Grenze International Journal of Engineering and Technology, Special Issue

A New Paradigm on Reduction of Human Bias in Software Testing Life Cycle

Raghav Sridhar[1] and Dr.Vinod Vijayakumaran[2]
[1-2]BITS Pilani – WILP
Email: sridhar.raghav29@gmail.com, vinod.vijayakumaran@gmail.com

***Abstract*—This paper will aim to help software professionals reduce human bias introduced into the software testing life cycle, with a new model of incremental testing. There is a basic tendency that professionals working in the software industry have; identifying familiar and relatable patterns in the workplace's processes vis-à-vis the outside world. The solution proposed in this paper attempts to fit in best practices coupled with incremental cycles of arithmetic progression into the process of test case development, execution and reporting, while performing Agile Software Testing and the standard Software Testing Life Cycle.**

***Index Terms*— Agile Software Testing, Best Practices, Optimization, Software Testing Life Cycle (STLC), Bias.**

I. INTRODUCTION

Software Testing has grown over the years from rudimentary testing methods into well-defined testing life cycles. Over the years, many good-practices and safe testing methodologies have been recommended to be implemented into the daily routines of software development and testing. Some of these can be roughly explained as [1]:

a. Software needs to be compartmentalized, i.e., broken into segments to achieve a deeper level of testing. This enables us to look at the finer aspects of performance within the software.
b. A third eye, i.e., a person not familiar with the software is preferred to run the tests on the environment, as it could point to new avenues, which were previously unexplored.
c. Performing tests during the development process will enhance code quality in the long run, by aiding developers to fix the software before it even reaches live-systems.
d. Software testing should be performed in a controlled environment, to study the various metrics under various scenarios such as heavy stress, threat response, generic standard levels etc.

Once Agile concepts were introduced into the software testing life cycle, paucity of time became a very real issue; maintaining quality standards and keeping with the deadlines have seen a marked decrease in the following of good-practices of testing and quality assurance. Many novel methods are being introduced to keep in line with the high quality of code, similarly this paper aims to introduce a novel incremental software testing model, suited for medium to large teams. To plan and execute tests, software testers must consider the software and the function it computes, the inputs and how they can be combined, and the environment in which they will eventually operate [2]. During testing, a single participant's view need not always be the right approach to analyze a test scenario. This is proven when bugs are reported by a client after a product's

Grenze ID: 01.GIJET.4.3.26

release. This complacency in testing arises due to the short turnaround time available in the agile model, which in turn is caused by the inability of many teams to stick to certain quality standards. In this paper we propose a testing methodology that imbibes certain valuable software testing best practices such as, four eyes principle, breaking tests into small components, maximum test coverage, regression testing, customized testing etc. to increase the efficiency of testing in Agile, and to minimize the possibility of human bias.Broadly categorizing, Software Testing Process contains the following steps:

a. Strategizing and planning the testing life cycle
b. Designing the tests
c. Executing the tests
d. Recording the results
e. Test Closure

II. Current Approaches

A. Agile Software Testing

A software testing approach that follows the paradigm of agile software development is called agile software testing. Agile is an iterative development methodology, where requirements evolve through collaboration between the customer and self-organizing teams. Agile aligns developments with the customer's needs. Unlike the waterfall method, agile testing can begin testing at the start of the project with continuous integration between development and testing [3]. Agile testing is not sequential (i.e. it's not that it is carried out only after coding phase) but continuous. The Agile team works as a single team towards a common objective of achieving quality. Few of its principles:

a. Testing is not a phase
b. Testing moves the project forward
c. Everyone tests
d. Shortening Feedback Response Time
e. Clean code
f. Reduce test documentation

B. Software Testing Life Cycle

Software testing life cycle is a sequence of different activities performed by the testing team to ensure the quality of the software or the product. It starts as soon as requirements are defined or SRD (Software Requirements Document) is shared by stakeholders [3]. It provides a step-by-step process to ensure quality software. Software testing life cycle has the following different phases but it is not mandatory to follow all phases. Phases are dependent on the nature of the software or the product, time and resources allocated for the testing and the model of SDLC that is to be followed.

