Bachelor's thesis

Information Communication and Technology

2018

Kalyan Giri

UPDATING A TEST PLAN FOR AGILE DRIVEN SOFTWARE DEVELOPMENT

A Case Study

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Communication and Technology

2018 | 37 | 11

Kalyan Giri

UPDATING A TEST PLAN FOR AGILE DRIVEN SOFTWARE DEVELOPMENT

A case study

The goals of the thesis were to analyse the current software testing process of a company and to produce a Modular Test Plan on the basis of a survey and a case study of the company.

The theoretical part focuses on the detailed study of the current test plan in the company. Similarly, it also presents and analyses agile testing methodologies by studying relevant literature.

The practical part, however, identifies the challenges faced by the company during the testing process from the developers' point of view and provides an updated test plan for the company. To achieve this, an online survey was distributed amongst the development team of the company to validate their agile development process.

The survey results indicated that the company does not strictly follow agile testing methodologies for the smaller projects. Moreover, the projects have an insufficient number of testers for quality testing. Similarly, the test process followed is outdated since test automation tools are not being used. Also, the documentation is weak, and the interaction with customers is not sufficient.

The result of this thesis was a new and updated modular test plan that eliminates the above-identified issues and will be used for the next few years. However, in the future, further extensive research needs to be conducted, and a new test plan should be formulated according to the company's status at that moment.

KEYWORDS:

Waterfall, Agile, Test Plan, Continuous Integration, Agile Testing, Scrum

CONTENTS

APPENDIX

FIGURES

TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
ATDD	Acceptance Test Driven Development
BDD	Behaviour Driver Development
CEO	Chief Executive Officer
CI	Continuous Integration
CMS	Content Management System
IoT	Internet of Things
NFC	Near Field Communication
PHP	Hypertext Preprocessor
PM	Project Manager
PMT	Project Management Tool
PO	Project Owner
QA	Quality Analyst/Assurance
R&D	Research and Development
SDLC	System Development Life Cycle
SME	Small and Medium Enterprise
UAT	User Acceptance Testing
UI	User Interface

1 INTRODUCTION

Several factors determine the quality of a product: the amount of time spent, materials/machines used, people's skills, money invested, and so on. It is crucial for any software to pass the quality test before being delivered to the end customers. The test verifies the quality and future of the product which is essential for the manufacturers, the owners, the designers, and the developers.

Software Testing is a process of executing a program with the intention of finding a problem, also known as a bug. Today, there are several tools and processes available to ensure the quality of the product in the software development business. Among them, the agile software development process is the most popular, mostly, due to its history of successful results. Numerous companies and industries have been successful in flourishing their business due to the agile development process.

The purpose of this thesis is to create a new test plan for the testing phases of the company's software development process. Chapter 1 introduces the thesis and its Commissioner. Chapters 2 and 3 of this thesis include qualitative research on agile methodologies and software testing to gain insight into agile software development and testing processes.

Chapter 4 analyses the positive and negative aspects of the current test process and test plan of the company. In Chapter 5, a survey helps to understand the development and testing process of the companies and to measure the agile testing skills of the testers.

Chapter 6 of the thesis provides a new and updated modular test plan with a few recommendations on the testing process and tools of the company. The seventh chapter concludes the thesis by providing a summary of the project.

1.1 Case Company's Background

The case company is a mobile and web application development company. Back then, the company's key focus was responsive and CMS based website development. After receiving positive feedback from different clients, the private company changed itself to limited liability company. Currently, the company is providing a technical solution to small and medium-sized businesses around the Nordic region. The company specialises in mobile, web, interactive screen and even IOT solutions for various industries, such as the pharmaceutical industry, with the aim to provide automation on medical facilities in Finland. It has worked in mobile and web solutions for real estate, e-commerce and many more.

Currently, the company is working on projects involving NFC, beacons and different IOT based applications. There is a team of 27 in Nepal which includes developers, designers, the Project Manager, QAs, and Testers; and a team of 3 in Finland. The team in Finland focuses more on R&D and scaling up the business, whereas most of the development happens in Nepal.

The Company's software development process is an agile development process. They have always mentioned it during the introduction of the company to new trainees and customers before the start of new projects. Of recent, the company is facing some challenges like late delivery, critical bugs, negative customer review and long project completion time.

2 AGILE SOFTWARE DEVELOPMENT AND METHODOLOGIES

Software development is an iterative logical process that aims to create a computer coded or programmed software to address a unique business or personal objective, goal or process. Software development is generally a planned initiative that consists of various steps or stages that result in the creation of operational software. (Techopedia.com 2018).

Agile software development/engineering is a combination of a philosophy and a set of development guidelines. Agile is a project management methodology having short development cycles called "sprints" to target continuous improvement in the development of a product or service (Alexander 2018). It focuses on client satisfaction with early delivery, motivated team, informal working methods, and simplicity. It also encourages active and continuous communication between software developers and customers. Software engineers and other venture partners practice it, for instance; directors/managers, clients, and end users.

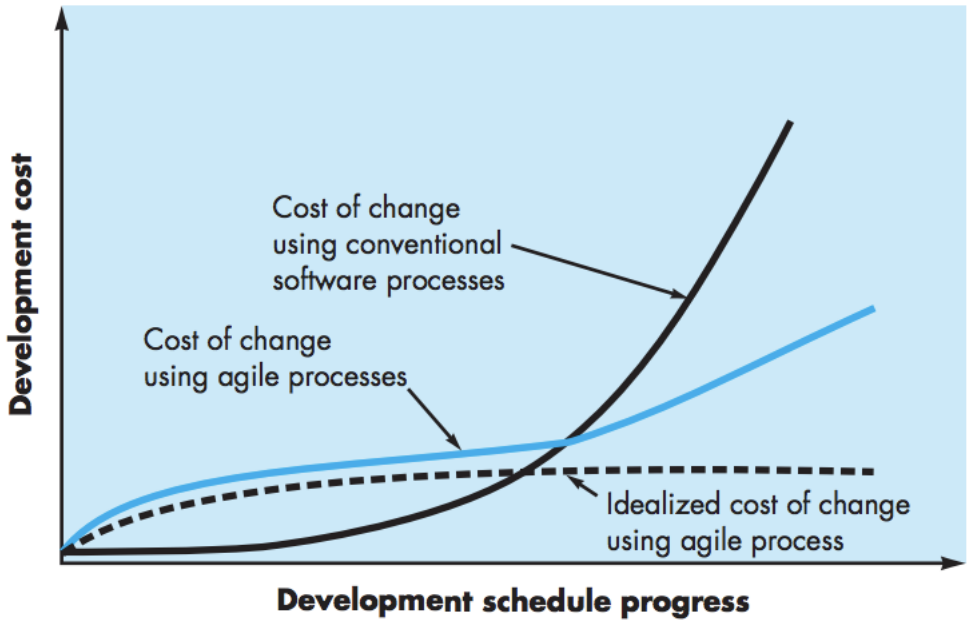

Figure 1. Agile Software Development schedule progress (Pressman 2014, 69).

Agile development is productive mainly from two perspectives; Development cost and Development progress. In Figure 1, comparing the cost of change using agile process with Cost of change using conventional software process, the value and importance of agile driven software development can be identified. Taking an example of a software ABC, which is in the validation testing phase, and one of the product owners wants a significant functional change which requires substantial component development, new

designs, and API integrations. Agile states that if the product development strictly follows the agile process and its methodologies, then, as shown in Figure 1, it allows a development team to push changes efficiently without affecting the time. (Pressman 2014).

Agile methodology is a strategy used while developing a software. Testing services have a critical role in the production and success of software. Agile methodologies do the fragmentation of the whole project by listing essential features and divide output into several cycles. By doing this, it helps the development team to deliver a product, inefficient manner, through short, interactive and iterative session. The process involves having large numbers of a short delivery cycle, which helps in receiving the feedback and make a product better and better in every development cycle. (Alexander 2018)

2.1 Scrum

Scrum is a subset of Agile. Scrum is a widely used, agile product development strategy- a collection of rituals, values and team roles implemented in combination to create iterative work products (Morris 2018). The iterative development process is suitable for large and complex projects. These kinds of projects need to be fragmented and divided by features priority. There is always a fixed length iteration called a sprint, which lasts from 1 week to 2 weeks or more. It is the period given to developers, designers, and team members to produce something by the end of it. After reviewing the product again, team members plan for next sprints, and this goes on until the final product is released. Scrum has four stages for every sprint, i.e., sprint planning before starting the sprint, daily-standup during the sprint to know what team is working on, sprint demo after finishing the sprint to show what has been done and finally, the sprint retrospective. Different materials like charts/graphs are used to indicate the progress of the plan.

According to Denning (2015), Scrum is the most popular process for implementing agile in any project. Daily standup and meetings make it easier for product owner and team members to predict their product future. Every member has a clear vision of who is doing what. This eliminates any misunderstanding and helps find issues, which can be resolved before it hampers the whole project. It helps to provide the report to understand the budgeting of the project. It also lets the user see results in every sprint which is very important for testing and end user points of view because they are the decision maker. Every small mistake and issue can be resolved early on before they become too big and expensive to fix. Scrum teams always have the tasks divided by the group themselves, and no project manager is dividing the responsibility in the scrum. Scrum emphasises teamwork and helps to empower each other to become independent.

Despite having benefits, Scrum has some shortcomings as well. There is always a small problem, which due to feasibility in the project, a user might always ask for continuous

changes and modifications which will delay the product date and increase expenses. Proper experience and adequate knowledge about the agile development process is required in the team to run the Scrum process smoothly. It needs the regular involvement of individual team members for the daily standup.

2.1.1 Bug Correction process in Agile/Scrum

In the Agile software development process, every project is divided into small fragments called sprints. The primary reason to have these sprints is to find out the status of the project and test the delivery. A bug is a flaw which prevents the application from functioning correctly. It must be fixed immediately during the sprints before it evolves to a more significant issue and hampers the whole product requirement, making it unacceptable. To reduce the bugs, the team must work together from the beginning of the project. There is no bug-free software/product, and achieving it is almost impossible. Every bug correction process must follow Scrum. Scrum defines that when a bug is found, it must be treated as product backlog and put in following sprints according to the priority. (Sherman 2017)

2.1.2 Roles of testers in Scrum

In Scrum, there are three significant roles Product Owner, Scrum Master, and a Team (Amarinfotech 2018). The product owner is the decision maker; Scrum Master is a maintainer, and a Team is a group of the designers and programmers. In a formal Scrum process, there are no testers. Depending upon the project there are dedicated testers in some Scrum projects. Testers are responsible for following activities during the scrum.

Sprint Planning

During sprint planning, the testers are responsible for picking user stories from the product backlog for testing. They then should estimate the time, which gives a clear idea about how long it will take to finish the testing (Bartlett 2015). The testers are also responsible for prioritising the user stories and must have an idea about the goal of the project. (Guru99 2018)

Sprint

The testers are responsible for various activities during the sprint. This includes helping a developer during unit testing and also validating user stories because it is preferable to do it together than the developer alone. This helps save time and money. The testers are responsible for managing the backlog and prioritising user stories. It also includes analysing the backlog tasks and putting the uncompleted task to the next sprint. In every sprint, testers write automation scripts to perform automatic testing. This ensures that everything has passed through testing. After testing, the testers have to analyse the result and submit a report to the product owner or stakeholders. The testers also perform different non-functional testing. They are also responsible for discussing with the product owner regarding the acceptance criteria for the acceptance test. Thereafter, the testers also perform UAT and declare the end of the sprint. (Guru99 2018)

Sprint Retrospective

In the end, the testers review the full sprint, note the feedback from the stakeholders, and analyse what worked well and what did not work well in the sprint. It is a time to explain the weakness and determine solutions for them (Bartlett 2015). Hence, they will declare the end of a sprint and let the developers, or the scrum master, know when to start a new sprint. (Guru99 2018)

3 AGILE SOFTWARE TESTING

Agile testing is a software testing technique which strictly follows the agile methodologies. It has several advantages over typical testing methods. In agile, test cases are written before coding in order to prevent future bugs and to know the correct timing for testing. It saves time and money and does not need much documentation because of using Scrum. In agile, there is continuous customer feedback which helps minimise any errors, and a daily meeting helps determine the issues well in advance. Every agile tester must have explicit knowledge of agile software development. Agile testing is not a phase because it is continuous, and testing happens continuously throughout the development process. Agile testing moves the project forward which is quite the opposite to conventional method because in conventional method testing is regarded as a quality gate, but agile provides continuous feedback on an ongoing basis until the product meets the demands.

