Test Case Quality: Issues and Limitations in Agile Software Development

Samera Obaid Barraood[1,2], Haslina Mohd.[1] and Fauziah Baharom[1]

[1]*Universiti Utara Malaysia, Malaysia, {samera_obaid@ahsgs.uum.edu.my, haslina@uum.edu.my, fauziah@uum.edu.my}*

[2]*Hadhramout University, Hadhramout, Yemen, {sammorahobaid@gmail.com}*

ABSTRACT

In the Agile software development environment, continuous changes in user requirements lead to an increase in the importance of a testing process to demonstrate a quality product. As test case is a cornerstone of the testing process, it is important to emphasize the high-quality construction of the test cases. Hence, testing process should be adequately planned and evaluating the quality of test cases can help to explain some important issues associated with software testing. However, findings from literature and the critical analysis of empirical studies revealed that less academic research has investigated the test case quality in Agile software development process. Therefore, with a specific reference to scrum methodology, the purpose of this paper is to identify the problems of test case quality in Agile software development by reviewing the existing work concerning testing quality in Agile. This paper has made a useful contribution by illustrating and clarifying the shortcomings of test case quality in agile projects and pointing out the factors that help to improve it.

Keywords: Agile software development, Agile testing, Scrum, test case quality.

I INTRODUCTION

The nature of test case construction is to obtain the necessary software coverage under testing (Tran et al., 2019; Yamaura, 1998). It is important to get good test cases that have high chance to expose unknown defects at low cost, ability to increase performance and robust to meet the users' requirements (Gómez et al., 2016; Kamde et al., 2006). This effort will consequently result in producing quality software (Adlemo et al., 2018; Tran et al., 2019). Assessing the quality of test cases is necessarily important in order understand how much testing is required and where potential testing attempts should be carried out (Ahmed et al., 2016). The test case quality (TCQ) is therefore capable of evaluating the quality of a software system (Pfaller et al., 2008). A quality test case is referred to a test case that has a high chance of revealing defects in a minimum effort, providing more detailed results, increasing system performance at a lower cost, and has a high chance of detecting unknown defects (i.e. the higher the quality of a test case, the more the potential to detect failures) (Gómez et al., 2016; Kamde et al., 2006).

The continuous changes in Agile software development methods requires many efforts to be performed on testing activities (Beck, 2003; Humble & Farley, 2010). The efficiency of testing activities depends largely on the TCQ, which directly defines the quality of testing (Causevic et al., 2012; Lai, 2017b). Although TCQ appears to be an effective solution for exposing software defects, in Agile methodology it still has some issues and problems that need to be studied and addressed. Therefore, it is important to identify the shortcomings of TCQ in the Agile testing process, to understand what the underlying issues are, and to identify the potential solutions suggested that require further investigation.

Hence, this paper begins with discussion on the concepts in TCQ and highlights the importance of this concepts in assuring software quality. The second section introduces the Agile software development process. Section 3 is the overview of Agile testing, and testing activities in scrum. Section 4 discusses some current and previous identified issues and limitations of TCQ in Agile software development and proposed solutions while section 5 concludes the paper.

II AGILE SOFTWARE DEVELOPMENT

Agile software development has become a preferred method for developing software with an increased adoption by companies worldwide to meet software complexity and evolving user demands (Matharu et al., 2015; Penmetsa, 2017). Agile software development methodology is a process for workable software which basically divide an entire project into manageable small sizes that can be separately handled for time items change risks and time control (Rajasekhar & Shafi, 2014). Unlike in the traditional development paradigm, Agile process do not have separate coding and testing phases (Gil et al., 2016). The term mostly used in Agile process is iteration. There are cross-functional teams that typically work in all development areas, that is, design, coding and testing in each iteration (Crispin & Gregory, 2009; Javed et al., 2019).

Several iterations, each one the same length, may be needed to deliver an entire theme or epic (Crispin & Gregory, 2009). Customers provide feedbacks to the system, in the form of stories, for a developed and tested iteration. The stories of the customers only stopped when the levels of functionalities required are delivered (Olausson et al., 2013). A new feature may require multiple iterations. Every iteration has to be fully integrated and carefully tested as a final production release (Penmetsa, 2017). Since the iterations make the software development effective and efficient to meet requirements of the customer and contribute in the success of the project, it also make the development process a little more complicated and time-consuming (Javed et al., 2019). The reason for this complication is that each iteration in Agile software development contained many activities (Javed et al., 2019). Short iterations in Agile software development implies that there must be an efficient testing process to avoid too much time being spent in the iteration for test preparation rather than on the actual tests running (Olausson et al., 2013).

