
expoQA®26
MADRID 26th, 27th & 28th May
expoqa.eu

Improving accessibility shifting testing left

Nohelia Borjas
Test Automation engineer

Thibault Schnellbach
QA team lead

Our context

eCommerce Platform

Product organization

50+ product teams

Web / Mobile SAP

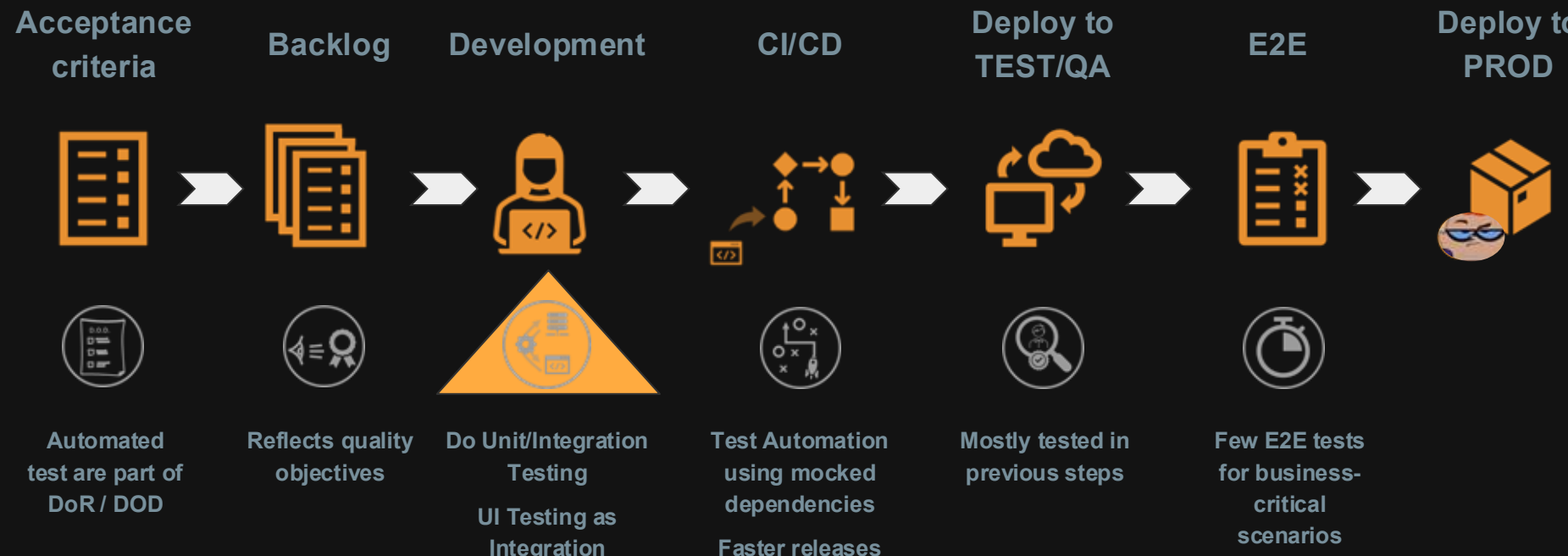
The goal

Test early to develop and release with confidence

The strategy

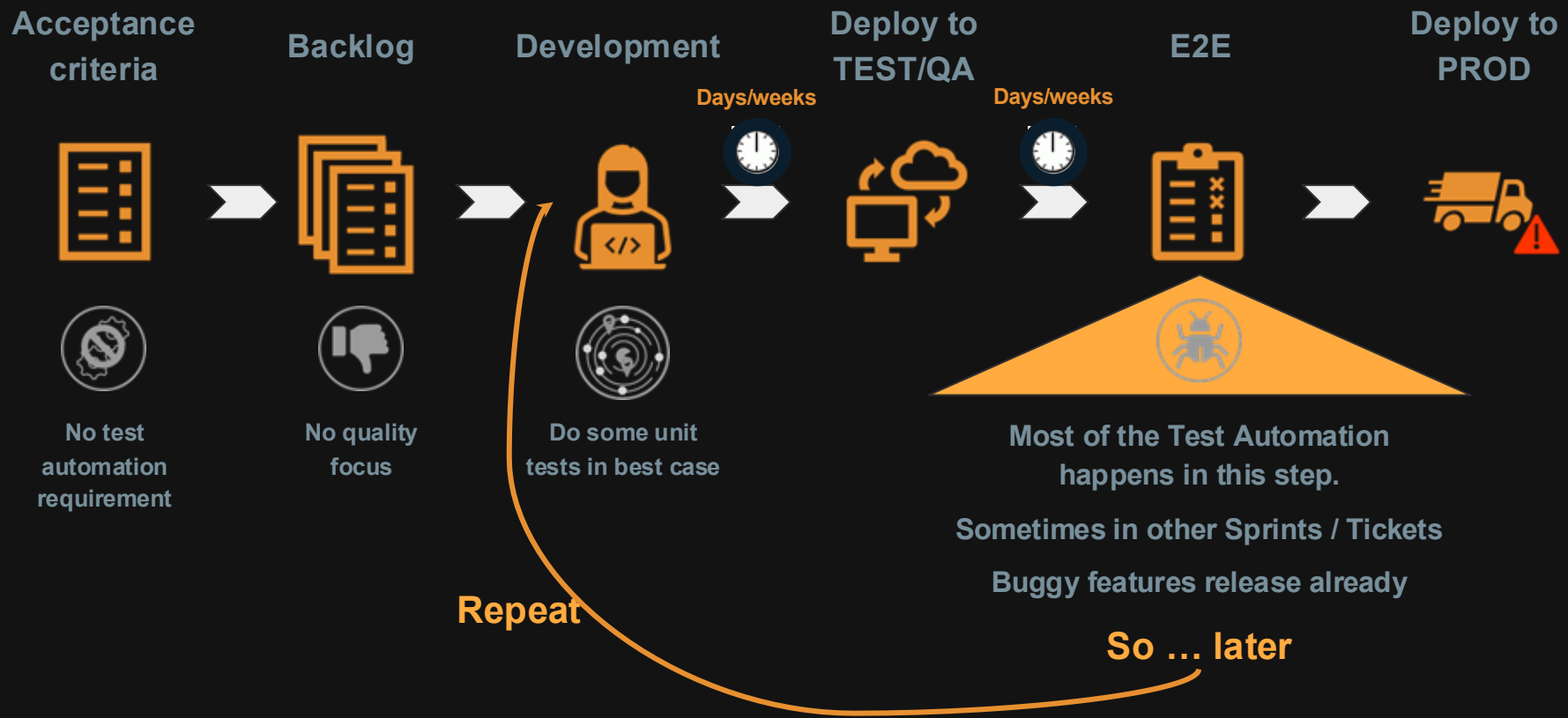
Shift Left Test Automation

Target development journey



Actually …

Current development journey



Current pain points

Slow & Late Feedback

Test automation implemented late → No early feedback for developers

Faulty Test Environments

Over-reliance on E2E tests and Test Data → slow, buggy test environments

Ice Cream Cone rather than Pyramid

Issues found after deployment → buggy code released

Impact in delivery

Slow & Late Feedback

Faulty Test Environments