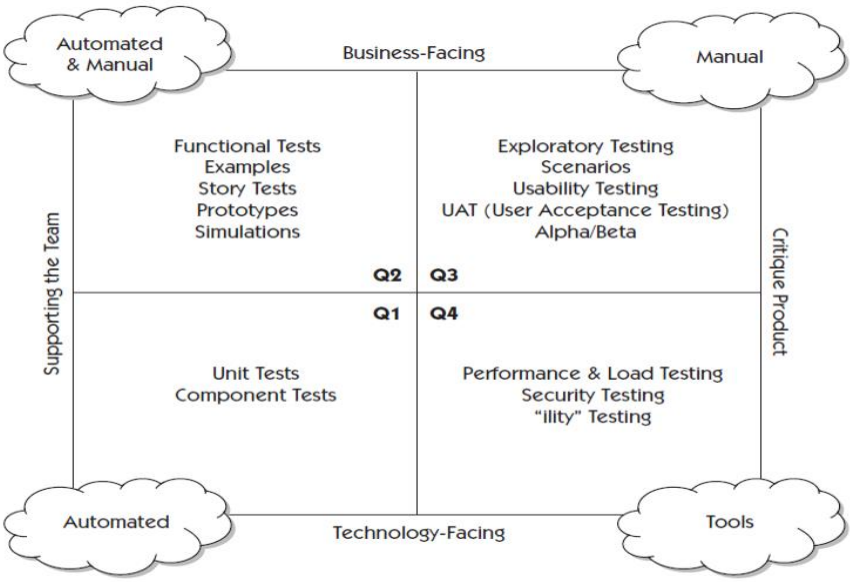

Figure 2. Agile Testing quadrants (Crispin and Gregory, 2014).

Figure 2 shows the agile testing quadrants. Agile testing quadrants help to categorise the different type of testing that needs to be performed. Q1 is linked with Automated testing and covers the tests such as Unit Tests, Component Tests and API tests. The primary objective is to enhance the Quality of the product by proper management of source code. Q2 is either manual or automated. Functional tests, story tests belong to this quadrant. Skilled testers collaborate with clients to understand the delivery of the product. Q3 is associated with manual testing such as Exploratory testing, Usability testing and UAT testing. It is all about verifying the product. Q4 covers test such as Performance and load testing, Security testing and ility testing – Stability, Reliability and Scalability. It is related to automated testing. Tools such as Jmeter, jConsole is used for this testing. (Cigniti 2016)

In agile software development, everyone is a tester, and he/she performs different types of tests like programmer/developer performs unit testing, performance testing whereas marketing and end customer do Black box/acceptance testing. If there are problems in the code or some features, the sprints help to identify and fix them right away. The daily standup and every two-week sprint help to minimise documentation. The significant difference between agile testing and conventional testing is that, in traditional testing; testing is performed after execution while in agile testing, testing is done during execution. Agile testing always supports testing by priority because it is not possible to test everything, and every sprint is related to each other which helps to adopt the changes.

3.1 Agile Testing Methods

Agile testing gives priority to the early involvement of testers in the development process to have a high level of insight into the requirements and goals. It helps to encourage collaboration of testers, development team and product owner. Agile testing focuses on frequent testing which might slow down the overall testing process.

3.1.1 Behaviour Driven Development (BDD)

Behaviour Driven development starts with an initial requirement collection based on end user and performs a test that is human readable. It begins with a functional specification using the Gherkin (Given, When, Then) syntax (Knight 2017). It helps to guide the developers, testers and product owners to move forward with other features. BDD requires an automation strategy that results in high-level efficiency. BDD is different from Waterfall testing because Waterfall testing involves test cases to be written early on, and perform it during the end of the development cycle, whereas BDD especially focuses on testing along with the development of the feature. BDD is for a project which is more focused on product features and has to prioritise user experience as a critical requirement.

In Waterfall software testing, it is testers who are responsible for writing test cases, but in BDD approach it is the business owner/product owner who writes a test case. BDD methodologies require some skills which should be learned by the business analyst like using Gherkin synthesis to write test cases directly. It is also necessary to know how to use different testing tools like Cucumber to define criteria. All the documents should be well managed and streamlined to keep the entire process lean. (Carmichael 2017)

Table 1. An example of a feature definition in BDD.

Feature:	add an item to cart *(As a user, I must be able to add an item to the shopping cart once I click add to cart.)*
Scenario:	item added successfully
Given	the item page is opening
When	I select the item and click Add to cart
And	I click to cart
Then	Item is in the cart added successfully

3.1.2 Acceptance Test Driven Development (ATDD)

In Acceptance Testing, the test is created first, and then the code is written with an intention to pass the test. Compared to Test Driven Development, in ATDD tests are typically customer-facing acceptance tests. The idea behind this type of development is that the user's view of the product is as crucial as the functionality. In ATDD, all the necessary input from the customer is collected and then, the same data is used to create the Acceptance Criteria. To create different manual and automatic test cases, the same Acceptance criteria is used and finally coding starts. All in all, ATDD is a test first approach, not a requirement driven process. ATDD helps minimise the significant areas for misunderstanding by removing the condition of the developer to define the product's future (Esquivel 2018). In comparison to BDD, ATDD directly focuses on customer views on how the product will be used. Since there is a direct connection between development and customer use, it prevents the need for re-design and many new releases.

ATDD is quite challenging compared to other methods since it requires frequent customer engagement. It is best suited for those teams where the customer is equally engaged as a developer and focuses more on user experience and customer adoption. Every single user story should always have questions like "Will a customer use this system like this?". Some popular tools for ATDD are FitNesse, Cucumber, JBehave, Concordion and Easyb. In conclusion, the ATDD approach provides clear goals for a developer to understand how the product is going to be and how they need to work. (Carmichael 2017)

3.1.3 Exploratory Testing

Exploratory software testing is a unique and friendly approach to testing. Testers do not follow any test steps or documentation but use an app in clever ways to break it. Every stage of testing is not documented; testers only keep a record of defects. In exploratory testing, planning, designing and execution is done at the same session, which makes it easier to produce a report. It is quite flexible since the testers can always think of themselves as an end user and test it like them. In this kind of testing, the testers can investigate and look for other opportunities, in comparison to Manual Script Testing which is strict and focused on documentation explaining steps and direction. (Carmichael 2017)

Adopting Exploratory testing is relatively easy and quick. It makes it easier for testers to find more defects in a short period. It is best suited for development teams which are under a time limit that need to identify the tests to run without any support from developers and organisations who want to be sure that they do not miss anything during the trial (Kohl n.d.). It requires highly skilled and experienced testers since it is dependent upon inspectors.

3.1.4 Session-Based Testing

Session-based testing is based on exploratory testing; however, it is more structured and organised. It helps to reduce the complete dependency on testers for testing. This kind of testing has a specific time frame, focused testing session and frequent testing reporting. It also needs to cover five PROOF points: Past, Results, Obstacles, Outlook and Feelings (Eriksson 2015). PROOF explains what has happened in the past, what was achieved, what are the obstacles to the process, what needs to be fixed and the overall feeling regarding the testing. Like Exploratory testing, this testing is also easy to adopt and launch. Session-based Testing best suits a team that is quite handy with Exploratory testing but faces challenges like unorganised and wants to improve accountability through the process. This testing needs the involvement of testers and managers. In Session Based testing, testers are very clear about the software they are testing and the need for the proper documentation. The involvement of the manager helps to overview the testing status and the development process. (Carmichael 2017)

3.2 Automatic Software Testing

Software development practices change over time, so do tools and technologies. These sorts of changes aim for better productivity, quality, customer satisfaction, shorter delivery times and a successful business. Software testing is an essential role in achieving these objectives. According to World Quality Report 2017-2018, there has been an enormous increase in test automation and adoption of agile and methodologies (Buenen and Muthukrishnen 2018). In manual testing, a human being is responsible for single-handedly testing the functionality of the software, whereas automated tools perform automatic testing. Automated testing is well suited for larger projects which need a recurring testing mechanism. Automation tools perform a progression of preplanned situations with expected outcomes. They will log a failure and save it for a human to check and decide. Professional bug tracking software not only tracks bugs but also produces a report, severity, priority, date of creation and a mechanism to solve it. It is essential to understand that Test Automation demands a considerable investment of money and resources.

Test Automation is the best way to enhance the effectiveness, efficiency and coverage of the testing. It fulfils all the gaps created by manual software testing for the testing team. Automated testing helps testers test for the same issue again and again, which is impossible with manual testing. Automated testing also increases test coverage since, in manual testing, lengthy and repetitive testing is always compromised. It can run on multiple computers at the same time going through its memory contents, file contents and internal program states to check if the product is producing a result as expected or not. It allows testers to run thousands of different test cases, thus saving much time. Automated testing also helps the developers to catch and fix the issues before the product goes through Quality Analyst. Since it is easy to perform repetitive testing with automated testing, the team can focus on the more challenging tasks and increase the confidence of individual team members. (Smartbear 2018)

3.3 Continuous Integration

Continuous Integration is a widely established development practice in the software development industry, in which members of a team integrate and incorporate development work (e.g., code) frequently (Shahin, Ali Babar and Zhu, 2017). It includes automated software building and testing. It does not get rid of bugs; however, it makes it easy to detect and get rid of them.

CI helps to continuously integrate the changes and runs a set of the tests by the instructions fed to the CI. It emboldens developers to share their code and unit tests by combining their changes into a shared version control repository after small changes are made, or the task is completed. Committing change starts an automated build system, which grabs the latest code from the shared repository and builds, tests, and validates the full master branch (Guckenheimer 2017).

Jenkins server is an open-source and easy-to-configure CI tool. It offers an understandable way to set up CI for almost any language and source code repository using pipelines, as well as automating other routine development tasks (Heller 2017). However, Jenkins needs some pre-configuration like creating a script for individual steps.

4 TESTING OVERVIEW

4.1 Current Testing Process of the Case Company

Different types of agile test practices are applied in the company. In order to understand how these test practices are applied, it is necessary to understand the testing process. The Testing process starts from planning, followed by designing test cases, preparations for execution, and evaluating the status till test closure.

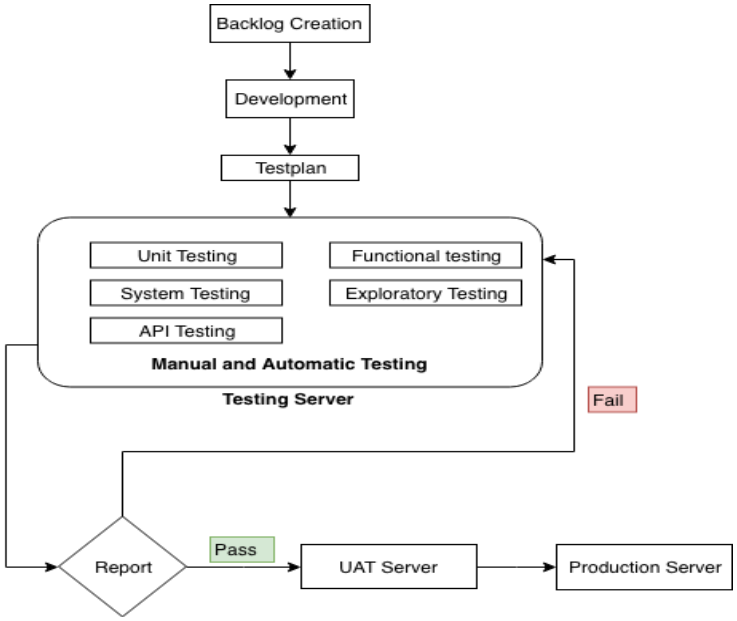

Figure 3. Agile Test Process Overview.