A number of Agile methods has been reported in the software domain literature which have been adopted by the software community because of advantages which include focus on quick software delivery, changing requirements and customer satisfaction (Kayes et al., 2016). Notable amongst these methods are Extreme Programming (XP) (Beck, 1999), Scrum (Schwaber & Beedle, 2002), Feature Driven Development (FDD) (Palmer & Felsing, 2001), Adaptive Software Development (Highsmith, 2013), Crystal methods (Cockburn, 2004), Agile Unified Process (Ambler, 2005), and Dynamic Software Development Method (DSDM) (Stapleton, 1997), however Scrum is the most commonly adopted method in Agile software development (Aamir & Khan, 2017; Kayes et al., 2016). As reported in the 14th State of Agile (StateOfAgile, 2020) that the most organizations adopt Scrum (58%) and when calculate this percentage with the hybrid methodologies that include Scrum, it becomes 85% of organizations use Scrum. Therefore, it is important to show the scrum process and activities as example to show the development process and activities in Agile software development methods.

Scrum is one of the Agile software development iterative and incremental methods. It has been developed for managing the systems development process. It is an empirical approach applying the ideas of industrial process control theory to systems development resulting in an approach that reintroduces the ideas of flexibility, adaptability and productivity (Schwaber & Beedle, 2002). It proposes continuous adaptation of the project planning, using cycles called sprints, where each sprint is a time-boxed lasting for between one to four weeks as well as each sprint produces a new version of the product with new features (Gil et al., 2016). The unique features of Scrum according to (2015) are collaboration, daily meetings, product backlog, sprint backlog, and roles. The Scrum team should have skills in designing, developing, testing, and documenting the product (Anwer et al., 2017). See Figure. 1 illustrates the scrum process from (Javed et al., 2019).

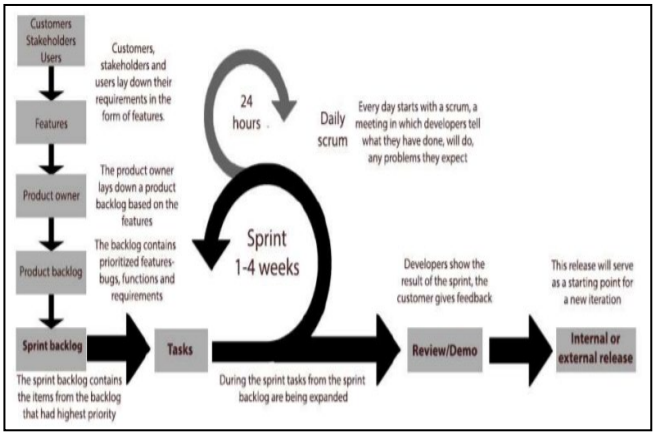

Figure 1. Development Process in Scrum.

The process and standards of scrum process are properly followed by the organizations (Tahir, 2019). And due to the people are involved in the development process, who are product owner, scrum master, developers, and quality assurance engineers (testers) (Abrahamsson et al., 2017; Kayes et al., 2016). Sprints are planned by selecting items from a product backlog, estimating the effort needed to complete each item selected for the sprint, competition, product quality, and available resources (Anwer et al., 2017). During sprints, the team groups up every day for 15 minutes or less for a daily scrum meeting, where the status of the tasks is tracked and they take the corrective action for any speed interruption (Anwer et al., 2017; Matharu et al., 2015). In this meeting, team members tell what they did yesterday, what they would be doing tomorrow and the blocks and obstacles they would face (Anwer et al., 2017; Matharu et al., 2015).

III AGILE SOFTWARE TESTING

Brian Marick provides a philosophy of Agile testing as "a style of testing, one with lessened reliance on documentation, increased acceptance of change, and the notion that a project is an ongoing conversation about quality" (Leffingwell, 2010). Rajasekhar and Shafi highlighted the aim of testing in both Agile and traditional method is same, but the difference is the team constituent, where the testers in Agile are required to give quality infusion support through the entire team (Rajasekhar & Shafi, 2014). The early feedback from testing is the good thing for testers in Agile projects as this helps developers to identify the

issues at an early development stage (Tripathi & Goyal, 2014).

In Agile software development, the whole team is responsible for the quality, every one of the team also can write test cases not only the testers (Laing & Greaves, 2016). This helps the team comprehend that testing is an activity all of them need to be involved in (Laing & Greaves, 2016). Agile testing has five differences with traditional testing (Laing & Greaves, 2016), as represented visually in testing manifesto form in Figure 2.

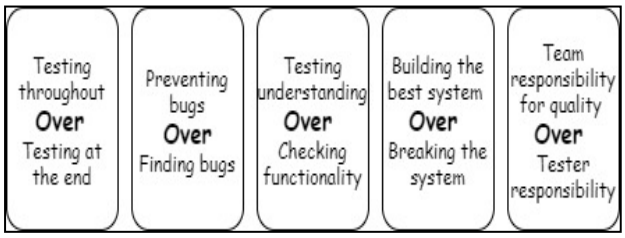

Figure 2. Testing manifesto

In Agile software development, testing starts at the project beginning and it is recognized as an integral part of software development with coding (Khan et al., 2016). The testing is done in each iteration, after user stories are prioritized and selected which tasks start to be achieved, it is immediately tested and released to the customer and when all tasks are developed and delivered, all tasks are integrated and tested (Rajasekaran & Dinakaran, 2015). Software testing is vital part of any project which can be designated as a component of quality assurance. Testing process has more value for demonstrating quality product in Agile environment (Harichandan et al., 2014).

