

45426: Teste e Qualidade de Software

CI cycle

Sérgio Freire

v2025-05-06

Learning objectives

DevOps in a nutshell

Explain the core concepts of Continuous Integration practice

See CI in action

Learn how CI can be integrated with a test management tool

How to analyze the results and its impacts in Jira

BONUS: How to overcome some testing challenges even with CI

The cost of bugs

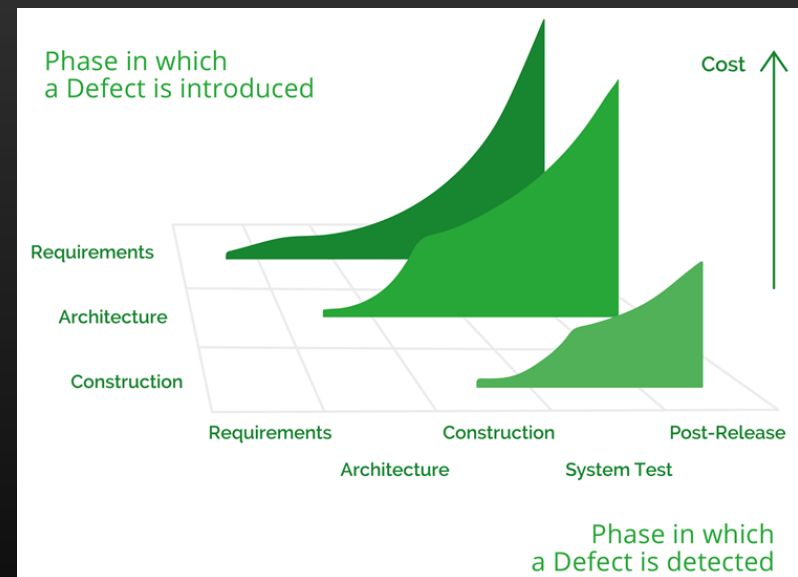
The cost of fixing bugs grows exponentially depending on the phase in which they are detected.

The fix cost also depends on the stage where bugs are initially introduced.

⇒ **Test early**

⇒ **Test continuously**

⇒ **Avoid technical & testing debt**



(adapted from) Code Complete, Steve McConnell, 2004

Speed vs Quality

Are they real enemies?

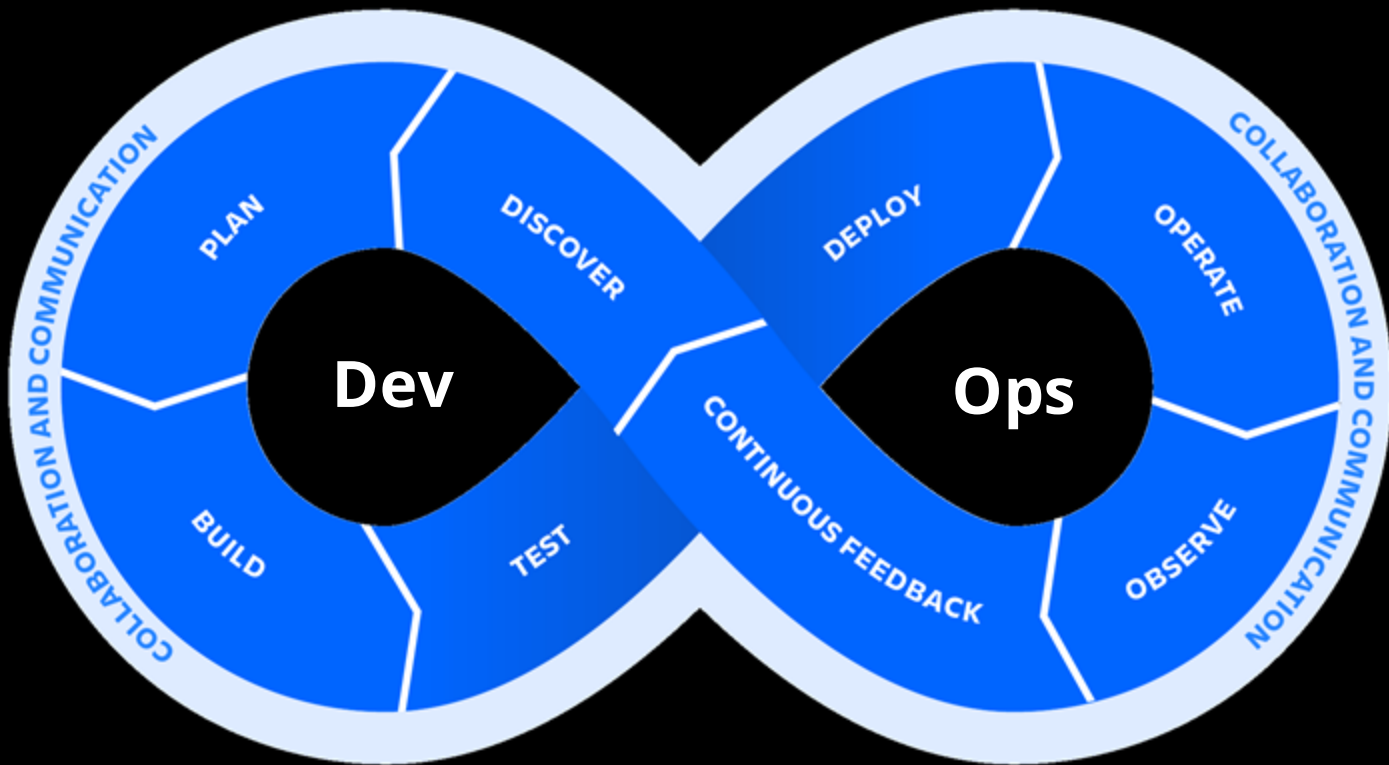


These can help you out:

- ✓ Agile
- ✓ DevOps
- ✓ Continuous Integration
- ✓ Automated Testing
- ✓ Continuous Testing
- ✓ Continuous Delivery

Release often, with **confidence**. Motivates the teams and makes customers happy.

DevOps



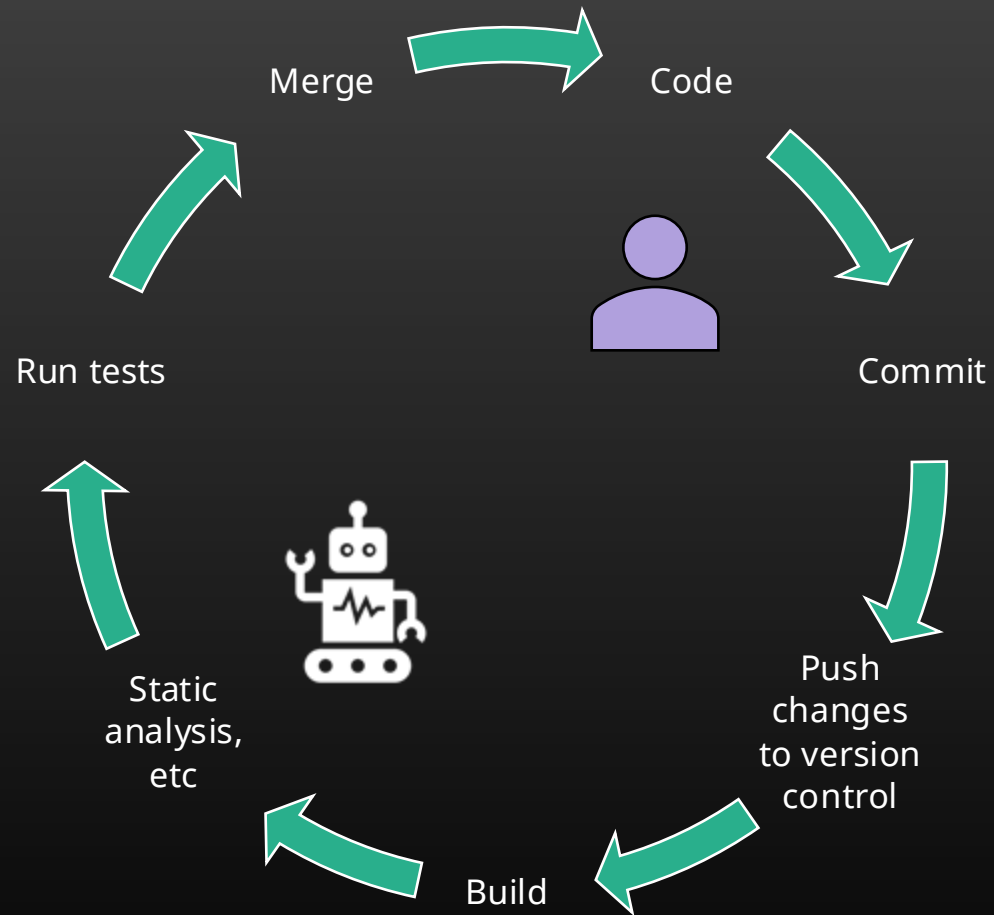
Continuous Integration

Continuous Integration is a software development practice where each member of a team merges their changes into a codebase together with their colleagues changes at least daily. Each of these integrations is verified by an automated build (including test) to detect integration errors as quickly as possible.