Figure 3 shows an overview of the software testing process of the company. A Project Management Tool, called JIRA is used in order to create a backlog. It contains all the tasks related to the project. The backlog is created after the requirement collection phase. A set of relevant information is provided to the developer, and on the basis of user stories, developers start to code and release the result in every two weeks because of the Scrum development process.

Along with development, the QA and PM start to create the test plans for the project. The Scrum master conducts a workshop if any issues arise in the development and testing process, to make sure that the team is working in the agile development process. While making a test plan, the QA verifies the tools, testing server, types of test reporting process, audiences, test principles and resources which are then further verified by the PM. This process also involves the Scrum master and developers in order to prioritise

the features to be tested and released first. Then, one of the testers works closely with the developers and the Project Manager to write a test case for performing functional testing. The test case is also used for customer validation and UAT validation.

Automation experts also get involved in the workshop to initialise the automation process for the project. Most of the testing is performed automatically to save effort and time with the help of different automation tools like PHPUnit, Postman and Selenium to automate Unit testing, API testing and Exploratory testing. These tests are run in the Testing Server Proper planning leads to a successful result, and test plans play a vital role in software testing, so preparing a quality test plan is one of the most challenging and time-consuming processes.

It is the QA and developers' job to verify the result of each testing. They are responsible for analysing the report and making a decision. If the test passes, it goes to the UAT server for User Acceptance Testing. There, PO and end users are provided with the test cases in order to perform a test for Quality Assurance. If the test fails, it will remain in the test server, and the developers start to go through development/coding to fix the issues. Once the customer verifies the product from the UAT Server, it is pushed to the Production Server. Every sprint meeting, the development team tracks their progress to identify the weak points.

4.2 Testing tools

In the list below, brief descriptions of the tools are given with their functionalities and recommended version or other alternatives.

Unit Testing: PHPUnit

The company mostly works with PHP on the backend and Angular on the frontend. PHPUnit is an open source, programmer-oriented testing framework and it is one of the more popular tools for writing unit tests for PHP. PHPUnit is a framework independent library for unit testing PHP (Carter 2018). It is an automated testing framework solely for PHP. It can be used via the command line and comes with a test class which can be customised according to the requirement. The best-case scenario for a good result is testing code should be written before the application code.

Recommendation: Upgrade to the latest version because the current version will stop receiving bug fixes at the beginning of 2019 (Bergmann 2018).

API Testing: Postman

Postman is an easy-to-use Rest client that is available as a Chrome plugin with native versions for both Windows and Mac. It helps make API development faster and better with its Graphical User Interface. It provides detailed API documentation and saves the history of API request (Colantonio 2017). It is easy to share since a developer can package all the requests and send it to another developer for review. The company has used Postman in most of their API-centric project development.

Recommendation: Katalon Studio because it supports both SOAP and REST APIs whereas Postman handles only REST requests (Aldaine 2018).

PMT Tool: JIRA

JIRA is a dominant issues tracker and agile project management tool developed by the Atlassian team. It is a single app with both features. Using Jira is helping the company plan and organise the tasks, workflow, documents and reports for the agile team (Confluence.atlassian.com 2018) more efficiently. In the company, the PO has access to the Jira Admin portal and controls access and permissions, which helps to have control over the development team.

Recommendation: None

4.3 Functional Testing

The company performs functional testing manually by following a test-case. A set of test cases is written on the basis of User Stories and Acceptance Criteria. Then, the testers perform the tests manually in the mentioned test environment. This is all done in the Testing Server. An example of a test case is given in Table 2.

Test cases help the testers from both the Development team and Client side to perform the functional testing of a product. Test Area gives an overview of the specific tested module. The Section shows the related entity. Test Summary provides a summary of the test. Preconditions explain what needs to be ready before performing the testing. Test Procedure explains the steps. Finally, Expected Results describe the outcome of testing. If the outcome matches the expected result, then the test passes; otherwise, it fails.

Table 2. End-user login Functional Test Case.

End-user login Functional Test Case.

Test Area	Login Page
Section	End User
Sub - Section	Content
Test Case ID	Log_01
Test Summary	Verify the content of Login page
Preconditions	A User should receive a link to the login page.
Test Procedure	Open the link on the browser
Expected Results	All the login fields are available 1.Email Field 2.Password Field 3.Login Button 4.Show password 5.Forgot Password Link

4.4 Analysis of the Current Test Plan of the company

The use of the current test plan in the most recent projects was reviewed thoroughly in order to identify possible issues, problems, best practices of the company. This was done by reading the test plan and comparing it to the actual outcome of the project.

The major issues of the current test plan are:

- Non-compliance with agile testing methodologies
- Inadequate involvement of PM in testing
- The length of the testing process due to dependence on customer and stakeholders
- No automation process in testing
- Some technologies are inadequate
- Development of a test plan by unqualified testers
- Documentation missing before and during the testing
- Lack of testing resources
- Inconsistent use of PMT tools, e.g. Google Sheets for test case instead of JIRA

On the other hand, some of the good practices include:

- Adoption of agile development and testing methods
- Developers and QA are working parallelly
- All testing methods are well explained

After analysing the company's test plan, it can be concluded that there is a need for an updated Test Plan.

5 COMPANY SURVEY

5.1 Goals and Methodologies

A survey was designed to get an overview of the different types of agile development and testing methods used by the company, and the challenges they faced while doing so. It also gathered information on the current test plan and its use.

In order to achieve these objectives, a Google Forms survey with 21 questions was sent out via email to the 27 employees of the company; both in Finland and Nepal. A total of 17 employees responded; all of them from the Software Development Field. The survey was created in English because the working language of the company is English. Some of the respondents did not answer all the questions which affected the quality of the research.

5.2 Survey and the Analysis

The survey was divided into four parts. The first part (Question 1-4) included questions regarding the background information of the respondents. The goal here was to understand the roles and responsibilities of the respondents in the company.

The second part (Question 5-8) was only for those respondents who have either never used, or have rejected using Agile methods in any project while working in the company. This part was to find out the main reasons for people rejecting Agile methodologies.

The third part (Question 9-11) was for those who have used Agile Methodologies in the company. This part was to get an overview of the kinds of projects people are involved in using Agile Methodologies. This part plays an essential role in the test plan for introducing other types of testing on the basis of projects.

The fourth part (Question 12-21) was for those who are currently using Agile development and testing Methodologies in their current project in the company. The goal here was to learn about the agile development and testing methods used in their current projects, a division of testing resources for different types of projects, and the respective length of the projects. It also gives an overview of the various challenges faced by the development team while following agile as a software development process. This part of the survey helps in the test plan for deciding if there is a need for more testing resource or not.

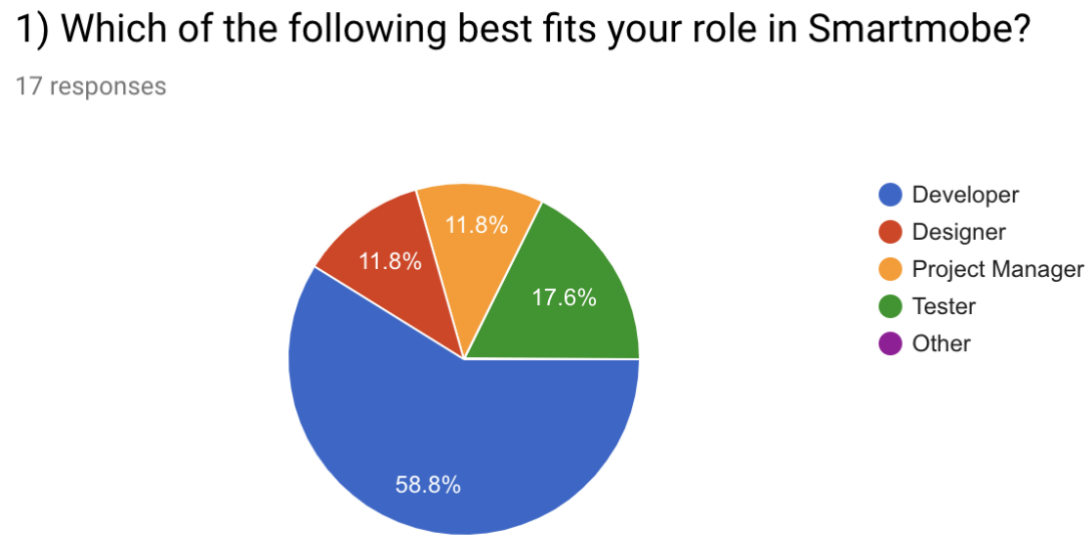

Figure 4. Roles of respondents.

Question 1 in Figure 4 asked the employees about their role in the company. Its purpose was to find the professional diversity of the company. The options for the answers included Developer, Designer, Project Manager, Tester and others. From the responses, it is evident that more than half of the respondents (58.8%) were developers. Similarly, 17.6% of them were testers and the remaining were project managers and designers in equivalent fraction (11.8% each).

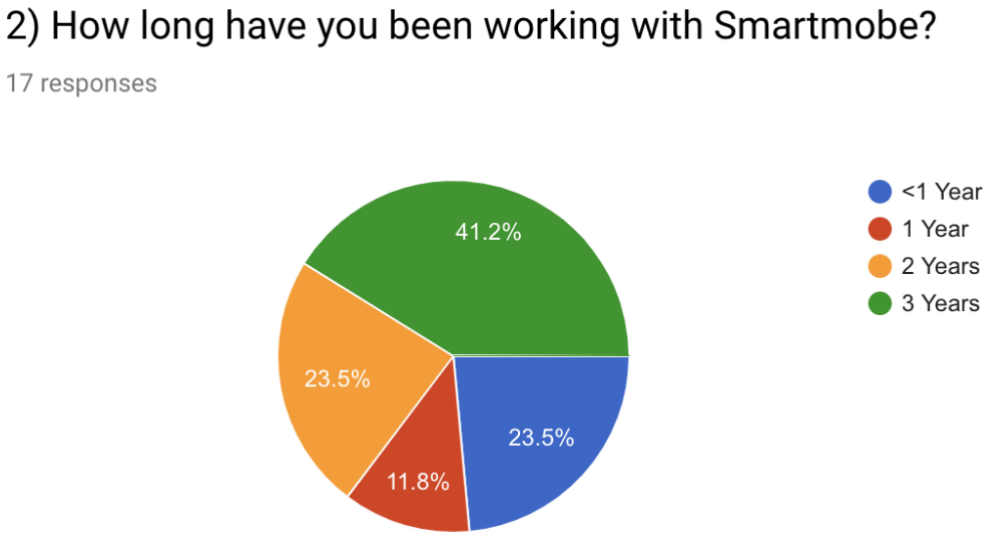

Figure 5. Respondents' work experience with the company.

Question 2 in Figure 5 asked the employees regarding the total time they have worked for the company. The purpose of this question was to analyse the experience of the employees within the company. The answers revealed that the majority of the respondents (41.2%) had joined the company more than three years ago. Moreover, the number of respondents who joined two years ago and those who joined less than a year ago were equal (23.5% each). Similarly, only 11.8% responded that they joined the company a year ago.