Compliance to checklist and requirement documents is not a strict obligation in Agile testing. The goal is simply to comply with basic necessities for completing the requests of the customer (Penmetsa, 2017). The continuous changes of requirements of customer increase the importance of software development and testing practices in Agile software development methods (Penmetsa, 2017). Agile testing enables the organization to be nimble about uncertain priorities and requirements (Penmetsa, 2017). The need for large numbers of tests is magnified in Agile software development practices, that require extensive testing to be performed (Beck, 2003; Gay et al., 2016). Hence, the lack of testing resources leads to poor quality (Chomal & Saini, 2014; Rajkumar & Alagarsamy, 2013).

The testing tasks in Agile methods should be prepared properly to cater for continuous changes in the requirements (Yu, 2018). In order to nimbly test a software system during Agile software development, it is crucial to identify what to test (e.g., requirements) and how to test it (i.e., test cases) (Olausson et al., 2013). The requirements are normally discussed by the developers and testers in order to identify the acceptance criteria test cases that need to be designed (Penmetsa, 2017). The requirements in Agile are described as user stories, which is formulated as one or two sentences in the everyday language of the user (Crispin & Gregory, 2009; Lai, 2017a). Each story is written on a small 3 by 5 inches paper note card to guarantee that it is not too lengthy (Crispin & Gregory, 2009). A well written user story will describe what the desired functionality is, who use it for, and why it is useful (Lai, 2017a). Correct, complete and consistent user story contents can help generate good test cases (Lai, 2017a).

Agile testers are responsible to plan and estimate user stories in a product backlog and specifying acceptance criteria to create test cases for each user story before they can be considered for inclusion in an iteration (Black, 2017; Kayes et al., 2016). Therefore, at the start of all testing processes, it is significant to document and execute test cases. The corresponding relationship of test cases and user stories should be one-to-many, which implies that, a single user story may be attributed to multiple test cases. The complexity of the user story may therefore increase the number of test cases (Aamir & Khan, 2017). Olausson et al. (2013) provides an example of user story, related acceptance criteria and test cases, as illustrated in Figure 3.

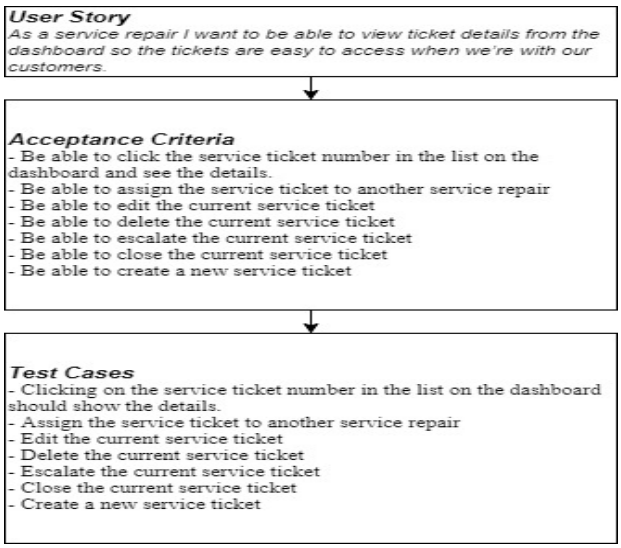

Figure 3. A user story, acceptance criteria and test cases

In Agile software development, test case quality is concept highly regarded as an important testing activity where it is one of the quality features that directly define the quality of testing that can lead to quality software and rapid delivery (Causevic, Punnekkat, et al., 2012; Lai, 2017b). The test cases in Agile must be developed as the requirements evolve

(Lewis, 2009). The continuous change of requirements and projects long duration calls for changing as well as increasing the test cases (Beer & Felderer, 2018). As a part of the Agile process, the incremental and iterative development process of test assets have to be properly handled (Olausson et al., 2013).

Initially, any new feature is not known very well for the Agile team, so typically it needs to run test cases on all defined acceptance criteria. When the feature has been completed, it should be tested according to the created test cases and that it works as expected. After that, there is a need only to run tests for requirement changes validation, which means how to know the tests to run are necessary for the team. Speeding up the process of testing to keep up with the short iterations is another aspect of the Agile story worth of note (Olausson et al., 2013). Testing activities in Agile will be explained in this paper through Scrum testing activities as follows.

Every cycle (sprint) in scrum affects testing. The testing starts during initial stage of a sprint (Kayes et al., 2016). Where, unclear requirements are clarified, system test cases are written, and test data are prepared by the tester with the product owner (Kayes et al., 2016), because the importance of testing in Scrum, Kayes emphasizes role of a tester in Scrum process. To show clearly the activities of the testing in Scrum, the tester role in Scrum will be explained.