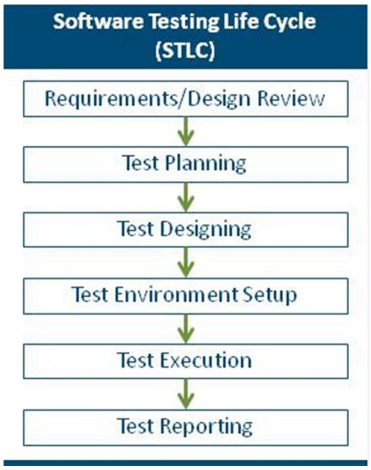

Fig1. Software Testing Life Cycle Phases [4]

As we see in Fig1. There are six major steps or phases in the software testing life cycle which are described below:

a) *Requirement Analysis:* When the SRD is ready and shared with the stakeholders, the testing team starts high-level analysis concerning the AUT(Application Under Test). During this phase, test team studies the requirements from a testing point of view to identify the testable requirements. The team may interact with various stakeholders to understand the requirements in detail. The requirements must be either functional or non – functional. Automation testing feasibility for the given AUT is also done in this phase.
b) *Test Planning:* The team plans the test strategy and approach. The test tool selection, test effort estimation, resource planning and determining roles and responsibilities etc. are performed in this phase. During this phase, the test team also try to identify the metrics, the method of gathering and tracking those metrics.
c) *Test Case Development:* The phase involves the creation, verification, and rework of test scripts. Test data is identified or created and is reviewed and then is reworked if needed.
d) *Environment Setup:* The test environment decided the software and hardware conditions under which a product is tested. Test environment set-up is one of the critical aspects of the testing process and can be done in parallel with test case development phase. The testing team may not be involved in this activity if the customer/development team provides the test environment in which case the test team is required to do a readiness check of the given environment.
e) *Test Execution:* During this phase, the testers carry out the testing based on the test plans and the test cases prepared. Bugs will be reported back to the development team for correction and retesting will be performed. This involves documenting the test results, log defects for failed cases and retest the defect bugs.
f) *Test Cycle Closure:* This is the last phase of the STLC where the testing team will meet, discuss and analyse testing artefacts to identify strategies that have to be implemented in future, taking lessons from the current test cycle. The idea is to remove the process bottlenecks for future test cycles and share the best practices for any similar projects in future.

III. ISSUES

The current method of application of testing has many underhand issues, i.e. ones that do not seem relevant at the beginning but may have a significant impact down the road on the working patterns, and the results garnered from performing the tests. Some major drawbacks of the current system of testing [5]:

a) Agile testing is defined by the time available in the sprint, making it a very short period for all the tests to be completed thoroughly, hence proving to be too expensive at times.
   Testing can become unstructured in Agile model, because of the changing requirements and the dynamic nature of the environment, it requires to be impartial.
b) Loss in focus with many test cases being handled at once by individual quality associates.
c) Applying continuous testing can be a challenge, as there might be specific periods with gaps while there is a lack in the understanding of the test scenarios and user stories, difference in feedback given to developers etc.

IV. PROPOSED APPROACH

As mentioned, there are many drawbacks to not following good practices. To better understand the proposed model, it is essential to have a basic understanding of the concept of an “Arithmetic Progression”. An arithmetic progression is a sequence of numbers such that the difference between successive terms is a constant value [7]. We shall use this concept to assign test cases in a sequential but incremental manner until every tester has covered all the test scenarios.

The roles of the participants are as follows:

Quality Lead (QL) – The person who designs and assigns the test cases and is generally considered the point of contact for all Quality related information.

Evaluator – The quality associate, or tester, whose purpose is to pick up test scenarios and perform the tests.

A series of steps have been formulated, which are as follows:

1) QL spends 1 productive day before the start of the sprint documenting and designing viable test scenarios for the upcoming sprint. The QL then creates a test assignment document (Fig.2), which will be updated before the start of every iteration of testing.

Test ID	Iteration Count	Owner	Evaluator

Fig.2 Test Assignment Document

2) The Evaluator is calculated by the following formula:
Evaluator Index = Relative Row Index of Owner + Iteration Count
For example, in the first iteration, in a group of ten, the evaluator of the 10^{th} test case = 0 + 1.

When the test life cycle begins, The QL marks the iteration as 1 and there can exist a maximum of {1 ... N} iterations for every test case; with N being the number of Evaluators - 1. For Example, in a group of ten

4) The Evaluator picks one test case and updates the Test Report Document (Fig.3), filling the columns marked with the asterisk
(*) before starting the test scenario, and the others after executing the test scenario.

