A Comparison Study of Software Testing Activities in Agile Methods

Conference Paper · February 2021

CITATIONS
7

READS
5,246

3 authors:

Samera Obaid Barraood
Department of Computer Science, College of Computers and Information Technol…
15 PUBLICATIONS **31** CITATIONS
SEE PROFILE

Haslina Mohd Haslina
Universiti Utara Malaysia
69 PUBLICATIONS **464** CITATIONS
SEE PROFILE

Fauziah Baharom
Universiti Utara Malaysia
85 PUBLICATIONS **395** CITATIONS
SEE PROFILE

A Comparison Study of Software Testing Activities in Agile Methods

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

Nowadays the majority of companies in the world are adopting Agile methodology for developing their software products due to the methodology promises to deliver product faster with good quality. The most significant method for checking the quality of a product is software testing. However, in Agile development, software testing is very complex and still has challenges. This is largely happened because the Agile development does not concentrate much on software testing activities. It focuses on customer involvement, short iterations, and regular deliveries. This study is a comprehensive review of the current practices of software testing in the Agile methods. The comparison is made based on some criteria such as change during iteration, acceptance criteria, and quality assurance activities. The aim is to identify the similarities and differences between these methods especially in creating test cases. The study focuses on three common Agile methods which are XP, Scrum and Kanban. The review shows no difference in the techniques for designing test cases between these three methods. This result can contribute to help the developers and testers who adopt Agile methodology to follow the same rule of creating test cases based on the suitable technique in different Agile methods.

Keywords: Agile methods, Scrum, Extreme programming, Kanban, Agile testing, test cases.

I INTRODUCTION

Trends for testing software development methodologies demonstrate that the practices of agility are adapted to the workplace context as organizations that adopts more practices of the agile-like software development (Atawneh, 2019). Software testing ensures that what you get in the end is what you wanted to build as stated in the system requirements. Also, it able to identify faults and errors in the system which can increase the quality of the software and it checks out if there is an error in the system which can make software unusable (Sawant et al., 2012). The Agile methods makes the testing becomes an essential component of other parts of the development phases and ensures the continuous product quality (Gil et al., 2016). The agile methods have some similar and difference features, where many studies make a comparison between these methods to show the similar and different aspects, such as Al-Zewairi et al. (2017), Anwer, et al. (2017), Black (2017), Kumar et al. (2019), Merzouk et al. (2017), and Saleh et al. (2019). Agile methods insist in sharing common values and principles, short iterations, continuous communication among Agile team members, and frequent fast delivery of system under test (Brhel et al., 2015; Tahir, 2019). However, some of these methods are different in some points such as period of iteration, acceptance of changing during iteration, and number of team members. Nevertheless, the previous studies did not show whether there is a difference in testing activities especially creating test cases among Agile methods. Therefore, there is a need to check whether they are different in the process of designing test cases or not. In order to help testers to be aware about creating test cases in each Agile method. Thus, this study aims to investigate whether there are any differences in designing test cases among the Agile methods, but this study uses the most Agile methods XP, Scrum, and Kanban (Black, 2017; Srivastava, 2017) adopt in the business environment (Anwer et al., 2017; Saleh et al., 2019) to achieve this comparison.

The next sections of this paper explain comparison between the Agile approaches extreme programming, Scrum, and Kanban, followed by testing in Agile and ending with the conclusion of this study.

II AGILE SOFTWARE DEVELOPMENT METHODS

Agile methodology is a collection of values, principles, and practices that incorporates iterative development, test, and feedback into a new style of application development (Agile 101, 2019; Lewis, 2009). Agile Software Development (ASD) methods are considered lightweight methods that could employ an incremental and iterative lifecycle accompanied with short requirements and iterations, which could be modified within the development with broad participation by the customer (Atawneh, 2019; Boehm & Turner, 2005; Usman et al., 2014). Agile methods are increasingly being adopted by companies worldwide to meet increased software complexity and evolving user demands (Matharu et al., 2015). There are many benefits for adopting ASD methods, such as frequent delivery, customer satisfaction, transparency, flexibility, improved

productivity, better software quality, and predictability (Matharu et al., 2015).

Agile methodology is implemented by several ways. The use of suitable way is depended on the type of project. The most commonly ASD methods used are XP, Scrum, and Kanban (Black, 2017; Srivastava, 2017).