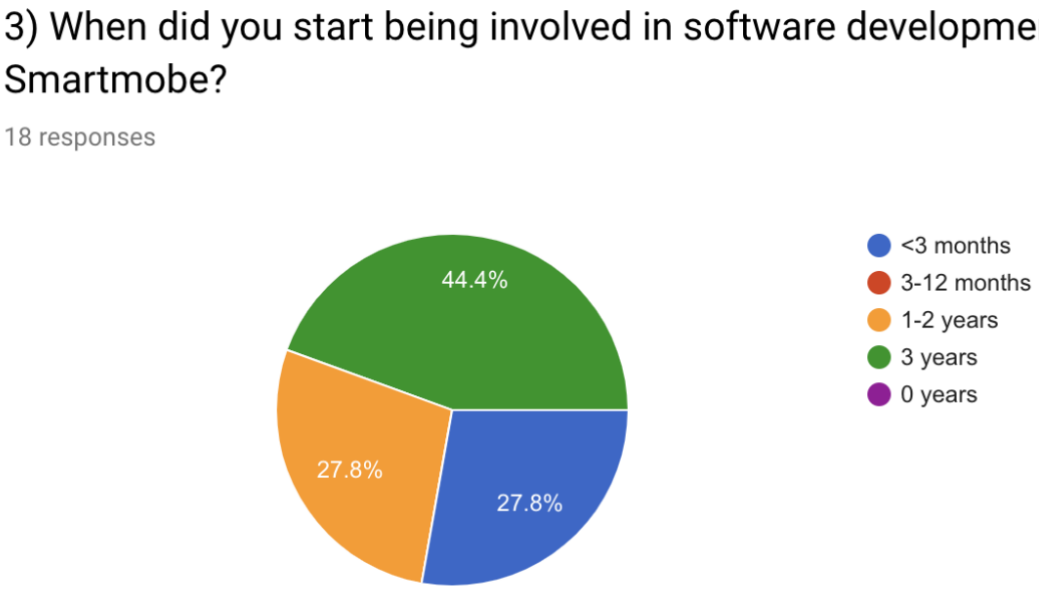

Figure 6. Respondents' experience in Software Development.

Question 3 in Figure 6 asked the respondents when they started being involved in software development in the company. Its purpose was to determine the experience of the respondents in software development projects in the company. The majority (47.1%) responded that they had more than 3 years of experience in working with the software development projects in the company. Similarly, 29.4% of them responded that they had 1-2 years of experience in the same. 23.5% responded said that they had less than 3 months of experience in working with a software development project in the company.

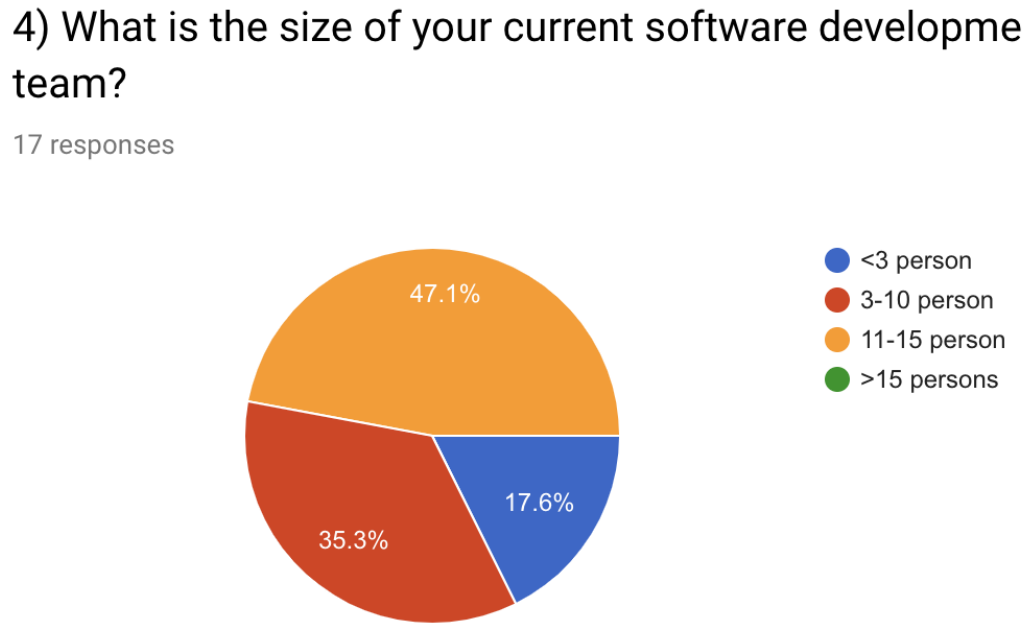

Figure 7. Size of Software Development team.

Question 4 in Figure 7 asked the respondents regarding the size of their current software development team. The purpose here was to determine the average team size in the company. Here, most of the respondents (47.1%) replied that their team size was huge and consisted of 11-15 persons. Similarly, 35.3% of them replied that their team consisted of 3-10 persons. Only 17.6% of the respondents admitted that their team consisted of less than 3 persons.

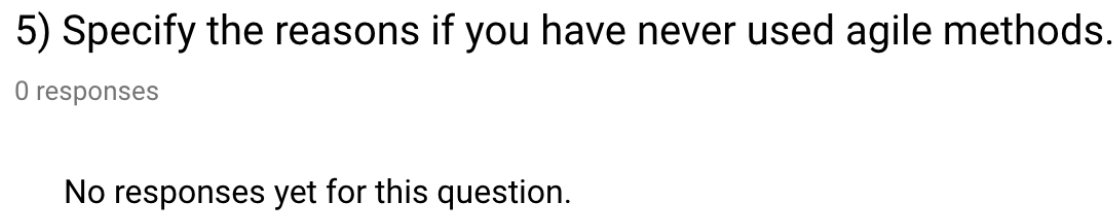

Figure 8. Reasons for not using agile development methods.

Question 5 in Figure 8 asked the reasons for never using agile methods. This was an open-ended question for the respondents to fill in whatever answer they determined best. However, none of the respondents replied to this question. The reason for this could either be that all the respondents have used agile methods or that the respondents did not want to fill in the text to answer the question.

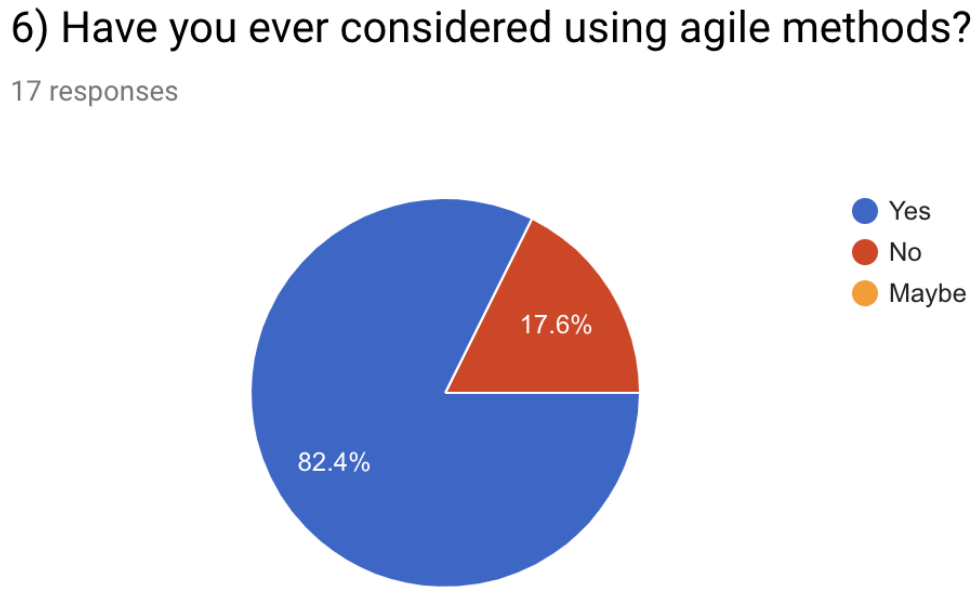

Figure 9. Consideration to use Agile methods.

Question 6 in Figure 9 enquired the respondents if they have ever considered using agile methods if they have not used it yet. Majority of the respondents (82.4%) responded that they have considered using agile methods and only 17.6% responded that they have never considered using agile methods.

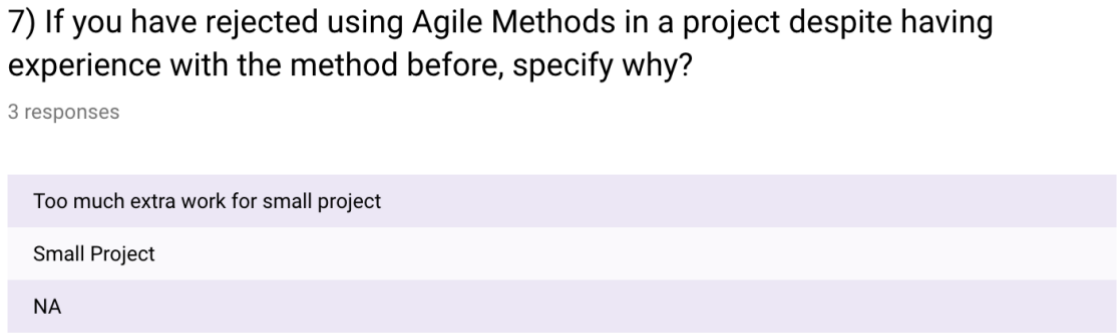

Figure 10. Reasons for not following Agile methods.

Question 7 in Figure 10 targets the employees who have rejected Agile Methods despite having experience before. Only three respondents answered the questions mentioning that some projects are too small to follow agile development methods.

Figure 11. Comments on Agile development methods and tools

Question 8 in Figure 11 asked the respondents to comment on agile development methods and tools. The purpose here was to understand the agile development methods and tools from the employees' point of view. This was an open-ended question where the respondents could answer whatever they wanted to. Most of the responses provided the general knowledge regarding the agile development tools and methods. However, a particular respondent replied that agile development has always benefited his/her company.

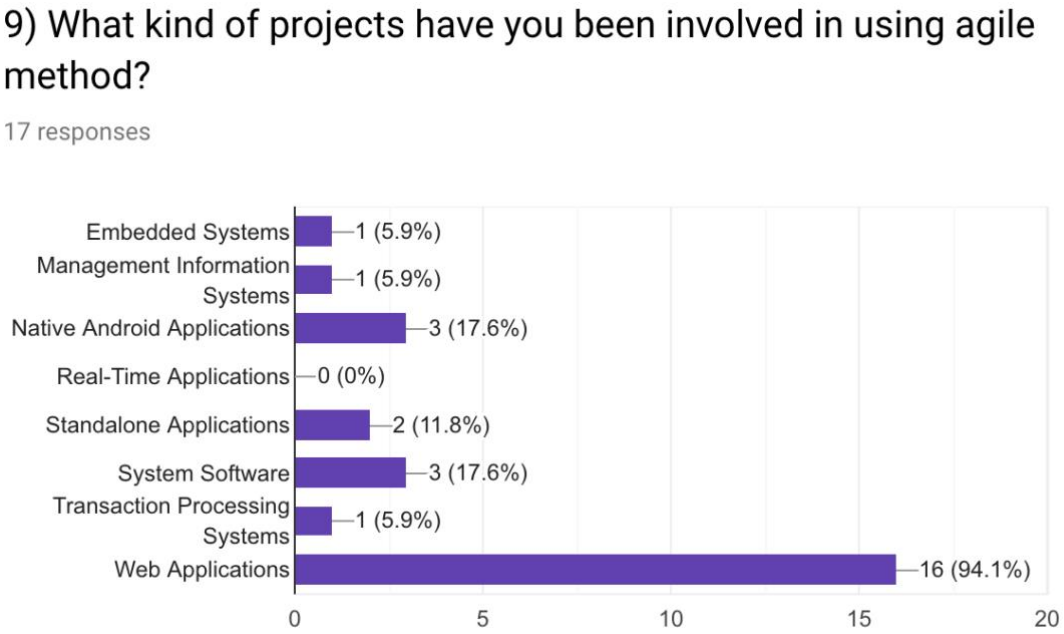

Figure 12. Types of Projects in the company.

