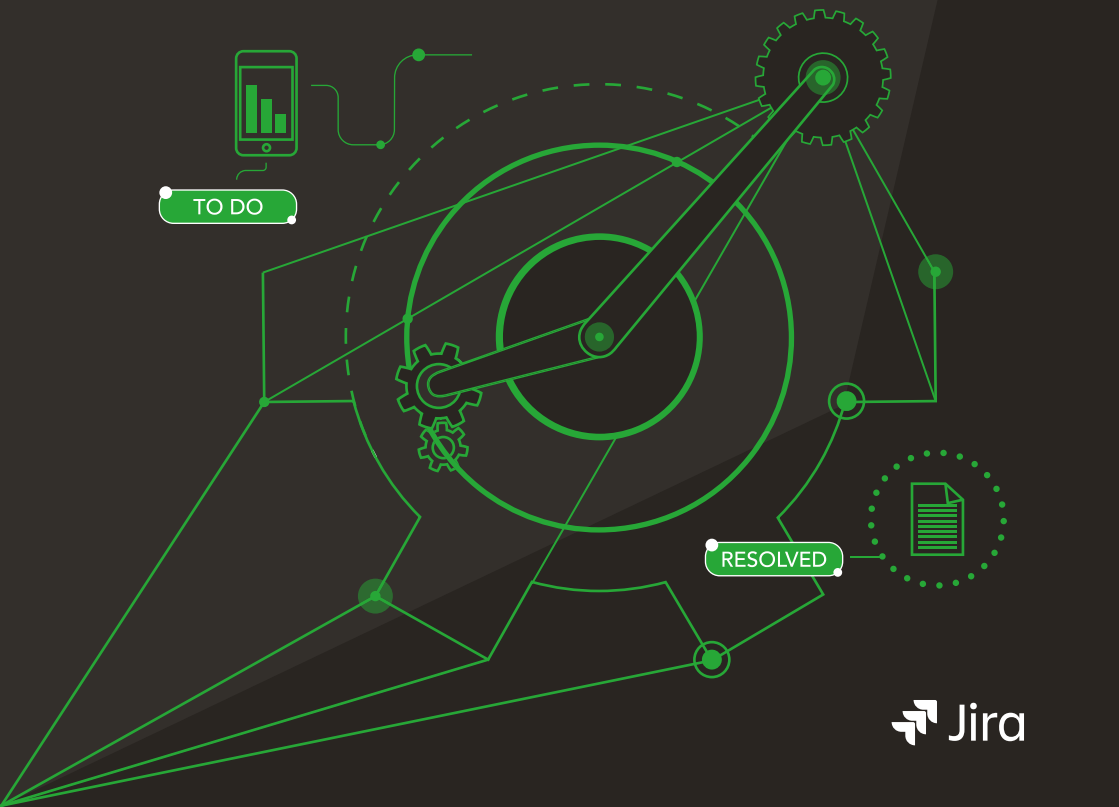
XRAY

Delivering better software using **Test Automation**



Jira

Content

Introduction

When companies today are trying to ship as fast as possible, test automation is becoming more and more essential.

According to a recent survey by PractiTest & Tea-Time with Testers, 85% of organizations have some kind of test automation in place. It's being used mostly for functional or regression testing but also for load testing, unit testing and **Continuous Integration**.

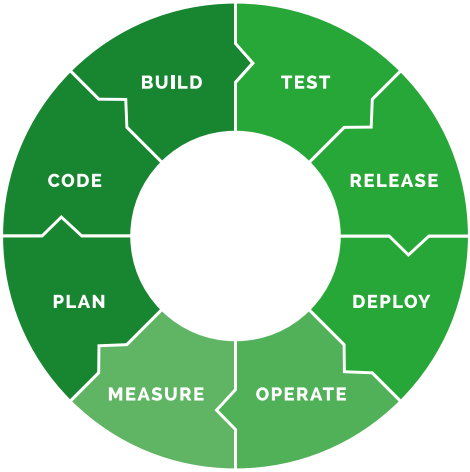
Similar to the build and deploy phases, testing is another part of the **Continuous Integration** and **Continuous Deployment** pipelines capable of being automated.

"Automation does not do what testers used to do, unless one ignores most things a tester really does.
Automated testing is useful for extending the reach of the testers work, not to replace it."

