Hybrid User Acceptance Test Procedure to Improve the Software Quality

Natarajan Sowri Raja Pillai
Department of Information Technology
Raak College of Engineering and Technology,
India
sowrirajacse@gmail.com

Ranganathan Rani Hemamalini
Department of Electrical and Electronics Engineering
St. Peters Institute of Higher Education and Research,
India
ranihema@yahoo.com

Abstract: *Fast-growing software needs result in the rise of quality software in technical and time challenges in software development and the impact the cost and scarcity of resources addressed by the companies. Thus, this research focuses on optimal implementation of the User Acceptance Testing (UAT) and the process generation integration. The Software Development Life Cycle (SDLC) was adapted to develop software and introduce the UAT process right from the initial phase of the software development. Additionally, it is devised to maximise time reduction by implementing the client testing in all the three processes. A High Capability to Detect (HCD) procedure has been incorporated in the problem formulation that has optimally identified sensitive bugs. A Modified Reuse of Code (MRC) is proposed for a feasible time-saving solution. The proposed UAT will provide an optimal solution in the software testing phases implemented earlier than black-box testing. The proposed UAT has significantly better production time, development cost, and software quality in comparison to other traditional UATs. The study's findings were corroborated by the output data from the UAT cases. The UAT ensures the quality of the product in the early phase of the development and implementation of the projects. This will minimise the risk during and post-implementation of bugs and achieve the target audience's needs*.

Keywords: *Black box testing, high capability to detect, modified reuse of code, user acceptance test.*

Received April 18, 2021; accepted October 21, 2021
https://doi.org/10.34028/iajit/19/6/14

1. Introduction

Software Testing is made up of several stages or phases. Software testing and its precursor, software development, is made up of multiple phases. All the phases mentioned above make up the two main basic lifecycles, namely Software Development Life Cycle and Software Testing Life Cycle. The stages involved in the Software Development Life Cycle (SDLC) are Business Analysis, Requirement Gathering, Requirement Analysis, Design, Development, Testing, Implementation, and Maintenance. The testing process in SDLC is done using the stages of the Software Testing Life Cycle (STLC). Unit testing, functional verification testing, system integration testing, system verification testing, and User/Client Acceptance Testing are the steps of the STLC (CAT).

Testing is a vital part of the software development process. The quality of the software testing is used to determine the quality of the software. Due to the crucial nature of testing, a lot of human effort and resources are being spent on planning and executing testing. The classic waterfall SDLC model is being considered throughout the length of this work since this model is widely accepted among major software companies which develop software. Irrespective of the SDLC model being followed, there is an emphasis on testing. Since the quality of the software plays a majorrole, keen care is being taken by all the companies to test for the quality. The last and end part in providing quality of developed software is the company's system verification testing. However, the customer or client or user is the actual person to use the software after implementation. On this consideration, CAT is being carried out at every firm. At this juncture, software quality is a requirement of the client or customer [5]. However, the client or the customer has not been involved throughout the software development and testing processes. The only phase where the customer gets involved is at the client or customer acceptance testing. The software passes numerous phases in the development life cycle and the testing life cycle.

Hence, one has to be very ardent on quality at the beginning than at the end. A quality product is a product that ensures quality right from its base. So, to ensure quality from its base, one has to test it from the base. This is a primary requirement for any user, developer, or other person involved in developing the software. The reason for this research is the query if the quality is being required from the base by the client or customer, why is the testing being carried out by the customer or client at the end of the testing. This kind of testing at the end leads to enormous usage of human resources, huge costs, and a lot more on time. The actual proposal of the research is to ensure user or client acceptance test right from the beginning of the development life cycle. The testing personnel must include clients and experts from the customer group. Any kind of formal test intervention from the client or customer improves the quality of the software product or project. Every development and testing phase can be included with a client or customer acceptance testing to resources, cost, and time.

This research work is organized in this paper as follows: section 1 describe about the need for software testing, followed by levels of software testing and motivation of this research. The detailed literature survey is covered in section 2. The problem specification of this research work is described in section 3. The adopted research methodology, comparative studies and results are discussed in section 4. Finally, the section 5 explain the outcome of this research work is given as conclusion.

1.1. Software Testing

The quality of the software depends on the role of the software development life cycle in the software development process. To meet the customer requirements, software quality is the top-level priority [28]. Regarding the software engineering area, Verification and Validation (V&V) methods are utilised to guarantee the quality of software items. The reason for V&V is to help the improvement of quality software frameworks. Software testing is a critical V&V movement that looks at the conduct of a software framework on a limited arrangement of experiments against the normal conduct [29].