Question 9 in Figure 12 asked regarding the different kind of projects the respondents have been involved in using agile methods. The purpose here was to analyse the scope of agile methods in different kind of projects. The answer choices included Embedded Systems, Management Information Systems, Native Android Applications, Real-Time Applications, Standalone Applications, System Software Transaction Processing Systems and Web Applications. The most popular choice was Web Applications with 94.1% of the respondents adhering to it. The second most popular choices were Native Android Applications and System Software with 17.6% each.

Similarly, Standalone Applications were the third most popular choice with 11.8% of the respondents adhering to it. Moreover, the fourth most popular choices were Embedded Systems, Management Information Systems and Transaction Processing Systems with 5.9% each. Real-Time Application was the least popular choice as no respondent chose it.

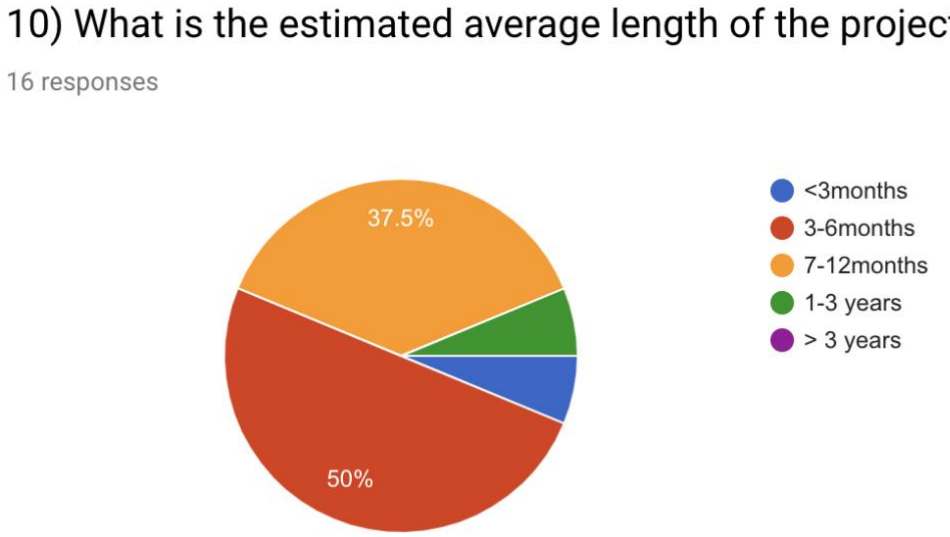

Figure 13. The average length of the Projects.

Question 10 in Figure 13 showed that 50% of the respondents replied that the average length of the project is 3-6 months whereas 37.5% responded that the project could last from 7-12 months. Remaining are either less than three months or more than a one-year project. This question provides an overview of the average project size of the company. It showed that most of the projects are medium projects of 3-6 months.

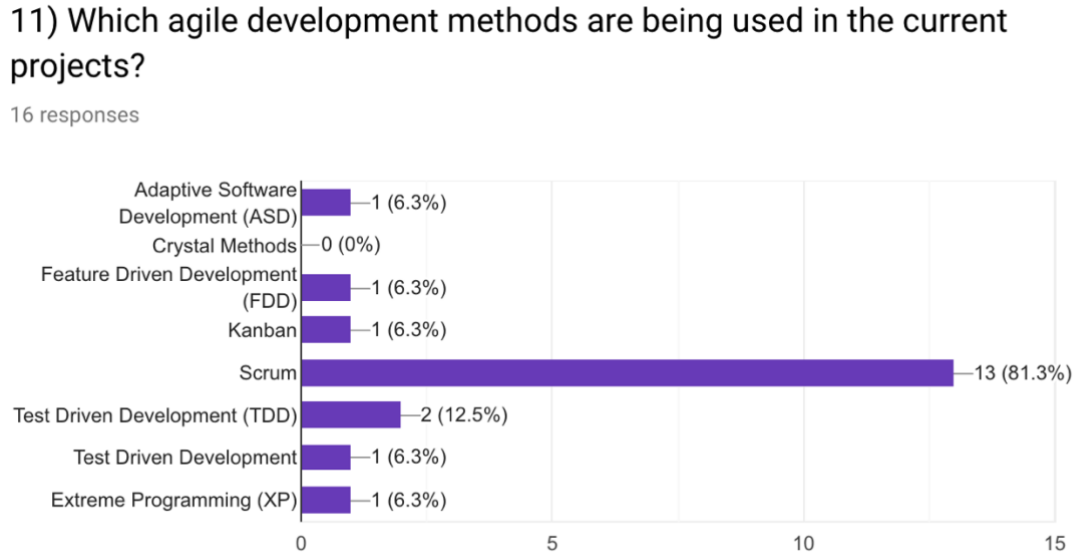

Figure 14. Agile development methods in the company.

Question 11 in Figure 14 enquired regarding the agile development methods used in the current projects. The answer choices were Adaptive Software Development (ASD), Crystal Methods, Feature Driven Development (FDD), Kanban, Scrum, Test Driven

Development (TDD) and Extreme Programming (XP). Scrum was the most popular (81.3%) agile development method. Test Driven Development (TDD) was the second most popular one with 12.5% of the respondents adhering to it. The remaining of the choices except Crystal Methods were equally chosen (6.3% each). Crystal Methods was the least popular answer with no respondent choosing it.

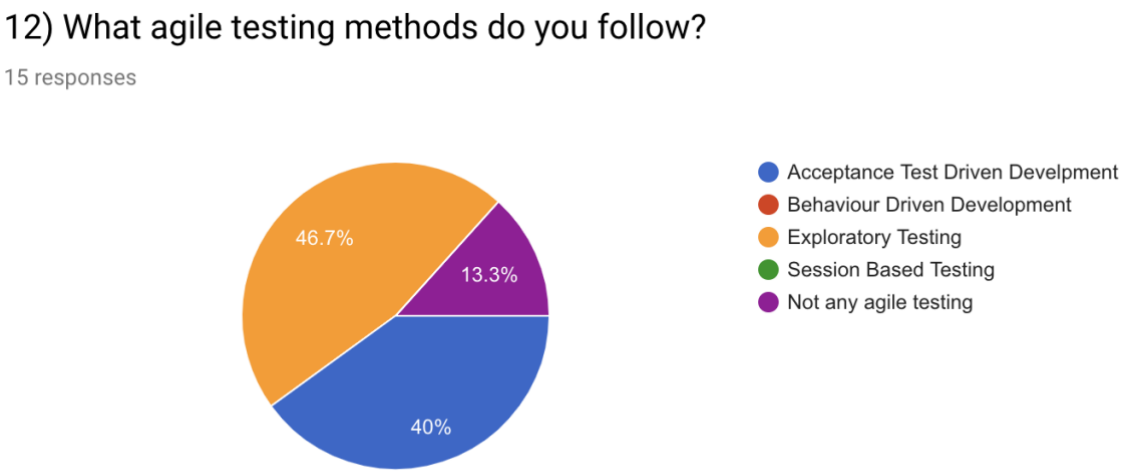

Figure 15. Agile Testing methods in the company.

Question 12 in Figure 15 showed that Exploratory Testing was the most popular (46.7%) agile testing method followed with Acceptance Test Driven Development closely behind (40%). 13.3% of the respondents said that they were never involved in the agile development method. Session-based testing and Behaviour Driven Development are not common in the company because the respondents did not choose them. This question helps to give an overview of the different agile testing methods that the company is following in their projects.

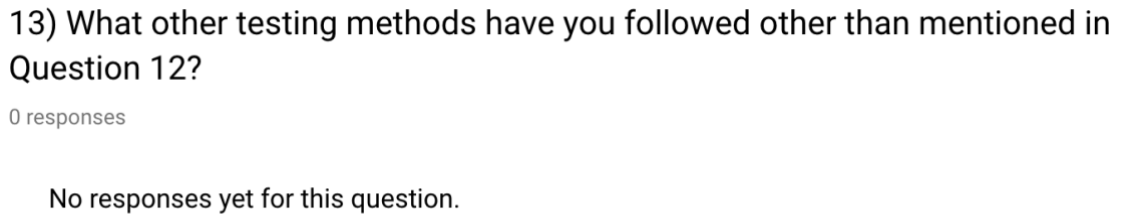

Figure 16. Other testing methods followed by the company.

Question 13 in Figure 16 asked to mention other testing methods followed other than those already mentioned in Question 12. The purpose of this question was to determine if there was any new or different testing method other than the ones mentioned above. However, no one responded to this question. This means that all the testing methods followed by the respondents have already been mentioned in the above question.

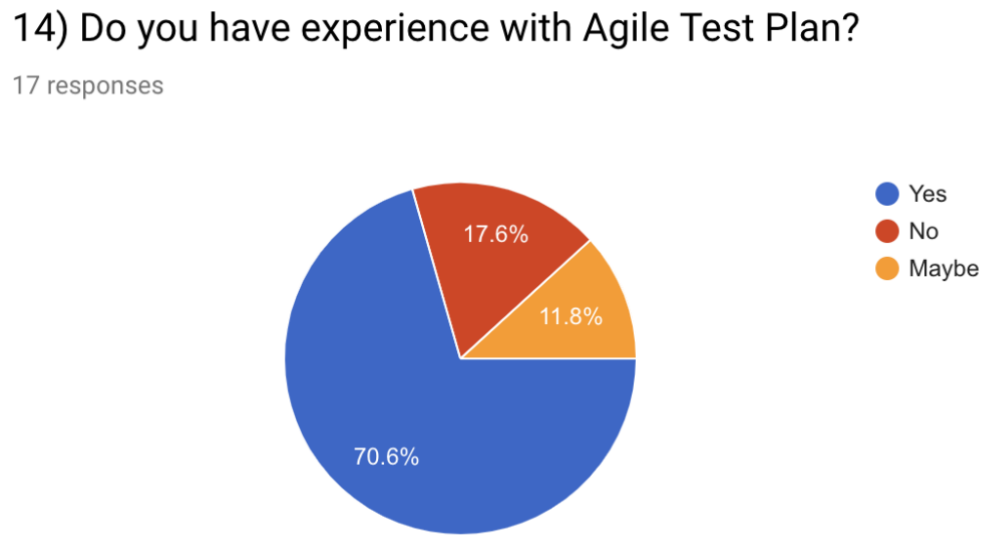

Figure 17. Respondents experience with Agile Test Plan.

Question 14 in Figure 17 asked the respondents if they have any experience with the Agile Test Plan. Its purpose was to determine the familiarity of the employees with the Agile Test Plan. The majority of the respondents (70.6%) responded that they have experience with Agile Test Plan. 17.6% responded that they do not have any experience with the same and 11.8% responded 'Maybe' to the question.

