Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

Design Thinking and its Concernment to Agile Testing: Emergence and Techniques

Parthasarathy P D [1],
[1.] Associate Developer,
SAP Labs Pvt. Ltd.,
Whitefield, Bangalore, India

Dr. Vinod Vijayakumaran [2]
[2.]Product Owner,
SAP Labs Pvt Ltd,
Whitefield, Bangalore, India

***Abstract* — Agile Software Testing (AST) has established its dominance in the software industry and is believed to be the most promising software testing approaches. Agile Software Testing(AST) follows the manifesto of agile methodology and follows the highest techniques and best practices of Software Testing Life Cycle (STLC). Fierce competition and the dynamic business environment have tempted the software industry to adopt agile as well as lean approaches in global software development(GSD) projects. Corporates are also adopting several newer quality standards in order to enhance their development methodologies. Although there is extensive research work carried out in this domain, the majority of the research discusses the theoretical aspects of quality standards. When these aspects are applied in the real-time projects, the results are not remarkable. The various reasons for this are lack of experience, lack of resources for testing, lack of domain knowledge, misinterpretation of the requirement, schedule slippage, improper testing methodology and so on. Meanwhile, Design Thinking (DT) has evolved as the key methodology that provides a solution-based approach to solving problems. It's extremely useful in tackling complex problems that are ill-defined, unknown or multidimensional. In this paper, we have proposed a software testing process model based on AST, STLC, and DT to bridge the gap between the theoretical aspects of quality standards and the practical aspects of quality standards. The proposed approach can act as a guideline to global software development practitioners for effective software development and ensure high-quality standards.**

Keywords* — *Agile Software Testing (AST), Software Testing Life Cycle (STLC), Design Thinking (DT).

I. INTRODUCTION

A business solution is a success only when it can solve the *exact* problem in an intuitive manner. In the agile world of rapidly changing requirements, there is a lot of lack of clarity on the problem that we are trying to solve. In the dynamic business environment, neither the development team nor the customer is exactly aware of the complete problem or big picture in hand. Thus, the product is developed and delivered in iterations. So, instead of one major release, many minor releases would be scheduled. Software testing is very important to ensure the quality of the product. Products with high-quality help in customer retention and happy end-users. To ensure high-quality various new standards and testing methodologies are being adopted in the industry. AST has been widely accepted in the industry and is believed to be the most promising software testing methodology. In the recent past, there has been extensive research in the areas of agile testing, software testing methodologies, automation, TDD and so on. But, a majority of these methodologies have not been remarkable when they are implemented in a live project. In 2016-2017, there are plenty of software failures which could have been prevented by effective testing. A widely-cited study [1] reports that inadequate testing methods and tools annually cost the U.S. economy between $22.2 and $59.5 billion. The same study notes that between 25 and 90 percent of software development budgets are often spent on testing yet satisfactory quality standards are not met in most cases. Hence, this manuscript attempts to bridge the gap between the theoretical aspects of testing with the practical ones by proposing a new software testing process model. This proposed model can act as a reference for software practitioners.

II. INDEX TERMS ELABORATED

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 [2]. 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. Agile testing has shorter time frames called iterations (say 1 to 5 weeks). Few Principles of Agile software testing are:

a) Testing is NOT a Phase: Agile team tests continuously and continuous testing is the only way to ensure continuous progress. Unlike in waterfall model where testing is considered as a phase, here it is not a phase but an ongoing continuous activity.

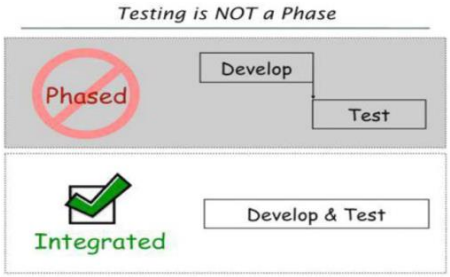

Fig1. Traditional approach VS Agile Testing approach [3]

b) Testing Moves the project Forward: When following conventional methods like the waterfall, testing is considered as the quality gate but agile testing provide feedback on an ongoing basis to make the product meet the business demands.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

c) Everyone Tests: In conventional SDLC, only test team tests while in agile including developers and QA's test the application.

d) Shortening Feedback Response Time: In conventional SDLC, only during the acceptance testing, the Business team will get to know the outcome of the development efforts, while in agile for each and every iteration, they are involved and continuous feedback shortens the feedback response time and cost involved in fixing is also less.

e) Clean Code: Raised defects are fixed within the same iteration and thereby keeping the code clean.

f) Reduce Test Documentation: Instead of very lengthy documentation, agile testers use a reusable checklist, focus on the essence of the test rather than the incidental details.

