What is “Shift Left”? Shift Left Testing Explained

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.

In the traditional software development model, requirements are kept on the left side of the plan, and the delivery and testing requirements are on the right side. The problem is that these practices can’t handle changing expectations and requirements, and they lead to negative outcomes for the business such as increased costs, increased time to market, and unexpected errors.
It has been shown that some 56% of all software defects emerge during the **requirement phase**, 27% in the **design phase**, and only 7% during the **development phase**. These figures were first **detailed** by Pulitzer Prize-winning author and IT consultant James Martin, **affirmed** by S.A. Kelkar of the Indian Institute of Technology in *Structured Systems Analysis and Design*, and **further confirmed** by STBC in *The Economics of Testing*. 
Get the IT Buyer’s Guide for Service & Operations

This IT Buyer’s Guide helps you understand what to look for and how to choose the right tools for IT service and operations.

Get the Buyer’s Guide

Now that we’ve established the phases in which defects emerge, it’s critical to evaluate the costs associated with such issues. A report from IBM System Institute revealed that defects identified and resolved during the requirement & design phase are about 100 times less expensive to fix than those discovered after the release of a product.

Source: IBM

Per IBM Security, research from the National Institute of Standards and Technology & the Ponemon Institute 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 $7600 to fix if detected during the production phase.

Source: IBM
http://www-01.ibm.com/support/docview.wss?uid=swg27048410&aid=1

In a Shift Left approach, testing is performed during the early stages by moving to the left in the project lifecycle. Developers need to focus on quality from the beginning, instead of waiting for errors & bugs to be discovered late in the SDLC (software development lifecycle). Shifting left enables project teams to test, provide feedback, and review changes & progress daily.
What is Shift Right?

A Shift Left testing approach may sometimes lack in providing 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, and ensure better test coverage. Shift Right initiates testing from the right i.e. post production. A completely built and functioning application is being tested 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 and collaborating with customers to determine what is working rather than spending time working from assumptions. Continuous feedback from users may help in responding better to unforeseen circumstances such as crash down, failures, etc.

Strategies for Shifting Left

Demand Planning

Test analysts need to engage with business and operational stakeholders, and provide a forward view of the demand. This enables planning and finalization of budget, resourcing, and test strategies ahead of time. It is an integral part of the Shift Left approach and 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. It is done with the purpose of finding defects early in the life cycle that could prove to be very expensive in the later
phases of the project. Requirement and design verification & validation is performed during design collaboration calls using appropriate checklists, and defects are logged into defect management tool.

**Unified Test Strategy**

It is an overall, high level strategy for testing end-to-end, i.e. from unit test through UAT, ORT and post deployment testing. The critical coverage will be all phases of quality control with clear responsibilities. It analyses dependencies on environments, stubs, automation and test data, ensuring that these could be fulfilled by respective teams.

**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, the priority for the test cases is decided based on the analysis done. The impact of failure would be discussed with the business analyst or designer, and the likelihood of failure would be discussed with the development team.

**Benefits of Shifting Left**

**Automation**

The increased levels of automation while shifting left can:

- Reduce human errors.
- Increase test coverage by running more tests at the same time.
- Give enough time for testers to focus on more inspiring and fulfilling tasks.
- Reduce issues that arise during production.
Quicker delivery

Testers can accelerate project delivery by finding defects earlier in the software development life cycle, minimizing the time between releases, and thereby improving software quality. An organized process can provide the most effective way of accomplishing tasks on time. Quality Assurance is focused on developing and documenting these processes, and then improving them.

Glitch free execution

Finding defects earlier in the requirement phase can make sure that your team is aware of all business requirements.

Satisfied business partners

If the team can deliver all the features faster with higher quality, lower costs and higher customer satisfaction, there is no reason to wait in adopting a Shift Left approach.

Shift Left for ITSM vs. DevOps

Shift Left is broadly classified into 4 types – traditional, incremental, Agile and model based testing. As it is a new buzzword in ITSM, let’s compare it with DevOps.

Shift Left in ITSM involves resolving support tickets as efficiently as possible by shifting tickets to the least costly source of support. It brings more complex work down to first line staff, and moves standard activities and repetitive work into web self service portals. Highly experienced staff members make their solutions more available to less experienced colleagues, so they can help customers using readily available answers.

Shifting left in ITSM is typically driven by Knowledge Centered Support (KCS). It is an effective IT support
mechanism that places the creation and sharing of knowledge at the heart of every IT interaction.

From a DevOps perspective, we need to look at Shift Left as an approach used to improve collaboration and communication by engaging stakeholders from functions that are to the right in the delivery pipeline, earlier in the lifecycle. Shifting left requires two key practices – Continuous Testing and Continuous Deployment. Continuous Testing is the process of executing automated tests as early as possible, along with service virtualization to mimic unavailable systems. In Continuous Deployment, the deployment pipelines are completely automated and they are triggered automatically for every change. Explore the differences between Continuous Delivery, Continuous Deployment, and Continuous Integration.

Best Practices for Shift Left Testing

- All stakeholders must have a clear understanding of the strategy, functional & business requirements, design, and test plan.
- Never allow your requirements to move into the design phase without a second pair of eyes reviewing every aspect of that requirement.
- Implementing a proof of concept allows you to evaluate the application’s ease of use. It also helps you to show the client how a project would perform.
- Developers may assume some responsibility for the test automation efforts to speed up the entire process. Behavior driven development is a useful way to persuade developers to begin creating tests and bridging the gap between developers and testers.
- Testers may take up a few developer tasks such as test automation code reviews. When they have the required domain knowledge and a holistic understanding of the
system, they can effortlessly focus on areas where defects could be found.

- Organize meetings with development team pertaining to defect symptoms and solutions. Report all defects and their impact in weekly and monthly status meetings.
- Record defects with their analysis and resolution in a central repository. This could be used as a baseline for upcoming software projects.
- It is essential to create a maintenance plan that provides a clear road map and direction.

How to Get Started with Shift Left

1. **Define full testing lifecycle:** Define the testing lifecycle with SDLC phases, tasks, and tools. Move relevant testing activities to the initial planning, requirement and design phases.

2. **Formal design techniques:** You can make use of test design techniques such as decision analysis, equivalence class partitioning and orthogonal analysis.

3. **Requirements validation:** Implement a formal inspection method to aid in early discovery of defects. These inspections may shorten the development phase by identifying missing requirements.

4. **Integration:** The next step is to integrate all the development, project management and operation processes with testing. It helps in identifying the estimated time to complete the testing cycle and prevent duplication of tasks.

5. **Quality controls:** Quality controls can assist project teams in determining the health of a project at each phase. The project manager has to take corrective actions to get control over requirements and scope changes.

6. **Planning releases:** It is recommended to plan departmental releases in advance as it helps in understanding possible delays and managing resources.
7. **Creating test cases:** Experts normally suggest developing test cases that could cover functional processes and operational patterns. It helps in reducing the number of test cases to be created in the future.

8. **Engage in Test Automation:** Both developers and testers can automate the entire build to test process across all the SDLC phases, which can infuse better integration between processes, and drive more confidence in each release.

9. **Continuous feedback mechanism:** Continuous feedback mechanism helps testers to provide feedback to developers, and errors tend to decrease overtime.

**Conclusion**

Shift Left simply means shifting integration testing to the left of its usual position in the delivery pipeline. To achieve integration testing early in the lifecycle, test virtualization becomes a prerequisite. Shift Left testing is a key enabler for increasing efficiency and quality. In this model, every team member holds an equal responsibility for producing high quality software.

Hence, Shift Left is much more than a process change – it is a mindset and culture shift.