The testers focus on ensuring the deliverables quality. Their role is started from the beginning of sprint to reduce the cost of the requirement and design errors (Kayes et al., 2016). The role of testers is more toward guarantee of product quality and not only for writing and run test cases (Kayes et al., 2016). Testers are more integrated to the development team (Harichandan et al., 2014). Itkonen et al. (2005) say that Agile software development can be benefited through a team of professional testers. Most quality assurance and quality control activities are skipped in Scrum because of the absence of a dedicated quality assurance team and its short cycles (Aamir & Khan, 2017). This short duration of sprint leads to Scrum team does not take quality into consideration as well as a developer cannot write a bug-free code when working under pressure (i.e., short duration of a sprint) (Aamir & Khan, 2017).

The activities of Scrum testing are illustrated in Figure 4. In the initial sprint phase, the tester writes test cases, prepares test data, clarifies requirements and coveys updated requirements to development team and ensures the environment of the test (Kayes et al., 2016). In the sprint period, he writes checklist which is a brief version of a test case. The checklist is written for sprint backlog, which is a set of prioritized items from product backlog. The checklist is executed via developers after the sprint backlog item is developed. The checklist assists to detect bugs early. The tester also assists the developers to write unit test cases and ensure the reviews of the code are done on time (Kayes et al., 2016).

During the sprint halfway, the tester shows to the product owner a sneak peek of the product, which is a demonstration of what has been done until that period. Finally, the developed features are testing under the tester responsibility based on the test cases which he writes. The tester deploys the sprint deliverables to a test server for regression testing when it became developed. The tester prepares test plan and run test cases in test server. When regression testing is completed, the developers perform the smoke test and the tester verifies the release (Kayes et al., 2016).

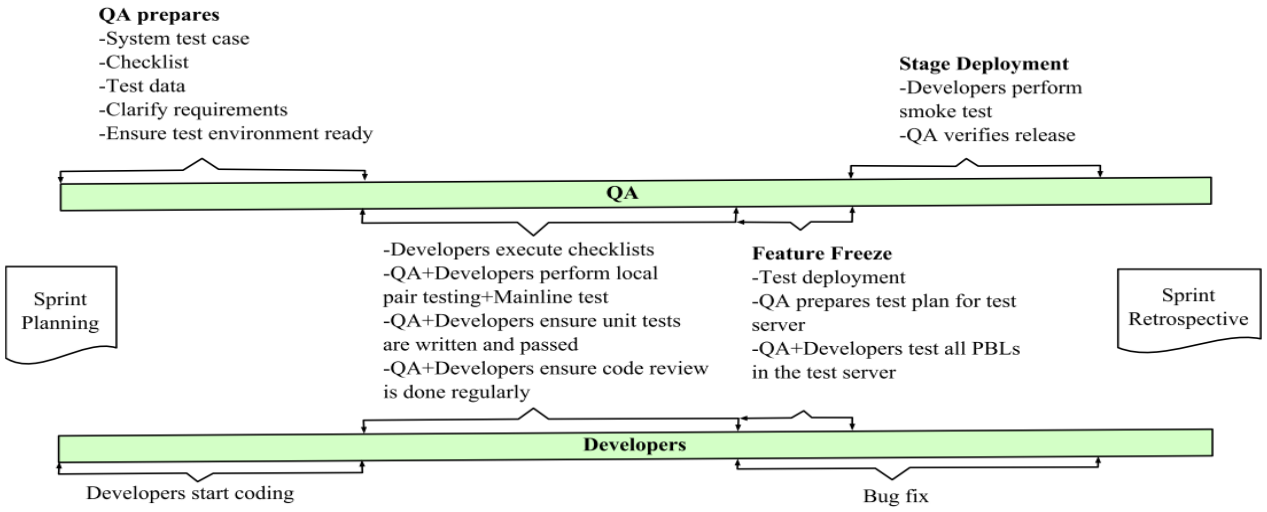

Figure 4. Scrum testing activities

IV ISSUES AND LIMITATIONS OF TEST CASE QUALITY

A test case represents the test instructions for a tester. It contains a set of conditions or variables under which a tester will determine whether a system satisfies the requirements or works properly (Bilal et al., 2017; Olausson et al., 2013). In Agile methodology, test cases are written by tester in an initial Sprint phase. He illustrates and clarifies the requirements for each user story just some days or hours before beginning the actual coding (Kayes et al., 2016).

The good-enough testing of software should have sufficient assessment of quality at a reasonable cost (Goeschl et al., 2010), and the good quality of test cases are very important for assuring the quality of software (Tran et al., 2019). Unfortunately, writing good test cases is one of the most difficult and time consuming testing activities (Serra et al., 2019). In writing test cases, it is notable for ensuring that testing could achieve a certain level of thoroughness (Romli et al., 2020). Missing test target, procedures, or expected result lead to reduce the quality of test cases (Jovanovikj et al., 2018).