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 is shared by stakeholders [4]. 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.

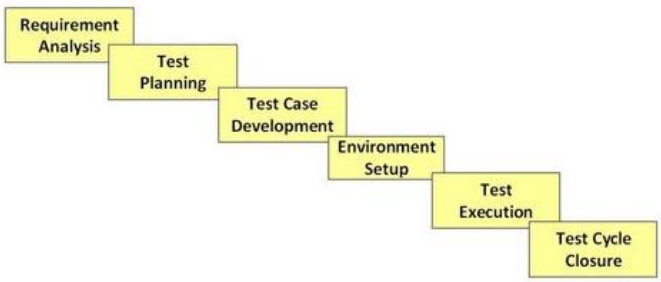

Fig2. Software Testing Life Cycle Phases [5].

As we see in Fig2. 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. 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 analyze testing artifacts 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.

C. Design Thinking

Design thinking means creating Innovation by combining diverse people, creative space, and an iterative approach. In a more formal way, Design thinking is a design methodology that provides a solution-based approach to solving problems. It's extremely helpful in tackling complex problems that are ill-defined or unknown, by understanding the human needs involved, by re-framing the problem in a human-centric way, by creating many ideas in a brainstorming session, and by adopting a hands-on approach in prototyping and testing [6].

In a nutshell, we like to think of this as "Humanizing IT". Meaning, in rather than focusing on systems, specs, or features, we want to show what's possible when starting with a focus on the end user and using a methodology called *Design thinking* to create innovative solutions. Design thinking is considered as the most promising process to solve any problem, it is not just confined to software but can be applied across industries and lines of business. It is proved that DT can also be applied to our life! [7].

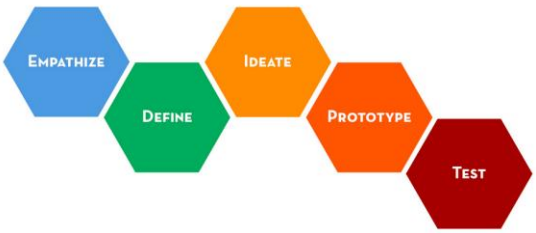

Fig 3. The five stages of Design Thinking [8].

As shown in Fig 3. There are 5 stages in the DT process.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

a) Empathize: Empathy is the foundation of a human-centered design process where we observe and engage with end users and immerse ourselves to uncover their real needs. In this stage, we look for issues that the end users may or may not be aware of. The three ways to empathize are to Immerse, Observe and Engage.

- *Immerse:* Become the end user and actually live their experience.
- *Observe:* Watching people is always fun but observing is about seeing user's actions and hypothesizing why they are acting a certain way.
- *Engage:* Getting out and talking to the end users is probably the most uncomfortable but potentially the most effective way to empathize if done right.

b) Define: During this stage, the information that was created and gathered during the empathize stage is put together. The team then analyzes the observations and synthesize them in order to define the core problems that the end user is facing. The problem statement is then re-framed in a human-centered manner. The most common technique is to find the core problem and articulating the problem statement by inserting information about the end user, their needs and our insights in the following template sentence[9]:

[End User . . . (descriptive)] needs [need . . . (verb)] because [insight. . . (compelling)].

c) Ideate: Ideate is the mode of our design thinking process in which we aim to generate radical design alternatives. Mentally it represents a process of "going wide" in terms of concepts and outcomes – it is a mode of "flaring" rather than "focus". In this stage, steps and ideas beyond obvious solutions and wild ideas are encouraged. The DT team creates a fluency (volume) and flexibility(variety) in the innovation options. The DT team get the obvious solutions out of their heads and think differently. The mission in this stage is to explore wild ideas while staying in the topic.

