FROM QUALITY ASSURANCE TO QUALITY ENGINEERING FOR DIGITAL TRANSFORMATION

Kiran Kumaar CNK

Capgemini India Private Limited, Inside Divyasree Techno Park, Kundalahalli, Brookefield, Bengaluru, Karnataka 560037, India

ABSTRACT

Defects are one of the seven prominent wastes in lean process that arises out of the failure of a product or functionality from meeting customer expectations. These defects, in turn, can cause rework and redeployment of that product or functionality again, which costs valuable time, effort, and money. As per the survey, most of the clients invest much time, energy, and money in fixing production defects.

This paper provides information about ways to move into quality engineering from quality assurance mode for digital transformation by diagnostic, Predictive & Prescriptive approaches, it also outlines the overall increase in quality observations, given QA shift left and continuous delivery through Agile with the integration of analytics and toolbox.

KEYWORDS

Diagnostic, Predictive & Prescriptive approaches, continuous delivery through Agile

1. INTRODUCTION

In this contemporary world, the quality of the product determines the furthering and sustaining of any software business. Currently, quality assurance (QA) plays a continuous and consistent role in improving the QA process in most of the software businesses to ensuring that quality of the product, i.e., is enhanced by reducing and eliminating defects as clients invest much time, effort and money in fixing defects. This process of quality assurance is achieved by applicationlevel testing, test automation, inward-focused and descriptive mode.

On the other hand, for the actual digital transformation, quality of the product must be concatenated and engineered rather than just only being assured. It is chieved by migrating from quality assurance to quality engineering. The framework of quality engineering comprises of customer-focused, diagnostic, predictive, prescriptive, Shift left & continuous delivery through agile.

Natarajan Meghanathan et al. (Eds): SIPP, BIGML, DaKM, SOEN, AISC - 2020
pp. 75-82, 2020. © CS & IT-CSCP 2020 DOI: 10.5121/csit.2020.100206

The main objective of implementation of quality engineering are as follows,

a. QA role in agility and DevOps

b. Integration of Analytics

c. Test driven development in agile mode

d. Structured Testing process & Adaptivity

2. QA ROLE IN 'AGILITY' AND 'DEVOPS'

Agile refers to an incremental and iterative approach that focuses on collaboration, continuous customer feedback, and small, rapid releases. DevOps is a concept of executing end to end engineering processes that focuses on constant testing and delivery, which brings in both development and operation team together.

In this environment of agility in combination with DevOps, the tester performs quality engineering and assurance in all stages of the Agile development phase as opposed to participating only at the end of the cycle. During Requirement Analysis & Grooming phase (Stage1), QA can synergize with Business Analysts to picture the client's viewpoint, to ensure quality at the requirement level, also to ensure that acceptance criteria have covered all validations points and to eradicate defect ambiguity profoundly. In the Planning & Estimation phase (stage 2), QA plays a vital role through effective communication & brainstorming that helps in arriving at accurate estimations and in the effective discharge of SCRUM events, including DEMO.

In the next Functional Analysis & Design phase (Stage 3), QA plays a pivotal role to prevent defects upfront, Which could be thoroughly established by keen story/ requirement analysis, understanding junit coverages from Dev team, Designing high-level test scenarios and providing walkthrough of the same to all stakeholders like BA, Dev, Architect, Product owner, etc., on a scheduled 'Test scenario walkthrough meet' and finally coming out with a master test plan document i.e., inclusion of unit, SIT and Regression test scenarios/cases based on the discussion which binds insights and inputs that brewed out in the 'Test scenario walkthrough meet'. This way, we ensure that all stakeholders are on the same page, removing discrepancies in approaches and eliminating requirement ambiguousness. Thus, filling up the gap in understanding in advance and effectively proceeding with a concrete plan.

In the implementation phase (stage 4), QA can effectively perform technical normalization by performing API / web service testing or smoke testing in Dev Env, to further minimize the proximity of defects. Finally, in the quality assurance phase (Stage 5), QA can engineer and assure the quality by incorporating different layers of testing like Build Verification Testing (BVT, without which QA do not proceed with SIT), full-fledged SIT, Regression (mostly automated) and smart testing like crowd testing. Refer to figure 1 for an overview of testing types and corresponding responsibilities. [1] [2]

Activities	Testing Types	DEV/QA
Unit testing	Unit testing is done by the developer in Local environment	DEV/QA
Smoke Testing	Smoke testing of developed system, by the developer, on the environment provided for testing post deployment	DEV/QA
Build Verification testing	To validate whether the basic functionalities are working as expected (if not, to reject the build and will not proceed to SIT)	QA
SIT	To carry out the complete Functional testing	QA
Regression	To validate that the existing functionalities are not impacted by the addition of new functionalities and defect fixes	QA/Dev