James Bach | Founder and CEO of Satisfice Inc

But why is **test automation** that relevant?
Can it really help deliver better software, at a faster rate?

We all know the demand to deliver faster and better software is constantly increasing. So, how do you adapt? More importantly, where do you start? Let's explore some options.



Why test automation?

2

As shown in *Gartner's World Quality Report 2016-17*, the benefits of **test automation** include quality, process, timing, awareness and costs related aspects.

Test automation:

· **Reduces time to ship** by validating builds faster;

· **Reduces testing effort** by running most of the tests as automated ones, thus decreasing the time needed for testing;

· **Reduces costs** not just by demanding less human resources for test execution, but also by detecting bugs sooner on the development lifecycle;

· **Makes continuous integration and continuous delivery** practices possible;

· **Avoids partial/cumulative testing.** When there is no time or resources left, user stories are validated independently on different code revisions, and the overall status of the user stories in a given cycle (sprint/version) is inferred;

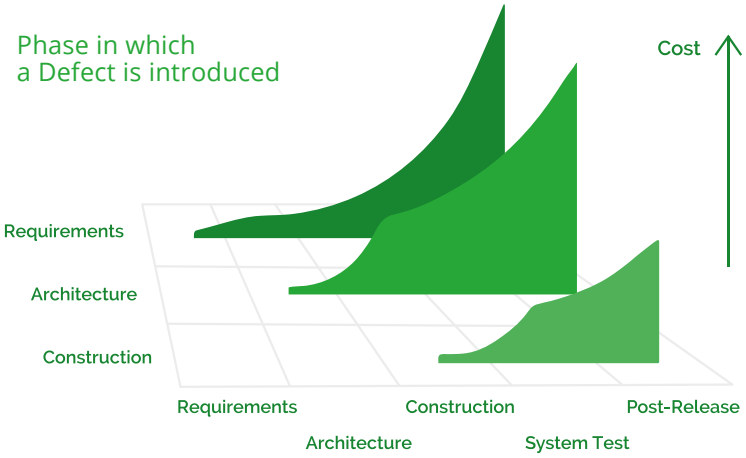
· **Makes regression testing banal** since automated regression tests can be run alongside other tests;

· Avoids human error by means of repeatable automated tests executed on clean "resettable" environments;

· Makes everyone happy, including developers, by detecting bugs as early as possible and avoiding greater problems afterwards.

Studies done by organizations such as NASA show that the cost of bug fixing grows exponentially depending on the phase in which they are detected. The fix cost also depends on the stage where bugs are initially introduced *(Code Complete, Steve McConnell, 2004)*.

Overall, bugs that are found later or introduced sooner have greater costs. Thus, by involving testers from the start, namely in the requirements specification phase and moving testing to the first development phases, by means of shift left testing practices, will help reduce overall bug fixing costs.



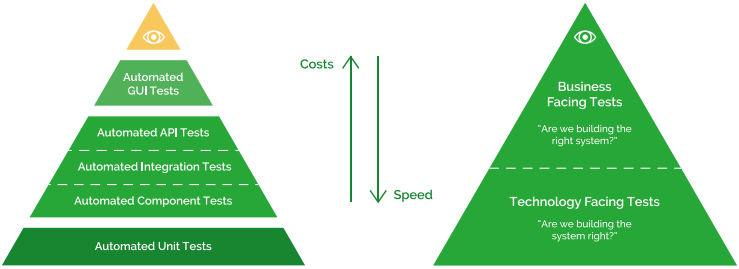
Test automation may help substantially here since it can be implemented early in the coding stage, in the form of unit tests.
With test automation you can implement component integration, functional and UI tests, to cover the whole testing spectrum.