A. Extreme Programming (XP)

XP has developed from long cycles of development in traditional methods (Beck, 1999). The XP aims at delivering useful concepts and ideas pertaining to the software engineering to "extreme" levels degrees (Beck, 1999). The XP method is "theorised" according to the key practices and principles that are being used (Beck, 1999). XP is described by some activities, values, principles, and practices. Activities such as listening (customer needs should be carefully listening by the developer), designing (class, responsibilities, and collaboration cards), coding (pair programmed and must be compliant with the development company's coding standards), testing (unit, system wide integration, and acceptance testing), planning (iterations and user stories), and managing (stand-up meetings) (Black, 2017).

The development in XP is guided by five values, which are communications between the projects team members, simplicity of activities, feedback from customer, system, and the team, courage the team members, and finally, for delivering a good software product, the respect between the team members is compulsory. Additional XP guidance are described as a set of principles: humanity, economics, mutual benefits, self-similarity, improvement, diversity (open-minded to suggestions), reflection, a continuous flow, opportunity (i.e., impediments as opportunities), redundancy (different approaches for problem solution), failure are normal (multiple versions), quality (should not be compromised), baby steps (short space of time), and accepted responsibility by team members (Black, 2017).

The team members in XP should follow 13 practices: 1) sit together, 2) skills and competences, 3) informative workspace, 4) energized work, 5) pair programming, 6) simple and clear user stories, 7) weekly cycle, 8) quarterly cycle, 9) slack the small non-serious stories, 10) ten-minute build, 11) continuous integration, 12) Test Driven Development (TDD), and 13) incremental design (increment in XP is smaller in size than increment in Scrum) (Anwer, Aftab, Shah, et al., 2017; Black, 2017; Matharu et al., 2015). XP values and principles influenced on most of ASD methods to follow it (Abrahamsson et al., 2017; Black, 2017). The XP lifecycle is described in Figure 1. Regarding testing, it is considered one of the major activities to ensure high quality product and high customer satisfaction (Al-Zewairi et al., 2017). XP using TDD, which is a type of unit testing in which test cases are written before coding development to pass these test cases (Beck, 1999).

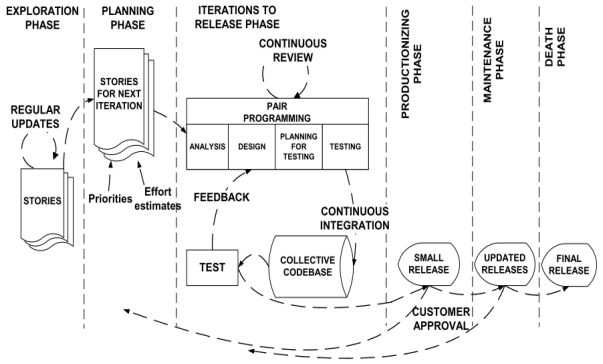

Figure 1. XP Lifecycle (Abrahamsson et al., 2017)

Following are the steps of TDD cycle as stated in (Shrivastava & Jain, 2010):

1) Write a test case for a piece of functionality, 2) Run all test cases to see the new test to fail, 3) Write corresponding code that passes these test cases, 4) Run the test cases to see all pass, 5) Refactor the code and 6) Run all test cases to see the refactoring did not change the external behavior.

B. Scrum

Scrum is ASD method aimed to improve team efficiency and dedicated for managing products (Black, 2017). Scrum puts forward iterations, roles, meetings, rules, and artefacts. There is no obligation to use specific practices, it is optional to team to decide their way to do things. Three things as a minimum should be available to implement Scrum; a wall for placing sticky notes, representing user stories, tasks and impediments; pens and blank sticky notes; and a set of cards to estimate the effort of implementation (Black, 2017). The main roles in Scrum are Product owner, Scrum master, and Scrum team (Anwer et al., 2017; Black, 2017). Figure 2 illustrates Scrum framework process.

