WHITE PAPER

Continuous Testing

It’s Time for Test to “Shift Left” in the SDLC

Sunil Chandrasekharan
Technology Services Director, FIS Global Automation and Testing Services

November 2018

Continuous Testing

This white paper examines the role of Continuous Testing in a Continuous Delivery model of software application development. This involves the use (and reuse) of automated test scripts and execution, as well as a "Shift Left" paradigm to move Test to begin on Day One of the software development life cycle (SDLC).

Software Testing Challenges and Opportunities

Challenge – We live in a digital era where rapid delivery of high quality, flexible and scalable applications is essential for organizations to succeed. Ever increasing user needs and expectations increase the complexity of the testing landscape. Software development is becoming more agile, but testing has been one of the slower phases of the SDLC to adapt. Most bottlenecks are due to traditional testing methods, antiquated processes, and tools that are just too slow to deliver the needed results. Low quality requirements, inadequate test data, laborious test environment setup, and manual process execution are contributing factors as well.

Opportunity – A Continuous Delivery model allows organizations to develop and deliver software faster, in a continuous fashion. However, the speed of a Continuous Delivery model is only as fast the slowest process in this model, and Testing – specifically traditional testing methodologies – has not kept pace with agile development speed. Testing teams do not have weeks to test, but rather hours and maybe minutes. Automating and identifying critical test scenarios are very important factors in improving testing speed and efficiency. Traditional methods place testing at the end of the development cycle, but integrating testing from Day One helps to create a pipeline that is capable of delivering tested software faster without compromising quality. The Continuous Testing process initiates testing early on in the SDLC, helping to identify business risk for the release and enabling the business to make quick informed decisions. Continuous Testing applies agile concepts to the Quality Assurance (QA) process and increases overall development efficiency and throughput.

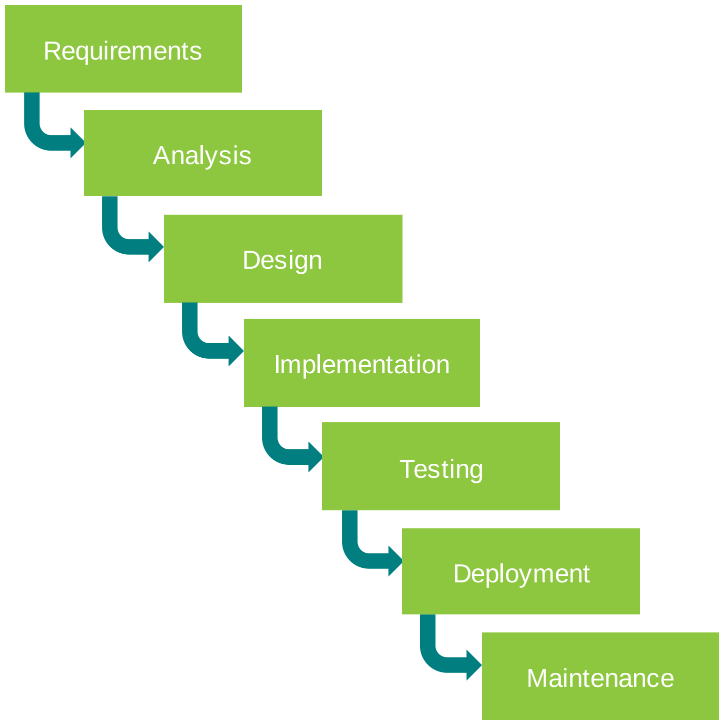
Benefits

Quality, speed and accuracy are the key benefits of Continuous Testing.[1] Performing testing early on in the development life cycle promotes quality from requirements gathering through development and system integration. Defects found are fixed early on, avoiding late rework and reducing cost while maintaining the user-desired functionalities. Software is accurately deployed faster and with less cost. Changes can be managed throughout the requirements, design, development, testing and deployment SDLC because these assets are integrated and functional from Day One, while also providing the flexibility to make changes and deploy them without negatively impacting the user experience.