Progressively mind start and basic software frameworks have made software testing a very vital movement. Software testing is directed through the software advancement and upkeep life cycle and ought to be upheld by a distinct and controlled testing process [27]. The activities of the testing process in software development are defect tracking, test logs, results evaluation, test execution, developing the test environment and test planning [30].

1.2. Levels of Software Testing

The levels of software testing are categorised into four levels. These levels are as follows [5],

- Unit Testing
- Integration Testing
- System Testing
- User Acceptance Testing

1.2.1. Unit Testing

The isolated modules are the portion of the software code that will be tested with the functionality of the requirements given by the clients. The developer itself will do this unit testing. All the developers will test their module as unit testing in the same project.

1.2.2. Integration Testing

After completing the unit testing, this type of testing will involve more than one or any number of small modules. This ensures the error-free software, data flow and flow of control after combining all the modules in a project. Integration tests will be done by the testers only.

1.2.3. System Testing

This type of test involves different two types of testing, functional and non-functional testing. This Test will ensure that overall software testing is against the requirement given by the client. The testing team will do this type of testing.

1.2.4. User Acceptance Testing

Customer requirements will be validated with this type of testing. The user acceptance test will ensure the customer's acceptance of what they want to do with the software. UAT test consists of two levels of testing, Alpha testing and Beta testing.

1.3. Motivation of the Research

This research work will motivate the software developers in different ways, as shown below,

- Finding the defects in the code will be increased to get more quality of the software.
- Applying for this framework, considering the cost of the software development will be reduced.
- The quality of the software will be increased.

2. Literature Review

Latorre [16] presented an approach for acceptance testing of the User Interface-equipped Internet of Things systems. The author’s approach to user acceptance testing is new for cloud-based applications. Mutation Testing was used, and the effectiveness of using this type of testing is enhanced. Test suites that are used at this Internet of Things (IOT) based system are adopted. Since the newly adopted suites are unfamiliar to prior users, they need hands-on experience to conduct the acceptance testing. As said before, the acceptance testing conducted is a new type, but the exact users could not perform the acceptance test.

Nuzha and Meenal [24] stated a new acceptance testing form. The acceptance criteria form has a Given-When-Then template. The criteria are divided into steps. The interdependencies are determined and numbered. The output of one step becomes an input for another step. A dependency tree of the entire steps of the testing is formed. The tree thus formed has a weighted tree wherever there is a decision that occurs. Hence the coverage of the Test is being generated. Through this template, the coverage of the tests is found, and therefore there is a vision being obtained of the quality of the software. All these above works on positive test cases and not likely on negative test cases.

Henard *et al*. [9] presented a combination of interaction and diversity-based techniques instead of the traditional white and black box techniques. The authors found the difference in quality between the older techniques and the new combinational testing technique. They discovered an overlap between the regression tests in the white box and the black box. By conducting white box regression tests at the initial level, it would be best for the quality of the software instead of releasing multiple versions of the same. Since the developers conducted the white-box tests, not many bugs would be found during the white box regression test.

Kochhar *et al*. [15] Unit Test-Driven Development (UTDD) and Acceptance Test-Driven Development (ATDD) are two product development approaches that have been used to create software. The costs involved in the development of the software by using these test-driven development techniques were economical. Even though the cost achieved for development was economical, there was a requirement for extensive help from the customer. There was a particular boost in the quality of the software while these techniques were used. As an additional requirement, this kind of development process requires more cooperative clients. The authors developed a knowledge-based advisory system that fulfilled the role of a "virtual quality editor". The system provided assessment results and suggestions based on the prerequisites provided [6]. Using programming language standards, the authors developed a code review technique for achieving maximum software quality. The program suggests whether the code quality can be improved and helps junior developers and students to achieve a good coding attitude [1].

Minhas *et al*. [22] presented an improved attempt by implementing the Business-Driven Acceptance Test methodology for software quality tests. Selective test cases and sceptical scenarios are being used to test the software. The uncertainty of the ad-hoc testing leads to the poor quality of the software. This particular methodology gained confidence at companies. Thus, users of the business gained technical knowledge of the software's functioning. According to the authors, white-box, black-box, and grey-box testing are the three most general and widely used software testing methods for identifying defects [5]. Yu and Pang [31] show that the improved uniform design strategy with all valid and invalid levels could be used to generate fewer test data than that of other selected strategies, and the test effects are appropriate to that of other strategies. Khan and Khan [13] have conducted a study on Smells in software test code: A survey of knowledge in industry and academia. He conducted a multivocal literature mapping (Classification) for scientific literature and practitioners. One hundred sixty-six sources were conducted. Test smells on surveying industry and academia. Liskin *et al*. [19] addressed the need for structured and reliable software testing. Their model (ExET) was used to set off important factors efficient and effective testing large-scale systems. They concluded that it is novel, actionable, and useful in practice.

