AgileLoad

White Paper

Performance Testing In Agile Process

Table of Contents

At the core of performance testing is the art of imitating real user behavior and activity and then evaluating how the application responds.

Introduction and Background Information

Performance testing is an essential activity in a software development life cycle. From initial planning to production analysis, application performance drives the development of better software iterations and releases. Application stakeholders, programmers and testers must make performance a primary consideration in all iterations of Agile development process.

This white paper can help you gain a closer understanding of how you can integrate performance testing into your Agile Processes. The paper summarizes the key activities and best practices involved in performance testing.

Issues Related to Performance Testing

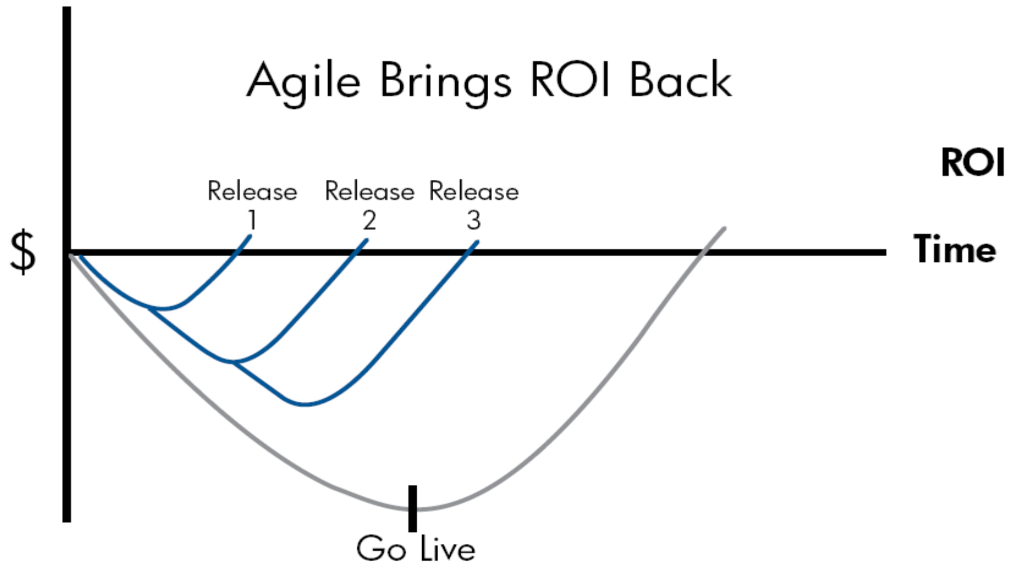
Over the past 2 decades, business application machines or devices have evolved from monolithic mainframe systems to distributed and composite systems. Development life cycle times are now measured in weeks and even days as opposed to years. These changes to the technology and development methodology both increase the pressure on test engineers to plan and finish performance testing. In addition to evaluating new systems, legacy application performance issues continue to impact test engineers. At the core of performance testing is the art of imitating real user behavior and activity and then evaluating how the application responds. With brand new systems, the test engineer is forced to make a calculated guess about the behavior of users based on use-cases and requirements. Even with the existing systems, it is a challenging task to perfectly model user behavior and interaction with the application.

Another performance related issue is reliably using data from a limited non-production environment to precisely predict how the system can perform in a more robust environment. Test engineers usually do their best to extrapolate and attempt to identify how the system can perform, but often have limited success forecasting performance.

An Agile Approach

As per the recent Forrester research report, interest in agile methodologies has been on the rise over the years. Agile approach improves the ROI (return on investment).

Around the world, organizations are working to produce high quality software in less time. In order to realize this goal, many enterprises have started adopting Agile methodology. This approach can help accelerate time to market, improve the quality of software, enhance productivity and alignment between various business purposes and the technologies. Agile process has tremendous momentum. As per the recent Forrester research report, interest in Agile methodologies has been on the rise over the past few years. Agile development breaks through the barriers of traditional waterfall model to software development to provide better value faster and improve the ROI (return on investment).