d) Prototype: Prototyping is converting the ideas and explorations from our mind into the physical world. A prototype is anything that takes a physical form – it can be a wall of post-it notes, a role-playing activity, space, an object, a model, an interface, a POC, or even a storyboard. This stage involves picking up the most feasible and innovative idea from the ideated list of ideas and converting the idea into the physical world. This step emphasis in failing quickly and cheaply i.e if the team is not able to convert an idea into a prototype they can fail early at this stage and re-iterate this stage with another idea rather than failing in the end.

e) Test: This phase gives the team to get feedback on the solutions that have been prototyped. The team aligns with the end users again and showcase their prototyped solution. They refine the solutions to make them better and continue to learn about the users. The strategy widely followed here are: *Prototype as if you know you're right, but test as if you know you're wrong.* The goal of this phase is to validate the prototype with the end-user, refining the solution and learn more about the end user.

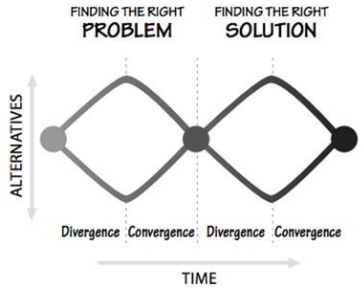

Fig 4. The Double-Diamond Model for DT [10]

As shown in Fig 4. The Double Diamond model and DT go hand in hand:

- The Empathize stage of DT → Divergence [in identifying the real problem]
- The Define stage of DT → Convergence [in framing the core problem]
- The Ideate stage of DT → Divergence [in brainstorming ideas]
- The Prototyping stage of DT → Convergence [narrowing down to feasible innovative ideas].

III. CURRENT APPROACH AND ISSUES

a) Current Approach

The current approach followed in the industry to enhance their quality standards is to adopt AST and STLC which are described in the previous section.

b) Issues

Though agile testing has been promising, it is still not the best testing process model as it is not well suited for ill-defined, unknown or multidimensional problems. Few major issues with the agile testing are the lack of constant involvement of end user in testing activities, no innovate ideas, no human-centric approach. Also, though AST emphasizes on parallel testing activities alongside development, it ignores the act of finding the real needs of the end user. AST starts with the requirements and goes about till the delivery, in parallel with the development phases but there is always a chance that the delivered backlog was not the real requirement of the end user. Such cases are not caught via STLC or AST and the team does not fail early in this case which is not the ideal case. Also, in the Agile world of rapidly changing requirements, there is a huge increase in such cases where there is a mismatch between the solution delivered and the real solution that the end user expects. Both STLC and AST act on SRD or the requirements gathered from the PO and stakeholders and do interact with the end user. Hence, there is dire need of a testing process model which can solve these pitfalls.

IV. PROPOSED APPROACH

As we saw in the previous section, there are pitfalls in the software testing process models which needs to be addressed as to achieve high-quality standards. Our proposed software testing process model combines the AST, STLC, and DT by embedding DT *stages* in STLC *phases* and following the resulting steps *iteratively*.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

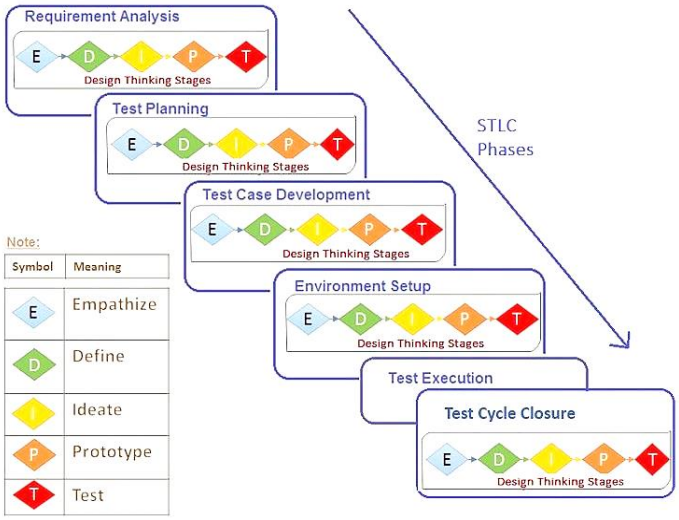

Fig 5. Proposed Software Testing Process Model

As shown in Fig 5, the five stages of DT are embedded in the STLC. Let's examine how DT is used in each phase: [E stands for Empathize, D for Define, I for Ideate, P for prototype, and T for Test in the below section]