In this paper, the authors have described and compared the two most important and commonly used software testing techniques for detecting errors: Black box testing and white box testing [29]. Nomura *et al*. [23] discussed the different aspects and types of testing as the Site Reliability Engineering (SRE) and Testing run alongside either by static testing and reviews or running the system/SW to confirm the compliance of requirements. In this paper, the authors have tested techniques and tools and described them as well as some typical latest research has also been summarised [11]. Arnicane [2] focused on the theoretical bounds of the size of test suites or the complexity of domain testing methods and included a subsumption hierarchy that attempted to relate various coverage criteria associated with the identified domain testing methods.

Coutinho *et al*. [4] analysed bibliographic of 1099 papers related to Agile Requirements Engineering with Software Testing (REST). They chose 14 of these for a more detailed study based on the systematic mapping principles. With test case design as a Software Testing methodology, test cases are an important artefact. Authors proposed an exponential software reliability model for fault detection with time variance [26]. They formulated multi-objective software reliability model for testing. According to a weighted cost function and testing effort metrics, they contrasted a multi-objective model with modules. Mei *et al*. [21] aimed to explore the regression testing for large-scale embedded software development. In their study qualitative part two large-scale companies taken for analysis with five software testing teams. They conclude that firms should reassess their regression testing strategy.

3. Problem Specification

The traditional user acceptance testing is usually conducted once the development is completed. At this stage, the user would be provided with all the existing bugs. Some checking conducted by the developer beforehand can only reveal flaws in the code. Any test carried out by the tester will only expose bugs in the defined technical specifications of a particular functionality [18]. Both the developers and the testers find bugs and communicate between themselves. Hence, the bugs get closed without the knowledge of the actual user. The end-user or client gets to know only during the completion of the software development [28]. This leads to much extra time, cost, and resources that would have to be utilised, and repetitive work occurs for the developer and tester. The entire team which gets involved in the developmental process undergoes all these issues [27]. When the whole team deals with the same problems, the software's consistency suffers. During the growth and testing process, retesting and regression testing are performed on instances not required [17]. The user would have a different requirement, but the developer and tester would be wasting time without the intervention of the actual user. A solution to this process is required. The occurrence in the development must be indicated as well as tested with the actual user or customer.

To overcome the above problem, the research conducted to reduce the process that occurs in the execution of test phases, find the bugs at the initial phase of testing, investigate, and update the client daily, which will reduce the time involved in the development of the software. Further, this work reduces the resources for the development and testing process, follows a framework for testing, and improves the overall quality of the software.

4. Research Methodology

The software testing life cycle involves a list of processes, but the processes undergo repetition. Repeating work cannot be intended for the workforce or afforded by the management. Also, an increase in time for completing work gives the developing companies a bad reputation from the client's perspective. Since the status updates show that the same glitches or defects keep cropping up, repetitive software development and testing cycles can be stopped [3]. So, a process cycle is followed at the unit testing level. The need for human capital participation in software testing exists at two key stages. System integration and system maintenance testing are the two steps of the development process [8]. The implementation process includes two steps: system integration and system maintenance monitoring. Until both system deployment and system evaluation testing, functional verification testing is performed [27].

At the beginning of the production process, this is a waste of time. The real user of the app is the consumer. The customer or client is the one who is in charge of providing software specifications, and the software is built based on those requirements. The findings of unit testing are rarely if ever, shared with the developer or customer [25]. The customer or client should be given a report. The customer or client is informed about the importance of paying attention to the specific requirement. CAT is often used [18]. As soon as a necessary vulnerability is discovered, the flaw or error is given domain clearance. A domain clearance is permission from domain or industry experts that must be received. The business/domain experts make the requirements change with proper approval and version control [8]. At the same time, the developers can work on other unit developments. Every testing phase involves a user acceptance test. This particular change in customer acceptance testing results in a requirement change during the implementation process. As a result, software production and testing became less complicated [8]. The cost, time saved, and human resources saved are boundless. A framework is designed and applied in order to obtain the above variables with high quality in the applications developed.

