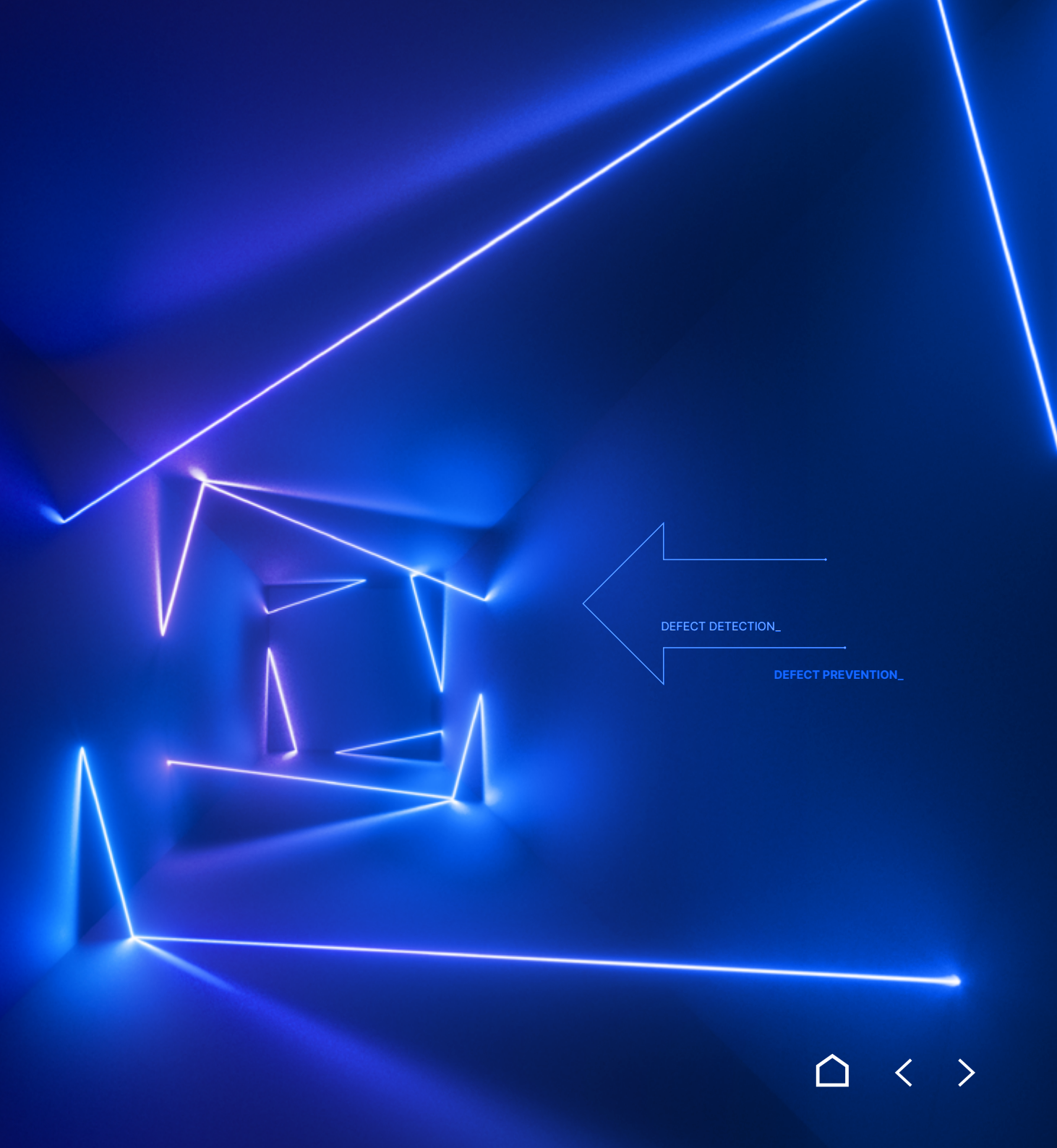
opentext™

ADM Market Insight:
Leveraging Shift-Left Testing in Performance Engineering

Introduction

As organizations adopt post-pandemic business models, they are also racing to play “catch up” with ongoing digital transformation initiatives. As a result, performance engineering teams are expected to release software faster, meet enhanced performance requirements, and deliver a superior customer experience.

The widespread adoption of Agile and DevOps methodologies has also led to dramatic changes in software development and testing. By adopting shift-left testing practices, developers and testers collaborate earlier in the software development lifecycle (SDLC) with a keen focus on defect prevention versus defect detection.



The Case for Shift-Left Testing

Traditionally, performance testing has been the final step in the SDLC—often completed during the User Acceptance phase, or even later. But waiting to identify bugs, defects, and other issues often delayed releases. In some cases, the software would be scrapped altogether due to the time and cost involved in fixing defects.

