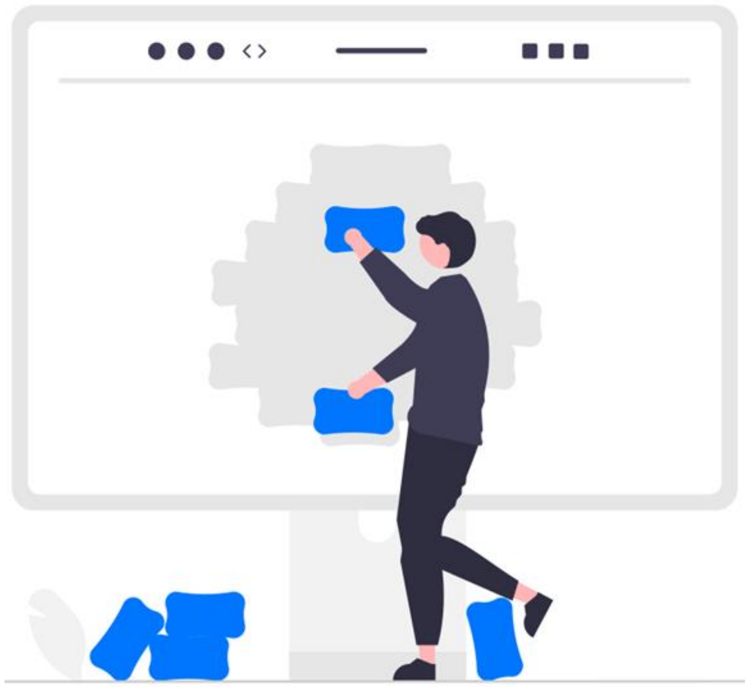
How to elevate your (OpenEuropa) contribution with automated tests

OpenEuropa Library Talk #4 – Adam Nagy

2 July 2024

Agenda

1. Introduction: Who we are
2. Introduction: Our types of session
3. Introduction: Connect with us!
4. Upcoming events
5. What are automated tests (including examples)
6. Types of automated tests
7. What makes a good automated test



Introduction: Who we are

Montana Franco
Product Owner OEL and Drupal Community of Practice Lead

Inés Mühlhofer
Drupal Community of Practice Lead

Sabina La Felice
Governance Lead

Monika Vladimirova
Events & Communication Lead

Hernani Borges de Freitas
Contribution Co-Lead

Joao Santos
DevSecOps & QA Lead

Daniel Moital
DevSecOps & QA Co-Lead

Raquel Fiahlo
Innovation Lead

Francesco Sardara
Innovation Co-Lead

Introduction: Our types of sessions

OEL Talks

Weekly Tuesday meetings to stay connected and informed in our Teams channel!

- Engaging Sessions **addressing everyone**
  - Guest Speakers
  - Trainings/Webinars
  - Lightning Sessions

Ask OEL

Weekly Friday meetings to ask all your questions on OEL in our Teams channel!

- For **developers or people with a technical profile**
  - Q&A Meetings
  - Tutorials & Demos
  - Technical Guidance

Introduction: Connect with us!

Our website

LinkedIn

On Teams

Our FAQ on Wikis

On drupal.org

Our contribution guidelines

You can also reach us via EC-OPENEUROPALIBRARY@ec.europa.eu

Upcoming events:

09/07/2024: **Introduction Drupal Starshot:** Introduction to the possibilities of Drupal Starshot– Cristina Chumillas

16/07/2024: **OEL Design System:** Introduction of the OEL Desing System – Kalte Lopez Gomez

Summer break

Overview

About me

- Joined in 2016
- Back-end developer
- EWPP Core team
- Drupal contributor
- Drummer

What participants will learn

- What are automated tests
- Why is it important
- Costs involved
- Types of automated tests
- Pros and cons
- Being productive with tests

Target audience

- Project managers
- Product owners
- Scrum masters
- **Developers**

What are automated tests?

Software testing technique that automates the process of validating the functionality of the software and ensures it meets requirements before being released into production.

No test is inherently useful just because it exists.

A test becomes useful when it fulfils its purpose.

You write tests to validate the intention behind the system.

Automated tests are developer-oriented documentation of the system and its design.

Yes, but...

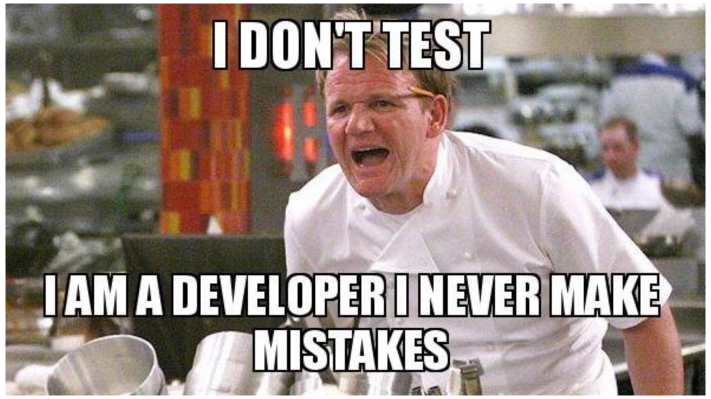
How can you know that ALL your code works if you don't test it every time you make a change?