4.1. UAT on Every Phase

The user's position in the software's UAT and the correlation of that specific user to be discovered. Any step of the production process includes face-to-face testing and self-managed testing [12]. The UAT has a time limit and a closely regulated atmosphere for testing. For maximum efficiency, the participants in the process must communicate face-to-face or through video conferencing. In this manner, the test scenarios are thoroughly studied, and the findings are automatically concluded [20]. As a result, the aim is to log any contact at any stage. Self-managed research aims to save time. Not everybody is expected to take the exam. Off-hours testing is done, but the UAT is done without fail at any point [14].

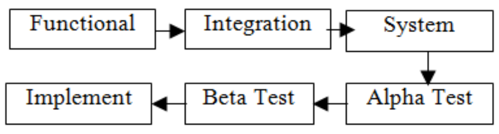


Figure 1. Two phases of UAT.

Figure 1 illustrates a divided two-phased UAT. Further, it can be stated that the design, control, management, and optimisation of these new processes and technologies, and their integration into the existing development process, pose significant technological challenges to ensure their reliability and safety, to improve and maximise their efficiency and cost competitiveness to provide quality software to residential, commercial, industrial, and transportation needs.

4.2. The Flow of End User Test Planning

- Teams from various departments are involved in the planning of the overall testing plan.
- Then lead to specific Alpha, Beta, and UAT.

The overall design of the testing plan is depicted in Figure 2 and shows the various steps involved in the software testing process and the user acceptance at various levels. The User Acceptance Testing step provided in Figure 2 can be explained in 2 phases that have been provided in Figure 1.

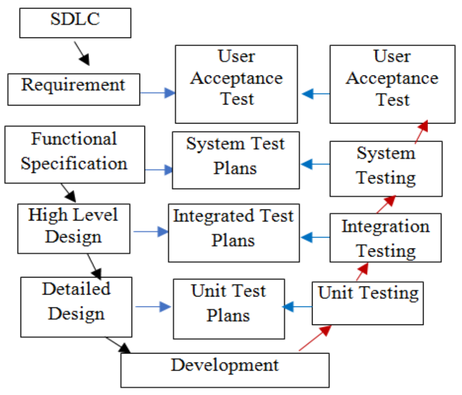


Figure 2. Design of test planning.

4.3. Modeling of UAT Framework

The objectives for optimum operation during UAT, users, assess the software to grasp mandatory responsibilities in real-world setups about conditions. UAT framework adopted in this paper is given in Figure 3. UAT group checks the inspected report one by one, includes significant remarks, and shares them repeatedly.

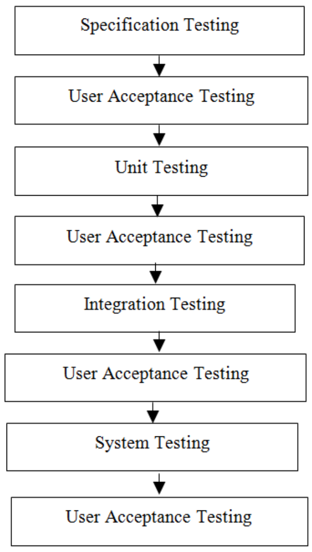


Figure 3. UAT framework.

Software Workflow with delay Problem

- The software that is considered is an IVR-based project.
- The computational flow for testing is described as follows.

- *Step*-1: Requirement document is converted to Test cases.
- *Step*-2: Implementation of unit testing with the generated test case from the requirement specification.
- *Step*-3: Completing the Alpha and Beta testing followed by the proposed UAT framework.
- *Step*-4: Determine the major factors to consider .

Reuse of Test cases-Reduces Re-writing time
Defects Detection-Domain Expert and Module segregation
Test Costs -Number of Resources involved
Test Cycle Time-Test time for KLOC (Thousands of lines of Code)
Where D_t = the defects found at early stages over time

- *Step*-5: Comparison of the results hence obtained.

Table 1 shows the input data utilised in this study for various testing scenarios.

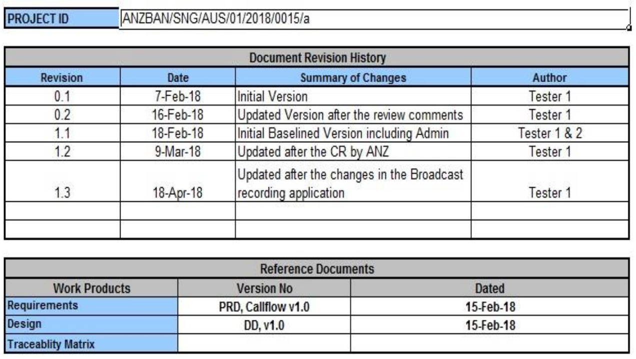
Table 1. Input Data used in this research.