Phase 1 - Requirement Analysis with DT:

- E → The AST team can empathize with the end users to understand their real issues via observing them, engaging them and making them use the existing solution and noting down the real pain points and on.
- D → The AST team after the initial stage can come up with the core problem statement in the template format as discussed in the previous sections.
- I → The AST team should ideate on all possible existing solutions, unobvious and innovative solutions for the requirement posed by the end user.
- P → The AST team could prototype the most innovative and feasible solution and create a paper prototype or wall post-it notes for the same.
- T → The AST team must validate their prototype with the end users and the developers for correctness and feasibility respectively. If the end user is not satisfied with the prototype solution it clearly indicated that the team has not understood the requirement can re-iterate the 5 stages of DT to get the requirement analyzed right.

Hence, there is progress in the STLC Phase unless the DT in a phase is complete and satisfactory.

Phase 2 - Test Planning with DT:

- E → The AST team must liaise and empathize with the QA who will be performing the quality activities for this requirement (who acts as an end user in this STLC phase) and understand their real needs in terms of the tools that are needed to test this requirement, domain knowledge needed, effort required and so on.
- D → The AST team can then create the problem the concrete problem statement, an example statement could be like: "QA and developers need a performance test mechanism because this requirement is an improvement backlog focusing on optimization and performance".
- I → The AST team needs to ideate on all possible ways to solve the problem defined in the previous stage. Going by the same example, the team would brainstorm on ways for performance testing, explore on automation and performance tools.
- P → The team can finalize on the approach and can prototype the same using a POC or a model. For our example say, they can finalize on a tool and try exploring more about it, judge the effort needed to use this tool etc.
- T → The team can again get back to validation where the QA for this requirement and the team decide on test strategies, effort management and see how this tool fits in for the testing of this requirement.

Phase 3 – Test Case Development with DT:

- E → The team gets immersed and engaged along with the QA on creating the test cases for the requirement. Potential problems in test case development are identified along with the outcome first 2 STLC phases.
- D → The team must come up with a core problem statement for test case development of this requirement. For our example by following the template, say we might have "Testing folks need thorough negative test cases for this requirement because there is a compelling need for the same in performance testing".
- I → The team brainstorms and comes up with as many as possible innovative test cases along multiple dimensions of the requirement.
- P → The team could use paper prototyping or post-it notes and POC to prototype the test cases found in ideation.
- T → The team would validate the prototyped test cases with the QA, technical stakeholders and the client side technical spokesperson. Any missed requirement or test scenario can be caught in this phase and can be refined. If needed the DT stages can be re-iterated for this STLC stage.

Phase 4 – Environment Setup with DT:

- E → The AST team can observe the systems landscape and environment setup of the customer and engage in discussions with them on the same. The team would understand their pain points and the exact scenarios where this requirement would be fulfilled and executed and if the end users can provide an environment for testing purposes.
- D → The team can define their core problem statement at this stage. An example could be "Testing folks need a system with 16GB RAM and 1024MB Heap memory because the real environment of the customer is similar to these specifications"
- I → The team could brainstorm ways to solve this issue. They can think out of the box and diverge their thoughts. In our example, it might be simulating a system, setting up a Virtual machine or a real one and so on.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

- P → The team could decide on the best feasible ideated solution and proceed ahead with a POC or an interface.
- T → The team must validate their prototyped solution if this satisfies their need. They can ensure that the prototyped solution is as close to the customer's environment. If the solution is not suitable for testing this requirement or not agreed by the customer, the team could redo the DT stages for this STLC phase or get a test environment from the customer.

Phase 5 – Test Execution:
In this STLC phase, there are no DT stages involved as by this phase the team is 100% sure that they are solving the right problem and they have also validated they are solving the problem in the right way via DT in test planning, test case development, and Environment setup. This phase involves just the execution of the test cases finalized in Phase 3 in the environment finalized in Phase 4. If there are bugs or undesirable outcome, as usual, they are documented and shared with the concerned developer and retested in the same conditions.