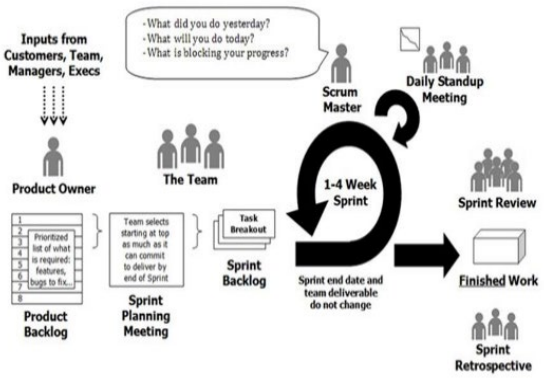

Figure 2. Scrum Process (Anwer et al., 2017)

Product Owner. Product owner is the person who directs the Scrum team toward "what to do next", includes and prioritizes the user stories in the product backlog, which should be accessible, clear, and transparent to all other members (Kayes et al., 2016). He has a responsibility for deciding feature criticality, quality characteristics and product validity (Black, 2017).

Scrum Master. The scrum is facilitated by a scrum master, who also helps to build high-value products with a development team. Scrum master ensures that practices and rules are implemented. He helps in managing and prioritizing the product backlog items. In addition, scrum master deletes all the impediments and outside distracting influences that come along in the process of sprint goals achievement (Black, 2017; Kayes et al., 2016).

Scrum Team. They are self-organized, cross-functional, ranged from 3 to 9 members, and they are responsible of converting the user requirements into active software through developing the code. Testing is also a team responsibility. The scrum team includes different skills members such as, testers, developers, architects and so forth and other specialists (e.g., a performance testing specialist) may join the team when needed and when their tasks are done, they are leave. The following Table 1 displays certain keywords in Scrum which have special meaning (Black, 2017).

Table 1. Scrum Keywords

Scrum Keyword	Description
Sprint	Fixed period of time (iteration) which usually ranged from 2 to 4 weeks as well as each sprint produces a new version of the product.
Velocity	A measure of the amount of work in a sprint a team can do. It can refer to the number of completed story points.
Product increment	A releasable product that resulted at the end of each sprint.
Product backlog	The source of the sprint content. The requirements are stored in product backlog in the form of user stories (product backlog items (PBIs)) that are not implemented yet. These user stories are ordered, where the more important become the first for implementation. During sprints, the product backlog not allowed to change, but the changes can be allowed during the release planning.
Sprint backlog	A set of high priority items from the product backlog which selected by the team during the sprint meeting. The items of sprint backlog also divided into tasks for execution.
Definition of done (DoD)	A product increment become a 'done' state when an agreed list of activities including testing is achieved at the end of a sprint.
Time boxing	The time needed for tasks implementation. Time needed for build user stories, develop, test, and time for meetings. If time short, the non-critical user stories move to the end of product backlog.
Transparency	It means every aspect of the Scrum process that affect the result should be visible to all team members involved in product development. The burndown chart is one examples of transparency.
Daily Scrum	It is also called daily stand-up meeting. It is a mechanism for progress reporting. The team members group up every day for 15-20 minutes, where the status of the tasks is tracked, and they take the corrective action for any speed interruption.
Burndown/burnup chart	Both charts are associated with Scrum. Both charts show how the team is progressing against its predictions.

Regarding testing, a high-level test planning is performed before write test cases to set the environment, budget, place, time and team members. So before delivering the product, unit, integration, regression and all non-functional testing are performed (Harichandan et al., 2014). All types of testing performed through test quadrant to get high product quality (Collins et al., 2012).

C. Kanban

The word Kanban comes from Japanese which means 'signboard' (Merzouk et al., 2017). It is like Scrum used for managing the products with an emphasis on continuous delivery on just-in-time. Kanban process is designed to assist teams by working together in efficient way (Black, 2017; Merzouk et al., 2017). In Kanban, three instruments are used, Kanban board, work-in-progress limit, and lead time.

Kanban board. On a board, several columns list items in different states. Each column represents a set of activities called a station, which represented as analyze, development, and tests, as illustrated in Figure 3. This Agile method also used sticky notes for symbolizing items, steps, and tasks. These sticky notes move from left to right when all activities of a station are done and there is a free slot in the next column. Thus, Kanban board helps in tracking the activities of testing.

Limit Work-in-progress (WIP). There is a limited number of tasks that can handle in each station. Therefore, in a time there is a limited number of user stories. This number of user stories is decided by the team with the contribution of testers depending on the test effort.