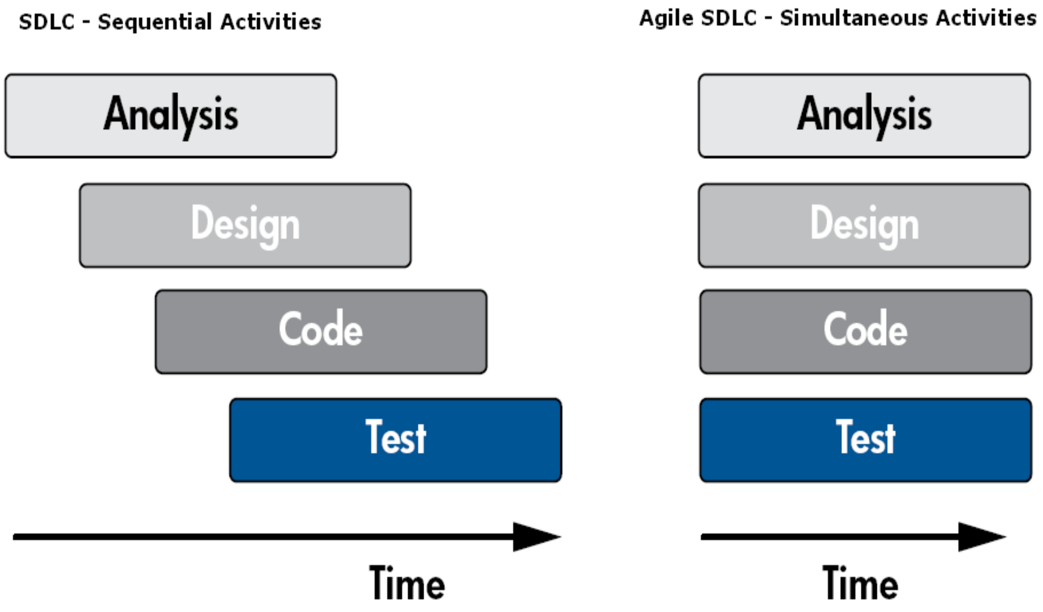
An Agile environment puts various teams to work and develop the product in a manner that is iterative and incremental. Agile methodology promotes close teamwork, appropriate reviews, more iterations and self-organization. This approach enhances rapid delivery of high quality software and enables a business practice that aligns software development with customer needs and company goals.

Performance testing that usually takes place at the end of the lifecycle in a waterfall model will move to the beginning in an Agile methodology, including the analysis and designing.

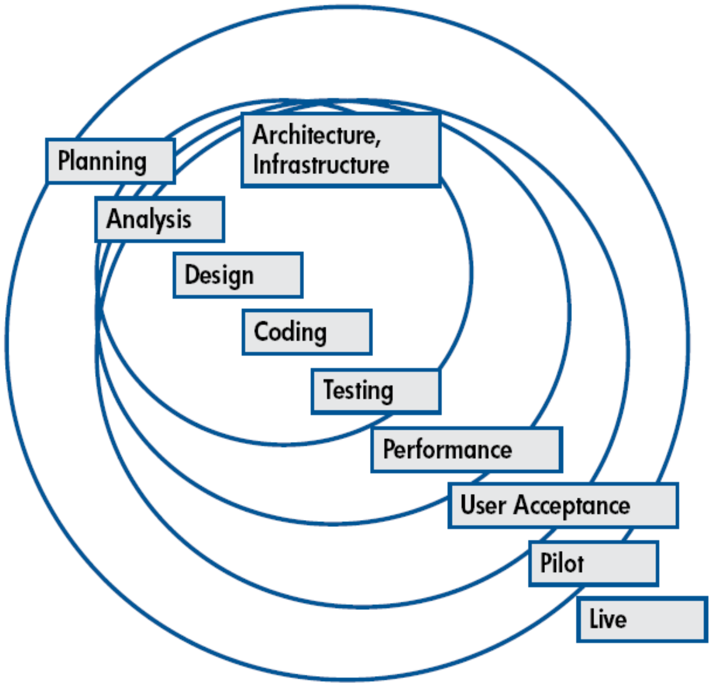
The performance of an application is directly proportional to its design and hence should be addressed at an early stage of the development lifecycle. Performance testing is considered throughout the Agile process, that is

Performance testing that usually takes place at the end of the lifecycle in a waterfall model will move to the beginning in an Agile methodology.

from the release planning stage onwards.
The traditional SDLC versus Agile SDLC – Activities chart is depicted below.