> *85% of bugs happen at the first stage of the software development life cycle. The cost of catching a bug later in the development cycle is 40x the cost of finding it during coding*

Where to start

3

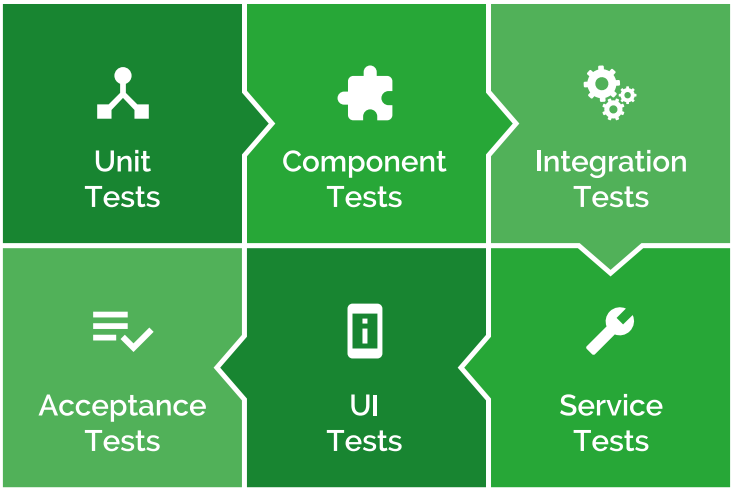
Mike Cohn's famous and widely-adopted Test Pyramid summarizes how the testing automation effort should be spent, depicting great focus on **unit tests** and significantly less on **UI testing**; special relevance should be given to "service tests" for validating the business logic.
WatirMelon presents a variant of this pyramid concept, giving focus on the different types of automated tests and their aim.



When we scale the pyramid up, costs go higher as the overall test run time. On the other side, tests at the bottom are simpler, faster to run and highly focused.

Some organizations tend to implement the inverted **Test pyramid** though, giving greater focus to **GUI tests**. This is mostly a side effect of a waterfall model or a shift right testing approach, where testing is done later in the development life cycle, at the end of Sprints or releases.

4 Types of tests and automation framework



Unit tests

Unit tests are easier and faster to implement and maintain, thus the preferable place to start with automation.
These type of tests will essentially evaluate if the system is being built correctly, and at the same time assess the code coverage. Each own language or platform has its own preferred unit testing frameworks, such as JUnit for JAVA or NUnit for C#.

Integration and component tests

Integration or component tests will evaluate if the individual parts that make up the solution are working and integrating with each other as expected.
They are relatively easy to implement and provide the means to prevent errors harder to track further ahead.

Service tests

Service or functional tests validate the business logic without involving the **UI**, thus their focus is more on the ***"Are we building the right system?"*** question.
These are a great fit to complement unit tests and can be run in a **continuous integration** scenario since their overhead is acceptable.
Some helpful **BDD (Behaviour Driven Development)** frameworks for this purpose include **Cucumber** or **RSpec**.

UI tests

On the other hand, **UI** or **end-to-end tests** are the ones that validate business' or user's demands simulating real usage. Typical examples of these make use of web automation frameworks such as **Selenium** for driving browser tests.

Browser or UI tests, in general, are easy to "sell" but have many drawbacks: they're expensive, hamper finding bug root causes, are slower and harder to stack into the continuous integration pipeline.

There are some other frameworks that allow complementary UI tests such as visual regression testing (e.g. PhantomCSS), which for some cases may be relevant.

Acceptance tests

On top, or in parallel, you may have acceptance tests as a way of formally ensuring the specifications are being met.

Cucumber, or any BDD tool, eases the writing of automated acceptance tests.
In fact, these tools distinguish the test specification from their implementation.

In Cucumber, for example, Tests are specified as a sequence of phrases written in natural language, which will be individually matched by the tool to find the proper test code. Each of the phrases may then be reused to specify another test scenario. BDD frameworks are great because they promote specification, closing the gap between business stakeholders, developers and testers.

Consolidate your results and deliver better software

5

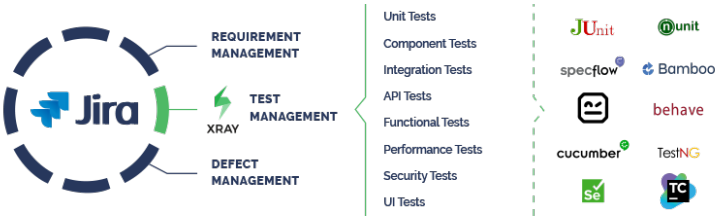
Choosing a **test management** tool that allows both requirement and test management, will make your team's life easier and give enhanced visibility into the progress of your projects.