UAT Traditional Framework (Defects Found)	Cost	Agile on Traditional UAT (Defects Found)	Cost	Proposed UAT with Agile Practices (Defects Found)	Cost
14	14	17	13	19	11
12	18	18	17	15	10
11	16	14	15	18	12
13	15	15	14	17	13
10	17	16	16	16	9
60	80	80	75	85	55

4.4. Test cases

- *Step* 1: Define Test Document: In the baseline version, the person updated the document, and the number of interactions is defined along with the occurrence dates.

Figure 4 shows the sample of Test case document initiation. This sample document contains the information about the Project ID, Document Revision histories such as Revision version, Date, a summary of changes, and Author. This document also contains Reference Documents such as work products, version No, and Date. This will help to reduce the tester time to complete the testing work and gets approval from the users when and where required.



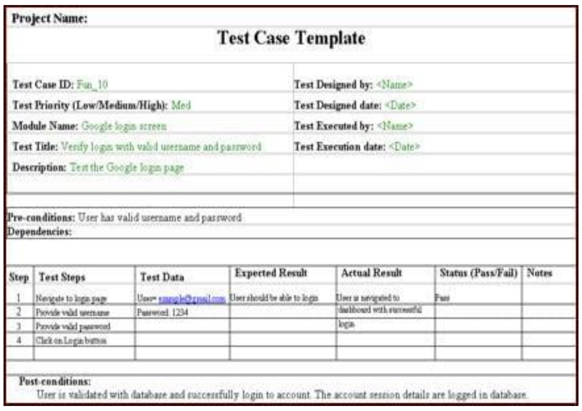
PROJECT ID	ANZBAN/SNG/AUS/01/2018/0015/a

Document Revision History			
Revision	Date	Summary of Changes	Author
0.1	7-Feb-18	Initial Version	Tester 1
0.2	16-Feb-18	Updated Version after the review comments	Tester 1
1.1	18-Feb-18	Initial Baselined Version including Admin	Tester 1 & 2
1.2	9-Mar-18	Updated after the CR by ANZ	Tester 1
1.3	18-Apr-18	Updated after the changes in the Broadcast recording application	Tester 1

Reference Documents		
Work Products	Version No	Dated
Requirements	PRD, Callflow v1.0	15-Feb-18
Design	DD, v1.0	15-Feb-18
Traceablity Matrix		

Figure 4. Test case document initiation.

- *Step* 2: Implementation: this research study concentrates on user acceptance testing, which is very useful in the software testing life cycle model. The proposed framework on the User Acceptance Test is to ensure meeting the client’s requirement at every stage of software development. The proposed framework will also support software testing automation tools. The main features of UAT will reuse the existing code wherever possible to increase the quality of software. Figure 5 shows the standard fields of the sample test case template, which will give details about the Project name, Test Case ID, Test Design by, Test Priority, Module Name, Test Title, Test designed to date, Test executed by, Test execution date and give the details of various step to be followed to complete the Test. The various steps show the Test Steps, Test Data, the expected Results, Actual results obtained, and status of the Test, whether Pass or Fail. Table 2 presents the comparison of existing framework with the proposed UAT.


Project Name:

Test Case Template

Test Case ID: Fun_10	**Test Designed by:** <Name>
Test Priority (Low/Medium/High): Med	**Test Designed date:** <Date>
Module Name: Google login screen	**Test Executed by:** <Name>
Test Title: Verify login with valid username and password	**Test Execution date:** <Date>
Description: Test the Google login page	

Pre-conditions: User has valid username and password
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to login page	User= sample@gmail.com	User should be able to login	User is navigated to	Pass	
2	Provide valid username	Password: 1234		dashboard with successful		
3	Provide valid password			login		
4	Click on Login button					

Post-conditions:
User is validated with database and successfully login to account. The account session details are logged in database.

Figure 5. Standard fields of sample test case template.

Table 2. Comparison of existing framework with Proposed UAT.

Methods	Defects Found	Cost	Software Quality
UAT Traditional Framework	60	80	70
Agile on Traditional UAT	80	75	80
Proposed UAT with Agile Practices	85	55	93

The UAT consists of different test plans, which are done at different levels of SDLC. This testing is included in all phases of the Software Testing Life Cycle to improve the software quality and complete the work on time with the optimised cost of the project. Each test plan is a sequence of the input process model, functional testing of the particular model, and corresponding output of the concerned process model. The test cases planned during the project plan to accompany all the testing plans to complete the project smoothly and within the planned duration. The test plans are automated in every phase of work and reduce the time required for unit-level testing. This is carried at every process level to achieve higher efficiency of the software product.