SDLC - Sequential Activities
Analysis
Design
Code
Test
Time
Agile SDLC - Simultaneous Activities
Analysis
Design
Code
Test
Time

One way of ensuring that performance testing is included early in the SDLC process is by having a clear definition of "Done" (as shown below) and its



adoption in the Agile implementation so that the application is tested for functionality, performance, and security by the end of each sprint.

Agile Performance Testing

In unit level tuning, tests are focused on discovering poorly performing methods. This is performed when the scalability and performance requirements are specified at method level.

This section describes an agile approach to managing application performance testing. As the term implies, the key to an agile approach is flexibility. To remain efficient and thorough in an agile environment, you may need to change the way you are used to managing your performance testing.

The objective here is to determine the performance bottlenecks in key business processes as early as possible during the project development cycle. You need to identify, isolate and fix performance bottlenecks at the component or code level.

Agile Performance Testing is a three phases approach as shown below:

Unit Level Tuning (for optimizing): Executing tests to isolate and fix bottlenecks at code level.

Component Level Tuning (testing components): Executing tests to isolate and fix bottlenecks in application components level.

Application Load Tuning (testing application flows): Testing the critical application flows for user experience under normal and peak loads.

Unit Level Tuning

Objective

- Tests are focused on discovering poorly performing methods.
- This is performed when the scalability and performance requirements are specified at method level.

Entry Criteria

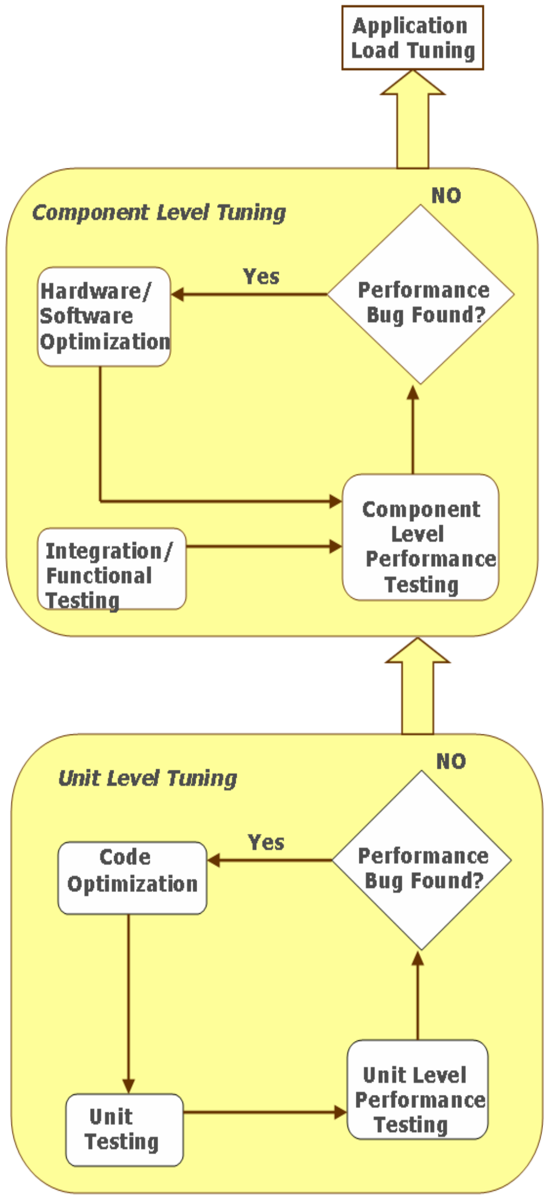
Unit performance tuning proceeds along with the development of the application and unit testing of the method to tune is completed.

Activity

- Identifying the methods.
- Scripts are prepared to invoke individual method.

In component level tuning, the major objective is to evaluate the system behavior when it is pushed beyond its breaking point.

- Methods are tested under normal load for response time while measuring CPU and memory utilization.
- If method does not show acceptable level of performance, it is fine tuned and test is re-executed.