Ice Cream Cone

IMPACT

Longer release cycles

More production incidents

Eroding trust in quality

WEAK CUSTOMER SATISFACTION

The goal

Test early to develop and release with confidence

3 main objectives

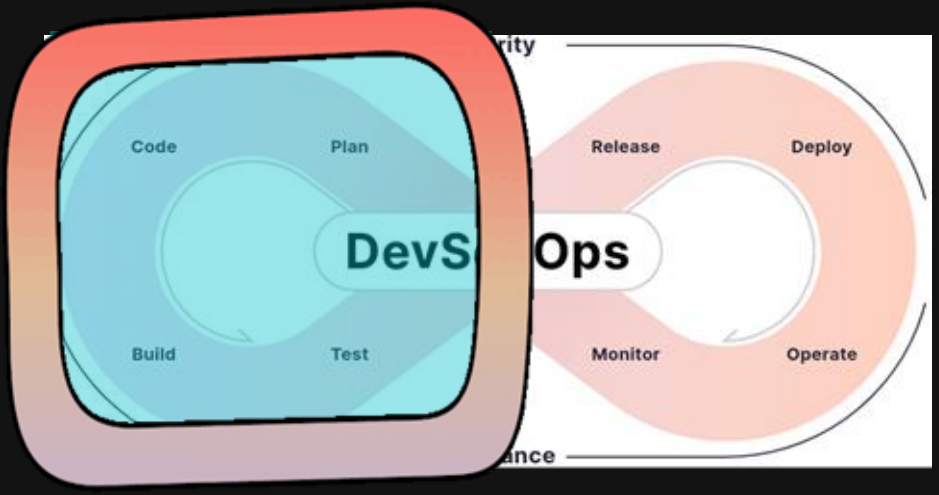
Balanced Test Pyramid

Fast, reliable & early feedback

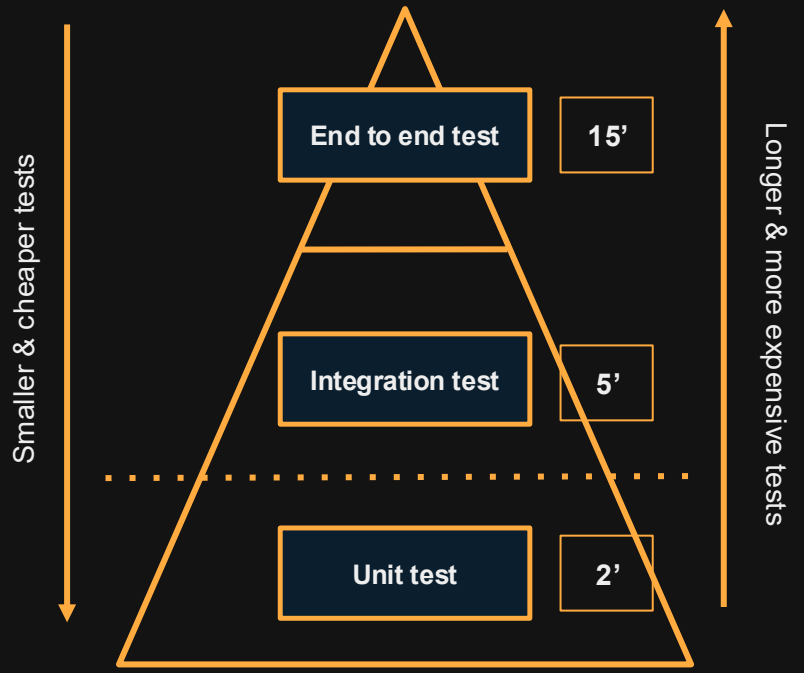
Team ownership

Secure SDLC

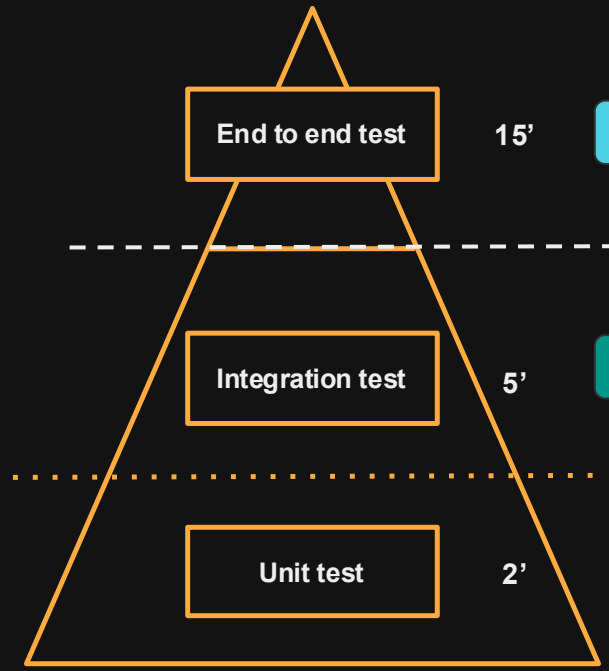
FOCUS HERE



Test pyramid



Test pyramid



Use less

Using real dependencies
Slow, complex

Use more

Using mocked dependencies
Fast, manageable

How we applied this approach to accessibility?

What's your Context?

Web App

Desktop App

Mobile App

Progressive Web Apps

What's your Context?

1. **What's your test automation stack?**
   Playwright, Selenium, Pytest, Junit, TestNG Maven, Gradle, Tox, Appium
2. **What are the a11y tools in the market for my stack?**
   Axe-deque library, Pa11y, CodeSniffer
3. **Can I integrate these results to my reports?**
   Allure, Extents, Built-in Reports
4. **What's the impact on my pipelines & deployments?**
   Azure, Github, Gitlab, Bitbucket

What is our Context?

Playwright

We used JAVA & integrated it with TestNG. We decided Gradle was the most potent tool to build it.

Axe-Deque

Open Source
Best-known source for Web Apps

Allure

Open Source
Great Visuals

Web App

Gradle, Java, TestNG...

Project set-up

BUILD

Gradle for dependency management and build.

Follows POM, dynamic-data-driven testing, JSON-based test configs, SLF4J logging.

CONFIGS & PROFILES

Multiple JSON configuration files for different locales/environments, used for test data & to extract and process configurations dynamically.

EXECUTIONS

In Parallel via TestNG and different XML files.
Sequentially via the gradle build/playwright runner.