Lead time. Kanban is utilized to improve the cycle time and tasks continuous flow via reduce the (average) lead time for the complete value stream. Thus, when complete a task, immediately the ticket

is transferred to the next station if there is any a free slot.

The test cases are designing earlier for development and the maintenance of it during development progresses can help in remove defects during the iteration. Kanban has a concept is Done-Done (Like Scrum has DoD) which point to that a user story cannot reach a completion state until complete the testing.

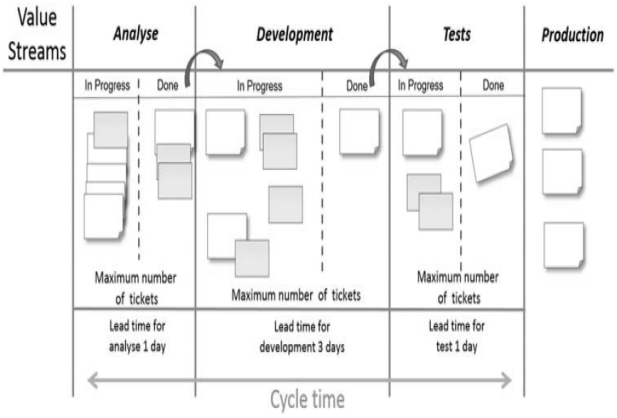

Figure 3. Kanban Lifecycle

III AGILE TESTING

Software testing is a quality assurance activity. It is an important part of any project which improve the quality and productivity of Agile projects (Gil et al., 2016; Nawaz & Malik, 2008). It is a series of processes that begin with requirements step in the early phases of product life cycle (Nawaz & Malik, 2008; Tekin & Cetin, 2012), hence lack of testing resources leads to poor quality (Chomal & Saini, 2014; Rajkumar & Alagarsamy, 2013). In Agile, a testing practice follows Agile principles and it prepared properly so as to cater for continuous changes of the requirements (Jammalamadaka, 2016; Yu, 2018b). It does not just mean testing on Agile projects but testing an application with a plan to learn about it as well as it is integrated into Agile development process unlike a traditional testing which is a phase (Anwer et al., 2017; Harichandan, Panda, & Acharya, 2014). However, it is similar aims with traditional testing, but it is different in the team structure. All team members are involved in Agile testing but with special contribution from professional testers (Kayes et al., 2016).

Agile testing process is based on the iterative methodologies and overcome the disadvantages of sequential models (Khan et al., 2016). All errors are corrected in each iteration after constant testing, obtaining clean code permanently (Gil et al., 2016). 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; Do, 2016). Testing in Agile can address these drawbacks that found in traditional testing, via adaption of frequent change of requirements and short iterations and releases (Yu, 2018a, 2018b). As well as via continuous feedback that redirect all the development process (Gil et al., 2016). The testers utilize essential information and they discard the irrelevant details (Gil et al., 2016).

As mentioned before, testing activities are achieved during each iteration. Starting from creating test plan, prioritizing user stories into product backlog, then creating acceptance criteria that for the testable user stories. Following by writing test cases based on the acceptance criteria, as illustrated in Figure 4. The created test cases should be easy, understandable, and reusable for all team members (Gil et al., 2016).

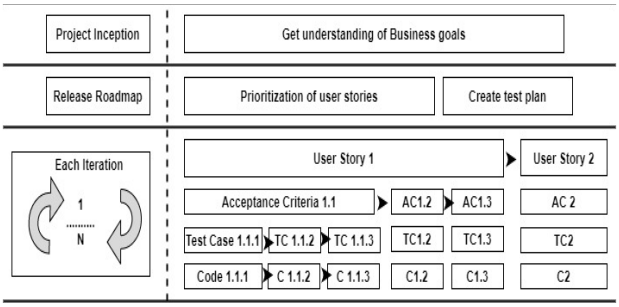

Figure 4. Testing Activities During Iteration Adapted from (Rajasekhar & Shafi, 2014)

ASD methods needs Agile testing practices for its implementation. Agile testing has been widely used in various test practice. The common strategies in testing practice are Test Driven Development (TDD), Acceptance Test Driven Development (ATDD), and Behavior Driven Development (BDD) (Rajasekhar & Shafi, 2014; Yu, 2018b). TDD is based on writing test cases followed by coding. Therefore, the actual tests start before the programming (Black, 2017). ATDD is depended on the collaboration of business customers, developers and testers in producing testable product requirements and to build high quality software in a more rapid way. The key point of ATDD is that it is driven by pre-defined acceptance criteria and acceptance test cases where each part of the program must pass an acceptance test before being merged into the master branch (Atawneh, 2019). Whereas, BDD is depended on the expected behavior of the software being developed. The BDD is often considered to be an extension to TDD and it provides a way to achieve modularity in the software development process (Atawneh, 2019).

