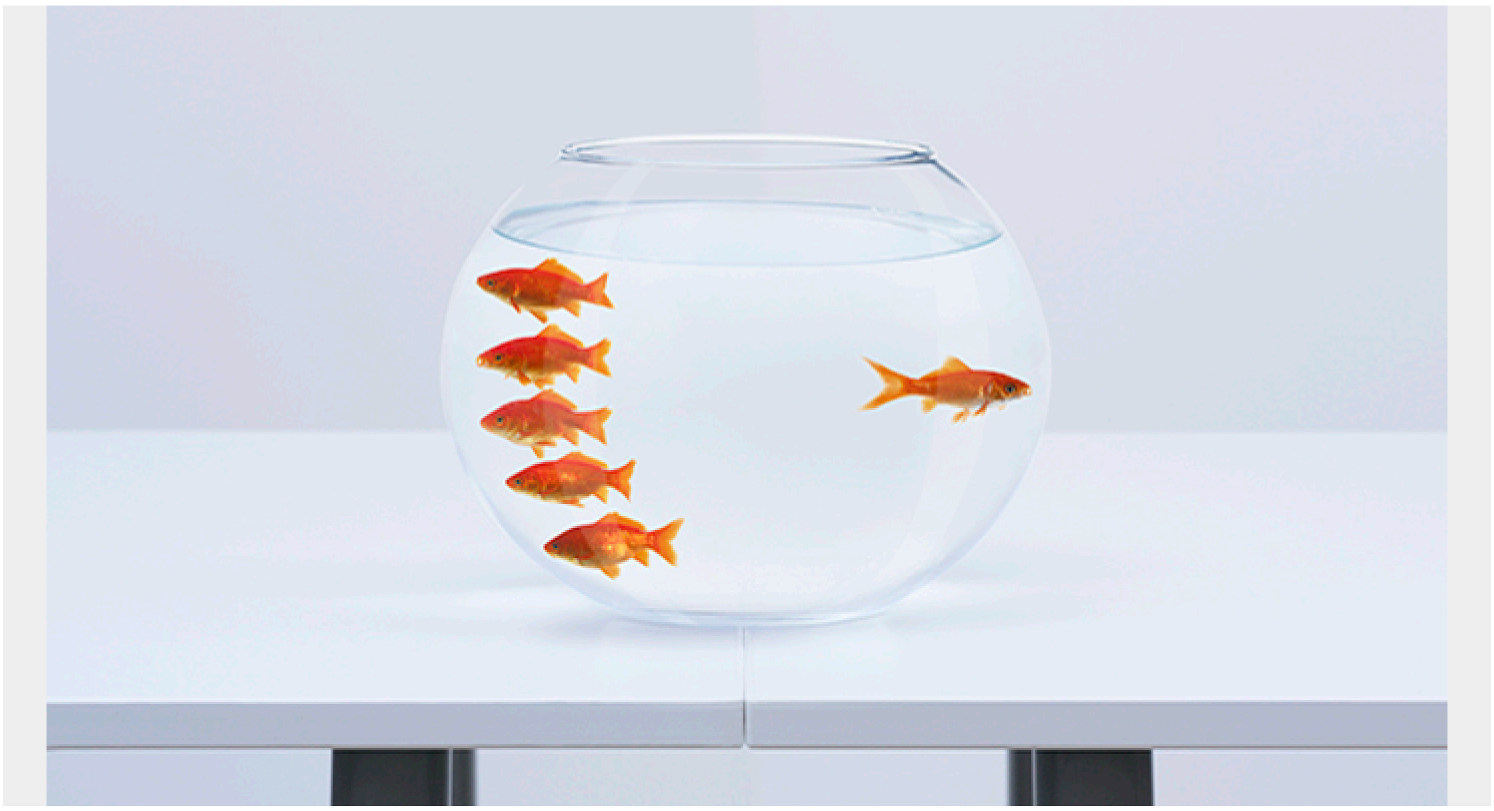
SHIFT LEFT TESTING: WHAT, WHY & HOW TO SHIFT LEFT



Agile practices, being adopted almost universally, necessitate faster and earlier testing in the software development lifecycle (SDLC). Bringing development and testing together early is commonly referred to as 'shifting left'.