Figure 1. Testing types and Responsibilities

3. INTEGRATION OF ANALYTICS

In this context, analytics enables testers to evaluate the performance of the test outcome. It can be tracked with the help of various potential metrics and parameters involved in the process of a test engineering exercise. In other words, the health quotient of the software product that is being tested is well being measured. The final status of the test results provides us with a perfect picture of the state of functionalities of the tested software. Refer to figure 2 to have an overview of prominent test analytics [3]

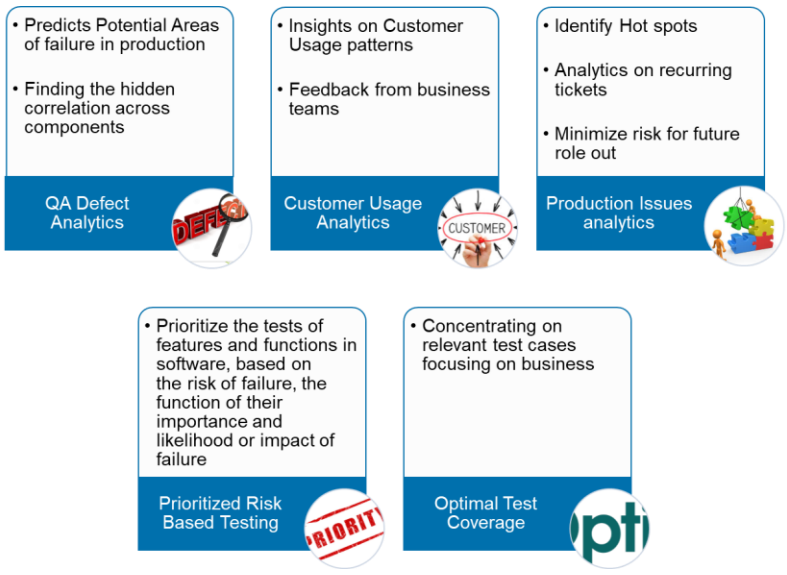


Figure 2. Prominent test analytics

4. TEST DRIVEN DEVELOPMENT IN AGILE MODE

Test-driven development (TDD) is one of the common practices of Agile core development. It is one of the engineering techniques for developing the software in collaboration where the code of programming design and its corresponding testing are executed in series of micro-iterations. It is an evolutionary approach that combines test first – Development next approach. The very purpose of this TDD is to focus on customer specification and not an end-phase validation primarily. Refactoring is also associated with this process, and it plays an important role in restructuring the code piece. Further, the tests should succeed to reduce complexity and enhance its understandability, maintainability, and clarity. Refer to figure 3 for an overview of the test-first development process. There are two levels of TDD. They are as follows, [4]

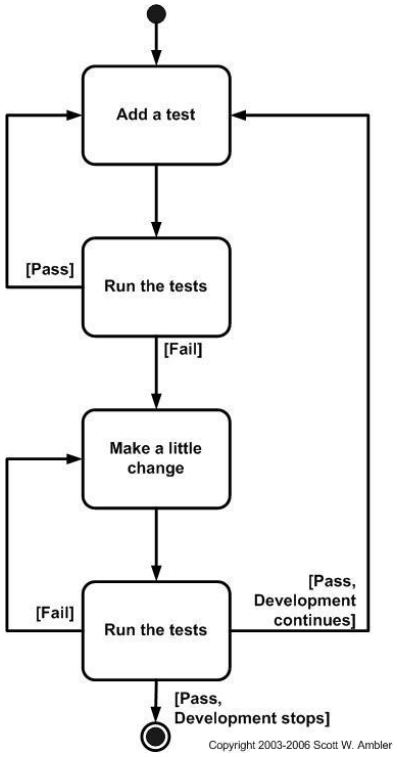


Figure 3. Test-first development process

4.1. Acceptance TDD (ATDD)

With ATDD you write a single acceptance test. This test fulfils the requirement of the specification or satisfies the behavior of the system. After that, write just enough production/functionality code to fulfil that acceptance test. The acceptance test focuses on the overall behavior of the system. ATDD also was known as Behavioural Driven Development (BDD).

4.2. Developer TDD

With Developer TDD you write a single developer test i.e., unit test and then just enough production code to fulfill that test. The unit test focuses on every small functionality of the system. Developer TDD is simply called as TDD.

