Automated software testing

Understand the differences between automated and manual software testing and learn how to plan an automated testing solution for your team.

BY MAX REHKOPF

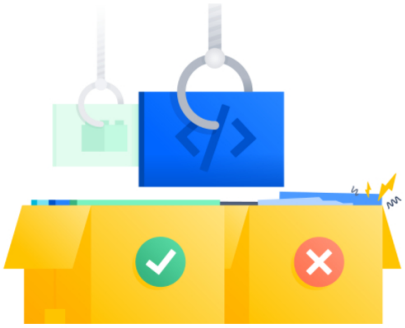
What is automated testing?

Automated testing is the application of software tools to automate a human-driven manual process of reviewing and validating a software product. Most modern agile and DevOps software projects now include automated testing from inception. To fully appreciate the value of automated testing, however, it helps to understand what life was like before it was widely adopted.

Back when manual testing was the norm, it was common practice for software companies to employ a full time QA team. This team would develop a collection of 'test plans,' or step by step checklists that assert a feature of a software project behaves as expected. The QA team would then manually execute these checklists every time a new update or change was pushed to the software project, then return the results of the test plans to the engineering team for review and any further development to address issues.

This process was slow, expensive, and error-prone. Automated testing brings huge gains for team efficiency and ROI of quality assurance teams.

Automated testing puts ownership responsibilities in the hands of the engineering team. The test plans are developed alongside regular roadmap feature development then executed automatically by software continuous integration tools. Automated testing promotes lean QA team size and enables the QA team to focus on more sensitive features.

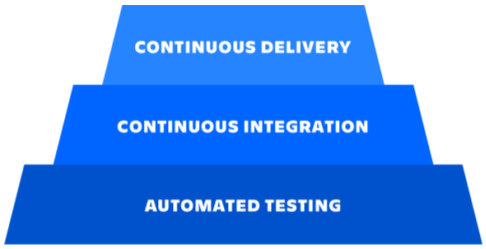


Why is testing automation important to continuous delivery?

Continuous delivery (CD) is all about delivering new code releases as fast as possible to customers. Automated testing is critical to that goal. There's no way to automate delivery to users if there is a manual, time-consuming step within the delivery process.

CD is a part of a greater deployment pipeline. CD is a successor to and also dependant on continuous integration (CI). CI is fully responsible for running automated tests against any new code changes and verifying that those changes don't break established features or introduce any new bugs. CD is triggered once the continuous integration step passes the automated test plan.

This relationship between automated testing, CI, and CD produces many benefits for a high velocity software team. Automated testing ensures quality at every stage of development by ensuring new commits do not introduce any bugs,so the software remains deployment ready at all times.



What kinds of software tests should be automated first?

End-to-End tests

Arguably the most valuable tests to implement are end to end (E2E) tests. E2E tests simulate a user level experience across the full stack of a software product. E2E tests plans generally cover user level stories like: “a user can login” “a user can make a deposit” “user can change email settings”. These tests are highly valuable to implement as they offer assurance that real users are having a smooth bug free experience, even when new commits are pushed.

E2E testing tools capture and replay user actions, so E2E test plans then become recordings of key user experience flows. If a software product is lacking any kind of automated testing coverage, it will get the most value by implementing E2E tests of the most critical business flows. E2E tests can be expensive up front to capture and record the user flow sequence. If the software product is not doing rapid daily releases it can be more economical to have a human team manually execute through the E2E test plans.

Unit tests

As the name implies, unit tests cover individual units of code. Units of code are best measured in function definitions. A unit test will cover an individual function. Unit tests will assert that expected input to a function matches expected output. Code that has sensitive calculations (as it may pertain to finance, health care, or aerospace) is best covered by unit tests. Unit tests are inexpensive and quick to implement and provide a high return on investment.