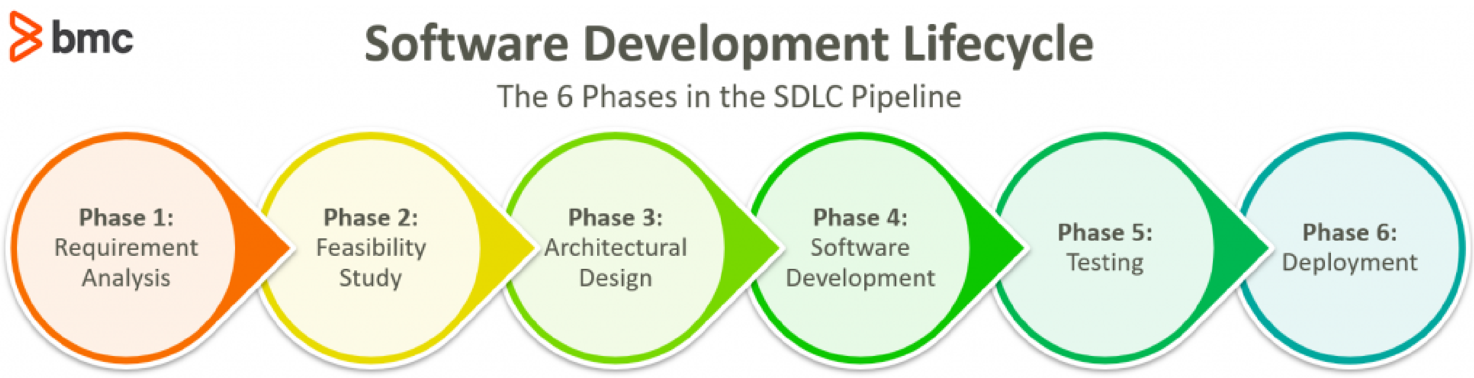
Traditional testing

A typical waterfall software development project would have seen testing occur immediately prior to release into production. This meant that when bugs or usability issues were inevitably found, the release would be delayed until these were fixed.

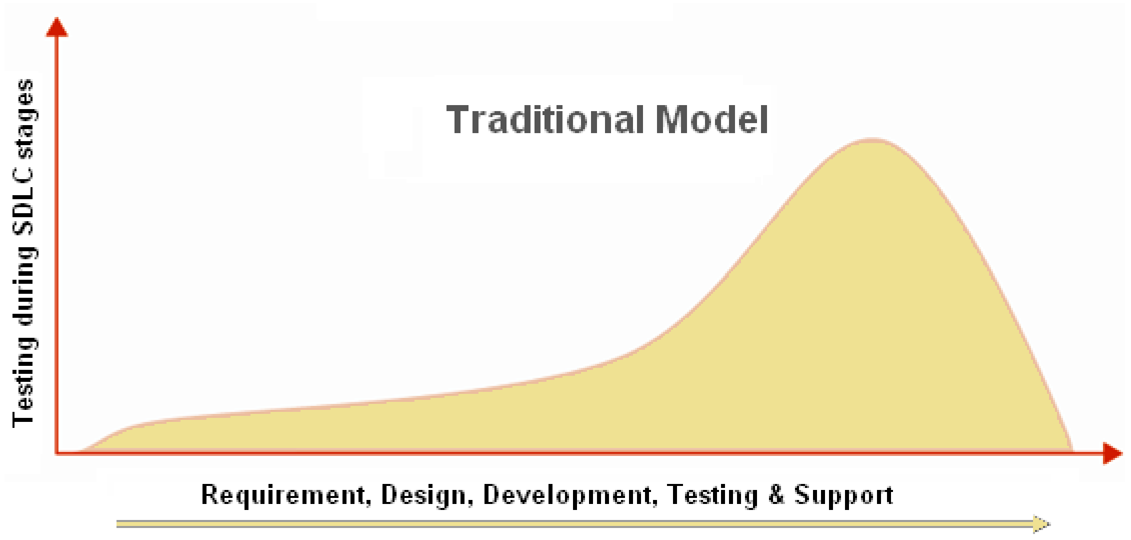
In this model, testing became a bottleneck that seriously impeded the ability of projects to deliver on time.

What's shift left testing?