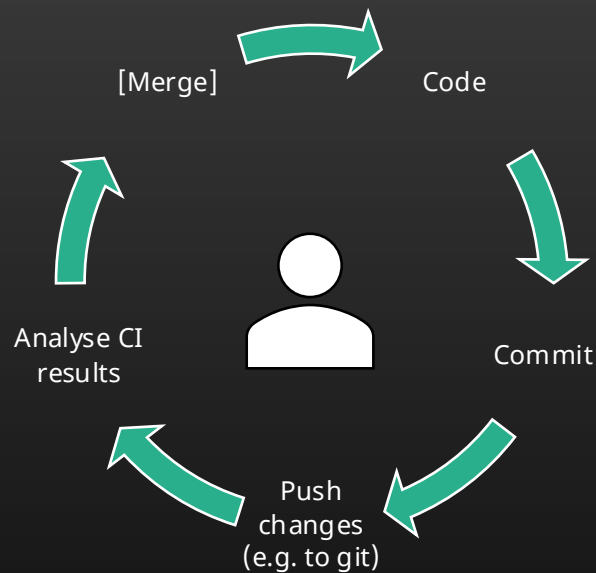
Teams find that this approach reduces the risk of delivery delays, reduces the effort of integration, and enables practices that foster a healthy codebase for rapid enhancement with new features.

Martin Fowler

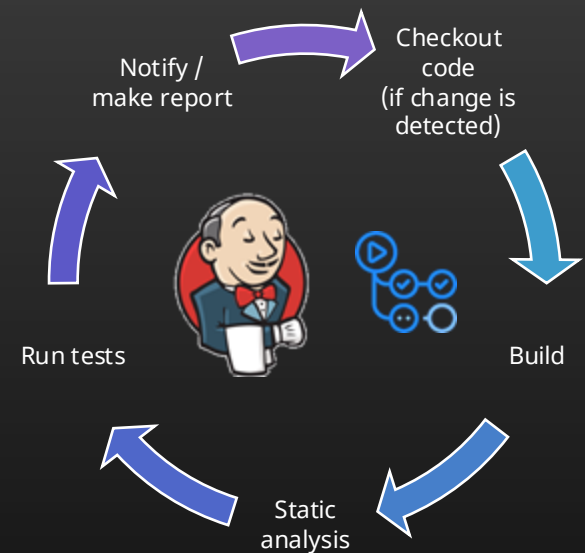
Continuous Integration



Continuous Integration



Developer



CI tool

(e.g., Jenkins, GH actions, Bitbucket pipelines, TeamCity, Gitlab)

Where do we integrate to and how?

- On "main"/"master" branch directly
 - Trunk-based development
- Using other branches, that we then merge if "all" is OK
 - GitHub or Git flows
- NOTE: in this case, we need to have short-lived branches!



Andrea L. • Following
Principal Software Engineer

19h (edited) ...

If you want to achieve CD, you need CI (Continuous Integration) and a prerequisite for CI is trunk-based development. Trunk-based development means that you are integrating your code into trunk multiple times a day and generating release candidates very often. That means that you need a solid testing infrastructure.



Build broken: who should fix it?

Usually, we follow the “You broke it, you fix it” rule.

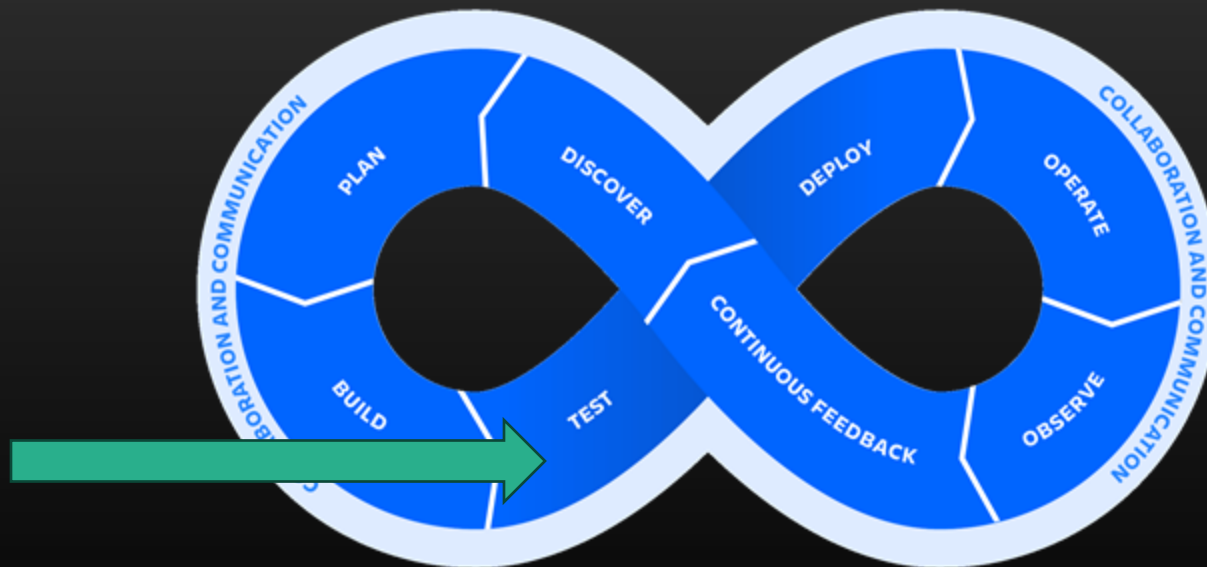
However, we should work as a team and improve quality together.



Continuous Integration

Ok... we need to integrate often and continuously, and it involves test automation; got it!

But is testing just like a “phase” in the DevOps infinity loop??



When (and how) do we test in DevOps?

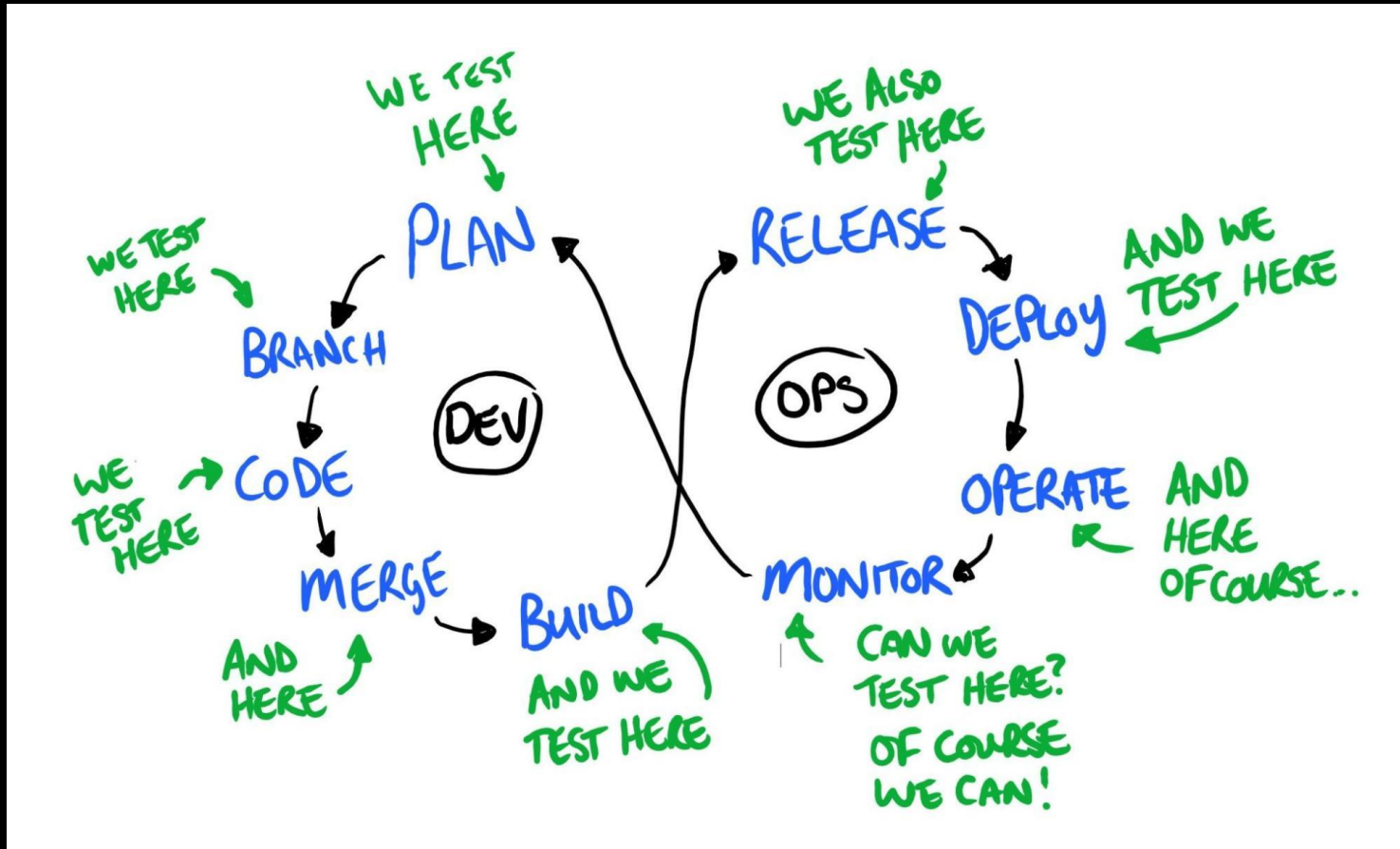
Whenever we implement the feature?

