SHIFT LEFT TESTING

BY: Birtukan Haregewoin
03/04/2025

Agenda

- Introduction to Shift Left Testing
- Types of Shift Left Testing
- Difference Between Shift Left and Shift Right Testing
- Benefits of Shift Left Testing
- How to Implement Shift Left Testing

A Real Example of a Company Disaster Due to Lack of Shift-Left Testing

One of the most infamous cases of a company failing because of poor testing practices is the **Knight Capital Group collapse in 2012**. The financial firm lost **$460 million in just 45 minutes** due to a software glitch that could have been prevented with proper shift-left testing.

What Went Wrong?

Knight Capital deployed a new automated trading system update without thorough testing. The software contained a critical error: an old, unused function called "**Power Peg**" was accidentally reactivated, causing the system to place unintended, massive stock orders.

Introduction to Shift Left Testing

- **Definition:** Shift Left Testing is a software development approach that prioritizes testing and quality assurance activities as early as possible in the development lifecycle. This helps identify and fix defects sooner, improving software quality and reducing costs.
- **Objective:** Detect and resolve defects early in the Software Development Lifecycle (SDLC).
- **Key Concept:** Move testing activities closer to the beginning of the project rather than waiting until the end.

Types of Shift Left Testing

- **Traditional Shift-Left Testing:**

  Testing is introduced later in the development cycle, leading to longer feedback loops.

- **Incremental Shift-Left Testing:**

  A balanced approach, introducing testing at medium intervals.

- **Agile/DevOps Shift-Left Testing:**

  Aligns with Agile and DevOps, emphasizing short feedback loops and continuous testing.

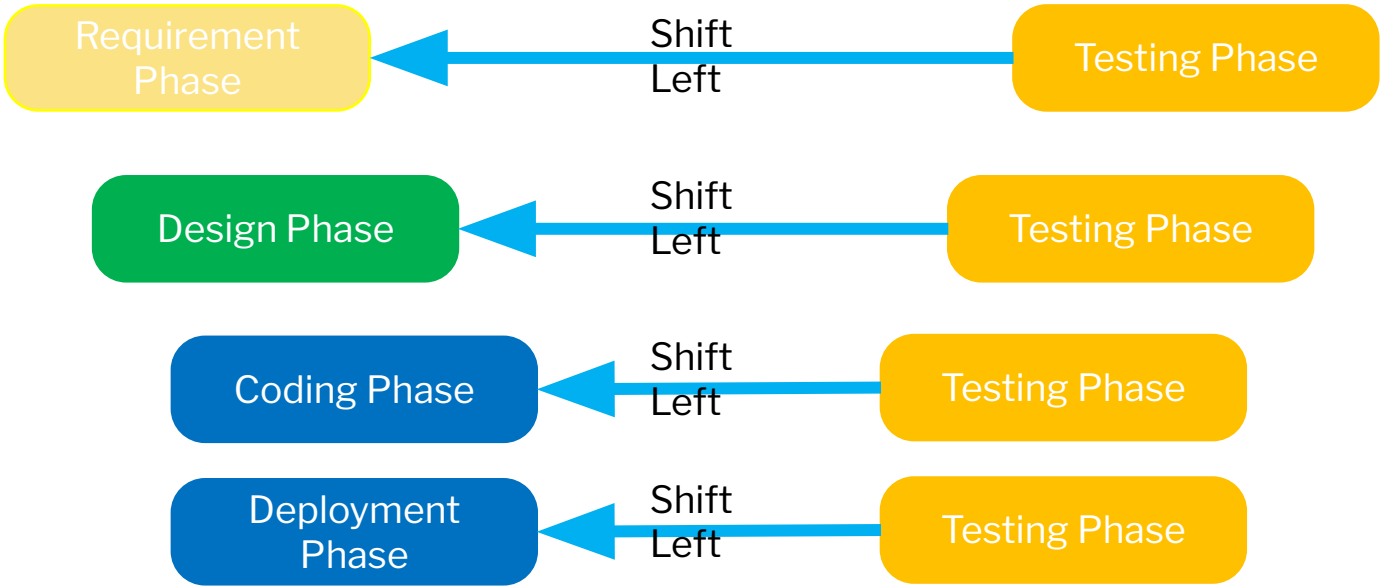
- **Model-Based Shift-Left Testing:**

  Focuses on early and continuous testing using models.

Testing Methods:

- **Unit Testing:** Verifies functionality of individual modules.
- **Integration Testing:** Ensures components work together correctly.
- **API & Contract Testing:** Validates service interactions in or micro services.
- **UI Testing:** Assesses complete functionality from a user perspective.

Shift Left Approach



Difference Between Shift Left and Shift Right Testing

Feature	Shift-Left Testing	Shift-Right Testing
Focus	Early defect detection & prevention	Real-world validation & user experience
Timing	Early in development lifecycle	After deployment to production
Goal	Improve software quality & reduce costs	Ensure reliability & performance in production

Benefits of Shift Left Testing

- **Early Bug Detection:** Reduces cost and effort of fixing defects later.
- **Faster Feedback Loops:** Enables quick fixes and continuous improvement.
- **Improved Quality & Reliability:** Ensures quality is built in from the start.
- **Reduced Time-to-Market:** Helps avoid delays caused by late-stage bug fixes.
- **Cost Efficiency:** Fixing bugs early minimizes rework and technical debt.
- **Better Collaboration:** Integrates QA, developers, and analysts early.
- **Supports CI/CD:** Automated early testing enables frequent, reliable releases.

Benefits of Shift Left Testing

How to Implement Shift Left Testing

1. **Understand Shift Left Concept:**
    - Traditional testing occurs late in development.
    - Shift Left Testing starts during requirements and design phases.
2. **Key Implementation Steps:**
    - **Integrate Testing Early:** Static testing, requirement & design reviews.
    - **Adopt Agile/DevOps:** Continuous testing, automated testing.
    - **Implement Test Automation:** Regression, smoke, API testing.

Advanced Implementation Strategies

- **Foster Collaboration:** Include QA in requirement & design phases.
- **Static Testing Techniques:** Code reviews, security analysis.
- **Shift Quality Ownership:** Developers responsible for unit t tests.

Tools & Cultural Changes for Shift Left Testing

1. Testing Tools:

- **Unit Testing:** JUnit, NUnit, pytest.
- **API Testing:** Postman, SoapUI, RestAssured.
- **UI Testing:** Selenium, Cypress, Playwright.
- **Performance Testing:** JMeter, Gatling.
- **Security Testing:** OWASP ZAP, SonarQube.
- **Monitoring:** Prometheus, Grafana.

Cont.

2. Cultural Changes Required:

- Break down silos between development and testing teams.
- Foster quality ownership across teams.
- Implement blameless postmortems for defects.
- Encourage continuous learning on testing techniques

Cont.

3. Measuring Success:

- Track defect escape rate (defects found in production).
- Monitor time to detect and fix defects.
- Measure test automation coverage.
- Track lead time for changes.

Thank you