The test cases help testers to find out problems in the requirements or in the design of software system (Kayes et al., 2016). The requirements in Agile projects are not sufficiently elaborated (Kārkliņa & Pirta, 2018). It can be inconsistent, incomplete and incorrect (Lai, 2015). In addition, both user stories and acceptance criteria, are not usually defined properly and Agile team does not emphasize on the quality standards which makes test cases difficult to be derived (Fischbach et al., 2020; Padmini et al., 2018). According to Uikey (2012), the test cases are never written upfront with the requirements or user stories. Again, there is a lack in traceability between test cases and related acceptance criteria (Fischbach et al., 2020). In addition, unsystematically acceptance tests cause excessive or incomplete test cases (Fischbach et al., 2020).

Although works have been identified in previous studies on Agile testing, however, efforts geared specifically on TCQ are very meager. Investigation into the result of the existing Agile TCQ models reveals a number of gaps that are still required to be filled, as majority of studies related to TCQ focused on traditional methods. In addition, there is misalignment in defining the TCQ among academy, industry, and practitioners (Tran et al., 2019). defining TCQ from practitioners' perspective is still lacking in empirical studies. Some studies (Adlemo et al., 2018; Bowes et al., 2017; Jovanovikj et al., 2018; Kamde et al., 2006; Kochhar et al., 2019; Tran et al., 2019) focused on the TCQ and most of them identified the factors based on practitioners' perspective. Unfortunately, these studies are conducted on traditional development methods. Where, the traditional development does not support requirement changing, not fast delivery, not iterative, and not incremental. Hence, these models can be difficult to apply for the current practices in Agile methods. Rajasekaran (2015) stated that the Agile team sometimes does regression testing repetitively without a clue on when to stop a particular sprint and deliver. Moreover, they reported that Agile methods (like Scrum) faced many testing issues such as inconsistent and inadequate unit testing, the huge and quickly changing in requirements.

On the aspect of continuous changes of requirements, Agile methodology was adopted by many companies nowadays (StateOfAgile, 2018). The requirement changes lead to changes in user stories, which lead to changes in the testing scope (Padmini et al., 2018). Changes of user stories lead to changes in test cases, and this consequently wasting a lot of time and resources (Beer & Felderer, 2018; Padmini et al., 2018). To address the issues of Agile testing, a number of researchers have undertaken many approaches to increase the quality of Agile testing.

For example, Shrivastava and Jain (2010) proposed automated test case for unit testing (ATCUT). This study specifically focused on the testability of test cases and its effects when applied in TDD as well as ATCUT design metrics are not sufficient to measure TCQ which is designed for unit testing, which has less bug finding effectiveness (25% to 30%) as compared to System testing (85%). Thus it may cause some problems related to the software quality (Rajasekaran & Dinakaran, 2015). Kayes et al. (2016) also proposed a metric called Product Backlog Rating (PBR) to measure and monitor the testing process in Scrum, but this metric need further evaluation and it is focused on the testing process not on the TCQ. Aamir and Khan (2017) proposed an enhanced quality-focused model of scrum via performing start-of-the-art testing activities in Scrum in which they account for a test backlog to sustain test cases and to deliver quality work. However, this study focused on the quality of product backlog to enhance the quality of product without referring to the quality of test cases which are used to catch the defects.

Fischbach et al. (2020) identified 16 quality factors for six Agile test artifacts. However, they focused on the Agile test artefacts in general and they proposed only one quality factor of unit test cases is code coverage which is not enough to measure the quality of test cases. Unudulmaz and Kalıpsız (2020) and Harichandan et al. (2014) proposed models to improve the Scrum process but they are not focused on TCQ. Causevic et al. (2012) conducted an

experiment to investigate the TCQ in TDD and traditional test approach by using three criteria which are not enough to measure TCQ in Agile software development as well as they used students as subjects in their experiment who do not have enough experience in this field.

In sum, these studies did not clearly address the issues of the quality of test cases in Agile software development. On the other hand, the study that clearly focused on TCQ in Agile is Lai (2017b). Lai proposed a test case quality measurement (TCQM) model based on four quality factors of TCQ which are qualified document, manageable quality, maintainable quality, and reusable quality. Even though, TCQM model is able to define the TCQ but there are some critical factors for the quality test cases which are still missing. For instance, complete and precise requirements which are crucial for writing effective test cases (Ahmad et al., 2019; Fischbach et al., 2020).

There have been studies focused on the efficiency of test cases (Adlemo et al., 2018; Kochhar et al., 2019; Shrivastava & Jain, 2010; Tran et al., 2019), effectiveness (Adlemo et al., 2018; Kochhar et al., 2019; Tran et al., 2019), readability (Adlemo et al., 2018; Bowes et al., 2017; Grano et al., 2018; Kochhar et al., 2019). Further, test cases should be repeatable to be high quality (Adlemo et al., 2018; Kamde et al., 2006). Also test case should be self-contained (Adlemo et al., 2018; Bowes et al., 2017; Kochhar et al., 2019), and understandable (Bowes et al., 2017; Jovanovikj et al., 2018; Kochhar et al., 2019; Shrivastava & Jain, 2010; Tran et al., 2019).