Test cases are designing by one of the testing techniques black box or white box, where, each one of has some techniques of designing test cases (Black, 2017). These techniques are used in traditional methodologies and in Agile methods as

well, but in Agile the documentation way is different (Black, 2017). In black box, test cases are designed only from the test item specifications without looking at the code (IEEE, 2008). In contrast, the white-box testing shows what happen inside the system, the tester has an insight about the details of the structure and source code inside the application which he uses it to design test cases (Honest, 2019; Irawan et al., 2018). The design techniques of test cases in black box like boundary value analysis, equivalence class partitioning, and decision tables (Black, 2017). Examples of white box techniques of designing test cases are control flow, basis path testing, loop testing, and data flow testing (Nidhra & Dondeti, 2012). In this paper we give the steps of designing test cases using equivalence class partition technique as example of black box testing (Nidhra & Dondeti, 2012).

1. Define the equivalence classes.
2. Write the initial test case that cover as many as valid equivalence classes as possible.
3. Continue writing test cases until all of the valid equivalence classes have been included.
4. Finally, write one test case for each invalid class.

As example of designing test cases in white box techniques is basic path testing steps (Nidhra & Dondeti, 2012). 1) The code is using for drawing the corresponding control flow graphs, 2) determine the cyclomatic complexity of resultant flow graph, 3) find the linearly independent paths, 4) prepare the test cases for each path one test case and for each test case it should define the input condition and expected output. These designing techniques are using in all Agile methods (Black, 2017).

XP, Scrum, and Kanban are commonly used with some differences and similarities. Since these methods are belonging to Agile so they have iterative and incremental nature but with different durations, continuous planning, clear definition of roles, and a workflow discipline. XP focus on engineering aspects of software project whereas Scrum and Kanban focus on management aspects. Table 2 shows the comparison between them. Some points of this comparison is extracted from a number of studies which include (Al-Zewairi et al., 2017; Anwer, Aftab, Shah, et al., 2017; Black, 2017; Kumar et al., 2019; Merzouk et al., 2017; Mohammad Almseidin et al., 2015; Nawaz & Malik, 2008; Saleh et al., 2019; Sophocleous & Kapitsaki, 2020). However, these studies have not highlighted testing activities and how designing test cases in different Agile methods.

Table 2: Comparison between XP, Scrum, and Kanban

Criteria	XP	Scrum	Kanban
Focus	Engineering aspects	Management aspects	Management aspects
Stages inside iteration	Analysis, design, planning for testing, testing	Analysis, design, evolution, testing delivery	Analyse, development, testing
Team size	2 - 20 members	5 – 9 members with Scrum Master and Product Owner	Undefined
Iteration/ Sprint duration	From 1 to 3 weeks	From 2 to 4 weeks	No specific period. It is measured based on the cycle time
Daily meeting	Yes	Yes	Yes
Requirements plan	The listing of prerequisites is done always	The requisites require listing based on the length of the run, each two, three or a month	The basics are done always every day/ hour
Change during iteration	Allowed without constraints	Changing not allowed if Sprint begins	Allowed without constraints
Acceptance criteria	Defined from user stories	Defined from user stories	Defined from user stories
Test cases	Designing based on acceptance criteria	Designing based on acceptance criteria	Designing based on acceptance criteria
Feedback	Span from minutes to months	Span over a month	Undefined
Testing	Performed in each iteration	Performed in each iteration	Performed in each iteration
Quality assurance activities	TDD, pair programming, continuous integration, unit testing, system testing, acceptance testing, coding standards, refactoring, collective code ownership, simple design, on-site customer, face to face communication, regular daily meeting, focusses and concentrates on leveraging	Unit testing, continuous integration, acceptance testing, exploratory testing, automation testing, regular sprint and daily meetings, coding and design standards, test cases are design based on acceptance criteria	A single user story is handled in an iteration, each user story is split into tasks, tasks split into sub-tasks, test is performed in each station, testing activities are traced by helping Kanban board, continuous flow, upfront test cases design, test cases are design based on acceptance criteria