Jira is great for this purpose, as you may ascertain in the free eBook Jira Software - Quick Start Guide for Test Management, since among other things, it:

- fosters collaboration;
- is highly flexible, extensible and can be effortlessly integrated with;
- keeps track of changes, ensures accountability and traceability;

> *Jira is great since it's highly flexible, extensible and can be effortlessly integrated with.*

Whether you're doing manual or automated testing, or a mix of both, at whatever level, it's important to have a consolidated overview of the status of your requirements and the progress of your testing. With a proper test management tool, you're able to answer ***"Did we test everything? Have we missed any important features?"*** and the ultimate question ***"Are we ready to go live?"***.



A tool such as Xray facilitates the management of both manual and automated tests, independently of the **testing frameworks** and automation libraries being used, providing clear visibility of the requirement coverage and test progress at any moment in time.

Whatever approach to the **test pyramid** you adopt and **testing frameworks** you use, your testing should be reflected in the quality of your software in a manageable way, so you know the risks of releasing a build beforehand.

There may not be a perfect build, but automation and automated tests will help you obtain better software, faster and consistently.
And reduce costs!

So, what are you waiting for?

Introducing

6

Test management with Xray

Xray is one of the Top 3 bestselling apps in the Atlassian Marketplace and winner of Atlassian's Fastest Cloud Growth in 2019.

More than 4.5 million testers, developers and QA managers trust Xray to manage 100+ million test cases each month. Xray is a mission-critical tool at over 5,000 companies in 65 countries, including 137 of the Global 500 like BMW, Samsung and IBM.

> *Xray issues perfectly map your testing activities, while inheriting the best of Jira.*

Enterprise testing features with Jira

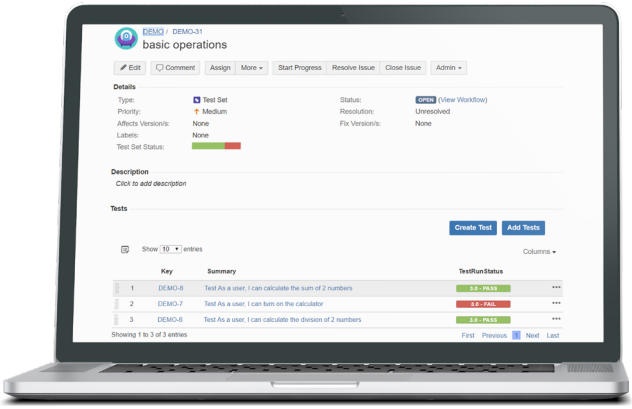
Xray uses Jira issues to represent most of the testing related entities.

Thus, you are able to leverage all Jira benefits into your tests, such as issue assignment, comments, screens and workflows customization, issue linking, custom fields and many more. Besides accountability, it's possible to audit changes using issue's history.

Xray is one of the few tools that assures **data consistency**, by storing test definitions for past executions, guaranteeing the integrity of previous test runs, no matter what you do afterwards.
Finally, Xray provides extensive configuration options, to fully adapt it to your specific needs.

Testing oriented

Xray allows you to maximize all your testing activities, from test design to reporting.
The test design phase includes the specification of both manual and automated tests using **Test issues**. Sometimes you'll need to ensure that some conditions are met before executing a Test. Xray allows you associate one or more Pre-Conditions and reuse them between tests.



Automated and Manual Tests

It does not matter how tests are implemented, Xray handles them in the same way, allowing users to have an integrated and consolidated view of test management and the overall requirement coverage.

Xray allows you to specify Cucumber automated tests in your natural language *(e.g. English)* using Gherkin. Xray even allows you to manage generic automated tests, no matter what automation framework you're using, really!

> *BDD with Cucumber is supported natively but tests may be automated in any framework*

```
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     |
```

Conclusion

In summary, test automation enables you to validate builds faster, reduce costs by detecting bugs sooner and even make regression testing banal.

It isn't a silver bullet, though. We recommend that you still use manual testing wherever and whenever appropriate and to complement your script-based testing with exploratory testing to find problems you wouldn't find any other way. If you use Jira for any of your projects, you are lucky.

Xray seamlessly integrates with Jira and is an enterprise-proven comprehensive **test management tool** capable of handling the different types of tests and approaches.

Thanks for reading our eBook!

Links

www.getxray.app

getxray.app