Test ID *	Test Scenario *	Test Duration *	Expected Outcome *	Outcome / Remarks	Revalidation Flag (default: false)

Fig.3 Test Report Document

5) The Evaluator now executes the test case. In case the evaluator runs into any question or doubt, at any point during the evaluation, the revalidation flag is marked as true.
6) In the case of encountering a bug, the revalidation flag is marked as true, and the bug is forwarded to the dev. Team for any fix.
(This step will aid in eliminating any bias that the Owner of the test case would have introduced had they executed the test case.) Once the bug is fixed, any regression testing is performed by the owner of the test case, ensuring the removal of any bias introduced by the Evaluator.
7) If at this point the Owner faces a new dilemma pertaining to the test scenario, the revalidation flag is marked as true.
8) After an Evaluator completes a test case, the Test Assignment Document is visited to verify which test case is taken up by said Evaluator next.
9) Once all the test cases generated are complete, the iteration is incremented by 1.

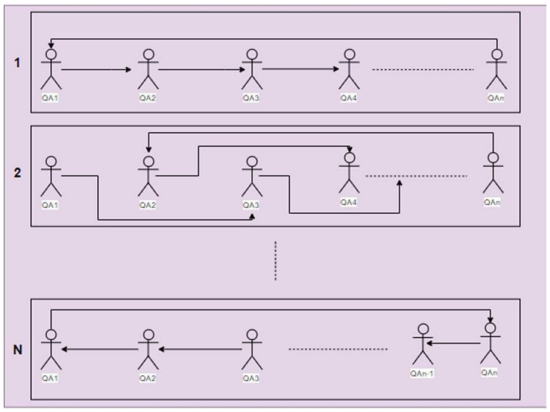

Fig.4 Bias Reduction Life Cycle

10) Finally, at the end of the Lifecycle process, the issues marked with the revalidation flag as true, are discussed and resolved as a team.

11) The QL's initial document is revisited, and all the tests completed are marked.
12) If newer test cases are created, steps 4 to 11 are performed until arrival at closure of the testing life cycle.

Fig.5 Iterative and Incremental Test Pattern [6]

V. CHALLENGES

a) Time-Consuming – The proposed model is very demanding on the group's time and requires a big commitment to collaboration on the duration of the project. The goal here is to deliver quality software rather than on quantity. Excellent sprint planning and Quality dedication is needed when using this model.
b) Mindset Challenge – Bringing in change always comes at a cost. Engineers find it difficult to switch onto new practices from the existing ones due to convenience.
c) Reduced Adoption – There is a downward trend in adoption of new methods to reduce Bias in software due to increasing deadline demands with short turn-around time to product development.
d) No comparative model – Even with the advancements in software testing, human bias is not quantifiable, and hence is unexplored territory in the realm. Most other software testing model deals only with absolutes and not with psychological aspects of software testing.

VI. CONCLUSION

Since Software quality is the key to the success of an agile development team's delivery and product's success, there are plenty of newer process models and strategies coming up as an aid to achieving this. This paper presented a new software testing process model which ensures the minimization of Human Bias, by many eyes reviewing the testing process.
This paper is not aimed at providing a mathematical proof of being better than other models , rather, to allow users to explore a more elaborate procedure to cover all the bases while testing.

REFERENCES

[1] DevOps and Testing ,Best Practices ,[Online] Available: https://devops.com/13-best-practices-successful-software-testing-projects/
[2] J. A. Whittaker "What is Software Testing? And Why Is It So Hard?" [Journal] IEEE Software pp. 70-79 2000
[3] Guru99, Agile Testing – a beginner's guide, [Online], Available: https://www.guru99.com/agile-testing-a-beginner-s-guide.html
[4] Software testing fundamentals, Software Testing Life Cycle, [Online], Available: http://softwaretestingfundamentals.com/software-testing-life-cycle/
[5] Software testing fundamentals, software testing life, [Online], Available: http://softwaretestingfundamentals.com/software-testing-life-cycle/
[6] Codoid, Challenges of agile testing, [Online], Available: https://codoid.com/challenges-of-agile-testing/
[7] Wolfram – Arithmetic Sequence, [Online], Available: http://mathworld.wolfram.com/ArithmeticProgression.html
[8] Ulf Eriksson, Iterative Approach, [Online], Available: https://reqtest.com/wp-content/uploads/2016/05/atlc2-300x224.png