

ABSTRACT

This paper addresses the manifesto of Agile Testing and why software and testing companies should adopt it instead of standard software theory and testing procedures. The paper also talks about the issues in standard software testing processes and how agile testing helps overcome these through small cycle releases. It also discusses how agile testing increases the **rate of delivery** with smaller increments, helps improve product quality and above all reduces waste?

INTRODUCTION

“Agile Testing” is a term loaded with promise and freedom as well as fear and loath – most of all, there is uncertainty simply because it’s a new concept in the field of Information Technology.

Traditional development is based on predictable defined processes. However, to be predictable requires us to know everything up front. Getting there requires extensive amount of time to figure out all requirements, define work, estimate the work, plan out how the work will be accomplished while taking possible risks into consideration. Once the plan has been created, we try to control changes through rigid change control processes in order to avoid scope creep that will threaten the plan.

When viewed from a linear perspective, the approach starts by examining the software development project as a whole. This includes finding the reasons, why stakeholders have chosen to include performance testing in the project, and the value that performance testing is expected to add to the project. The results of this examination include the team’s view of the success criteria for the performance testing effort.

Once the success criteria are understood at a high level, an overall strategy is envisioned to guide the general approach to achieving those criteria by summarizing what performance testing activities are anticipated to add the most value at various points during the development life cycle. Those points may include key project deliveries, checkpoints, sprints, iterations, or weekly builds. Frequently, while the strategy is evolving, the performance specialist and/or the team will begin setting up a performance-test environment and a load-generation environment.

With a strategy in mind and the necessary environments in place, the test team draws up plans for major tests or tasks identified for imminent performance builds. When a performance build is delivered, the tasks should be executed in priority sequence (based on all currently available information), appropriately reporting, recording, revising, reprioritizing, adding, and removing tasks and improving the application and the overall plan as the work progresses.

The process can compare two or more devices or programs in terms of parameters such as speed, data transfer rate, bandwidth, throughput, efficiency or reliability.

Performance testing can also be used as a diagnostic aid in locating communication bottlenecks. Often a system will work much better if a problem is resolved at a single point or in a single component. Effective performance testing can quickly identify the nature or location of a software-related performance problem.

Performance Testing is not to “Find Bugs”

With the ever growing need for quick access to data, and with the large volumes of data at our disposal, the need for high performance systems and applications has become increasingly vital.

The goal of performance testing is not to find bugs, but to eliminate bottlenecks and establish a baseline for future regression testing. Ideally, the software under test is already stable enough so that this process can proceed smoothly.

Valuable time can be wasted due to poorly designed hardware layout or badly developed application. Slow data transfer rate may be inherent in hardware but can also result from software-related problems, such as:

- Multiple applications running at the same time
- A corrupted file in a Web browser
- A security exploit
- Heavy-handed antivirus software

Performance testing of an application under development aims to thoroughly address a multitude of performance affecting factors to result in an enhanced user experience when using an application.

Various techniques, depending on the application type, project duration, available performance testing tools and resources i.e., budget and time constraints are taken into consideration when devising the most suitable performance testing strategy.

Why Agile Testing

Agile testing takes a new avatar of standard software testing methodologies in the pursuit of working towards rapid software deliveries to customers. With the rapid advent of technologies clients expect their business applications to be super productive in order to harvest the investment returns within short turnaround time. The word “Agile” itself goes with the meaning “move quickly” and so does the testing. In agile testing, no conventional testing practices are applicable where one has to wait until the entire development cycle activities are completed. Instead the testing process is closely intact with the development and is done in parallel to as and when a piece of code is developed.

In the conventional development process which is based on phases, each phase goes through thorough and lengthy validation before triggering the next stage.

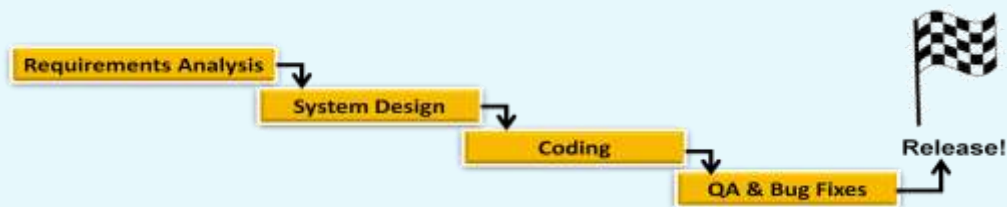


Figure 1

The concept illustrated in Figure 1 does is not faulty by definition. If we can accurately spec out the exact product all at once, then this is great. The problem is the other 99% of software projects which end up looking more like this.