[1] https://www.ca.com/content/dam/ca/us/files/white-paper/continuous-testing-for-continuous-delivery.pdf

Traditional Testing

While rigorous testing is recognized as a very important function needed to deliver quality software, it is often blamed for being too slow or inefficient. This is not just the Test team's fault. Traditional testing processes and tools are manual, slow and not easily integrated with deployment. Traditional testing follows a top-down approach where completion of one phase leads to another phase, an approach in which it is inherently difficult and time-consuming for development teams to make and control changes.[2]



Traditional Testing Process

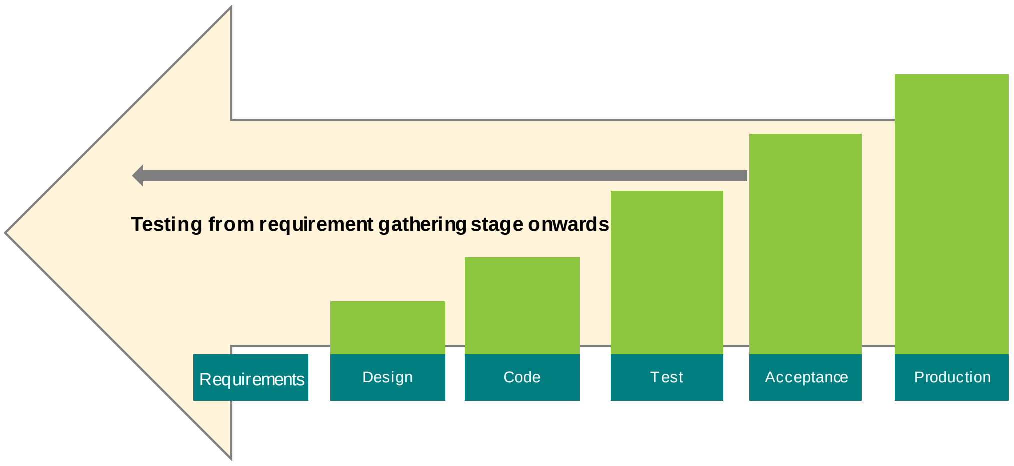
Traditionally testing occurs at the end of the software development life cycle, causing delays in finding defects and fixing them. Functional testing often takes weeks or months to complete due to unavailability of test data, environmental issues, and/or inadequate requirements. Automated regression testing must wait until all the functional changes are complete. Performance testing then occurs after all functional testing is completed and system integration is in place. Focusing testing at the end of the development cycle can become particularly problematic if performance issues are identified at such a late stage.

[2] https://devops.com/continuous-testing-devops-bottleneck/

“Shift Left” Approach

The “Shift Left” approach is based on starting testing early in the life cycle.[3] Traditional testing processes involved a lot of waiting: waiting for testing work, waiting on requirements, waiting for new builds with new features and development in order for regression testing to begin. Thanks to application program interfaces (APIs) becoming a major component in applications, there is no longer the need to wait for the user interface (UI) to be finalized before testing can begin. Testing can be injected in all parts of the project, and this enables the project team to focus on quality from Day One. Teams are able to identify and address issues and bottlenecks early on, reduce development costs, and deliver quality tested products to market rapidly.

Performance testing can also be started early on and performed throughout the development, build, deployment and final testing phases. This enables the development team to identify and fix performance issues at their onset, using advanced technologies and methodologies to streamline the overall development schedule and ensure the software meets scalability and reliability objectives.



“Shift Left” Approach to Testing[4]

Automation Is Key

Automation is key to implementing Continuous Testing and providing an end-to-end testing solution that can be integrated with the Continuous Delivery pipeline. Automation helps improve testing speed and efficiency. API, functional, system integration and performance testing all need to be automated and integrated to optimally improve efficiency and provide fast and reliable test results. With advancements in automation tools and technology, the creation of automation scripts can start from Day One instead of waiting for all the functionalities to be delivered. Automation tools can test the API layer directly without going through the UI, thereby reducing wait times. Most automation tools integrate with the DevOps platform, thereby improving the testing speed and providing results faster. Triggering of automated scripts and collection of test results enable a truly end-to-end automation process with no manual intervention needed. In addition to enabling Continuous Testing in the SDLC, automation scripts can be reused in different phases of testing – base product, implementation, production. Re-usability of artifacts reduces

[3] https://softwaretesting.cioreview.com/cxoinsight/to-shift-left-or-to-shift-right-continuous-testing-is-everyone-s-responsibility-nid-23923-cid-112.html

[4] http://360-degree-technosoft.weebly.com/blog/make-your-mobile-app-development-bug-free-with-shift-left-testing

the implementation time and improves the quality of the software delivered. It also enables production readiness and proactive monitoring (health check) of production systems thereby reducing or even eliminating downtime.