The main goal of ATDD and TDD is to specify detailed, executable requirements for your solution on a just in time (JIT) basis. JIT means taking only those requirements into consideration that are needed in the system. So, increase efficiency and providing accurate coverage. Refer to figure 4 to know how acceptance TDD and development TDD work hand in hand.[4]

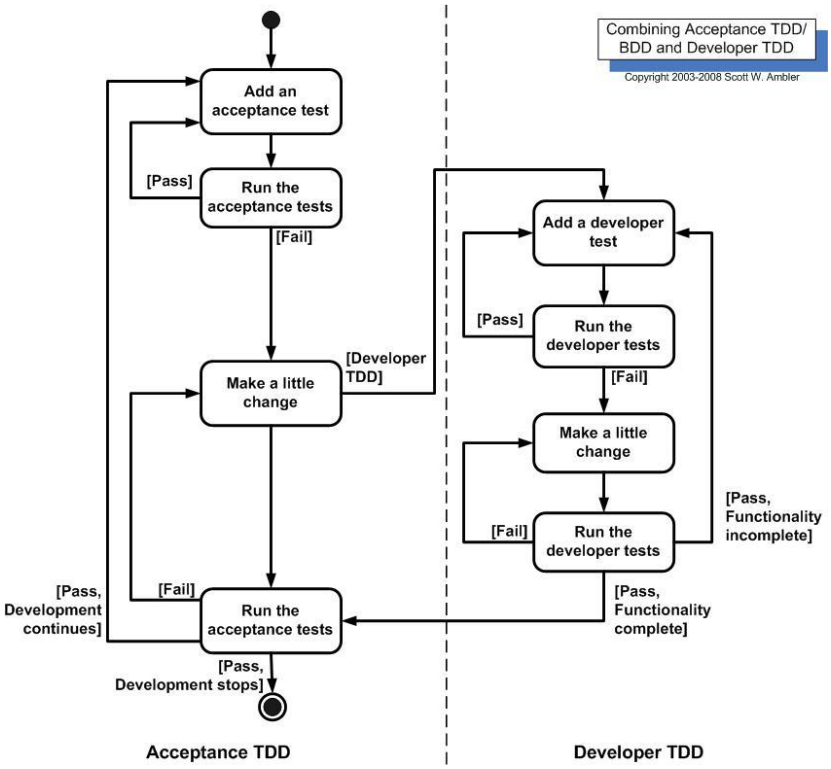


Figure 4. How acceptance TDD and developer TDD work together

5. STRUCTURED TESTING PROCESS & ADAPTIVITY

The structured testing process is one in which we would come up with a master test plan that comprises of test scenarios and test cases of unit testing, SIT and UAT managed together. The primary objective of designing a master testing plan document is to enable inhibition-free coordination and communication between different testing levels. By managing all testing activities on the same platform, we would be able to have a check on a few major aspects. First, it provides transparency between different phases of testing as we get to know what exactly covered as part of each testing phase. This way, we could carry out precise validation rather than exhaustive testing with redundancies. Second, we could eliminate any miss in scenario coverage

with a preconceived assumption. Finally, it further strengthens the core principle of agile. i.e., collaboration. Figure 5 illustrates the structured testing process.

Adaptivity is more of a combating process where testers from all phases pool together and learn from experience and response to changes instantly. It also ensures the QA team of different phases to stay on the same page with a clear understanding of in-scopes and out of scopes. This would eventually enable QA to provide strategic inputs on business functionalities on further development. Figure 6 illustrates the flow of adaptivity.