PACKAGE STRUCTURE

Similar to any framework that follows the POM pattern

- `config (configuration handling)`
- `factory (for object creation and test data generation)`
- `utils (supportive utilities like OrderGenerator, Payment, etc.)`
- `webui (contains page object models for UI testing)`

Build.gradle file

```
plugins {
    id 'java'
    id 'io.qameta.allure' version '2.12.0'
 }

 repositories {
    mavenCentral()
    maven { url 'https://jitpack.io' }
 }

 dependencies {
    implementation "org.testng:testng:7.10.2"
    implementation "com.microsoft.playwright:playwright:1.49.1"
    implementation "com.deque.html.axe-core:playwright:4.10.1"

    testImplementation "org.testng:testng:7.10.2"
    testImplementation "io.qameta.allure:allure-testng:2.29.1"
}
```

Use your a11y tool in your flows

```java
@Step("Run accessibility checks") 6 usages  Nohella Borjas +1
private SoftAssert runAccessibilityCheck(Page page, String pageName) {
    var axeBuilder = new AxeBuilder(page);
    var accessibilityScanResults = axeBuilder.analyze();
    var issueMessage = accessibilityScanResults.getViolations();
    int issueCount = issueMessage.size();
    SoftAssert softAssert = new SoftAssert();
    Allure.addAttachment( name: pageName + " Accessibility Scan Results", accessibilityScanResults.toString());
    if (issueCount > 0) {
        Allure.step( name: "⚠️ Number of accessibility issues found on "
                + pageName + ": " + issueCount, Status.FAILED);
        for (int i = 1; i <= issueMessage.size(); i++) {
            Allure.step( name: i + ". Issue: " + issueMessage.get(i - 1), Status.FAILED);
        }
        log.warn(pageName + " has " + issueCount + " accessibility issues.");
        System.out.println("##accessibility-warning## Accessibility violations detected.");
        softAssert.fail(pageName + " has " + issueCount + " accessibility issues.");
        accessibilityIssues.put(pageName, accessibilityScanResults);
    }
    return softAssert;
}
```

```
import webui.CheckoutAddressPage;
import webui.CheckoutLoginPage;
import webui.HomePage;
import webui.OrderConfirmationPage;
import webui.OrderSummaryPage;
import webui.PaymentPage;
import webui.PaymentSelectPage;
import webui.ProductPage;

@Epic("Accessibility Testing")  borjasn +1 *
@Feature("Accessibility Verification for Checkout Pages")
@Slf4j
public class AccessibilityTests extends SequentialTest {

    private ProductPage pdp;  6 usages
    private CartPage cartPage;  4 usages
    private CheckoutLoginPage clp;  5 usages
    private CheckoutAddressPage cap;  6 usages
    private PaymentSelectPage psp;  5 usages
    private OrderSummaryPage osp;
    private OrderConfirmationPage oc
    private PaymentPage pap;  2 usages
    private final Map<String, Intege

    @Test  Nohelia Borjas +1
    @Description("Verify accessibili
    public void verifyCartOverlayAcc
        setupCartOverlay();
        verifyOverlayAccessibility(p
    }

    @Test(dependsOnMethods = "verifyCartOverlayAccessibility")  Nohelia Borjas +1
    @Description("Verify accessibility compliance for the Cart page")
    public void verifyCartPageAccessibility() {
        setupCartPage();
        runAccessibilityCheck(cartPage.page(), pageName: "Cart Page");
    }

    @Test(dependsOnMethods = "verifyCartPageAccessibility")  Nohelia Borjas +1
```

Add Allure/TestNG Annotations

Annotation

```
@Test  Nohelia Borjas +1
@Description("Verify accessibility compliance for the Cart Overlay")
public void verifyCartOverlayAccessibility() {
    setupCartOverlay();
    verifyOverlayAccessibility(pdp.page(), overlayClassName: "pca-the-overlay__container");
}
```

Continuous Quality: Test Automation Flow

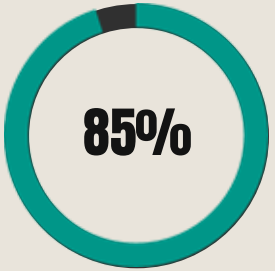
Deployment Pipeline

- **Unit Tests**
  Code-level verification
- **Integration Tests**
  Service-to-service validation
- **Deploy to Prod**
  [AUTO-ACTIVATED]
  Automated release

Test Pipeline

- **E2E Tests**
  User flow validation (Playwright)
- **A11y Tests**
  Accessibility audit (Axe-Deque)

E2E Quality Gate



THRESHOLD PASSED
Automation suite provides confidence for production stability.

Let’s try this now on the EXPOQA web app!

expoQA®26

MADRID 26th, 27th & 28th May

Thank you for attending

expoqa.eu