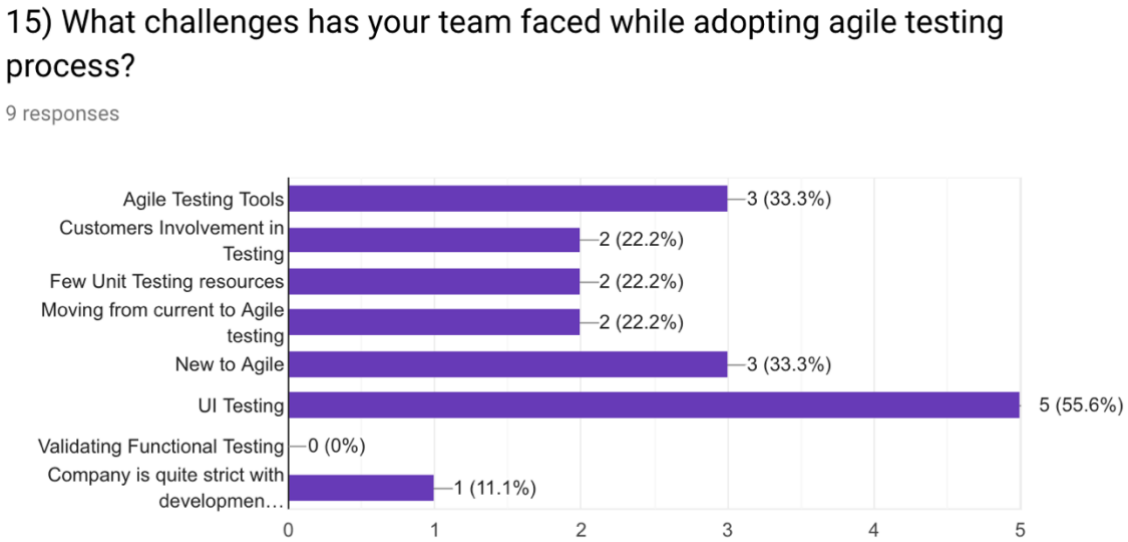

Figure 18. Challenges faced by respondents while adopting agile testing process.

Question 15 in Figure 18 asked the respondents regarding the challenges faced by the team while adopting agile testing process. The primary objective of this question was to determine the most common challenges faced while adopting agile testing process. According to the responses, UI testing was the most popular (55.6%) challenge. Similarly,

33.3% of the respondents chose to move from current process to agile in Figure 15, and it is clearly shown that more than half (55.6%) of the respondents faced challenges while adopting agile methodologies for UI Testing.

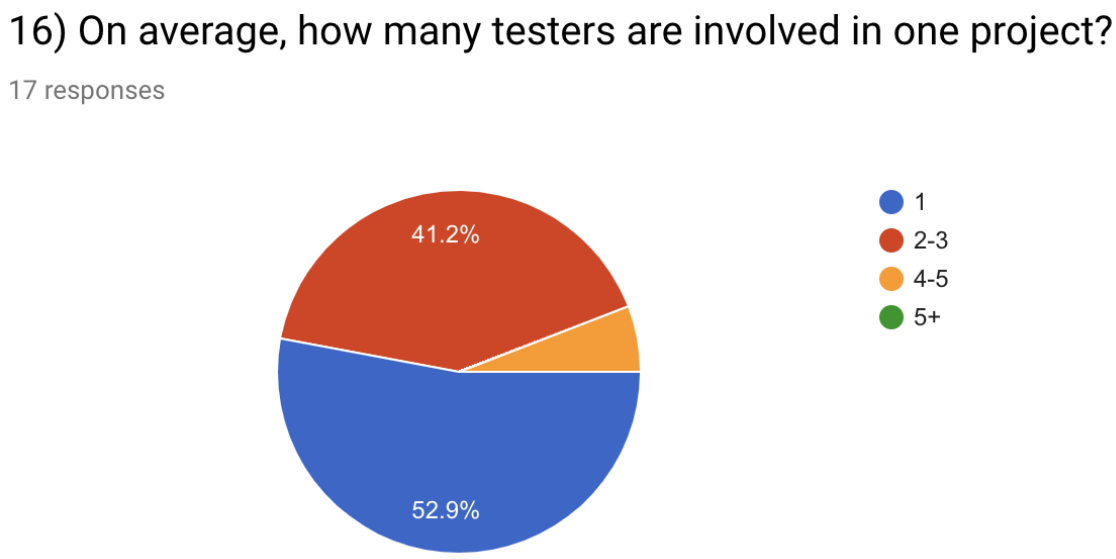

Figure 19. Number of testers involved in a project.

Question 16 in Figure 19 asked the respondents regarding the number of testers involved in a project. The responses showed that more than half of the respondents (52.9%) replied that 1 tester is involved in a project. Of the respondents, 41.2% concluded that there are 2-3 testers involved in a project. Moreover, 5.9% of the respondents answered that 4-5 testers were involved in a project. More than 5 was the least popular choice with no respondent choosing it. It showed that company distribution of testing resources depends upon the product size but still, a medium size project demands at least 3-4 testers.

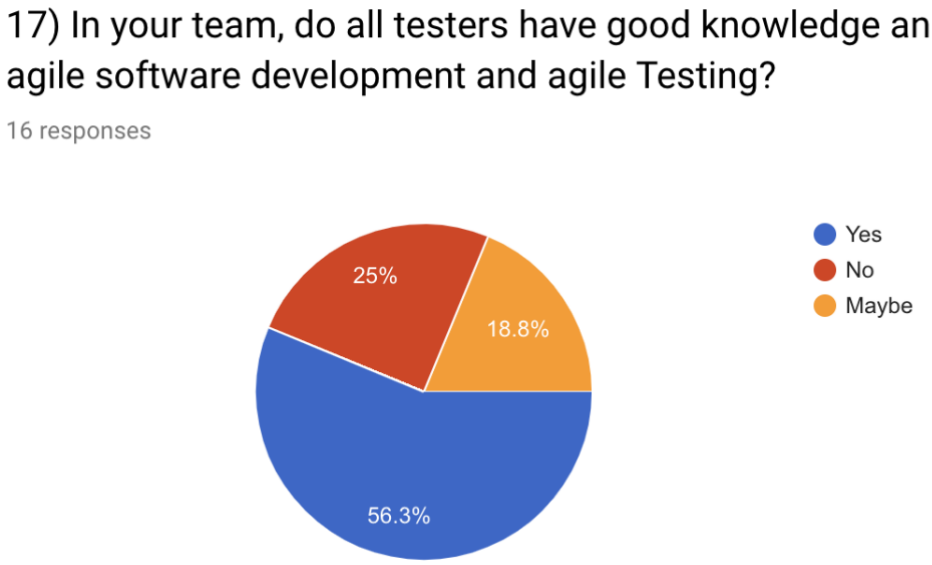

Figure 20. Testers' skills in Agile testing and Development.

Question 17 in Figure 20 asked if all the testers have good knowledge and skills regarding agile software development and agile testing. According to the responses, more than half of the respondents (56.3%) responded that all the testers have good knowledge and skills regarding agile software development and agile testing. Similarly, 25% of them replied the opposite and 18.8% of them either didn't have knowledge regarding this since their answer was neutral.

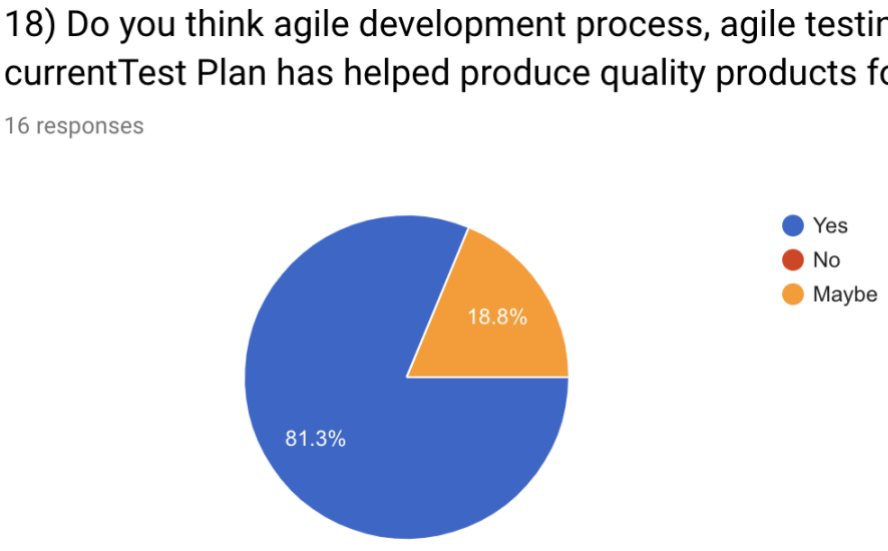

Figure 21. The effectiveness of agile and current test plan.

Question 18 in Figure 21 asked the respondents if the agile development process, agile testing and the current test plan has helped produce quality products for the company.

The majority of the responses (81.3%) were positive, and only 18.8% of the respondents gave a neutral answer. None of the respondents denied that agile development process, agile testing and the current test plan has helped them produce quality products for the company.

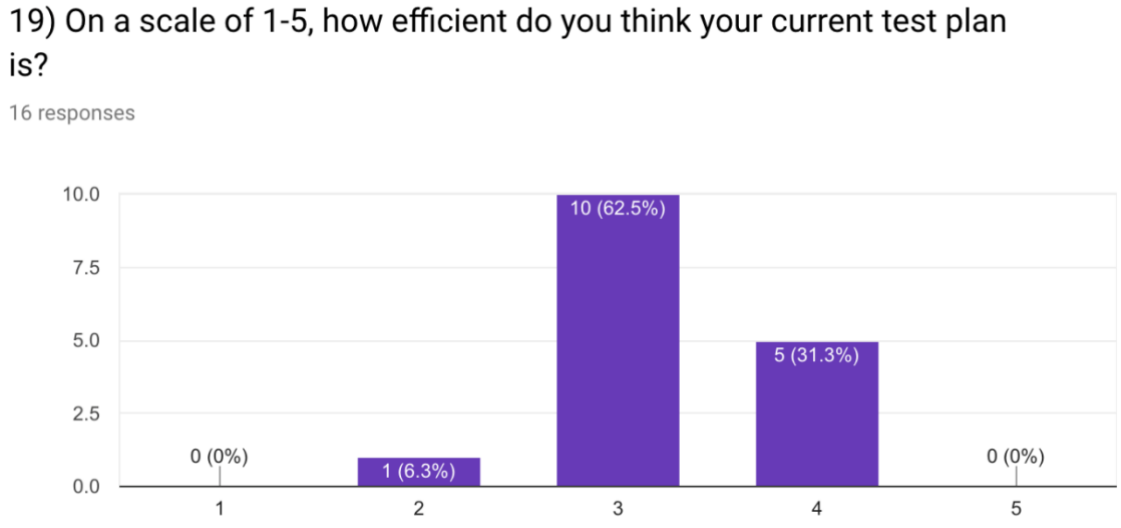

Figure 22. Efficiency of the current test plan.

Question 19 in Figure 22 asked the respondents to rate the efficiency of the current test plan on a scale of 1-5- From the chart, it can be clearly seen that the majority of them (62.5%) rated 3 to the current test plan. Similarly, 31.3% of the respondents gave a rating 4 to the test plan. However, 6.3% gave only 2 out of 5. However, none of the respondents gave neither the lowest score or best score to the test plan. This proves that the respondents are not happy regarding the current test plan.

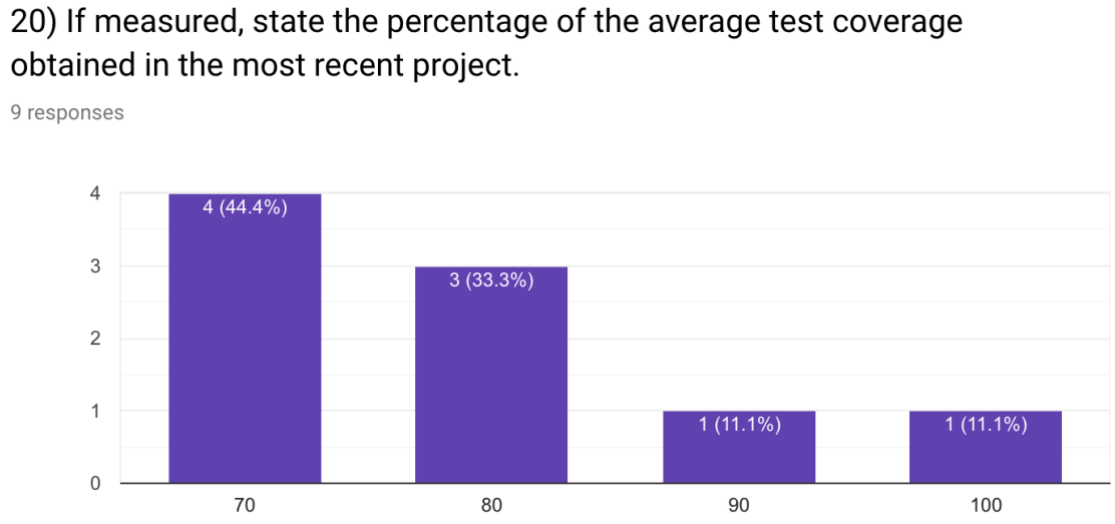

Figure 23. Average test coverage in the most recent project.