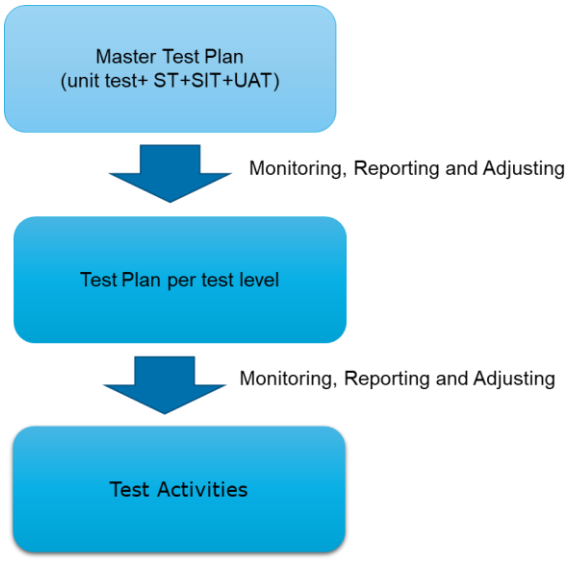


Figure 5. Structured testing Process

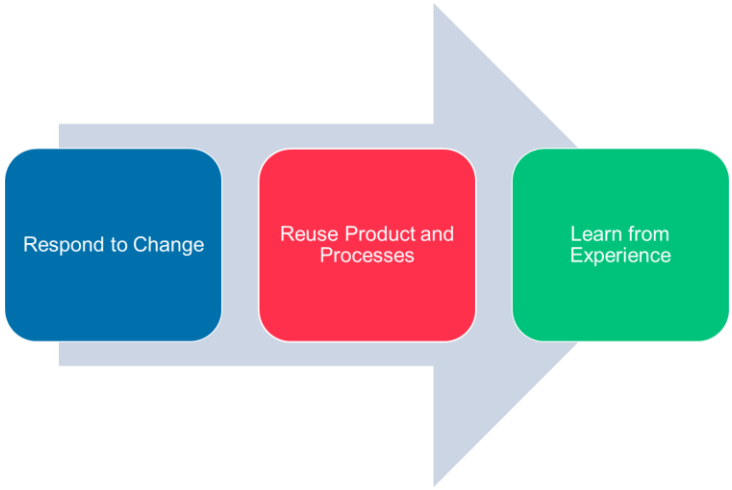
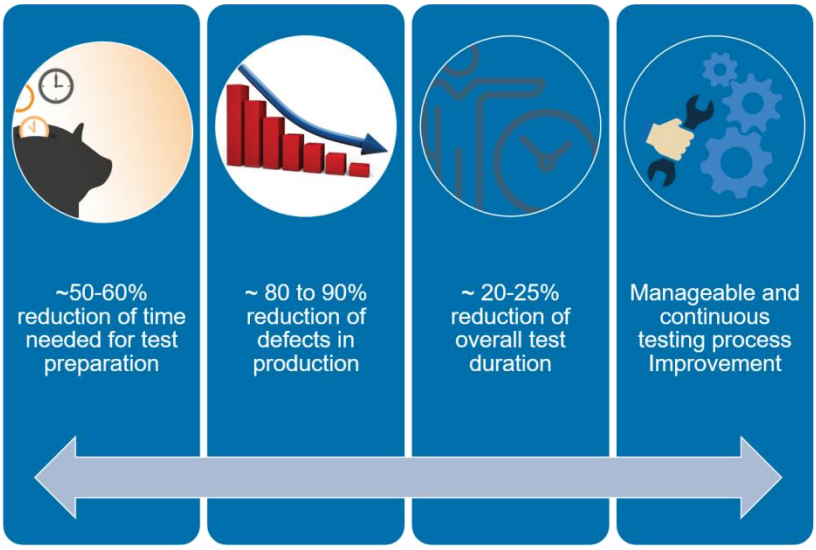


Figure 6 illustrates the flow of adaptivity

6. RESULTS FROM THE PILOT PHASE

Having run a pilot phase by implementing this approach across two digitally transformed projects of team capacity 17 and 9 each in the automotive domain, we have observed the below illustrated results. As it goes by, there is almost 50 – 60 % of the reduction of overall time spent in the test preparation phase that includes data settings, test scenario design, test case design, review and review comment incorporation. Next, to be keenly observed, there is almost an 80 to 90% reduction in the defect leakage in production environment due to systematic working in a preventive defect mode rather than defect deduction mode. Also, almost 20 – 25 % of the time-saving in the overall test duration. Finally, this approach persists in providing us with a channel to manage and improve the testing process periodically and dynamically.



7. CONCLUSIONS

Hereby I conclude that by employing the above suggested quality engineering framework, we can dramatically increase the quality of the deliverables. This model also strongly advocates for the saying '(Defect) Prevention is always better than cure.' This, on the other hand, helps in continuous testing process improvement and in testing advancements like automation, Robotic Automation process, etc. Finally, the tester gets transformed as 'Quality Engineer and assurer' by acquiring complete knowledge from functional, business, and technical spheres.

8. ACKNOWLEDGEMENTS

I would like to thank all my peers, managers, and mentors for their support, directly and indirectly, helping me to in making this article

9. REFERENCES

[1]Tridibesh Satpathy, (2017) A Comprehensive Guide to Deliver Projects using Scrum -Third Edition, SCRUMstudy™, a brand of VMEdu, Inc.

[2]Saliya Sajith Samarawickrama, (2018) "Continuous scrum: a framework to enhance scrum with Devops, 2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer)IEEE

[3]Harsh Vardhan (2019), "Applying Data Analytics to Test Automation", Stickyminds, TechWell Corp.

[4]Max Guernsey Iii, (2013) Test-Driven Database Development: Unlocking Agility-1st Edition, Net Objectives Lean-Agile Series, Addison-Wesley Professional

AUTHORS

Kiran Kumaar CNK works as a Senior Agile Quality Engineer with Capgemini India Pvt Limited. Currently, he is working on an onsite assignment at Capgemini, Netherlands. He holds B. Tech & MBA (Executive program in project management). He possesses over 10 years of experience in IT industry, predominantly in Insurance, automotive and public sectors. He is a certified Scrum Master and certified in ISTQB (advanced level) as well. Unequivocally, he possesses good presentation and communication skills that allowed him, from offshore, efficiently collaborate with different onshore clients and stakeholders across countries like US, Germany, Poland, Morocco, France, UK & Netherlands.