Phase 6 – Test Closure with DT:
- E → The team understands the real need of the end user at this stage for this requirement which in this case is to see the test results and also perform a round of testing to verify correctness and get the feedback of the whole test process for this requirement.
- D → The team should come up with the template problem statement which may be something similar to "Team needs to arrange for beta testing and lessons learnt meeting because customer feedback is vital".
- I → The team does the brainstorming session to come up with beta testing plan and way to conduct the lessons learnt meeting and ways to present the lessons learnt document.
- P → The team decides on the plan for beta testing, mode of meeting and the presentation way and prepare the prototype which in this case is the slide deck for lessons learnt meeting, pushing the test scripts for beta testing and so on.
- T → The beta testing is conducted and the lessons learnt meeting is steered to mark the closure of the STLC of this requirement.

This integrated and systematic testing process model ensures that there is no gap in the requirement and the delivery thus ensuring high-quality standards.

V. CHALLENGES

a) DT Mindset – As the proposed model heavily uses the DT principles, there needs to be mindset change from systems, specs, delivery to end-user needs and satisfaction. As DT involves a lot of interaction and multiple disciplinary teams, there needs to be a human-centric thought process, and end-user focused mindset to achieve success.

b) Time-Consuming – The proposed model is very demanding on the developer'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 is needed when using this model.

c) More End-User involvement – The proposed model involves end-user involvement in almost every stage and needs good commitment and support from the end users. This commitment is very engaging, rewarding and ensures delivery of right product but if the end users work in silos and do not support in the process, the proposed model does not work well.

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 embeds DT stages in the STLC phases and recommends to perform the phases iteratively. This manuscript is aimed at guiding and forming a hub for all agile practitioners in achieving high-quality standards in their product delivery.

VII. ABBREVIATIONS

- AST – Agile Software Testing
- STLC – Software Testing Life Cycle
- DT – Design Thinking
- GSD – Global Software Development
- TDD – Test Driven Development
- QA – Quality Associate
- AUT – Application Under Test
- SRD – Software Reference Document
- POC – Proof of Concept
- PO – Product Owner

REFERENCES

[1] NIST Govt Report, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, [Online], Available: https://www.nist.gov/sites/default/files/documents/director/planning/report02-3.pdf

[2] Guru99, *Agile Testing Guide*, [Online], Available: https://www.guru99.com/agile-testing-a-beginner-s-guide.html

[3] Mohd Hamid, "*Agile Testing*", International Journal of Scientific and Research Publications, Volume 5, Issue 9, September 2015.

[4] Softwaretestingfundamentals, *Software Testing Life Cycle*, [Online], Available: http://softwaretestingfundamentals.com/software-testing-life-cycle/

[5] Guru99, *STLC – Software Testing Life Cycle*, [Online], Available: https://www.guru99.com/software-testing-life-cycle.html

[6] Interaction Design Foundation, *What is DT and Why is it so popular*, [Online], Available: https://www.interaction-design.org/literature/article/what-is-design-thinking-and-why-is-it-so-popular

[7] Carlye Lauff, The Jornal Blog, "*Applying Design Thinking to your life*", [Online], Available: https://blog.usejournal.com/applying-design-thinking-to-your-life-4ff87047b332

[8] David Terrar, Enterprise Irregulars, *"What is Design thinking"*, [Online], Available: https://www.enterpriseirregulars.com/125085/what-is-design-thinking/

[9] Rikke Dam and Teo Siang, "Stage 2 in the Design Thinking Process: Define the problem and interpret the results", Interaction Design Foundation.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

[10] Don Norman, The Design of Everyday things, "*Design Thinking – The Double Diamond model for design*", Published by Basic Books, Perseus Books Group, 2013 – P 220

[11] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 7th Ed., McGraw Hill International Edition,2017

[12] Ian Sommerville, *Software Engineering*, 9th ed., Pearson, 2014

[13] Paul C Jorgensen, "*Software Testing: A Craftsman's Approach*", CRC Press, 2013, 4th Edition.

[14] Aditya P Mathur, *Foundations of Software Testing*, 2nd Edition, Pearson, 2014,p.223-300

[15] BK Madhu, Megha Jigalur, V Lokesh. "*A study on Agile Software Testing: Emergence and techniques*", African Journal of Mathematics and Computer Science Research Vol. 3(11), pp 288-289, Nov 2010.

[16] Joey Aquino, "A Crash course in Stanford's Design Thinking?", [Online], Available: https://joeyaquino.wordpress.com/2012/05/23