Before?

After?!

... and using just Test Automation during CI?

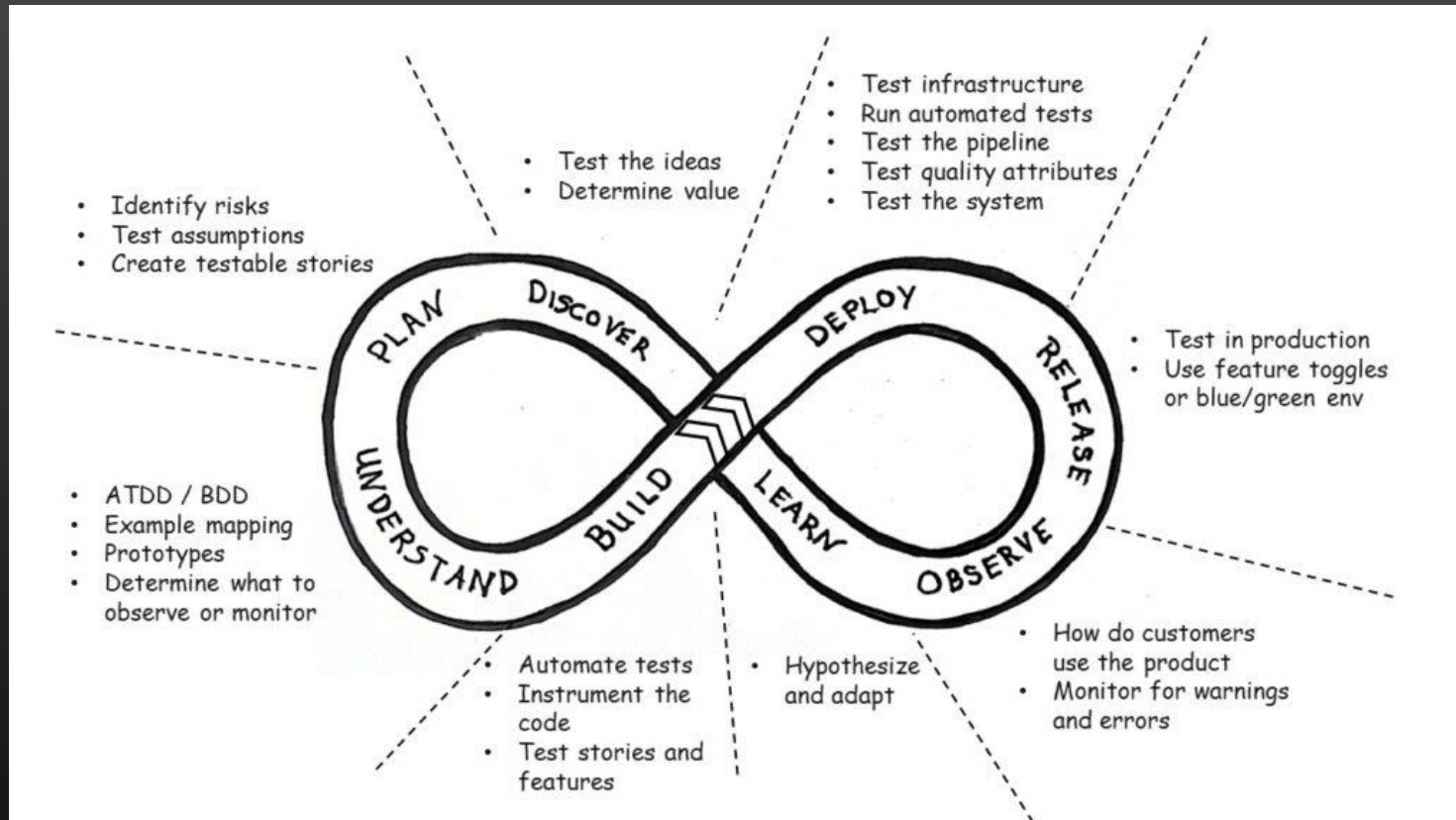
When do we test in DevOps?



Continuous Testing in DevOps model by Dan Ashby

<https://www.linkedin.com/pulse/continuous-testing-devops-dan-ashby/>

Holistic Testing (Agile Testing)



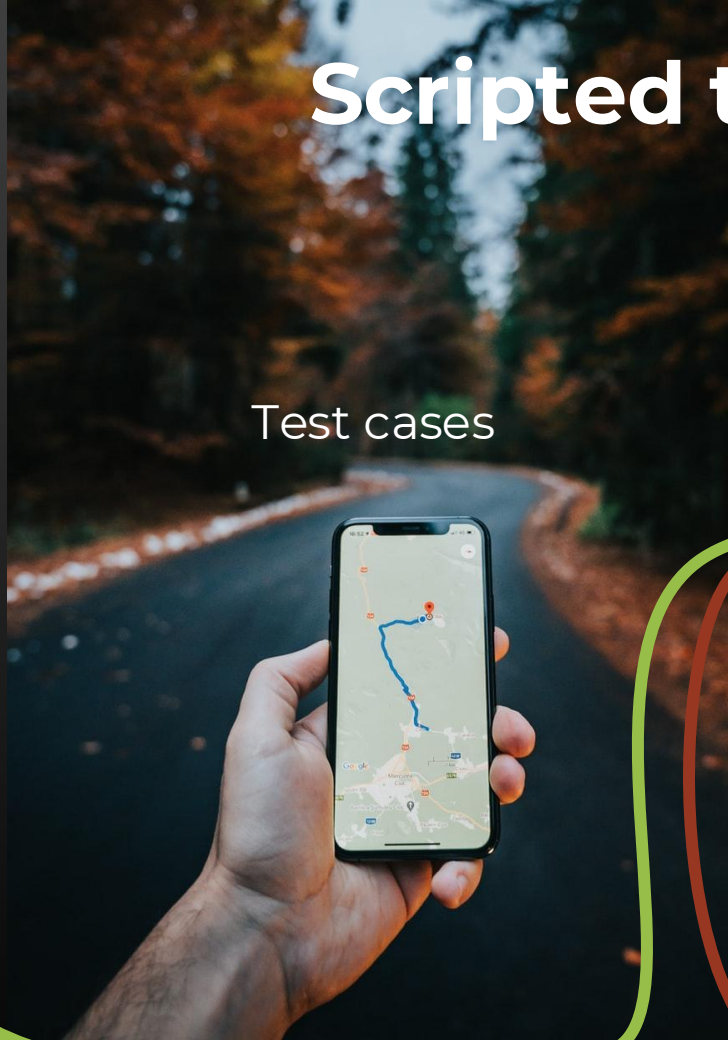
When we test, we need to consider all types of testing, not only the ones we think a tester is responsible for. It includes automation, exploratory testing, or any other type of human-centric testing. It involves the whole team, the product organization, and even the customer. We need to consider testing from a holistic point of view. - Janet Gregory

<https://janetgregory.ca/testing-from-a-holistic-point-of-view/>

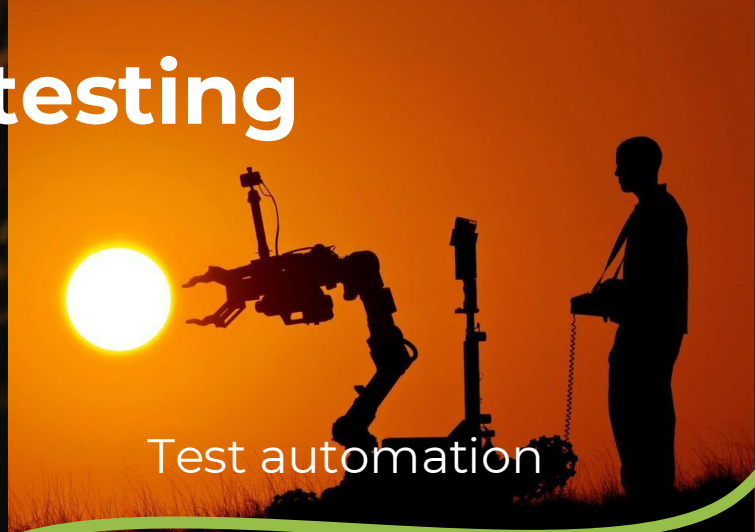
Testing approaches?