	the quality of software, slack the small non-serious stories, test cases are design based on acceptance criteria		
Testing issues	No documentation for user plan, short iteration (one or two weeks), unstable requirements, lack frequent communications, short period of time	Most quality assurance activities are skipping due to absence of a dedicated quality assurance team; acceptance criteria, user stories, and DoD are not defined properly, ignoring negative paths when designing test cases, 50% of test cases are not written based on requirements, unit testing is inadequate and inconsistent, quickly requirements changes causes increasing number of test cases and testing speed, high cost and time of regression testing, lack communication between tester and developer, unbalance between the meetings and documentation, less test documentation may lead to early Sprint output.	As other Agile methods issues, most test cases are not defined based on requirements, lack communication between testers and developers, less documentation causes difficulty in designing test cases in high level requirements

The XP, Scrum, and Kanban have iterative and incremental nature but with different durations. These ASD methods with several features and aspects to support projects that need short or long period of time to be finished.

It is noticed in Table 2 that testing activities are integrated with other parts (i.e., analysis, design, develop) of the ASD methods process (e.g., XP, Scrum, and Kanban). Testing is a very important part and implement good practices and follow the whole-team approach (Gil et al., 2016; Srivastava, 2017). It is achieved effectively in each iteration of the ASD process. Testing activities such clarifying requirements, preparing test data, and writing test cases in all software development methods have the similar aims, which is detection, prevention, demonstration, improving quality, verification, and validation (Chauhan, 2010; Kaplesh & Pang, 2020; Kayes et al., 2016; Rajasekhar & Shafi, 2014). However, in ASD it should take into consideration the volatility of requirements, the whole team sharing in testing process, and iterative and incremental life cycle. The testing activities support several principles, practices and values of different ASD methods (i.e., XP, Scrum, Kanban), such as continuous integration, incremental, acceptance criteria, and accepting changes during the development. The increments in these methods requires test cases to validate its functionality and to validate the whole system operations. Test cases which is the main part of testing activities constitute based on the acceptance criteria, which are extracted from testable user stories in all these methods (Black, 2017; Kayes et al., 2016). XP, Scrum, and Kanban methods use the same strategy to write test cases, which written before coding. Therefore, designing test cases from user stories is similar in ASD methods. However, the big issue is that many companies do not create test cases based on the requirements (i.e., user stories) (Sophocleous & Kapitsaki, 2020; Uikey & Suman, 2012) and this causes several problems of testing quality, other issues is displayed in Table 2.

IV CONCLUSION

The main result of this comparison between Agile methods are observed that these methods (XP, Scrum, and Kanban) are different in some roles and practices but in testing activities they are similar, where designing test cases depends on the user requirements, which described in Agile as user stories. As well as they are using the same designing techniques of test cases. The role of software testing is very imperative in the development process of Agile projects for ensuring the quality of products. The nature of Agile that accept requirement changes, incremental, and iterative emphasize that the testing should be achieved in each iteration. An additional research is needed to achieve to support our work on the test cases in Agile methods to show its importance for gain high quality software. This study contributed to show that the testing activities (i.e., designing test cases) are not different in Agile methods.

REFERENCES

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

Agile 101. (2019). What is Agile Software Development? Agile Alliance. https://www.agilealliance.org/agile101/

Al-Zewairi, M., Biltawi, M., Etaiwi, W., & Shaout, A. (2017). Agile Software Development Methodologies: Survey of Surveys. Journal of Computer and Communications, 05(05), 74–97. https://doi.org/10.4236/jcc.2017.55007

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.

Anwer, F., Aftab, S., Waheed, U., & Shah, S. S. M. (2017). Agile Software Development Models TDD , FDD , DSDM , and Crystal Methods : A Survey. International Journal of Multidisciplinary Science and Engineering, 8(2).

Atawneh, S. (2019). The Analysis of Current State of Agile Software Development. Journal of Theoretical and Applied Information Technology, 97(22).

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

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.

Black, R. (2017). Agile Testing Foundations An ISTQB Foundation Level Agile Tester Guide. BCS Learning & Development Ltd.