The new framework of UAT in the software testing model will increase the confidence of clients' software that meets their requirements at every stage. This will be tested along with the client approval of every stage and reduce enormous time on user-level functional testing of the software. This ensures that developed software is stable and robust in work. UAT is done through the Black Box testing techniques to test the users' point of view, and evaluation is done at every stage of software development. The client satisfaction is increased, and their confidence levels are also increased in terms of their requirements are met with their expectations. The development team and clients have more communication to improve the quality of the software. Thus the requirement definitions are improved through acceptance tests and are approved by the clients.

- *Step* 3: Comparison of Results and Discussion: the data for the testing procedure is applied in a web-based application. The user acceptance testing is conducted, and the results are compared. The findings show that considerations such as production time are short, capital costs are limited, and software quality has increased. The obtained results from the agile type of testing are vague and do not show any procedural change that occurred at the end of the completion of the software development [7]. The reports that got traversed between the users and the development-testing team did not have any order. The collected data are evaluated and compared to current processes. The following segment tabulates and explains the comparison data.

The results are compared with the existing methods like UAT Traditional Framework and Agile Traditional UAT and Proposed UAT with Agile Practice in Figure 6. The obtained results are depicted in Figure 6 and compared. The results show that the UAT with the Agile practice offered the increased or high software quality is found, and similarly, reduced the cost with more defects are found in this method and got rectified. So, the proposed hybrid method improved software quality, found more defects and reduce total project cost.

- Limitations: most existing acceptance testing solutions need a Graphical User Interface (GUI) prior to the creation of tests. Furthermore, simple changes in the user interface might cause GUI-based tests to fail. Many parts of GUI-based tools are immature. Selenium tests may also fail if the driver of the component being tested changes. In the event of a modification request, much upkeep is necessary. It's possible that automating a non-stable feature may result in much maintenance.

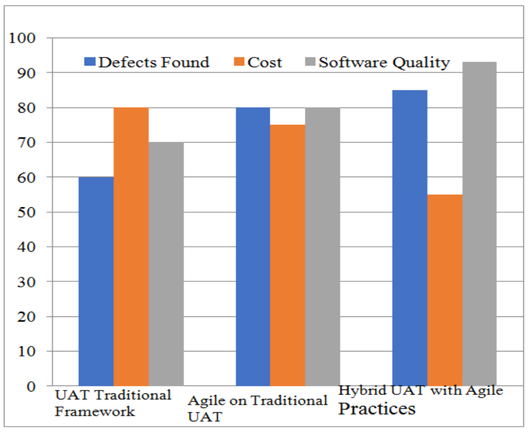


Figure 6. Comparisons of preexisting traditional frameworks.

This project data are collected from in-house project developers and programmed using Excel software. The experiment was conducted with the software for UAT traditional framework, Agile on Traditional UAT and our proposed Hybrid model which is the combination of traditional and Agile process.

5. Conclusions

This research discusses and contrasts the various software testing methods used in the software development industry [3]. The usefulness of the techniques varies depending on the type of programme and database associated with it. There are different types of testing, with a white box and black box testing being the most often used. Even the approaches vary concerning the application. The characteristics are depicted in the results that are taken from one of the web-based application development processes.

This research work consists of the comparisons of the existing frameworks with the proposed frameworks. Compared with the traditional UAT frame and the agile framework with the proposed hybrid UAT framework. Proposed hybrid UAT framework combination of traditional UAT with the Agile framework. In this work, the number of defects found is more than the others, and the cost of the work to be completed is very low with others. However, the quality of the software increases, as shown in the graph and table.

The Acceptance Test Priority specification and review have been completed [10]. The research results and review of the daily research data was compared to the testing technique used previously. The consistency of the tested papers and the requirement adjustment that happened is investigated. It has been discovered that the production time, development cost, and software quality are all significantly better than the waterfall model testing. The study's findings were corroborated by the output data from the User Acceptance Test cases.

- Scope of Future Study: this research result demonstrates the software's and the production team's improved results. In the future, the framework's output will be examined for other interventional testing and automated using automated testing methods. Further research was needed to look at ways to minimise the amount of time and money spent on production.