Performance Testing

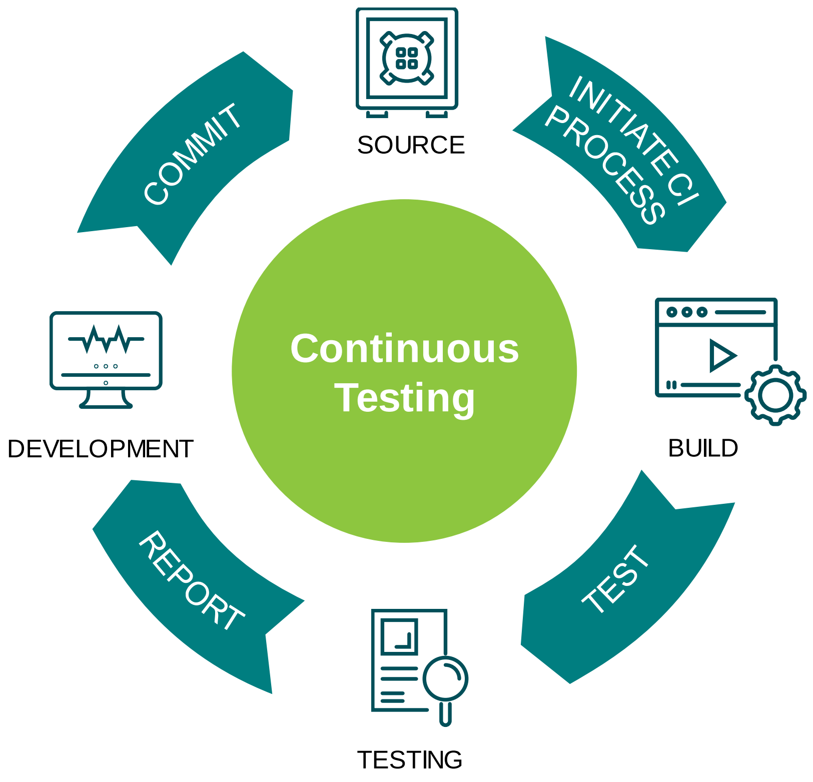
Performance testing analyzes application performance under load and stress. It has always been considered a final testing activity, performed once the entire system is in place and regression testing has completed. However, the business cannot wait to have performance test results at the end of the life cycle: Application scalability and performance are critical factors that need to be understood early on in the project and monitored throughout. Development teams need and want to understand application scalability and performance under load as they are building the solution. Providing the performance test results early on helps agile development teams build their applications more robustly and ensures each solution is capable of scaling to meet consumer demand.

There are three stages where early performance tests can help – code development, test and deployment. As such, performance testing needs to “Shift Left” to uncover any performance issues early on, and “Shift Right” to monitor actual user transactions within applications. With real-time monitoring tools, the user-reported issues gathered from a live environment help build and improve the performance tests for future iterations.

Continuous Testing

What is Continuous Testing, specifically? Think of it as a continuous process such as a manufacturing assembly line where all the processes are automated and each task is performed in a workflow one after the other[5]. Continuous Testing follows a similar workflow, as shown below:

[5] http://www.professionalqa.com/continuous-testing



Continuous Testing

For example, once QA deployment is completed, smoke test begins. Once the smoke test passes, the workflow proceeds to functional/regression testing, and if that meets the “pass” criteria then the performance test is run. If any of these testing activities do not meet the pass criteria, a report is generated with error details and the workflow stops the release of the build until the issue is resolved and a new build is deployed.

It is critical to identify the business risk associated with each release candidate. Automated tests are designed to be efficient and predict the business risk accurately. In addition to the automation of testing, the key element for Continuous Testing success is to identify the test set that will predict the impact to the end user.

Reporting of test results provides key information regarding the business risk. Dashboards and reports with detailed information of the testing results are essential to perform analysis and identify corrective actions to mitigate risk. Historical test result information and dashboards showing test result trends help quickly identify and resolve issues.

The goal of Continuous Testing is to “Shift Left” to engage testing Day One and find defects earlier in the software development life cycle, enabling the release of higher quality tested and scalable software at a quicker pace. Continuous Testing plays an important role in the Continuous Delivery ecosystem by helping take an idea from design to production at a faster speed without compromising quality. To achieve Continuous Delivery, organizations need to adopt Continuous Testing.