Question 20 in Figure 23 asked to state the percentage of the average test coverage obtained in the most recent project. Almost half of the respondents replied that the average test coverage obtained in their most recent project is 70 percent. Similarly, 33.3% of them responded that the average test coverage for their most recent project is 80%. The number of respondents who believed that the average test coverage for their most recent project is 90% and 100% are 11.1% each.

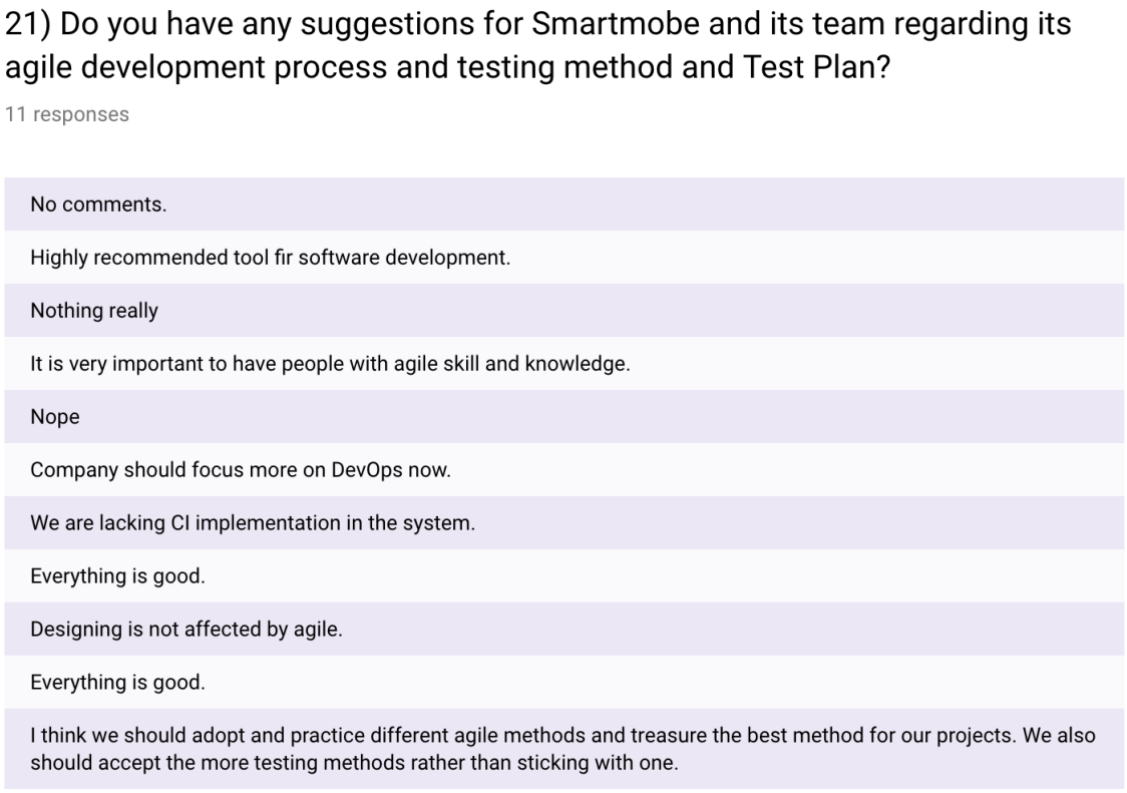

Figure 24. Respondents suggestions for the company

Question 21 in Figure 24 prompt the respondents for any suggestions for the company and the team regarding its agile development process, testing methods and test plan. This was an open-ended question for the respondents to pour in their thoughts and suggestion. Some of the meaningful answers were that the company should now focus more on DevOps and CI implementation in the system. It has also been suggested that more testing methods should be adopted rather than just sticking to one.

6 NEW TEST PLAN AND PROCESS

The new test plan is based on issues and good practices identified during test plan review and survey to employees. In the new test plan, new testing types and methods are introduced in order to help the company produce more efficient results. Also, different testing types and testing levels are introduced, and more information is provided regarding the roles of testers in the different stages.

6.1 New Testing Process

A new testing process is defined, with the help of the Lead QA and Product Manager and is recommended for software testing in the company. Figure 25 gives an overview of the entire testing process and introduces Continuous Integration in the company.

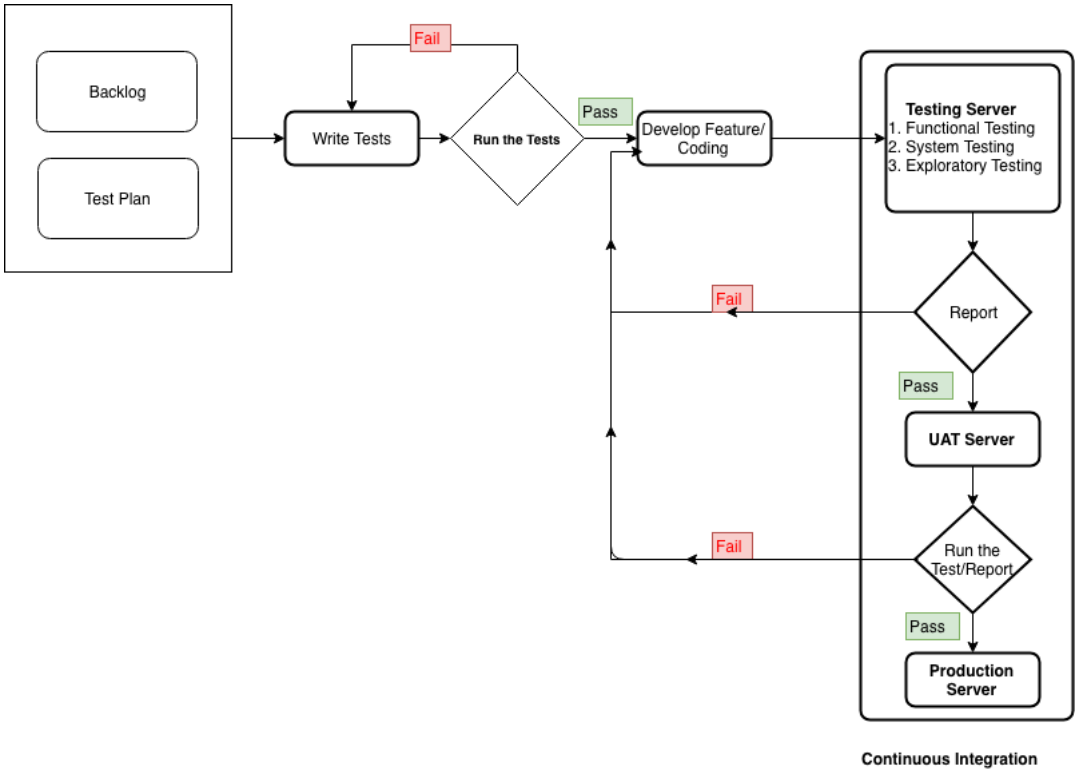

Figure 25. Recommended Testing Process.

The Backlog and Test Plan are started parallelly in the early phase of development. Initial creation of the Backlog gives a clear vision of the project to the development team, while the preparation of a test plan lets the testers be involved in the development process from an early phase. Therefore, compared to the current development process, the test plan and Backlog development should be done simultaneously for better results. The testers and developers start writing the test for Unit and API testing whereas the Project manager (PM) and other team members start creating a backlog from user stories which

are collected from the clients. After finalising the Test, the developers begin to write a code to pass the test. This process does not let developers go out of scope. Similarly, it always verifies that the completed tasks are tested. The PM and Lead QA must analyse all the processes in different sprint meetings and make sure everything is in the plan. When Developers finish a small set of functionalities or changes in their private workspace, they should then be committed to the shared repository, and the CI servers will monitor the repository and check out the differences and run unit and integration tests.

Implementing CI will help the development team catch the bugs earlier in the development cycle, which results on mitigating expenses. For this, a tool like Jenkins is advised. Jenkins is free, so from a cost point of view there is nothing to lose for the company, but setting up a Jenkins server does require time and dedication.

New test processes demand more testing on different stages, and sometimes the same testing is done repeatedly.

6.2 New Test Plan

Development of the updated new test plan (Appendix) is based on the current test plan. Few changes such as legitimate involvement of testers in different stages of software development, enhancing testing tools and roles of team member has been updated. The test plan strictly follows the company's testing policies and guidelines. The new Test Plan targets the development team and the customer. It also defines the roles and responsibilities of the team members.

All the information collected from the analysis of the responses to the survey was used to develop the new test plan. The new test plan (Appendix) is divided into 16 different sections/components. Each component is briefly explained. This test plan handles the testing at a generic level and defines components like testing resources, environment, scopes and schedule. Details of the individual components are not considered in this thesis.

The sections of the new test plan are discussed below:

6.2.1 Introduction

Section 1-4 of the Appendix gives an overview of the key functions of the project and explains the general purpose of the test plan in the project. It provides the limitations and specifications for the scope of the project. This is so that the user of the document can understand the details of the project or what the project is about. The limitations and specifications provide the boundaries within which the user should work. The purpose explains the goal of the test plan such as; the objective of testing, testing tools, testing resources, testing process and test execution time for the project. Understanding the purpose of the test plan will keep the user on the right track for the duration of the testing process.

Section 3 of the Appendix lists the audience who will be using this document to understand the product, test process and their roles in the testing process. Knowing the audience will help decide the content and amount of technicality of the document. Usually, the test plan is made for both technical and non-technical people, so it is a good idea to avoid technical content. Section 4 of the Appendix informs the overall objectives of the testing process such as, finding defects and making sure the software is bug-free before the release. A list of features to be tested are mentioned, and the goal of the tests based on the listed features are defined.

6.2.2 Test Assumptions

This section will list the assumptions of the test process made during the preparation of the test plan. In order to make sure that all the test activities are successfully and timely completed, all the assumptions must be validated. It also lists the dependencies on the resources. After analysing the current test plan of the company, it is found that the assumptions are not categorised on the basis of their importance and impact on the project. Therefore, in Section 5 of the Appendix, the assumptions have been divided into two parts:

- Key assumption

  Assumptions which have a higher impact on the project and must be prioritised.

- General assumption

  Assumptions that will have a minor impact on the project.

6.2.3 Test Scope

This part of the test plan explains the scope of the testing, both in-scope and out-of-scope. It includes what system or part is being tested, priority of features, new features, features that will be tested in the next phase and features which are not tested at all. In the current test plan, only in-scope testing is explained. Test scopes are not categorised and explained in an unorderly manner which is quite lengthy and not easy to understand. In the updated test plan (Appendix, Section 6) a table has been introduced. The table is divided into 5 columns, and each column has information explaining the scope of testing.

6.2.4 Test Level

Section 7 of the Appendix highlights the details of the different types of testing that will be conducted. In this part of the test plan, all the mandatory test levels are mentioned. It was found that Unit testing and API testing were not mentioned in the old test plan. In the current testing process, all the testing is done either during the development or after the development. This does not follow the test-first approach. Even if the company does not follow the new test process, Unit and API testing must be mentioned in the test plan.