Scripted testing

Test cases



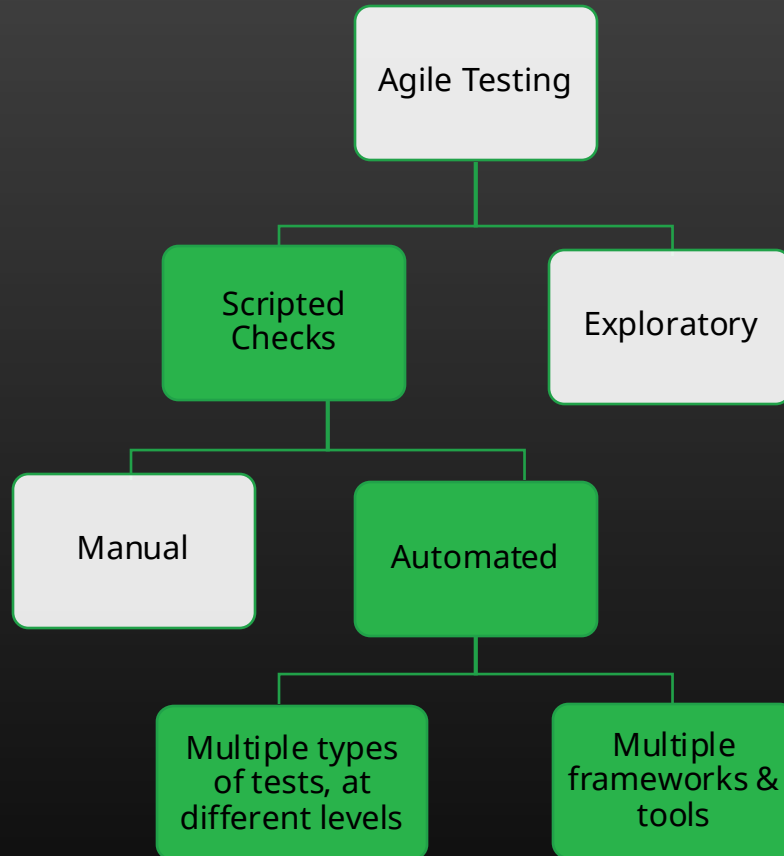
Test automation



Exploratory testing



Agile Testing and the testing approaches



Essential for..

- Avoiding regressions
- CI/CD



Each team “cooks” software differently

- With DevOps teams use automation and a bunch of tools
- Teams demand flexibility

**How to make
sure we don't
miss
information?**



Flexibility is key when adopting more tools

Source: Atlassian: State of the Developer Report 2022

How do we integrate all this?

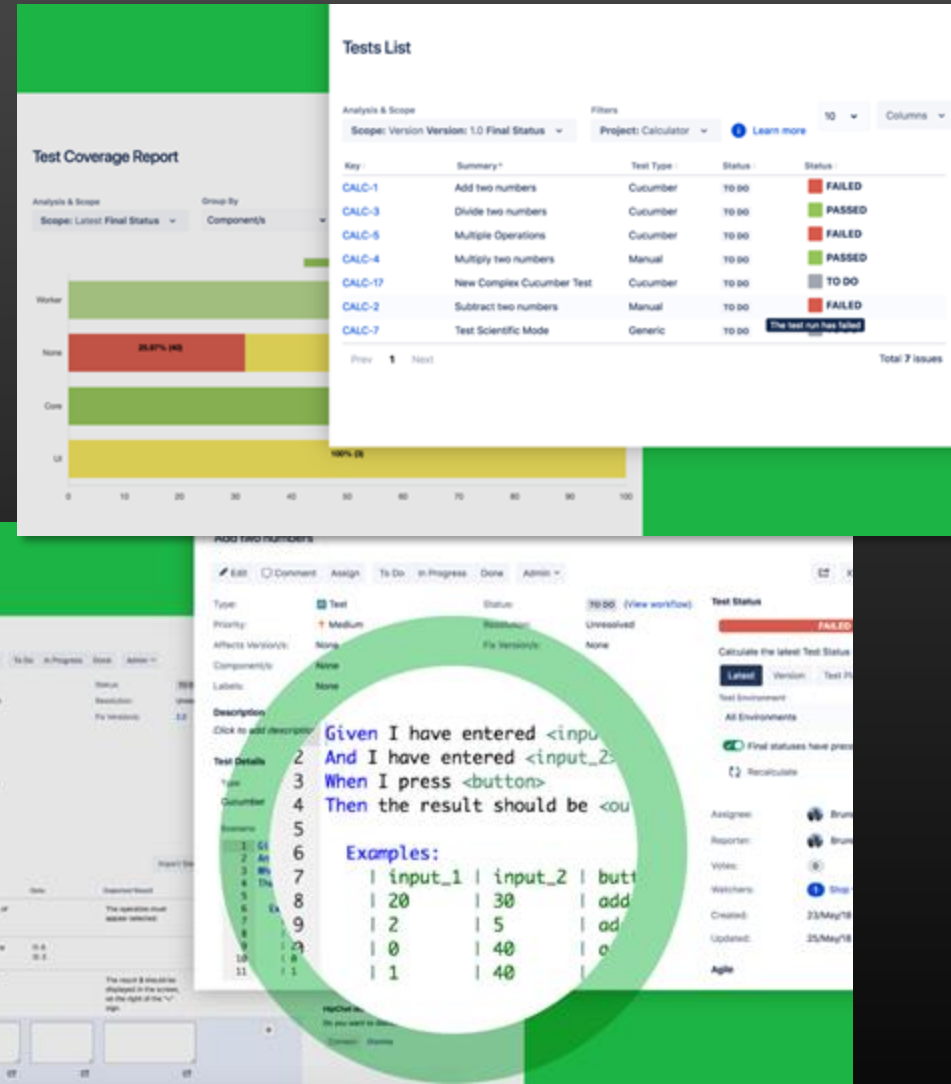
How to have a unified overview of testing?

- Different approaches with different adoption levels
- Different frameworks & tools

How to track its impacts on the deliverables managed on the issue tracker being used?

What is Xray?

- Test Management App for Jira
- Scripted and Exploratory Testing
- Organize, Plan & Execute Tests
- Report bugs
- Test Coverage visibility
- Complete traceability
- Automation friendly



Single-source of truth, for all team members.

Xray: "enhanced", consolidated testing in Jira

- Multiple test approaches
- BDD
- Data-driven testing

Test details

Test details Preconditions Test Sets Test Plans

Test Type

Cucumber

Scenario

- 1 Given I'm logged in on the website
- 2 When I navigate to the personal information page
- 3 And I update my details
- 4 Then I receive a notification that my account is updated

Dataset for Test BOOK-138

There are a total of 24 iterations to execute.

Create parameter Import

Combinatorial parameters

GRN Quantity

Yes 1

No 2

Select... 5

10

#	Item	Price	Rating	Author	In stock	Condition	Format
1	In Search of Lost Time	\$34	5	Marcel Proust	Yes	New	Paperback
2	One Hundred Years of Sol...	\$20	4.9	Gabriel Garcia Marquez	Yes	Used	Paperback
3	The Great Gatsby	\$39	4.7	F. Scott Fitzgerald	No	New	Kindle

New

Xray: trace the impacts and the source

Story/requirements ⇔ Tests ⇔ Runs ⇔ Bugs

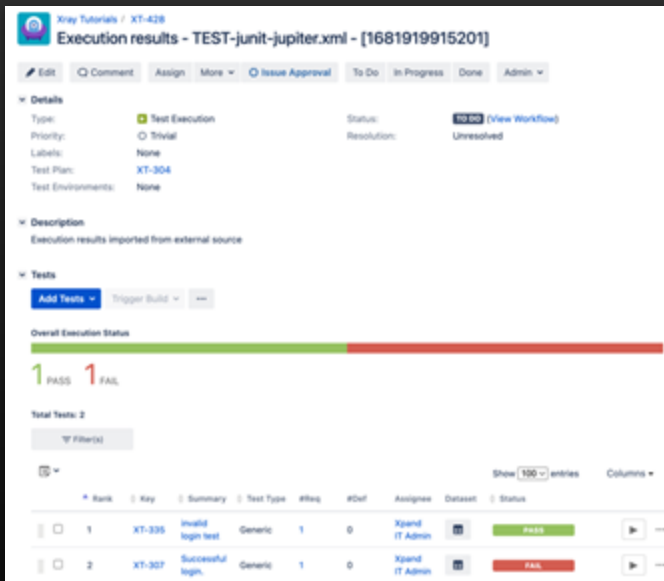
The screenshot displays the Xray Traceability Report interface. At the top, there's a header 'Traceability Report' with a help icon. Below it, filters for 'Scope: Version Version: v1.0 Final Status' and 'Project: Calculator Issue Type: Epic, Story Labels: MOT' are visible, along with a 'Show Test Runs' toggle. A 'QUICK FILTERS' section shows counts for OK (1), NOK (1), NOTRUN (0), UNKNOWN (0), and UNCOVERED (0). The main content area is divided into four tabs: REQUIREMENTS, TESTS, TESTRUNS, and DEFECTS. The REQUIREMENTS tab is active, showing a list of requirements. A red arrow labeled 'impacts' points from the right towards the requirements. An orange arrow labeled 'source (?)' points from the bottom left towards the right. The requirements list includes:

- CALC-1120 (7988): Fix Version/v: v1.0. As a user, I can perform restricted operations in my ... (NOK)
- CALC-2657 (7989): As a user, I can manage my profile details (UNCOVERED)
- CALC-795 (7989): Fix Version/v: v1.0. As a user, I can login the web application (NOK)
- CALC-797 (7988): Valid Login using Gherkin style (PASSED)
- CALC-1449 (7988): Fix Version/v: v1.0. Finished On: 5May22 03:15 PM. Executed By: Sérgio Freire. Test Environment/v: (PASSED)
- CALC-1428 (7988): Fix Version/v: v1.0. Finished On: 28Apr22 03:42 PM. Executed By: Sérgio Freire. Test Environment/v: (PASSED)
- CALC-1427 (7988): Fix Version/v: v1.0. Finished On: 27Apr22 04:02 PM. Executed By: Sérgio Freire. Test Environment/v: (PASSED)
- CALC-804 (7988): usability test (FAILED)
- CALC-2660 (7988): Fix Version/v: v1.0. Finished On: 25Jun22 03:19 PM. Executed By: Sérgio Freire. Test Environment/v: (FAILED)
- CALC-2661 (7990): login button too small (FAILED)

Xray: integration with test automation

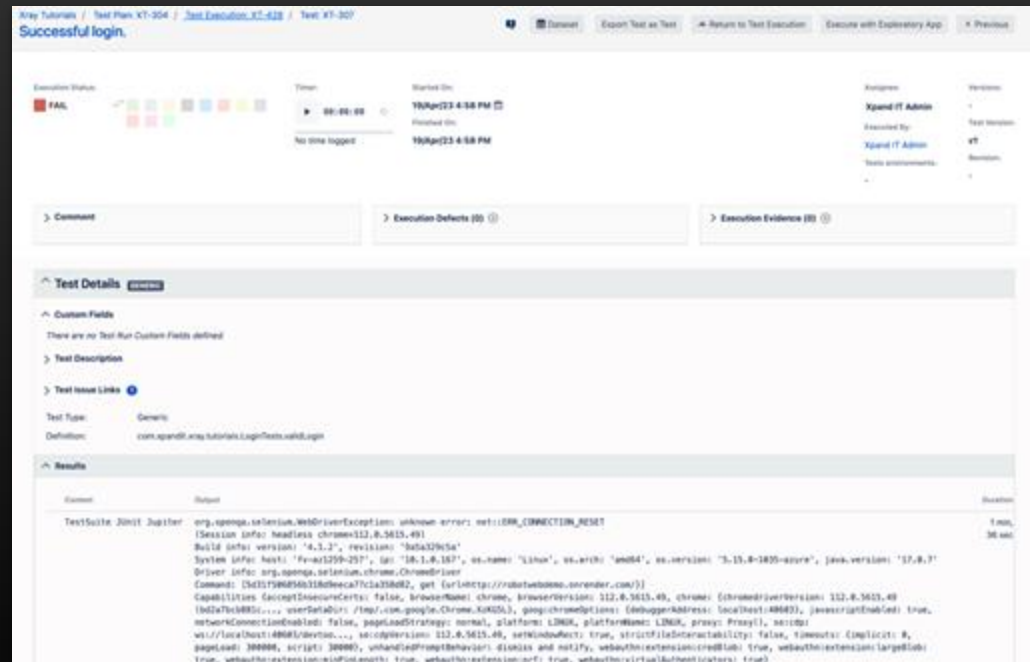
1. Track (have visibility of) test automation results in Jira, using Xray

Alongside other testing effort and available to all team members.



The screenshot shows the Xray Test Execution results for a test plan named "TEST-junit-jupiter.xml" with ID [1681919915201]. The interface includes tabs for "Details", "Description", and "Tests". The "Details" tab is active, showing the test execution status as "100% (View Workflow)". The "Description" tab shows "Execution results imported from external source". The "Tests" tab shows a summary of test results: "Overall Execution Status" with a green bar for "1 PASS" and a red bar for "1 FAIL". Below this, a table lists the test results:

Rank	Key	Summary	Test Type	#Req	#Def	Assignee	Status
1	XT-335	Invalid login test	Generic	1	0	Xpand IT Admin	PASS
2	XT-307	Successful login	Generic	1	0	Xpand IT Admin	FAIL



The screenshot shows the Xray Test Details for a test case named "Successful login" with ID XT-307. The interface includes tabs for "Details", "Description", and "Results". The "Details" tab is active, showing the test execution status as "FAIL". The "Description" tab shows "Execution results imported from external source". The "Results" tab shows a summary of test results: "Overall Execution Status" with a green bar for "1 PASS" and a red bar for "1 FAIL". Below this, a table lists the test results:

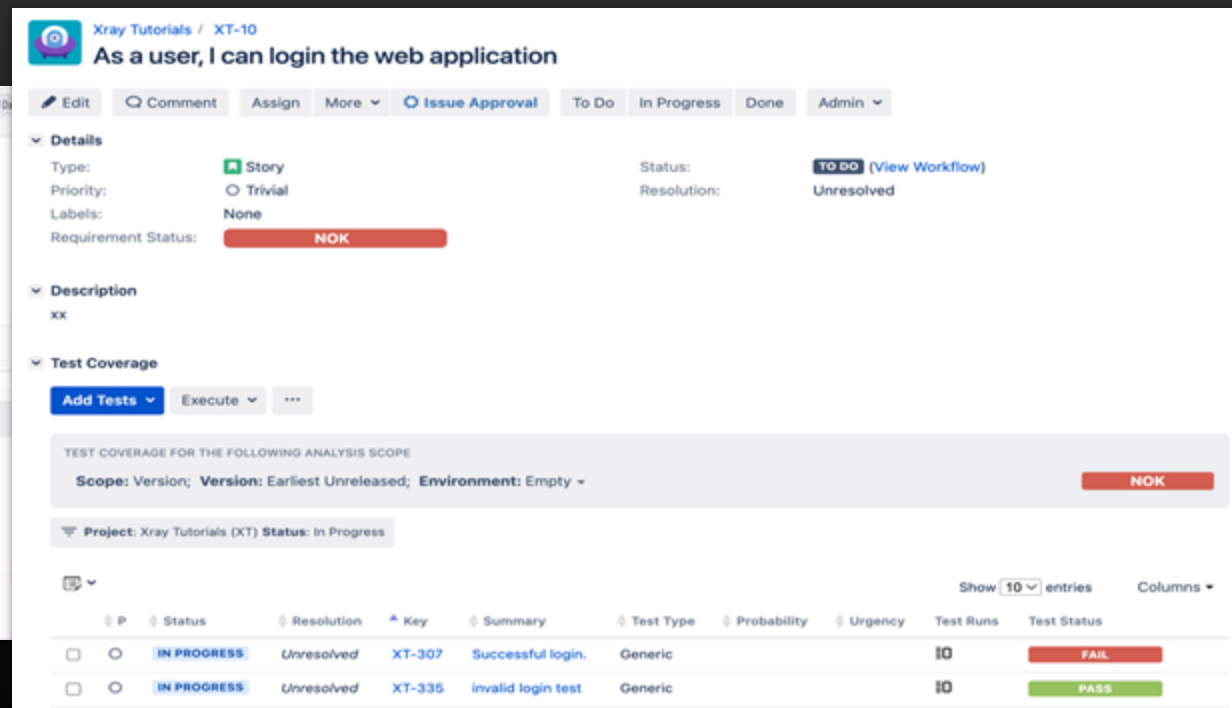
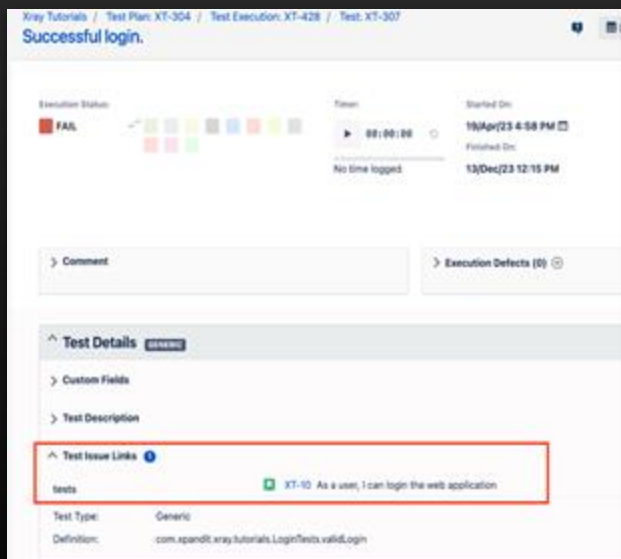
Rank	Key	Summary	Test Type	#Req	#Def	Assignee	Status
1	XT-335	Invalid login test	Generic	1	0	Xpand IT Admin	PASS
2	XT-307	Successful login	Generic	1	0	Xpand IT Admin	FAIL