References

[1] Abdallah M. and Alrifaee M., “A Heuristic Tool for Measuring Software Quality Using Program Language Standards,” *The International Arab Journal of Information Technology*, vol. 19, no. 3, pp. 314-322, 2022.

[2] Arnicane V., “Complexity of Equivalence Class and Boundary Value Testing Methods,” *Scientific Paper*, vol. 751, pp. 80-101, 2009.

[3] Causevic A., Sundmark D., and Punnekkat S., “Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review,” *in Proceeding of 4th IEEE International Conference on Software Testing, Verification and Validation*, Berlin, pp. 337-346, 2011.

[4] Coutinho J., Andrade W., and Machado P., “Requirements Engineering and Software Testing in Agile Methodologies,” *in Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, Salvador, pp. 322-331, 2019.

[5] Crispin L., “Driving Software Quality: How Test-Driven Development Impacts Software Quality,” IEEE Software, vol. 23, no. 6, pp. 70-71, 2006.

[6] Eldrandaly K., “A Knowledge-Based Advisory System for Software Quality Assurance,” *The International Arab Journal of Information Technology*, vol. 5, no. 3, pp. 304-310, 2008.

[7] Garousi V. and Küçük B., “Smells in Software Test Code : A Survey of Knowledge in Industry and Academia,” *Journal of systems and Software*, vol. 138, pp. 52-81, 2018.

[8] Haugset B. and Stalhane T., “Automated Acceptance Testing as an Agile Requirements Engineering Practice,” *in Proceeding of 45th Hawaii International Conference on System Sciences*, Maui, pp. 5289-5298, 2012.

[9] Henard C., Papadakis M., Perrouin G., Klein J., Heymans P., and Le Traon Y., “Bypassing the Combinatorial Explosion: Using Similarity to Generate and Prioritize T-Wise Test Configurations for Software Product Lines,” *IEEE Transactions on Software Engineering*, vol. 40, no. 7, pp. 650-670, 2014.

[10] Ieamsaard C. and Limpiyakorn Y., “On Integrating User Acceptance Tests Generation to Requirements Management,” *in Proceeding of International Conference on Information Communication and Management IPCSIT*, pp. 248-252, 2011.

[11] Janzen D. and Saiedian H., “Does Test-Driven Development Really Improve Software Design Quality?,” *IEEE Software*, vol. 25, no. 2, pp. 77-84, 2008.

[12] Kaur R., Kaur P., and Bahl K., “Acceptance Testing of Webapplication Using Jmeter,” *International Journal of Innovative Science, Engineering and Technology*, vol. 3, no. 4, pp. 353-355, 2016.

[13] Khan M. and Khan F., “A Comparative Study of White Box, Black Box and Grey Box Testing Techniques,” *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 6, pp. 12-19, 2012.

[14] Kim H., Ahmad A., Hwang J., Baqa H., Le Gall F., Ortega M., and Song J., “IoT-TaaS: Towards

a Prospective IoT Testing Framework," *IEEE Access*, vol. 6, pp. 15480-15493, 2018.

[15] Kochhar P., Thung F., and Lo D., "Code Coverage and Test Suite Effectiveness: Empirical Study With Real Bugs in Large Systems," *in Proceeding of IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering*, Montreal, pp. 560-564, 2015.

[16] Latorre R., "A Successful Application of A Test-Driven Development Strategy in the Industrial Environment," Empirical Software Engineering, vol. 19, no. 3, pp. 753-773, 2014.

[17] Leotta M., Clerissi D., Ricca F., and Tonella P., "Capture-Replay Vs. Programmable Web Testing: an Empirical Assessment During Test Case Evolution," *in Proceeding of 20th Working Conference on Reverse Engineering*, Koblenz, pp. 272-281, 2013.

[18] Li Z., Harman M., and Hierons R., "Search Algorithms for Regression Test Case Prioritization," *IEEE Transactions on Software Engineering*, vol. 33, no. 4, pp. 225-237, 2007.

[19] Liskin O., Herrmann C., Knauss E., Kurpick T., Rumpe B., and Schneider K., "Supporting Acceptance Testing in Distributed Software Projects with Integrated Feedback Systems: Experiences and Requirements," *in Proceeding of IEEE 7th International Conference on Global Software Engineering*, Porto Alegre, pp. 84-93, 2012.

[20] Mårtensson T., Ståhl D., Martini A., and Bosch J., "Efficient and Effective Exploratory Testing of Large-Scale Software," *The Journal of Systems and Software*, vol. 174, pp. 110890, 2021.