Integration tests

Often times a unit of code will make an external call to a 3rd party service. The primary codebase being tested will not have access to the code of this 3rd party utility. Integration tests deal with mocking these 3rd party dependencies and asserting the code interfacing with them behaves as expected.

Integration tests are similar to unit tests in the way they are written and in their tooling. Integration tests can be an inexpensive alternative to E2E tests however, the return on investment is debatable when combination of unit tests and E2E are already inplace.

Performance tests

When used in the context of software development ‘performance’ is used to describe the speed and responsiveness at which a software project reacts. Some examples of performance metrics are: ‘time to page load’, ‘time to first render’, ‘search results response time’. Performance tests create measurements and assertions for these example cases. Automated performance tests will run test cases across these metrics and then alert the team to any regressions or loss of speed.

What kinds of software tests should be done manually?

It is arguable that any tests that can be automated should be automated. It is a huge gain in productivity and human time cost. With that said, there are times when the ROI of developing an automated test suite is not worth it when compared to executing a manual test.

Exploratory testing

Automated tests are scripted and follow a sequence of steps to validate behavior. Exploratory testing is more random and tries unscripted sequences to find bugs or unexpected behavior. While there are software tools to establish a software exploratory testing suite, they are not fully mature and widely adopted yet. It can be much more efficient to assign a manual QA tester and use human creativity to explore how to break a software product.

Visual regression testing

A visual regression happens when a visual design flaw is introduced to software UI. This could be mispositioned UI elements, wrong font, wrong colors or more. As with exploratory testing there are tools out there to write automated tests to catch these regressions. These tools capture screenshots from various states of a software product and then use OCR to compare them to expected results. These tests are expensive to develop and the tools are not widely adopted. It can be much more effective to have a human look at something and see if there are any visual issues.

Building a test automation framework for your DevOps team

There is no all-encompassing solution for automated testing. When planning an automated testing solution for your team, there are a few key considerations to make.

Frequency of release

Software products that release on fixed intervals, such as monthly or weekly, may find manual testing is a better fit. Software products that release more rapidly will greatly benefit from automated testing since CI and CD are dependant on automated testing.

Available tools and ecosystem

Each programming language has its own ecosystem of complementary tools and utilities. Each type of automated test pattern has its own set of tools that may or may not be available in a particular programming languages ecosystem. Successful implementation of an automated testing pattern will require an intersection of the language and tool support.

Product market fit and code base maturity

If your team is working on building a new product which has not yet proven a target audience or business model, it may not make sense to invest in automated tests. Automated tests act as an insurance mechanism to restrict unexpected code regressions. If your team is moving at a high velocity it can be frustratingly expensive to have to update and maintain automated tests when the code is dramatically and rapidly changing

Make automated testing part of your CD pipeline

Automated testing is a standard modern software development practice. The best teams and companies use automated tests. CI/CD is dependant on automated tests and critical to helping the best teams ship reliable and robust software to their customers. Start exploring CI/CD solutions today.

The different types of testing in Software

Exploratory testing

Introduction to Code Coverage

What Is Continuous Deployment?

Microservices and Microservices Architecture

Bitbucket CI/CD tutorials

Continuous Delivery articles

MAX REHKOPF

As a self-proclaimed "chaos muppet" I look to agile practices and lean principles to bring order to my everyday. It's a joy of mine to share these lessons with others through the many articles, talks, and videos I make for Atlassian

TUTORIAL

ARTICLE

Integration Testing Tutorial

Learn how to run integration tests in this tutorial with Bitbucket Pipelines.

Try this tutorial →

The different types of testing in Software

Compare different types of software testing, such as unit testing, integration testing, functional testing, acceptance testing, and more!

Read this article →

CI/CD Topics

- Continuous Delivery
- Continuous Integration
- Continuous Deployment
- Pipelines
- Software Testing
- Microservices
- Tutorials

Sign up for more CI/CD articles and tutorials.

Email

email@example.com

Subscribe

Up Next
The different types of testing in Software →