Xray: integration with test automation

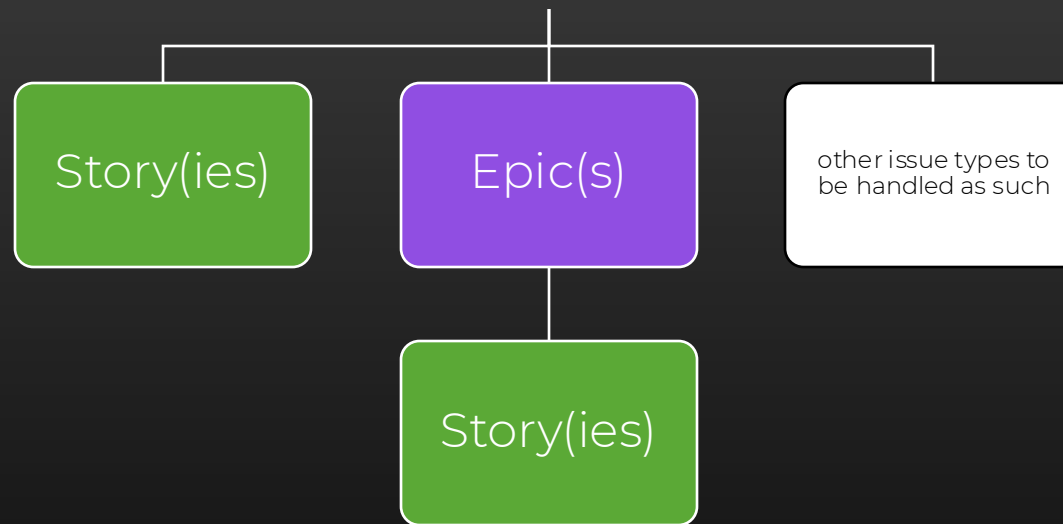
2. Track coverage related with test automation

a. What “requirements” are covered by automated tests

b. What is the status of those “requirements” given the test automation results



Xray: issues coverable with Tests



DEMO 1: Spring Tutorial

Unit and integration tests.
We'll see CI in action:

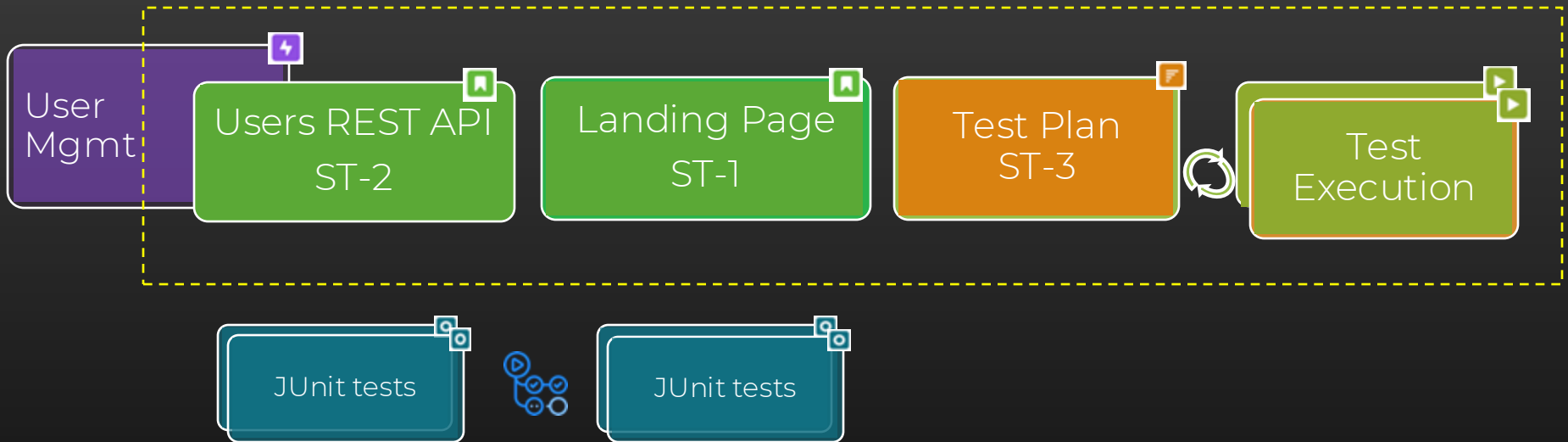
- Fix a bug
- Trigger build including tests
 - Multiple Java versions
- See feedback on PR
- Track results in Jira using Xray



<https://github.com/bitcoder/tutorial-spring>

Project: Spring Tutorial (ST)

Sprint 1



DEMO 2: A real-life example with “Xray Maven Plugin”

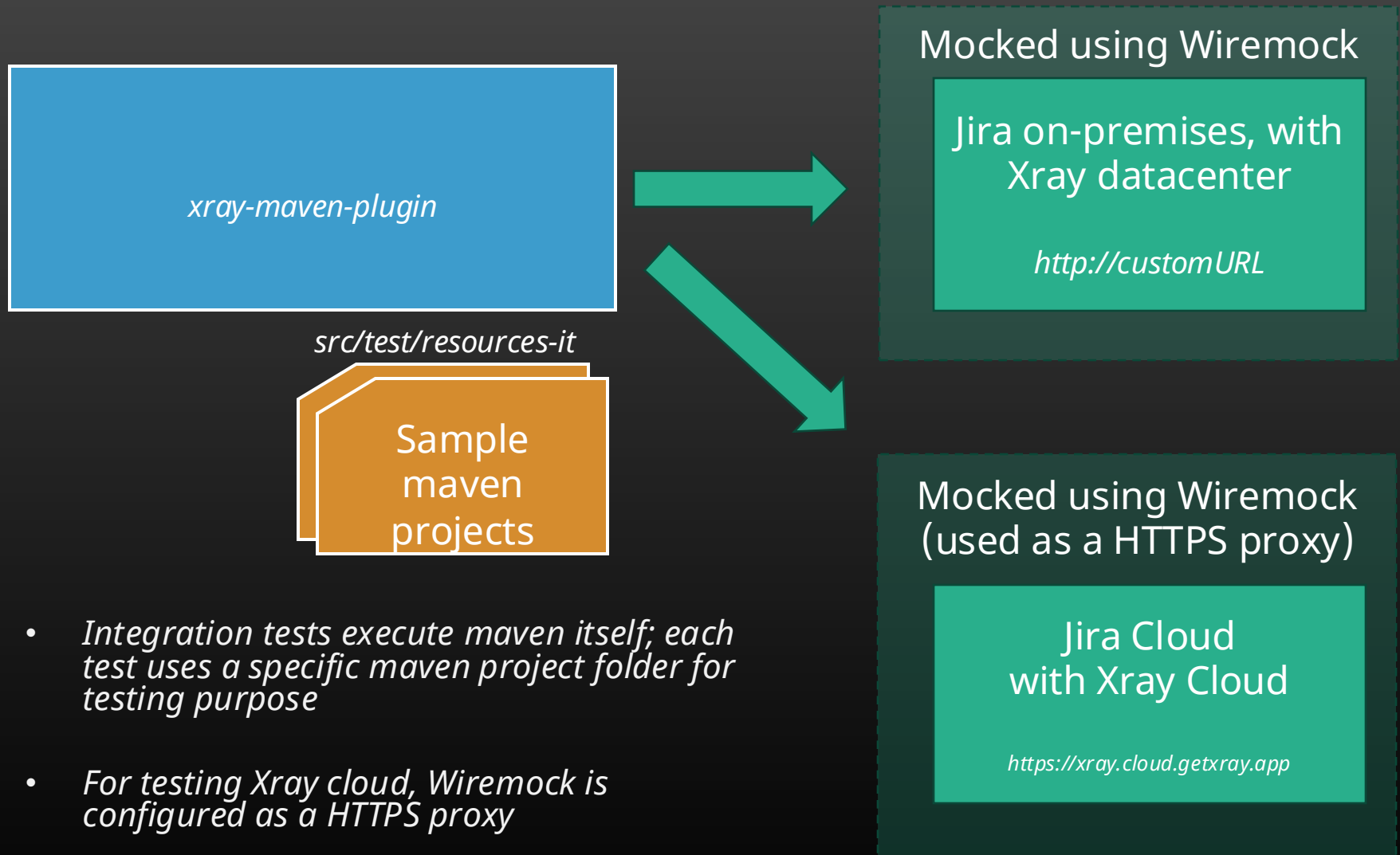
A maven plugin to upload test automation reports to Xray (on Jira). It's in essence a client library for Xray's REST API, with some additional logic.