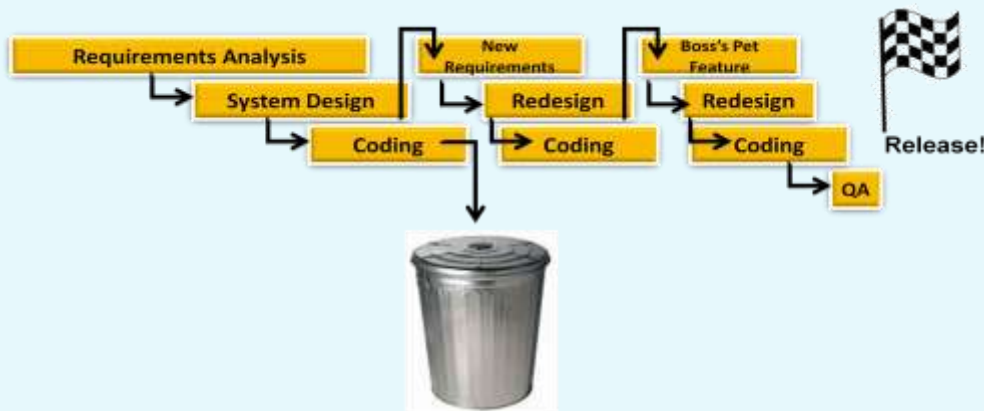


Figure 2

Agile testing does not emphasize rigidly defined testing procedures, but rather focuses on testing iteratively against newly developed code until quality is achieved from an end customer's perspective. In other words, the emphasis is shifted from "Testers as Quality Police" to something more like "entire project team working toward demonstrable quality."

At first, these Iterative methods went with small cycles, but remained with a fixed plan ahead of time: Fixed budget, strict plan, and predetermined feature sets – just instead of 1 major release, have many minor releases.

Agile takes this concept one step further: Iterative releases, with no fixed plan!

Each release adds a bit of value, allowing for a better viewpoint to decide what features to add next. Development works in small cycles, planning and designing only far enough ahead to keep the flow working and most importantly “Testing as part of the process, not the last step of the process”.

As agile software development process involves in “incremental” release, the testing should also be in incremental manner yet as quick as possible to test the code fragments as soon as they are ready, stable and take a shape from developer's unit-level or module testing. This is to showcase the customer a quality piece of the working product at regular short time intervals. So, agile testing is all about custom testing placing the customers' quality & delivery needs for the software in the first place rather than the software vendor's lengthy, regular testing lifecycle and the quality strategies.

Agile testing for increased customer satisfaction!

The agile testing drives the concept of “Test Driven Development” – testing analytics from the moment of procuring the requirement specifications document to develop a checklist, which covers basic nitty-gritty test scenarios to stabilize the primary functionality of a requirement, thereby helping the developers to eye on quality from the time of coding, keeping the code clean while covering the critical and fundamental test cases in their unit tests. With the test checklists already available even before the programming of the software starts, the testers are able to reuse them to easily manipulate and update the test cases for manual & automated testing, while managing the on-going requirement changes within the stipulated time frame.

Providing the space for reduction in resource utilization with re-useable test checklists and minimized manual testing by test automations using automation software testing tools and no need to compile heavy testing documentation, agile testing suits best to keep customers in loop & happy through incremental quality progress.

Agile Performance Testing over Core Performance Testing

Performance testing is a type of testing intended to determine the responsiveness, throughput, reliability, and/or scalability of a system under a given workload. Performance testing is commonly conducted to accomplish the following:

- Assess production readiness
- Evaluate against performance criteria
- Compare performance characteristics of multiple systems or system configurations
- Find the source of performance problems
- Support system tuning
- Find throughput levels

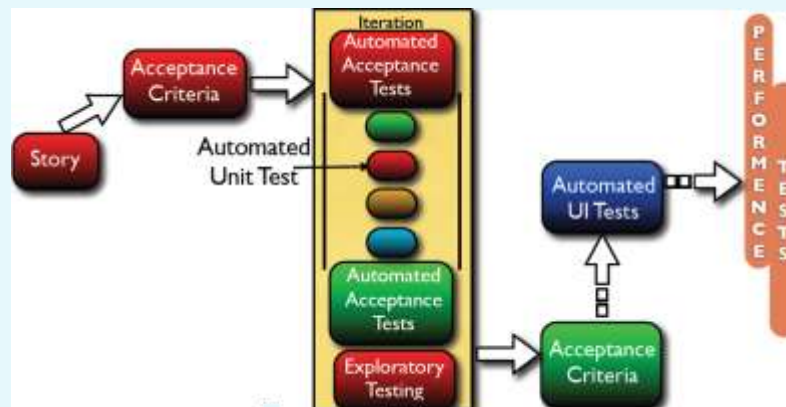


Figure 3

Agile testing is all about applying agile values and principles to testing. The value of agile testing lies in effective communication between developers, testers and the product owner. Testers write acceptance test cases and get them approved by the product owner. At the same time, coding is driven by the customer facing tests which is more likely to deliver better software. In the end a story is not complete until testing is finished. “Agile testing involves the set of good practices which help the agile team deliver high quality software”.

ENDING NOTES

Depending on the nature of the project, and the given times frames along with the availability of resources and test tools available, it is advisable to determine the most suitable performance testing strategy to adopt to successfully complete performance testing within time and budget constraints. The underlying philosophy in agile performance testing is to remain efficient by assigning different things to different teams whilst maintaining flexibility throughout the project.

The nature of performance testing makes it difficult to predict what type of test will add value, or even be possible. Yet, it holds an equally important place in the successful acceptance of an application which is robust, and able to deliver results. Thorough performance testing ensures that the application will deliver, without compromising quality and it is essential for every software to undergo thorough performance testing.

REFERENCES

1. Making Sense of UCD and Agile by Lane Halley 2006
2. Agile Overview-Embrace Uncertainty by Naresh and Bala
3. Agile Testing by Naresh Jain
4. Role of QA and Testing in Agile by Mayank Gupta of Global logic
5. <http://agiletesting.blogspot.com/2005/02/performance-vs-load-vs-stress-testing.html>