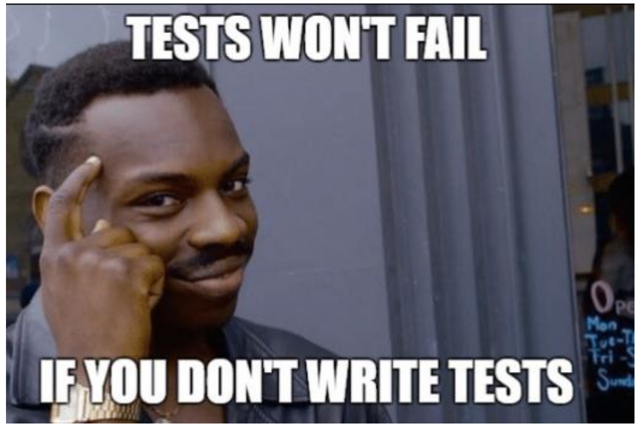
How can you test it every time you make a change if you don't have automated tests with very high coverage?

Developers spend 620 million hours a year debugging software failures, which cost companies $61 billion annually.

They spend an average of 13 hours to fix a single software failure.

According to a study published by Cambridge MBA students in 2020. (https://www.prnewswire.com/news-releases/study-software-failures-cost-the-enterprise-software-market-61b-annually-301066579.html)





Why? Because...

It proves that your code works, and you can be *nearly* certain it doesn't break anything.

Bug fix without a test is just a workaround.

Improves system reliability and overall quality.

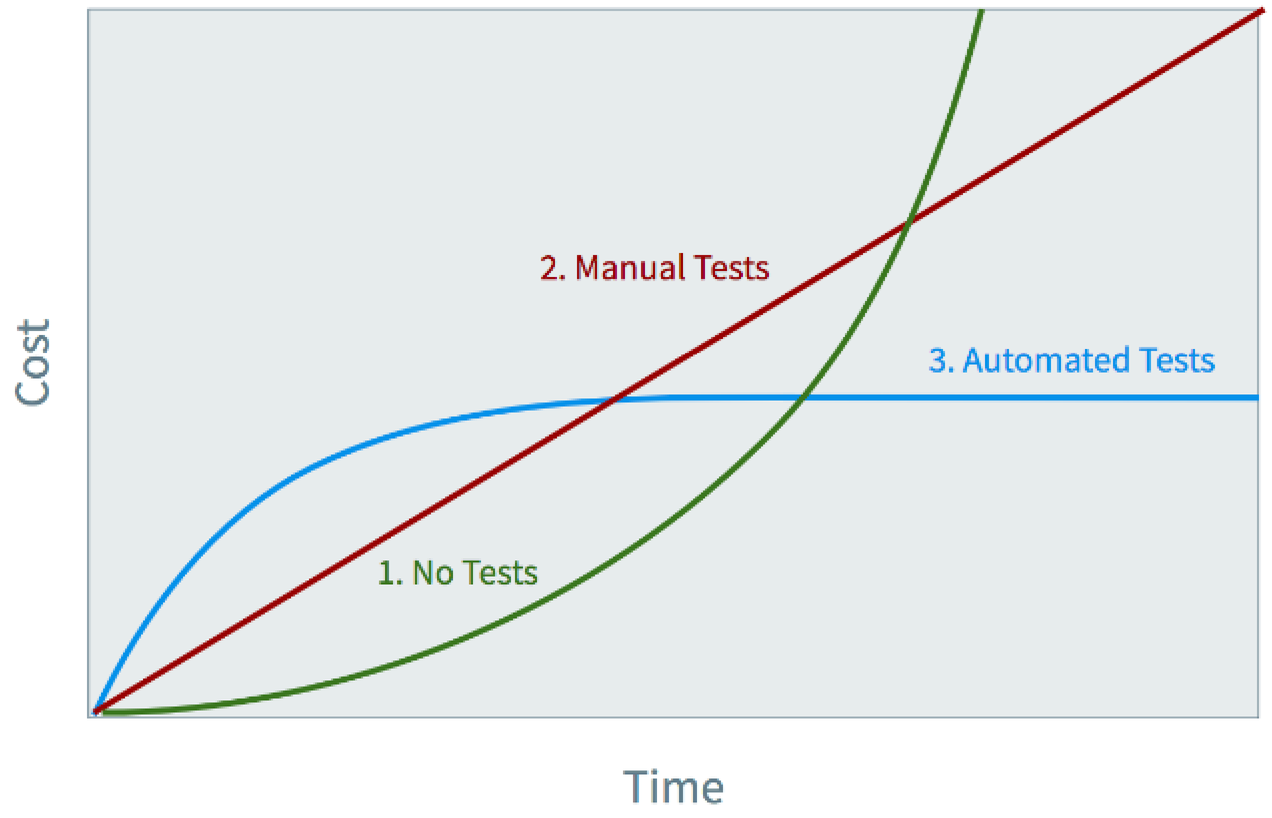
Speeds up the implementation of new features, faster Time to Market (TTM).