What we'll see:

- Info on the PR
- Merge
- Make a release with (some level of) confidence

<https://github.com/Xray-App/xray-maven-plugin>

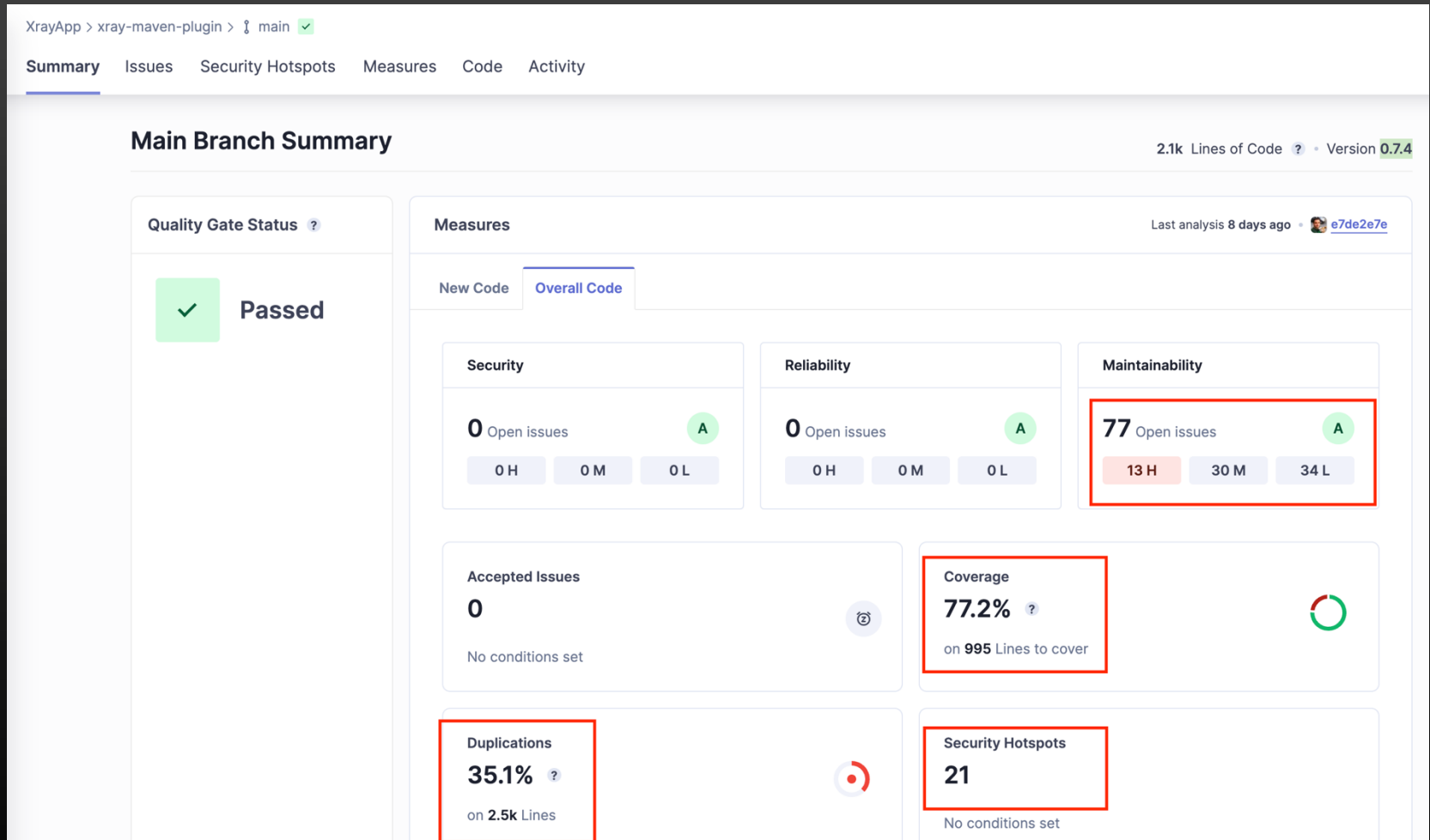
Xray Maven Plugin: architecture



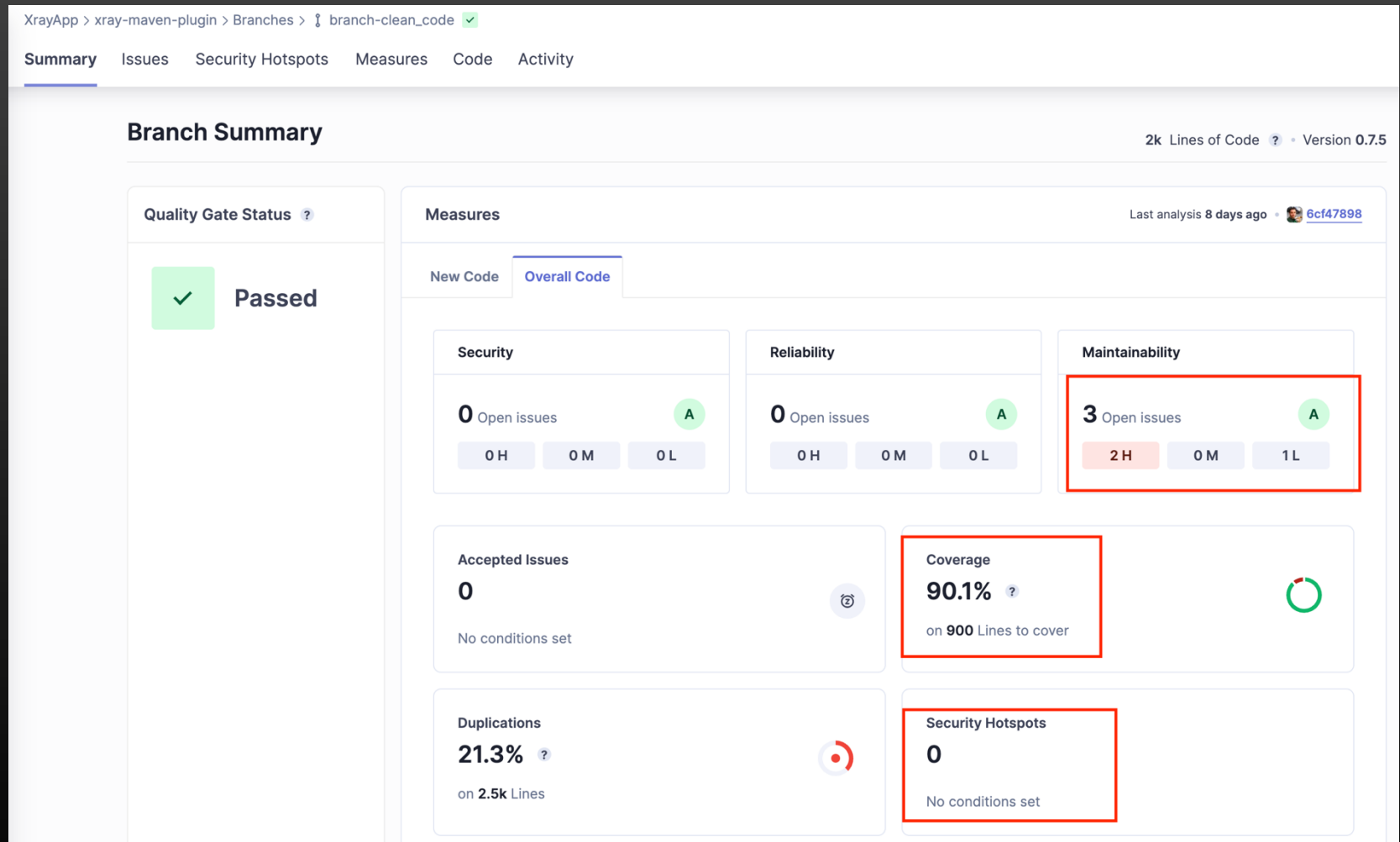
Testing challenges

- Java 8, 11, 17, 21
- 2 target apps in fact
- Maven “inside” maven
- Wiremock for integration tests

Before refactoring



After refactoring and adding more tests



Late refactoring is achievable but it's harder.

Bonus: Is Test Automation enough?

Can we rely only on the existing test automation set during CI? What may we be missing?