Boehm, B., & Turner, R. (2005). Management Challenges to Implement Agile Processes. Traditional Development Organizations.

Brhel, M., Meth, H., Maedche, A., & Werder, K. (2015). Exploring principles of user-centered agile software development: A literature review. Information and Software Technology, 61, 163–181.

Chauhan, N. (2010). Software Testing: Principles and Practices. Oxford university press.

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.

Collins, E., Dias-Neto, A., & Jr., V. F. de L. (2012). Strategies for Agile Software Testing Automation : An Industrial Experience. Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual, 440–445.

Do, H. (2016). Recent Advances in Regression Testing Techniques. Advances in Computers, 103, 53–77. https://doi.org/10.1016/bs.adcom.2016.04.004

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.

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

Honest, N. (2019). Role of Testing in Software Development Life Cycle Nirali. International Journal of Computer Sciences and Engineering, 7(5), 886–889.

IEEE, 829-2008. (2008). IEEE Standard for Software and System Test Documentation. IEEE Computer Society.

Irawan, Y., Muzid, S., Susanti, N., & Setiawan, R. (2018). System Testing using Black Box Testing Equivalence Partitioning (Case Study at Garbage Bank Management Information System on Karya Sentosa). International Conference on Computer Science and Engineering Technology.

Jammalamadaka, K. (2016). An Industrial Survey On The Test Automation Challenges In The Agile Scrum Teams. International Journal of Research in Engineering and Technology (IJRET), 05(10), 82–87. http://ijret.esatjournals.org

Kaplesh, P., & Pang, S. K. Y. (2020). Software Testing. In Software Engineering for Agile Application Development (pp. 189–211). IGI Global.

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

Kumar, R., Maheshwary, P., & Malche, T. (2019). Inside Agile Family: Software Development Methodologies. International Journal of Computer and Engineering, 7(6), 650–660. https://doi.org/10.26438/ijcse/v7i4.184190

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.

Merzouk, S., Elhadi, S., Ennaji, H., Marzak, A., & Sael, N. (2017). A Comparative Study of Agile Methods: Towards a New Model-based Method. International Journal of Web Applications, 9(4), 121–128.

Mohammad Almseidin, Khaled Alrfou, Nidal Alnidami, & Ahmed Tarawneh. (2015). A Comparative Study of Agile Methods: XP versus SCRUM. International Journal of Computer Science and Software Engineering (IJCSSE), 4(5), 126–129. http://ijcsse.org/published/volume4/issue5/p3-V4I5.pdf

Nawaz, A., & Malik, K. M. (2008). Software testing process in agile development. In Computer Science Master Thesis, Blekinge Institute of Technology. Citeseer.

Nidhra, S., & Dondeti, J. (2012). Black box and white box testing techniques-a literature review. International Journal of Embedded Systems and Applications (IJESA), 2(2), 29–50.

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 Projectl. The International Journal of Computer Science & Applications (TIJCSA) Volume, 1.

Saleh, S. M., Huq, S. M., & Rahman, M. A. (2019). Comparative Study within Scrum, Kanban, XP Focused on Their Practices. 2nd International Conference on Electrical, Computer and Communication Engineering, ECCE 2019, December. https://doi.org/10.1109/ECACE.2019.8679334

Sawant, A. A., Bari, P. H., & Chawan, P. . (2012). Software Testing Techniques and Strategies. Journal of Engineering Research & Applications, 2(3), 980–986.

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

Srivastava, S. (2017). Agile Development Testing Paradigms. International Journal Of Advance Research, Ideas And Innovations In Technology, 3(3), 915–923.

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

Tekin, O., & Cetin, G. B. (2012). Application test process in product life cycle. 2012 6th International Conference on Application of Information and Communication Technologies (AICT), 1–6. https://doi.org/10.1109/ICAICT.2012.6398483

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.

Usman, M., Mendes, E., Weidt, F., & Britto, R. (2014). Postprint Effort Estimation in Agile Software Development : A Systematic Literature Review. The 10th International Conference on Predictive Models in Software Engineering, 82–91. http://dx.doi.org/10.1145/2639490.2639503 N.B.

Yu, J. (2018a). Design and Application of a Testing Framework of Online Course Based on Agile. IOP Conference Series: Materials Science and Engineering, 394(3), 32099.

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