[21] Mei H., Hao D., Zhang L., Zhang L., Zhou J., and Rothermel G., "A Static Approach to Prioritizing JUnit Test Cases," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1258-1275, 2012.

[22] Minhas N., Petersen K., Börstler J., and Wnuk K., "Regression Testing for Large-Scale Embedded Software Development-Exploring The State of Practice," *Information and Software Technology*, vol. 120, pp. 106254, 2020.

[23] Nomura N., Kikushima Y., and Aoyama M., "Business-Driven Acceptance Testing Methodology and its Practice for E-Government Software Systems," *in Proceeding of 20th Asia-Pacific Software Engineering Conference*, Bangkok, pp. 99-104, 2013.

[24] Nuzha A. and Meenal H., "Framework to Software Testing and Types," *International Journal of Research*, vol. 05, no. 21, pp. 458-465, 2018.

[25] Pandit P. and Tahiliani S., "AgileUAT: A Framework for User Acceptance Testing based on User Stories and Acceptance Criteria," *International Journal of Computer Applications*, vol. 120, no. 10, pp. 16-21, 2015.

[26] Rafique Y. and Misic V., "The Effects of Test-Driven Development on External Quality and Productivity: A Meta-Analysis," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 835-856, 2013.

[27] Rani P. and Mahapatra G., "Entropy Based Enhanced Particle Swarm Optimization on Multi-Objective Software Reliability Modelling for Optimal Testing Resources Allocation," *Software Testing, Verification and Reliability*, vol. 31, no. 6, 2021.

[28] Shala B., Wacht B., Trick U., Lehmann A., Shala B., Ghita B., and Shiaeles S., "Framework for Automated Functional Testing of P2P-Based M2M Applications," *in Proceeding of 9th International Conference on Ubiquitous and Future Networks*, Milan, pp. 916-921, 2017.

[29] Uusitalo E., Komssi M., Kauppinen M., and Davis A., "Linking Requirements and Testing in Practice," *in Proceeding of 16th IEEE International Requirements Engineering Conference*, Barcelona, pp. 265-270, 2008.

[30] Verma A., Khatana A., and Chaudhary S., "A Comparative Study of Black Box Testing and White Box Testing," *International Journal of Computer Sciences and Engineering*, vol. 5, no. 12, pp. 301-304, 2017.

[31] Yu B. and Pang Z., "Generating Test Data Based on Improved Uniform Design Strategy," *Physics Procedia*, vol. 25, pp. 1245-1252, 2012.

Natarajan Sowri Raja Pillai is born in Puducherry, India and born on 2.07.1981. He is currently working as Head of the Department Information Technology and Placement Officer at Raak College of Engineering and Technology, Puducherry. He has completed his B.E. Electrical and Electonics Engineering in 2004 from Arunai Engineering College and completed M.Tech. Information Technology from St.Peters University. He completed his Ph.D. from St. Peter's Institute of Higher Education and Research, Chennai on Software Engineering in 2021. He is also completed MBA Systems from Annamalai University. He has been serving different institutions for the past 14 years at various levels and involved in software testing, teaching, student counseling, Training and Placements for the past 8 years. His specialisation is Software engineering, Data mining, Software Testing and upscaling technologies. He has published more 10 papers in the national and international journals. Dr. Sowri Raja Pillai is senior grade member in the Software Engineering field and has received several appreciations from Institute and society as well.

Ranganathan Rani Hemamalini is born in Kumbakonam, Tamilnadu, India and born on 15.01.1969. She is currently working as Professor and Head in the Department of Electrical and Electronics Engineering at St. Peter's Institute of Higher Education and Research, Chennai. She has completed B.E. Electrical and Electronics Engineering in 1990 from Alagappa Chettiar Government College of Engineering and Technology, Karaikudi and Completed M.Tech and Ph.D. in National Institute of Technology, Tiruchirappalli in Controls, and Instrumentation Engineering. She has been serving in the field of teaching for the past 30 years at various levels. She received BOYSCAST FELLOWSHIP award from DST and received Air India BOLT (Broad Outlook Learned Teacher) award from Air India. Under her guidance 10 candidates completed Ph.D. and guiding 13 Ph.D. students. She carried one DST project, Two AICTE and one MoEFCC sponsored project related to control engineering with total cost of Rs. 100 lakhs. She has Organised more than 30 Seminars/Conference/ workshop/FDP for engineering faculties which are sponsored by AICTE, DST, CSIR, BRNS, ICMR and DRDO. She has published more than 80 papers in the national/international journal and conferences. She is member of the Institute of Engineers (India), ISA, ISTE and IEEE.