Other limitations of Lai's TCQM model such as, it did not define the factors based on practitioners' perspective, nevertheless, it is very important to define the quality of test cases (Tran et al., 2019). In addition, Lai's TCQ model adapted Linear Combination Model (LCM), which does not define the measurement goal. Defining the measurement goals are important to clearly guide the practitioners in organizations to derive metrics for each factors (Fenton & Bieman, 2015).

In sum, majority studies utilize traditional development approaches but are unable to tackle the challenges and limitations of balancing the quality of software and rapid delivery. A second frequently encountered problem by prior studies is that there is no clear description how to measure and access the quality factors. Therefore, first, it is a need for further research on effective test cases quality based on very clear requirements and organizational goals. Secondly, since Agile software development is placing more emphasis on organizational goal and human expertise more research is needed to look into a wider or organizational artifact, that may strengthen the future findings.

V CONCLUSION

Software testing is a very important activity in Agile methods. A great influence on the testing process is writing and managing of test cases. It is crucial for Agile teams to understand the drawbacks of test cases, as this helps to write successful test cases in a short time. Therefore, this study concentrated on identifying the issues of test case quality in the Agile environment.

The test cases quality in Agile methods face some challenges such as the test cases are not written based on the requirements, limited requirements coverage, unclear requirements, less experience of the team members to write test cases, and missing some critical quality criteria that improve it such as requirement quality, tester experience, test case readability, understandability, specific, performance efficiency, independence, repeatability, and accuracy. These issues being clearly identified in this paper will serve as the basis for developing our test case quality measurement model that will aid in assessing quality test cases in Agile projects.

REFERENCES

Aamir, M., & Khan, M. N. A. (2017). Incorporating quality control activities in scrum in relation to the concept of test backlog. *Sādhanā*, *42*(7), 1051–1061. https://doi.org/10.1007/s12046-017-0688-7

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). Agile software development methods: Review and analysis. *ArXiv Preprint ArXiv:1709.08439*.

Adlemo, A., Tan, H., & Tarasov, V. (2018). Test case quality as perceived in Sweden. *5th International Workshop on Requirements Engineering and Testing (RET'18)*, 9–12. https://doi.org/https://doi.org/10.1145/3195538.3195541

Ahmad, A., Leifler, O., & Sandahl, K. (2019). Empirical Analysis of Factors and their Effect on Test Flakiness-Practitioners' Perceptions. *ArXiv Preprint ArXiv:1906.00673*.

Ahmed, I., Gopinath, R., Brindescu, C., Groce, A., & Jensen, C. (2016). Can testedness be effectively measured? *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 547–558.

Ambler, S. (2005). The agile unified process (aup). *Ambysoft, Http://Www. Agilealliance. Hu/Materials/Books/SWA-AUP. Pdf, Última Visita*, *14*.

Anwer, F., Aftab, S., Shah, S., & Waheed, U. (2017). Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum. *International Journal of Computer Science and Telecommunications*, *8*(2), 1–7.

Beck, K. (1999). Embracing change with extreme programming. *Computer*, *10*, 70–77.

Beck, K. (2003). *Test-driven development: by example*. Addison-Wesley Professional.

Beer, A., & Felderer, M. (2018). Measuring and Improving Testability of System Requirements in an Industrial Context by Applying the Goal Question Metric Approach. *5th International Workshop on Requirements Engineering and Testing Measuring*, 25–32.

Bilal, M., Sarwar, N., & Saeed, M. S. (2017). A hybrid test case model for medium scale web based applications. *2016 6th International Conference on Innovative Computing Technology, INTECH 2016*, 632–637. https://doi.org/10.1109/INTECH.2016.7845115

Black, R. (2017). *Agile Testing Foundations An ISTQB Foundation*

Level Agile Tester Guide. BCS Learning & Development Ltd.

Bowes, D., Hall, T., Petrić, J., Shippey, T., & Turhan, B. (2017). How Good Are My Tests? In *International Workshop on Emerging Trends in Software Metrics, WETSoM* (pp. 9–14). IEEE. https://doi.org/10.1109/WETSoM.2017.2

Causevic, A., Punnekkat, S., & Sundmark, D. (2012). Quality of Test Design in Test Driven Development. *2012 Eighth International Conference on the Quality of Information and Communications Technology*, 224. http://urn.kb.se/resolve?urn=urn:nbn:se:mdh:diva-18773

Causevic, A., Sundmark, D., & Punnekkat, S. (2012). Test Case Quality in Test Driven Development : A Study Design and a Pilot Experiment. *The EASE 2012*, 223–227.

Chomal, V. S., & Saini, J. R. (2014). Cataloguing most severe causes that lead software projects to fail. *International Journal on Recent and Innovation Trends in Computing and Communication*, 1143–1147.

Cockburn, A. (2004). *Crystal clear: a human-powered methodology for small teams*. Pearson Education.