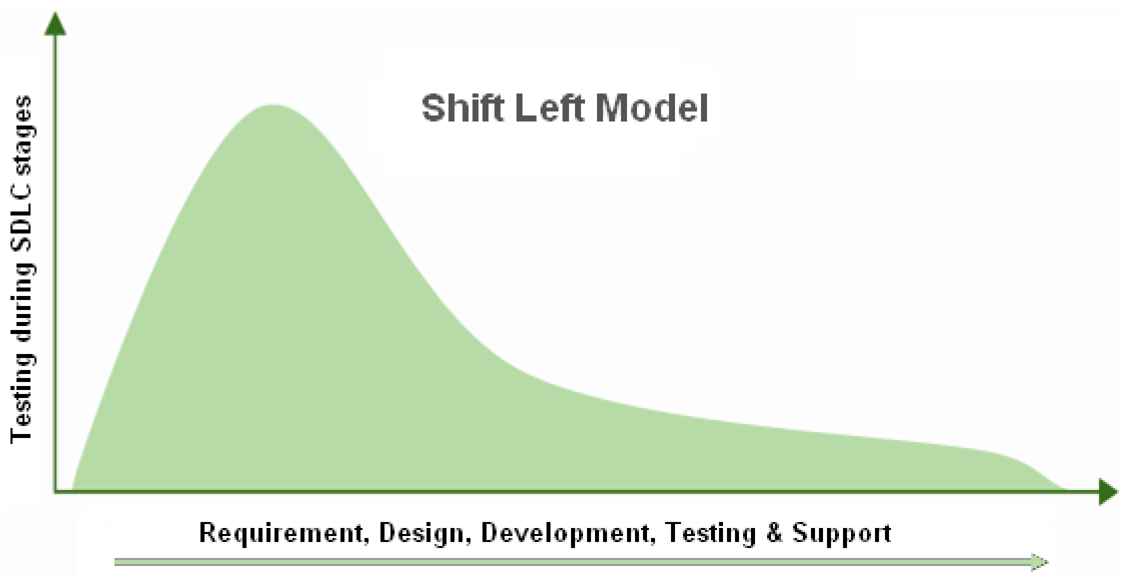
The easiest way to explain shift-left software testing is to think of the development cycle as a line running from left to right.


bmc
Software Development Lifecycle
The 6 Phases in the SDLC Pipeline
Phase 1: Requirement Analysis
Phase 2: Feasibility Study
Phase 3: Architectural Design
Phase 4: Software Development
Phase 5: Testing
Phase 6: Deployment

In the old model, testing only came into play on the far right of the line. Recognizing the bottleneck here, we now want to move the initiation of testing as far to the left as possible.



Shift Left is a practice intended to find and prevent defects early in the software delivery process. The idea is to improve quality by moving tasks to the left as early in the lifecycle as possible. Shift Left testing means testing earlier in the software development process.



Why Shift Left?

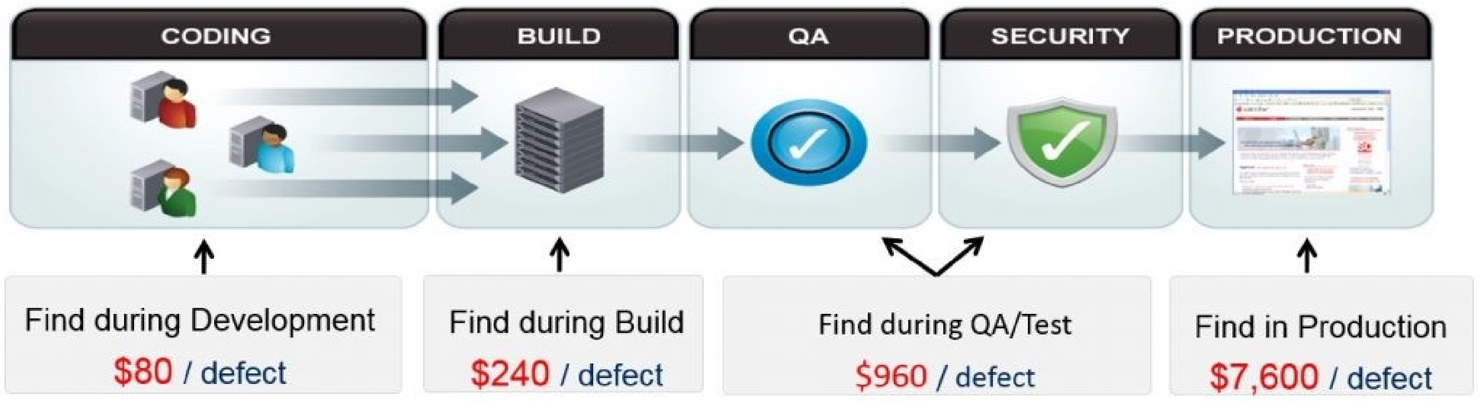
In the traditional software development model, requirements are kept on the left side of the plan, and the delivery and testing requirements on the right. The problem is that these practices can't handle changing expectations and requirements, resulting in negative outcomes for the business such as:

- Increased costs
- Increased time to market
- Unexpected errors

Cost alone is a very strong incentive for shifting your testing to the left. Estimates indicate that over half of all software defects could be identified during the requirements phase, with less than 10% emerging during the development phase of the lifecycle. The cost of resolving these defects works in reverse:

A defect that is removed after the product has gone into product will cost around 100 times more than one that is identified and removed during the requirements phase.

Research from the Ponemon Institute, in 2017, found that if vulnerabilities get detected in the early development process, they may cost around $80 on an average. But the same vulnerabilities may cost around $7,600 to fix if detected after they have moved into production.



The Shift left approach emphasizes the need for developers to concentrate on quality from their earliest stage of a software build, rather than waiting for errors and bugs to be found late in the SDLC. Shifting left enables product teams perform daily tasks like:

- Testing
- Providing feedback
- Reviewing changes and progress

Is Shift Left always appropriate?

A Shift Left testing approach may not always be able to deliver optimal performance and functioning in a real-world environment. In such situations, a Shift Right testing strategy may help to:

- Enhance customer experience
- Provide scope for implementation of test automation
- Ensure better test coverage

Shift Right initiates testing from the right, i.e., post-production. In this Shift Right practice, you'll test a completely built and functioning application to ensure performance and usability traits. Reviews and feedbacks from targeted users further help in enhancing the quality of the software.

An important characteristic of the Shift Right approach is a willingness to:

- Validate a hypothesis by trying out new solutions
- Collaborate with customers to determine what is working (instead of working from assumptions)

Continuous feedback from users may help in responding better to software failures.

How to move to Shift Left

There are some key strategies that will help you shift left with your software testing:

Demand planning

Test analysts will engage with business and operational stakeholders, providing a forward view of demand. Having this view enables you to—ahead of time—plan and finalize:

- The budget
- Resourcing
- Test strategies

Demand planning is an integral part of the shift left approach and provides a starting point for all other activities in the test lifecycle.

Static testing

Static testing is carried out in the early cycles of the project, and includes validation of requirements and design. The purpose of static testing is to find defects early in the life cycle that could prove to be very expensive to remove in the later phases of the project.

Use appropriate checklists to verify and validate requirements and design. Log defects into a defect management tool.

Unified test strategy

This is an overall, high level strategy for testing end-to-end—from unit testing through user acceptance testing (UAT), operational readiness testing (ORT), and post-deployment testing. The strategy will cover all phases of quality control, defining clear responsibilities.

A unified test strategy allows you to analyse dependencies on environments, stubs, automation, and test data—ensuring that the respective teams can fulfill the needs.

Risk-based analysis

Risk-based analysis is carried out to determine the impact and likelihood of failure for each test scenario. This approach is used for functional, non-functional and regression types of testing.

Once the test cases are established, decide the priority for the test cases based on the finished analysis. Discuss the impact of failure with the business analyst or designer. Determine the likelihood of failure from the development team.

Benefits of a shift left approach

There are several benefits that can be obtained by adopting a shift left strategy. Here are some of the most important:

Automation

Shifting left gives a greater ability to automate testing. Test automation provides some critical benefits:

- Much fewer human errors
- Increased test coverage (multiple tests can be conducted at the same time)
- Ability for testers to focus on more interesting and fulfilling tasks
- Fewer production issues

Increased delivery speed

Earlier means faster. When you find defects earlier in the production cycle, you can also fix them a lot faster. As a result:

- The time between releases can reduce significantly.
- The quality of software improves.

Increased satisfaction

Faster delivery of software with less defects is a major benefit of the shift-left approach.

If nothing else convinces you that this is a good move, then the smiles on the faces of your business partners should be all you need.

Learn from the choices Humana made when selecting a modern mainframe development environment for editing and debugging code to improve their velocity, quality and efficiency.

RELATED READING

- BMC DevOps Blog
- What Is Continuous Testing?
- Load Testing, Performance Testing & Stress Testing Explained
- Testing Frameworks: Unit Tests, Functional Tests, TDD & BDD Explained
- What is Parallel Testing?
- Patch vs Hotfix vs Coldfix vs Bugfix: Differences Explained