Component Level Tuning

Objective

- Here the tests are targeted towards finding bottlenecks in application tasks and activities.
- The major objective is to evaluate the system behavior when it is pushed beyond its breaking point.

Entry Criteria

The components are deployed on their respective application server.

Activity

- The components can be identified based on following characteristics:

    1. High business criticality and usage.
    2. Can be deployed and invoked independently.
    3. Deployed at application or middle tier.

- Scalability and response time requirement of each component is captured.
- Scripts are prepared to invoke individual component.
- Tests are executed for each component separately. With respect to two major areas of bottlenecks, the following two tests are executed:

    1. Concurrency Test: Component is tested for simultaneous requests under incremental user load, until failure is noticed.
    2. Test for Throughput: Component is tested for high number of hits per second with limited number of users.

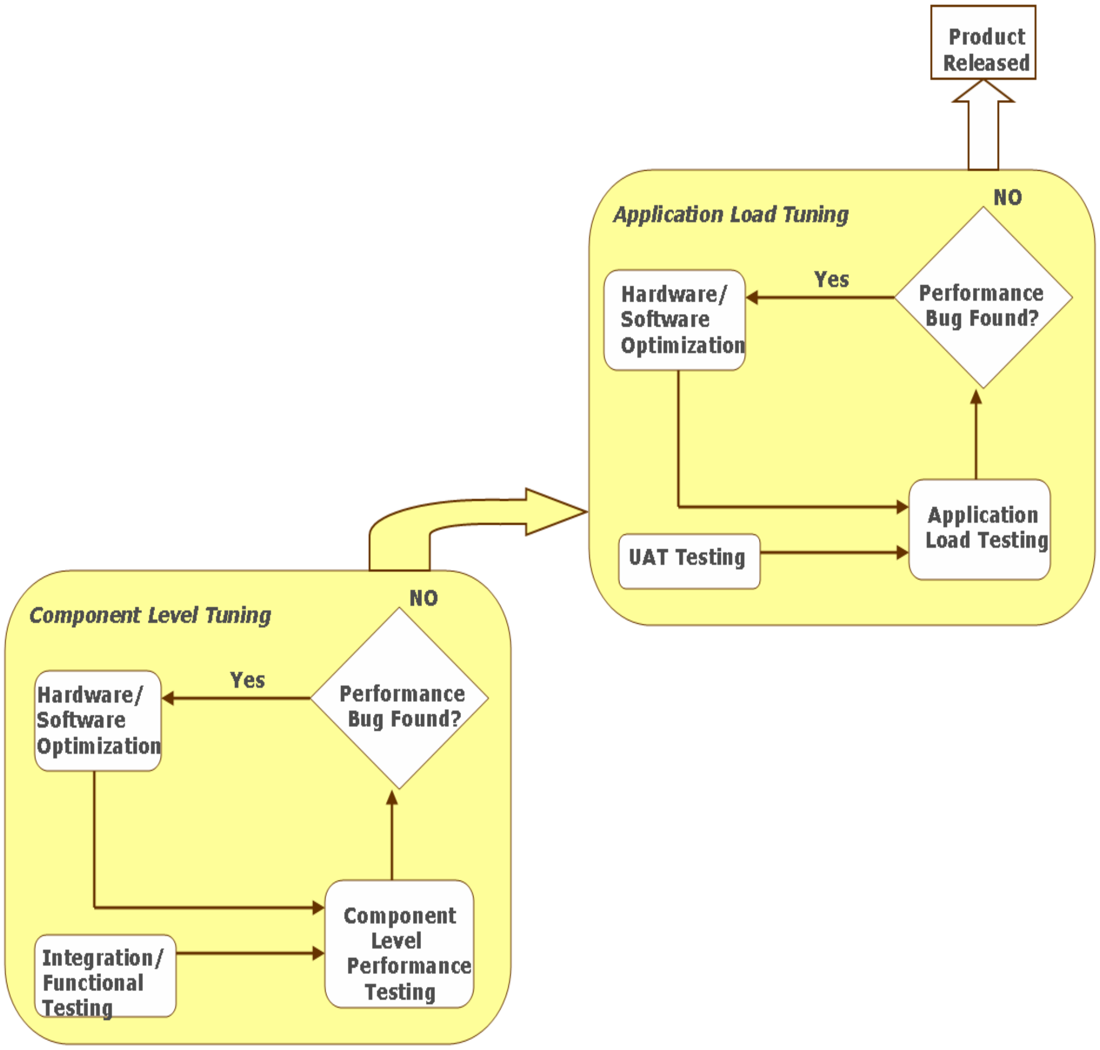
- With the help of above results, the component is tuned by optimizing hardware/software configuration and test is re-executed.