Crispin, L., & Gregory, J. (2009). *Agile testing: a practical guide for testers and agile teams* (1st ed.). Pearson Education, Inc.

Fenton, N., & Bieman, J. (2015). *Software Metrics A Rigorous and Practical Approach Third Edition* (Third Edit). Taylor & Francis Group, LLC.

Fischbach, J., Femmer, H., Mendez, D., Fucci, D., & Vogelsang, A. (2020). What Makes Agile Test Artifacts Useful? An Activity-Based Quality Model from a Practitioners' Perspective. *ESEM '20*. http://arxiv.org/abs/2009.01722

Gay, G., Rajan, A., Staats, M., Whalen, M., & Heimdahl, M. P. E. (2016). The effect of program and model structure on the effectiveness of mc/dc test adequacy coverage. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *25*(3), 1–34.

Gil, C., Diaz, J., Orozco, M., de la Hoz, A., de la Hoz, E., & Morales, R. (2016). Agile testing practices in software quality: State of the art review. *Journal of Theoretical and Applied Information Technology*, *92*(1), 28–36.

Goeschl, S., Herp, M., & Wais, C. (2010). When agile meets OO testing: a case study. *Proceedings of the 1st Workshop on Testing Object-Oriented Systems*, 10.

Gómez, O. S., Monte, B., & Monte, B. (2016). Impact of CS programs on the quality of test cases generation : An empirical study Categories and Subject Descriptors. *ICSE '16 Companion*. doi: http://dx.doi.org/10.1145/2889160.2889190

Grano, G., Scalabrino, S., Gall, H. C., & Oliveto, R. (2018). An empirical investigation on the readability of manual and generated test cases. *Proceedings of the 26th Conference on Program Comprehension*, 348–351.

Harichandan, Ss., Panda, N., & Acharya, A. A. (2014). Scrum Testing With Backlog Management in Agile Development Environment. *International Journal of Computer Science and Engineering*, *2*(3), 187–192. http://www.ijcseonline.org/pub_paper/38-IJCSE-00144.pdf

Highsmith, J. (2013). *Adaptive software development: a collaborative approach to managing complex systems*. Addison-Wesley. http://www.dorsethouse.com/books/asd.html

Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)*. Pearson Education.

Hynninen, T., Kasurinen, J., Knutas, A., & Taipale, O. (2018). Software testing: Survey of the industry practices. *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1449–1454. https://doi.org/10.23919/MIPRO.2018.8400261

Itkonen, J., Rautiainen, K., & Lassenius, C. (2005). Towards understanding quality assurance in agile software development. *ICAM 2005*.

Javed, K., Khan, A. H., & Tubbassum, L. (2019). Critical Analysis of Software Development Methodologies based on Project Risk Management. *INTERNATIONAL JOURNAL OF ACADEMIC RESEARCH IN BUSINESS AND SOCIAL SCIENCES*, *9*(12).

Jovanovikj, I., Narasimhan, V., Engels, G., & Sauer, S. (2018). Context-specific Quality Evaluation of Test Cases. *MODELSWARD*, 594–601.

Kamde, P. M., Nandavadekar, V. D., & Pawar, R. G. (2006). Value of test cases in software testing. *Management of Innovation and Technology, 2006 IEEE International Conference On*, *2*, 668–672. https://doi.org/10.1109/ICMIT.2006.262303

Kārkliņa, K., & Pirta, R. (2018). Quality metrics in Agile Software Development Projects. *Information Technology & Management Science (RTU Publishing House)*, *21*.

Kayes, I., Sarker, M., & Chakareski, J. (2016). Product Backlog Rating : A Case Study On Measuring Test Quality In Scrum. *Innovations in Systems and Software Engineering*, *12*(4), 303–317.

Khan, R., Srivastava, A. K., & Pandey, D. (2016). Agile approach for Software Testing process. *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*, 3–6. https://doi.org/10.1109/SYSMART.2016.7894479

Kochhar, P. S., Xia, X., & Lo, D. (2019). Practitioners' Views on Good Software Testing Practices. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 61–70. https://doi.org/10.1109/ICSE-SEIP.2019.00015

Lai, S.-T. (2015). A Maintainability Enhancement Procedure for Reducing Agile Software Development Risk. *International Journal of Software Engineering & Applications*, *6*, 29–40.

Lai, S.-T. (2017a). A User Story Quality Measurement Model for Reducing Agile Software Development Risk. *International Journal of Software Engineering & Applications*, *8*(2), 75–86.

Lai, S.-T. (2017b). Test Case Quality Management Procedure for Enhancing the Efficiency of IID Continuous Testing. *Journal of Software*, *12*(10), 794–806. https://doi.org/10.17706/jsw.12.10.794-806

Laing, S., & Greaves, K. (2016). *Growing Agile : A Coach ' s Guide to Agile Testing*. Growing Agile.

Leffingwell, D. (2010). *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional.

Lewis, W. E. (2009). *Software Testing and Continuous Quality Improvement Third Edithion*. Taylor & Francis Group, LLC.