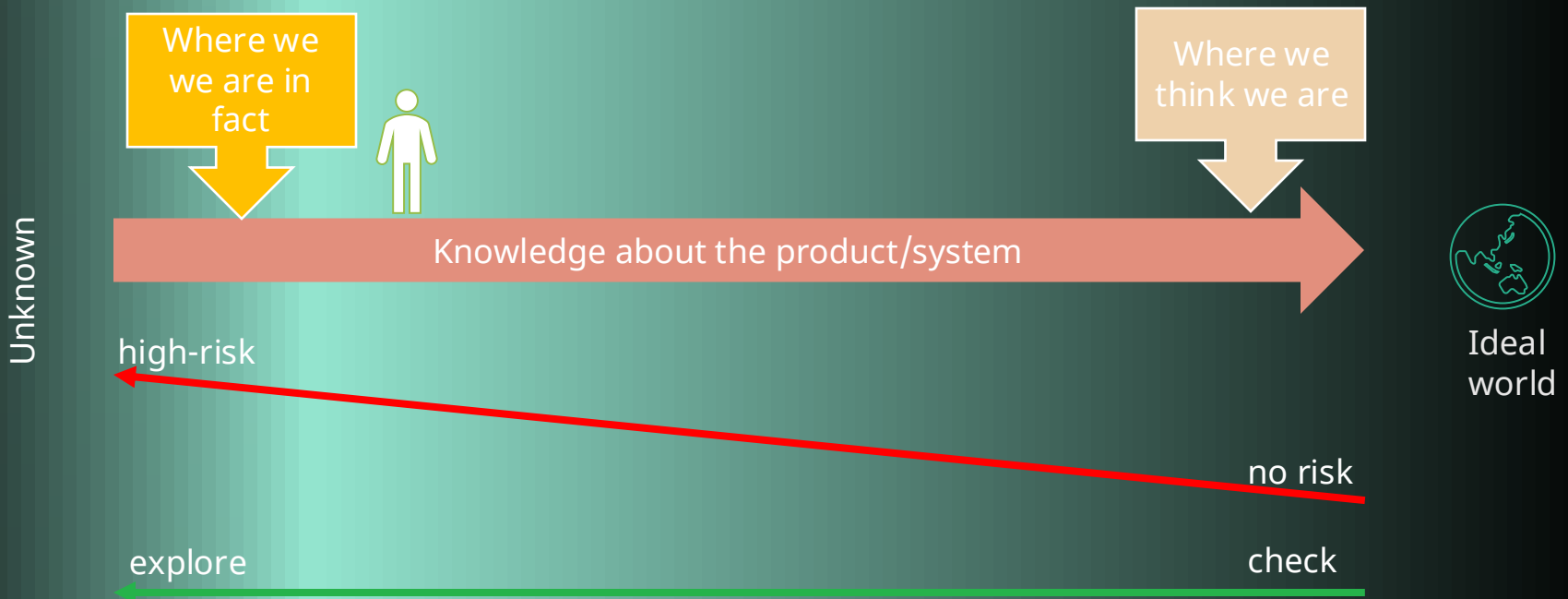
Test automation is a safety net



Image by Ben Hershey, Unsplash

- Give us some level of confidence
- The more and the thinner, the better
- It can break and also have bugs
- There will always be gaps
- Is the net targeting what matters?! (eg., wind, liquids, bear)
- Focused on the known-knowns

Exploratory testing to uncover unknown risks



Adapted from Callum Akehurst-Ryan

Space Shuttle Columbia (2001)



Photo by Eric Long

Endeavour (2011)



More than 1,000 toggles!

Sources for complexity (just a few!)

Many...

- Input parameters
- Values for these parameters
- Configurations and/or environments



A lot of scenarios to test!

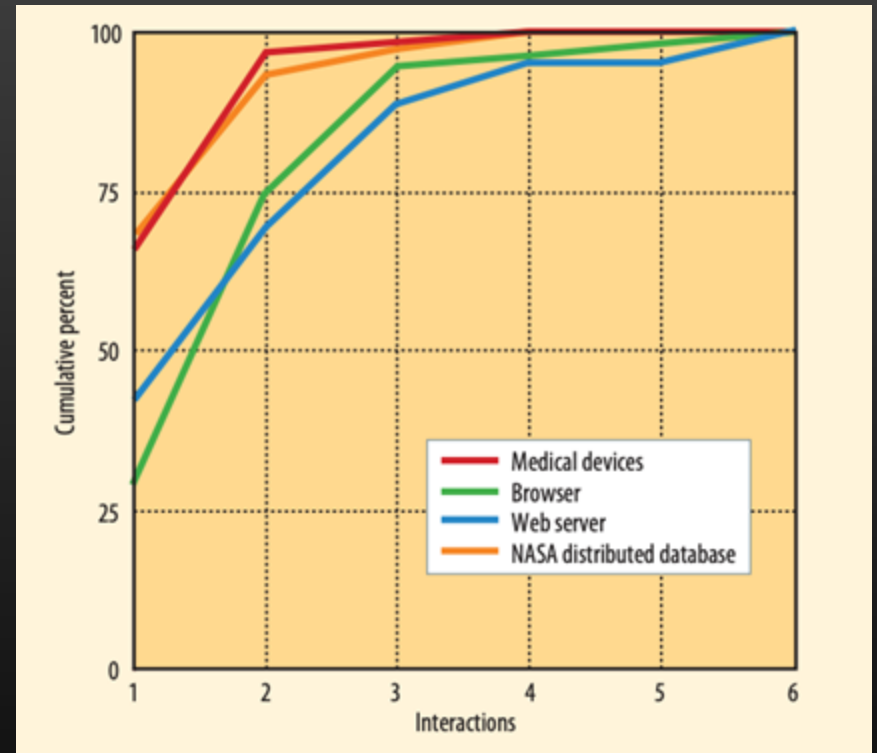
But can we test this?!

Even with test automation, it can simply be impossible (e.g., time, cost, resources).

Findings from the field

Most failures are induced by single factor faults or by the joint combinatorial effect (interaction) of two factors, with progressively fewer failures induced by interactions between three or more factors.

- NIST studies beginning in 1999 showed that across a variety of domains, all failures could be triggered by a maximum of 4-way to 6-way interactions
- Pairwise testing finds about 50% to 90% of flaws but it may not be enough



PRACTICAL COMBINATORIAL TESTING D. Richard Kuhn, Raghu N. Kacker, Yu Lei

Why does it happen?

1-way interaction fault

(pressure=60)

```
if (pressure > 50) {  
    // buggy code  
} else {  
    // ...  
}
```

2-way interaction fault

(pressure=1, volume=500)

```
if (pressure < 10) {  
    if (volume > 300) {  
        // buggy code  
    } else {  
        // ...  
    }  
} else {  
    // ...  
}
```

3-way interaction fault

(pressure=1, volume=500, admin=true)

```
if (pressure < 10) {  
    if ((volume > 300) && (admin == true)) {  
        // buggy code  
    } else {  
        // ...  
    }  
} else {  
    // ...  
}
```


What can we do then? Optimize!

Finding more bugs, with less effort, without forgetting what matters.

- Reduce number of tests
- Exclude scenarios/combinations
- Ensure most error-prone combinations are tested
- Ensure critical/important scenarios are tested
- Reduce number of necessary environments (software and/or hardware related)

Sneak Peak



Can you spot the bug?

...

```
loanAmount = eval(this.state.details.loanAmount);
interestRate = eval(this.state.details.interestRate / 1200);
numberOfMonths = eval(this.state.details.termLength * 12);
result = eval(loanAmount * interestRate) /
    (1 - Math.pow(1 + interestRate, numberOfMonths * -1)).toFixed(2);
```

Welcome to LCB (Low-Cost Bank): Loan & Mortgage Calculator

Please Select Type

Find Mortgage

Loan Amount	Interest Rate (%)	Term Length (Yrs)	Age	Children
5000	0.5	1	40	2

Calculate **Clear All**

✓ Result: Mortgage = \$Infinity

Based on code from [Kunal Panchal](#)

In sum...

- To release faster and with confidence, we need CI
- CI is a key practice, part of DevOps
- Test Automation (TA) is a core part of CI
- TA is essential to enable fast feedback loops
- Static code analysis as part of CI to ensure code quality
- CI is a practice; it needs to be exercised and respected
- Testing techniques (e.g., pairwise-testing) and exploratory testing can improve our testing outcomes and efficiency

References

- Martin Fowler's articles
 - Continuous Integration
 - <https://martinfowler.com/articles/continuousIntegration.html>
 - Continuous Delivery
 - <https://martinfowler.com/bliki/ContinuousDelivery.html>
- Wiremock
 - <https://wiremock.org>
- Xray (Test Management)
 - <https://www.getxray.app>
 - <https://docs.getxray.app/display/XRAYCLOUD/TTT%3A+Automation>
- Agile & Holistic Testing
 - <https://agiletester.ca>
 - <https://agiletester.ca/quick-tips-for-agile-testing>
 - <https://janetgregory.ca/testing-from-a-holistic-point-of-view/>
- Pairwise testing, using PICT (open-source tool):
 - <https://github.com/microsoft/pict>
- State of the Developer Report 2022
 - <https://atlassianblog.wpengine.com/wp-content/uploads/2022/03/atlassian-state-of-the-developer-report-pdf.pdf>