By adopting shift-left testing, software developers and testers collaborate throughout the SDLC. Testers understand the software requirements, design, architecture, and functionality from the very beginning. Because testing is part of the development process, all team members work together to identify and resolve issues faster and earlier, reducing the time and cost involved.

Why Shift-Left Testing in Performance Engineering?

Shift-left testing bridges the gap between development and testing teams—allowing for seamless collaboration among teams, tools, and processes through continuous feedback loops. In other words, QA and performance engineers aren't solely responsible for software quality and user experience. A shift-left approach involves every team working together to:

+ Maintain high levels of performance that enhance the customer experience, keeping up with the pace of business.
+ Manage end-to-end performance.

Six Shift-Left Testing Facts

Keep the following in mind while developing your strategy:

1.
Collaboration is essential. Shift-left testing requires involving team members from diverse disciplines at every stage of the SDLC. This approach shifts focus from defect detection to defect prevention.

2.
Shift-left testing expands testers' roles in the SDLC. They work proactively with the team to plan and build an effective testing strategy that accounts for the long-term vision of the product.

3.
A shift-left approach allows testers to test the software design first and through the customer experience lens.

4.
Testing throughout the SDLC empowers developers to take more ownership of their code and the final deliverable.

5.
It empowers testers to adopt test-driven development (TDD) and behavioural-driven development. These practices help prevent the induction of defects into the software.

6.
Shift-left testing also works well with the Agile Testing Manifesto. This approach supports Agile Scrum teams, which include testers and other stakeholders in regular stand-up calls, review meetings, and other communication forums. Scrums provide testers with a detailed analysis of the software and rapid feedback to prevent defects from being grounded into the software.

Benefits of Shift-Left Testing

Shifting left introduces testing earlier in the software development process. This practice results in more efficient, comprehensive testing while also improving software quality and customer satisfaction:

+ **Improved solution design—**Teams often have a sharper focus on quality when testing starts earlier. Having additional stakeholders responsible for QA can also lead to fresh perspectives and potentially new design alternatives.
+ **Earlier bug detection—**In shift-left testing, potential glitches and performance issues are identified sooner and can be addressed more efficiently instead of as an afterthought "once the critical stuff gets done."
+ **Time and resource savings—**Shift-left testing empowers teams to identify and address any defects early, reducing the cost of fixing them.
+ **Faster time to market—**When teams detect bugs and other performance issues earlier, they can fix them faster. Quickly resolving these problems can accelerate time to market and significantly reduce the time between releases.

Shift-Left Testing and the Agile Manifesto

Shift-left testing supports the Agile Testing Manifesto. It encourages teams to focus on:

+ Testing throughout versus testing at the end.
+ Preventing bugs versus finding bugs.
+ Testing understanding versus checking functionality.
+ Building the best system versus breaking the system.
+ Extending responsibility for quality to the team—not just testers.

Time and Resource Savings

The cost to fix an error found after product release was four to five times as much as one uncovered during design, and up to 100 times more than one identified in the maintenance phase.

+ The Systems Sciences Institute at IBM

Harmonize Your Shift-Left Goals

The more teams shift left, the more they can confidently produce high-quality software at speed and at scale. The challenge is that it has increasingly expanded testing demands. With performance engineering, developers are responsible for ensuring that applications are tested for performance, eliminating the need for teams to go back and refactor an entire application.

While shifting left requires cultural and organizational change, using tools that automate or streamline shift-left initiatives can facilitate adoption. To harmonize efforts, you need to choose techniques and tools that fit existing workflows, integrate with preferred tools, and automate as much as possible.

OpenText's primary solutions have an analogous developer-centric solution that natively integrates with common IDEs and with their counterparts. These solutions facilitate shift-left testing without increasing workloads to accelerate adoption:

+ **UFT Developer: Easily create tests for continuous testing and integration.**
+ **LoadRunner solutions: Natively run scripts and reuse assets in all LoadRunner family solutions. Developers can script, model scenarios, run performance tests, and quickly analyze the results without leaving their familiar development ecosystem.**
+ **PulseUno and Fortify: Inspect code as changes are checked in, the relevant security validations occur, and vulnerabilities are identified and fixed before release.**
+ **ALM Octane: Provide a central hub that includes integrated backlog, defect, and DevOps pipeline management.**

In addition, OpenText performance engineering solutions offer a proactive, continuous testing discipline that delivers four key advantages:

+ **Balance and prioritize responsibilities across developers, testers, and performance engineers.**
+ **Broaden integration of performance into the CI/CD process.**
+ **Monitor performance from build to production.**
+ **Continuously analyze and efficiently collaborate across teams.**

Learn how the LoadRunner family of performance engineering solutions can help your teams shift left to deliver better software quality and an optimal customer experience.

Learn More

opentext™