Matharu, G. S., Mishra, A., Singh, H., & Upadhyay, P. (2015). Empirical study of agile software development methodologies: A comparative analysis. *ACM SIGSOFT Software Engineering Notes*, *40*(1), 1–6.

Olausson, M., Rossbreg, J., Ehn, J., & Sköld, M. (2013). Pro team foundation service. In *Apress*. Apress.

Padmini, K. V. J., Kankanamge, P. S., Bandara, H. M. N. D., & Perera, G. (2018). Challenges Faced by Agile Testers: A Case Study. *2018 Moratuwa Engineering Research Conference (MERCon)*, 431–436.

Palmer, S. R., & Felsing, M. (2001). *A practical guide to feature-driven development*. Pearson Education.

Paruch, L., Stray, V., & Blindheim, C. B. (2020). Characteristic traits of Software Testers. *Evaluation and Assessment in Software Engineering (EASE 2020)*. https://doi.org/https://doi.org/10.1145/3383219.3383270

Penmetsa, J. R. (2017). Agile testing. In *Trends in Software Testing* (pp. 19–33). Springer Science+Business Media. https://doi.org/10.1007/978-981-10-1415-2

Pettichord, B. (2000). Testers and Developers Think Differently Observations on understanding and utilizing the divergent traits of key players on the software team. *Software Testing and Quality Engineering*, *2*, 42–47.

Pfaller, C., Wagner, S., Universit, T., & Wiemann, M. (2008). Multi-Dimensional Measures for Test Case Quality. *Software Testing Verification and Validation Workshop, 2008. ICSTW'08. IEEE International Conference On*, 364–368.

Rajasekaran, V. A., & Dinakaran, M. (2015). Issues in Scrum Agile Development Principles and Practices in Software Development. *Indian Journal of Science and Technology*, *8*(35). http://www.indjst.org/index.php/indjst/article/view/79037/67402

Rajasekhar, P., & Shafi, R. M. (2014). Agile Software Development and Testing: Approach and Challenges in Advanced Distributed Systems. *Global Journal of Computer Science and Technology*, *14*(1).

Rajkumar, G., & Alagarsamy, D. K. (2013). The Most Common Factors For The Failure Of Software Development Project‖. *The International Journal of Computer Science & Applications (TIJCSA) Volume*, *1*.

Romli, R., Sarker, S., Omar, M., & Mahmod, M. (2020). Automated Test Cases and Test Data Generation for Dynamic Structural Testing in Automatic Programming Assessment Using MC/DC. *International Journal on Advanced Science, Engineering and Information Technology*, *10*(1), 120. https://doi.org/10.18517/ijaseit.10.1.10166

Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum* (Vol. 1). Prentice Hall Upper Saddle River.

Serra, D., Grano, G., Palomba, F., Ferrucci, F., Gall, H. C., & Bacchelli, A. (2019). On the effectiveness of manual and automatic unit test generation: ten years later. *Proceedings of the 16th International Conference on Mining Software Repositories*, 121–125.

Shrivastava, D. P., & Jain, R. C. (2010). Metrics for Test Case Design in Test Driven Development. *International Journal of Computer Theory and Engineering*, *2*(6), 952–956.

Sophocleous, R., & Kapitsaki, G. M. (2020). Examining the Current State of System Testing Methodologies in Quality Assurance. *Stray V., Hoda R., Paasivaara M., Kruchten P. (Eds). Agile Processes in Software Engineering and Extreme Programming. XP 2020.*, 240–249. https://doi.org/https://doi.org/10.1007/978-3-030-49392-9_16

.

Stapleton, J. (1997). *DSDM, dynamic systems development method: the method in practice*. Cambridge University Press.

StateOfAgile. (2018). *13th Annual Atate of Agile Report*. www.stateofagile.com

StateOfAgile. (2020). *The 14th annual state of agile report*.

Tahir, M. (2019). Agile Software Development Methods. *Technology*, *1*(1), 10–20.

Tran, H. K. V., Ali, N. Bin, Börstler, J., & Unterkalmsteiner, M. (2019). Test-Case Quality–Understanding Practitioners' Perspectives. *International Conference on Product-Focused Software Process Improvement*, 37–52.

Tripathi, V., & Goyal, A. K. (2014). Agile Testing Challenges and Critical Success Factors. *International Journal of Computer Science & Engineering Technology, 1 (5)*, *5*(06), 632–638.

Uikey, N., & Suman, U. (2012). An empirical study to design an effective agile project management framework. *Proceedings of the CUBE International Information Technology Conference*, 385–390.

Unudulmaz, A., & Kalıpsız, O. (2020). TMMI Integration with Agile and Test Process. *ACM EASE Conference (EASE'20)*. https://doi.org/https://doi.org/10.1145/3383219.3386124

Yamaura, T. (1998). How To Design Practical Test Cases. *IEEE Software*, *15*(6), 30–36.

Yu, J. (2018). Design and Application on Agile Software Exploratory Testing Model. *2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2082–2088