More frequent releases results in high return of investment (ROI).

Lose all fear of making changes.

Keeps your production code flexible.

The Cost of Testing Over Time



From https://www.karllhughes.com/posts/testing-matters

Real life examples

Aggregator module - used by over 5000 sites
Release 2.1.0 - Update path overrode the homepage path, making aggregator views as homepage. We detected early on thanks to over-night builds. It was fixed the same day.

Entity Clone module - used by over 35,000 sites
2.0.0-beta5 released a major bug creating multiple orphaned entities and PHP fatal errors. Quick response and bug report is essential for the community.

RoleAssign module - used by over 18,000 sites
Release 2.0.1: Undefined array index "base_form_id" errors.

Real life examples - on the positive side

MaxLength module
2.0.x - minimally tested, fixes and features came very slow.
2.1.x - broad testing was implemented, and support for both CKEditor 4-5 versions.
3.0.x - now the module is used by over 46,000 sites and the bug list minimal.

SpamSpan module
2.x - almost no tests and used by over 14,000 sites.
3.x - broad test coverage introduced, and 19 merge requests with tests were done within weeks. By doing this, we identified critical bugs and made the module stable of the community.

These are good examples on how to start testing at a later stage in a SDLFC.
Identify the most critical functionalities and start from there.

Types of automated tests

The two main testing types are:

- **Functional automated testing** – these tests verify whether an application's features match the expectations outlined in the requirements. An example of functional testing is checking whether you can log in and log out.

- **Non-functional testing** – this focuses on testing all requirements aside from business applications and spans accessibility, performance, security, and usability testing. An example of a non-functional test is checking whether an application can handle high traffic.

Types of automated tests

Performance tests – these verify the overall performance of a system by checking response times and stability during the load.

Security tests – here, the application is checked for security issues and vulnerabilities that could be exploited by an attacker, leading to data loss or leaks.

Accessibility tests – these verify if applications can be accessed by people with different demands – including individuals with vision, hearing, or other impairments – and those using aids like screen readers.

Types of automated tests

Usability tests – these focus on testing whether a system is easy to use and intuitive. Usability tests are performed by real users or take place under conditions matching real use; insights and feedback are collected.

UI tests – these focus on the graphical interface users interact with, including testing on different device types and resolutions.

Smoke tests – here, it’s about checking critical functionalities and paths to make sure the build is stable and further testing can take place.

Regression tests – these involve running functional and non-functional tests again, ensuring no new issues were introduced.

Different testing levels

Static – linters, formatters, and type checkers are employed to catch typos and coding standards.

Unit – focuses on testing the smallest isolated piece of code.

Integration – here, independent modules or components are tested together to make sure they work as expected.

Functional – focuses on testing API communication, ensuring requests are properly handled and responses meet expectations.

End-to-end – verifies multiple components at once by interacting with the GUI and testing specified user stories.

Types of automated tests in Drupal

Most testing is done under the **PHPUnit** framework.

Unit - tests with minimal dependencies, bare PHP, it is very fast.
Example: spamspan – TwigExtensionUnitTest.php

Kernel - tests with a bootstrapped kernel, and a minimal number of extensions enabled.
Example: oe_content- TimelineFieldTest.php

Functional - tests with a fully booted Drupal instance. The installation takes time.
Example: oe_media - DocumentMediaTest.php

FunctionalJavascript - tests that use Webdriver to perform tests of Javascript and Ajax functionality in the browser thanks to Mink framework with Selenium.
Example: oe_media - MediaCreationFormWidget.php

Types of automated tests in Drupal

End to End tests - BEHAT

Behat is an open-source PHP framework that is used to automate testing by leveraging Behaviour Driven Development (BDD). A BDD is focusing on continuous communication and simple text stories. With BDD, the tester builds test cases in human-readable language. The language used by the Behat tool is Gherkin, which is a business readable and domain-specific language.

Example: oe_media – image.feature

Types of automated tests in Drupal

End to End tests - Existing Site Test

Sometimes you need to test a whole installation profile with complex interdependencies, and you need to have certain configurations in place or third-party libraries in place like on a real site. Drupal Testing Traits allows writing tests for an already installed site. Same as the functional tests in Drupal.

In EWPP we are using a combination of Existing Site tests with BEHAT.

Since BEHAT is not checked by business we stopped writing new ones.

In EWPP from 185 586 lines of code, 113 546 lines are only tests.

Common pitfalls

Test code is as important as the production code.

Unmaintainable tests will cripple the development cycle.

Separation of development and testing process is wrong.

Not prioritizing continuous integration.

Over-reliance on End-to-End testing.

What makes a good test?

Three things:

1. Readability
2. Readability
3. Readability

Perhaps it is even more important in tests than in production code.

Say a lot in a few expressions.

QA should find nothing!

European Commission

Q&A

OEL Talks