Critical application flows are tested for user experience under normal and maximum loads in application load tuning.

Application Load Tuning

Objective

- Here, critical application flows are tested for user experience under normal and maximum loads.
- The major objective is to capture the performance metrics and verify whether the performance objectives are met.



Developers should take the responsibility for unit-level, component-level & integration-level testing in an On Demand model.

Entry Criteria

Acceptance Testing is completed and application is stable.

Activity

- The critical application flows are identified here.
- Scripts are prepared for the flows identified.
- Application is tested for normal load and maximum load. Server and client metrics are captured.
- As per the results, the application is tuned by optimizing hardware/software configuration and test is re-executed.

The Three Models: Levels of Success

Iterative testing leads to constant awareness of the application performance and therefore more confidence in meeting release dates.

The three types of models - On Demand, On Retainer and Full Immersion, with varying levels of success are described below.

On Demand

This model can also be called as a Center of Excellence. The model is the functional equivalent of outsourcing to an internal team, but this is where most companies that are trying to integrate their performance testing into agile processes start. This is because the performance testing was actually an On Demand model before the Agile transition. Some of the steps to be handled for this model are mentioned below:

- You need to make use of On Demand services periodically.
- The performance targets, goals and objectives should be a standard module of each user story.
- Developers should take the responsibility for unit-level, component-level, integration-level testing and tuning.
- A full time team member has to take overall responsibility for managing performance related tasks.

On Retainer

On Retainer model can be used as an interim step between the On Demand and Full Immersion models. Some organizations may not have the required testers, performance testing environments and testing tools to support the Full Immersion model.

Here, each development project is assigned a specific performance tester. That performance tester is also assigned to two or more development projects. Though this model provides more performance testing expertise to individual project and more domain knowledge to the performance tester, the tester may fall short of becoming a fully integrated contributor to the team. As a result, the tester works independently, but provides guidance periodically. By following the same steps as listed in the above model and with the help of two additional steps as mentioned below, this model can provide more value.

- The performance tester should be available to the team at times that is critical to the performance evolution of the project.
- This model may require more performance testers or tools than On Demand model.

Performance testing in an agile environment allows you to manage the testing process in a highly flexible way.

Full Immersion

If the performance is really important to the value of the product and reputation of the company then this model should be the goal of every team in an Agile process. Here, full time team members are specialized in both delivering and testing performance and have responsibility for managing and coordinating performance related activities throughout the lifecycle of product development.

Risk Mitigation Benefits

Agile Performance Testing has positive outcome in risk mitigation with respect to:

Code Tuning: Unit level tuning is done early - so no need for code optimization at later stage on testing lifecycle.
Application Rejection: Reduces risk of the application being rejected because of issues such as memory leaks, hard coded values in code and database locking.
Earlier Detection: Reduces the effort and duration for performance tuning and retesting.
Release Dates: Iterative testing leads to constant awareness of the application performance and therefore more confidence in meeting release dates.
Performance Bugs: Less number of performance bugs in subsequent phases.
Performance Testing Efforts: Performance script can be re-used thus saving 60% of the efforts of performance cycle.

Conclusion

Hence, Performance testing is one of the single biggest catalysts to significant changes in architecture, code, hardware and environments. You can utilize this to your benefit by exploring the performance issues.

Performance testing in an agile project environment allows you to manage the testing in a highly flexible way. Iterative testing approach leads to better code that is optimized for performance. In particular, this approach allows you to revisit the project vision and reprioritize tasks based on the value they add to the performance test at a given point in time.