While the number of testers for a specific test depends upon the size of the project, this part of the test plan specifically assigns the type and skill of the testers for each test. It also highlights the necessary tools for the type of test. The goal of this section is to answer the why, who, what, when, where and how of a specific test type.

6.2.5 Test Schedule

This part of the test plan provides the overall schedule of the testing. This section of the test plan is entirely new and is equally important as the other components of the test plan. It provides information about the types of testing for specific test components and deadlines to complete the testing. It helps to estimate the product deployment.

6.2.6 Tasks, Resources and Responsibilities

Sections 9 and 10 from the Appendix include information regarding the tasks, resources and responsibilities involved in the testing. They also give a brief introduction of the tasks and their importance. Some examples of the tasks are; test case writing, system setup documentation, test summary preparation. All the resources are listed on the basis of the roles involved in the testing. In the new test plan, resources and

responsibilities are presented in a tabular form where one of the columns list the resources and, another column explains the responsibilities of those resources.

6.2.7 Testing Environments and Tools

Sections 11 and 12 of the test plan from the Appendix show which type of testing is conducted in each environment. It is recommended to revalidate the testing while migrating from one test environment to the other. All the automation tools that will be used during the testing process are mentioned in this part of the test plan. Also, a list of bug tracking and management tools are listed.

6.2.8 Test Criteria

The test criteria, section 13 in the Appendix, makes it easier for the QA team to recognise the requirements to be met and the tests to be performed during several phases. It also notifies when to start, pause and stop the particular test activity.

There are three different criteria mentioned in the new test plan:

- Entry Criteria

Information regarding the things that need to be done before starting the testing, such as the test plan, test cases and test tools which must be available in order to start testing.

- Pause Criteria

Information regarding the conditions which will halt the testing process such as, a massive amount of bugs making the whole application crash and unavailability of testers.

- Exit Criteria

Compulsory conditions to be met in order to proceed with the implementation, like, no defects and completion of all in-scope tests. Documents such as test logs, test incident report log and test summary report are the output achieved through this criteria.

6.2.9 Defect Management

Section 14 in the Appendix explains the process how the defects will be categorised and handled from start to end after classifying them on the basis of their types. The classification of the defects depends upon the tools used to track the defect. This section

is essential for the testers to understand because it rates the defects using different categories. It will help avoid arguments and conflicts while categorising the defects during the project. In the company, JIRA is used as a defect tracking tool and, on the basis of JIRA, specific categories are provided.

Figure 26 gives an overview of the life cycle of the defects. The life cycle depends on the testing process and the testing tools, which vary from project to project.

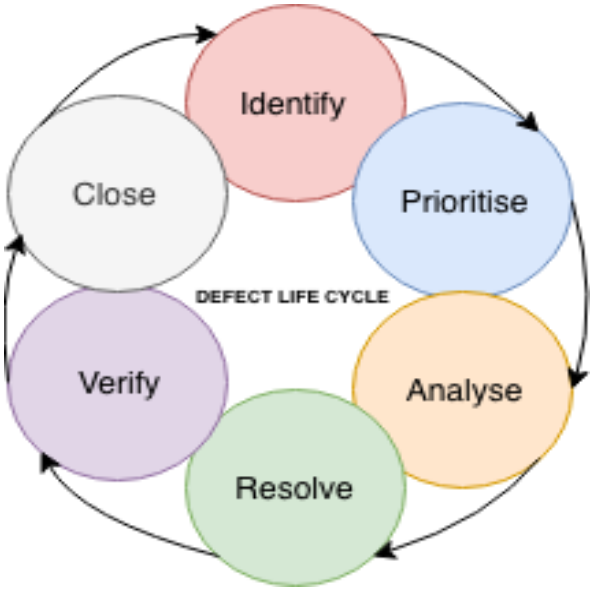

Figure 26. Defect Life Cycle.

6.2.10 Deliverables and Approvals

Sections 15 and 16 specify the role and name of the person who will be responsible for approving the test plan and the list of the documents that will be provided before, during and after testing. In the current test plan, only after Testing are the deliverables mentioned. This is not enough because during the testing process there are many documents that need to be created and must be approved by the PO. However, in the updated test plan (Appendix) the testers must provide test deliverables before and during the testing.

7 CONCLUSION

The main aim of this thesis was to analyse the current test plan, testing and development process of the company and to develop a new and improved modular test plan which will help the company achieve efficiency and competency in further software development. This has also helped the management team understand the internal testing process of the company along with the team members' views on it.

The analysis of the current test plan and test process was carried out by gathering information from various sources such as web articles, journals, books, and by comparing it with the outcome of the projects. A survey was designed with the aim of finding out the different types of software development, testing processes and challenges faced by a team while doing so. Different types of agile testing methods were also analysed and discussed.

It was found that there are some areas in need of improvement such as; new testers must be trained and tested before onboarding in a project. This can be done by providing a demo project to the new testers and see if they are fit for a project or not. More testers should be involved in a project. Although agile says no; documentation like developer documentation and deployment documentation should be provided to the customer. Test cases for Unit and API testing can be written during the preliminary phase of development, and coding for the product should be done in order to pass those test cases.

In the new test plan, some relevant information on agile testing, roles of testers and types of testing has been updated. A testing automation process such as Continuous Integration has been included as well. The Test Plan will be verified by the thesis commissioner and the CTO of the company and will be implemented under their guidance in future projects.

In the future, if someone wants to extend the research, it is recommended they study the impact of the new test process and test plan. This can be done after some time, after which the company will be following the new testing strategy. More development can be done in areas such as setting up a CI server and work with CI.

REFERENCES

Aldaine, A. (2018). Top 10 API Testing Tools (Details & Updates Done for You!). [online] Medium. Available at: https://medium.com/@alicealdaine/top-10-api-testing-tools-rest-soap-services-5395cb03cfa9 [Accessed 10 Nov. 2018].

Alexander, M. (2018). Agile project management: A comprehensive guide. [online] CIO. Available at: https://www.cio.com/article/3156998/agile-development/agile-project-management-a-beginners-guide.html [Accessed 8 Nov. 2018].

Amarinfotech. (2018). Agile Scrum Testing Process | Role Of QA In Agile Scrum. [online] Available at: https://www.amarinfotech.com/roles-responsibilities-agile-tester.html [Accessed 24 Oct. 2018].

Bartlett, J. (2015). *Agile Testing: The Role of QA in Agile - TestLodge Blog*. [online] TestLodge Blog. Available at: https://blog.testlodge.com/the-role-of-qa-in-agile/ [Accessed 4 Nov. 2018].

Bergmann, S. (2018). Release Announcement for Version 7 of PHPUnit – The PHP Testing Framework. [online] Phpunit.de. Available at: https://phpunit.de/announcements/phpunit-7.html [Accessed 8 Nov. 2018].

Buenen, M. and Muthukrishnen, G. (2018). World Quality Report 2017 - 18. [online] Available at: https://www.sogeti.com/globalassets/global/downloads/testing/wqr-2017-2018/wqr_2017_v9_secure.pdf [Accessed 16 Oct. 2018].

Carmichael, E. (2017). Agile Methodology: The Complete Guide to Understanding Agile Testing - QASymphony. [online] QASymphony. Available at: https://www.qasymphony.com/blog/agile-methodology-guide-agile-testing/# [Accessed 19 Nov. 2018].

Carter, A. (2018). PHPUnit: What, Why, How?. [online] Andy Carter. Available at: https://andy-carter.com/blog/phpunit-what-why-how [Accessed 24 Oct. 2018].

Cigniti. (2016). Test Automation & Agile Test Quadrants. [online] Available at: https://www.cigniti.com/blog/agile-test-automation-and-agile-quadrants/ [Accessed 20 Nov. 2018].

Colantonio, J. (2017). 16 Open Source API Testing Tools For REST & SOAP Services. [online] Test Automation Made Easy: Tools, Tips & Training Awesomeness. Available at: https://www.joecolantonio.com/12-open-source-api-testing-tools-rest-soap-services/ [Accessed 19 Oct. 2018].

Confluence.atlassian.com. (2018). Jira Software overview - Atlassian Documentation. [online] Available at: https://confluence.atlassian.com/jirasoftwarecloud/jira-software-overview-779293724.html [Accessed 17 Sep. 2018].

Crispin, L. and Gregory, J. (2014). Agile testing. Upper Saddle River, N.J: Addison-Wesley, p.98.

Denning, S. (2015). Agile: The World's Most Popular Innovation Engine. Forbes. [online] Available at: https://www.forbes.com/sites/stevedenning/2015/07/23/the-worlds-most-popular-innovation-engine/#592f35b57c76 [Accessed 25 Oct. 2018].

Eriksson, U. (2015). How to set goals for session-based testing (SBT) | ReQtest. [online] ReQtest. Available at: https://reqtest.com/testing-blog/how-to-set-goals-for-session-based-testing-sbt/ [Accessed 24 Oct. 2018].

Esquivel, G. (2018). Differences between TDD, ATDD and BDD. [online] gaboesquivel.com. Available at: https://gaboesquivel.com/blog/2014/differences-between-tdd-atdd-and-bdd/ [Accessed 3 Nov. 2018].

Guckenheimer, S. (2017). What is Continuous Integration? - Azure DevOps. [online] Docs.microsoft.com. Available at: https://docs.microsoft.com/en-us/azure/devops/learn/what-is-continuous-integration [Accessed 24 Oct. 2018].

Guru99. (2018). Scrum Testing Beginners Tutorial: Roles, Artifacts, Ceremonies. [online] Available at: https://www.guru99.com/scrum-testing-beginner-guide.html [Accessed 22 Nov. 2018].

Heller, M. (2017). What is Jenkins? The CI server explained. [online] InfoWorld. Available at: https://www.infoworld.com/article/3239666/devops/what-is-Jenkins-the-ci-server-explained.html [Accessed 19 Oct. 2018].

Knight, A. (2017). BDD 101: Writing Good Gherkin. [online] Automation Panda. Available at: https://automationpanda.com/2017/01/30/bdd-101-writing-good-gherkin/ [Accessed 3 Nov. 2018].

Kohl, E. (n.d.). What is Exploratory Testing? - James Bach - Satisfice, Inc.. [online] Satisfice.com. Available at: http://www.satisfice.com/articles/what_is_et.shtml [Accessed 2 Oct. 2018].

Morris, S. (2018). Tech 101: What Exactly is Scrum?. [online] Skillcrush. Available at: https://skillcrush.com/2017/06/28/what-is-scrum-project-management/ [Accessed 4 Nov. 2018].

Pressman, R. (2014). Software engineering. 7th ed. New York: Mcgraw-Hill, pp.68-69. Available at: http://dinus.ac.id/repository/docs/ajar/RPL 7th_ed_software_engineering_a_practitioners_approach_by_roger_s._pressman_.pdf [Accessed 13 Sep. 2018].

Shahin, M., Ali Babar, M. and Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review of Approaches, Tools, Challenges and Practices. IEEE Access, 5, pp.3909-3943.

Sherman, K. (2017). Testing in Agile: A Quick Tutorial on Defects, Bugs and Everything in Between. [online] LinkedIn. Available at: https://www.linkedin.com/pulse/testing-agile-quick-tutorial-defects-bugs-everything-between-sherman/ [Accessed 20 Nov. 2018]

Smartbear. (2018). Why Test Automation? Automated Testing Benefits and Tips. [online] Available at: https://smartbear.com/learn/automated-testing/ [Accessed 20 Nov. 2018].

Techopedia.com. (2018). What is Software Development? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/16431/software-development [Accessed 4 Jul. 2018].

APPENDIX

Test Plan V 1.0

Project: ……..

Prepared By: Kalyan Giri (Lead QA)

Table of Contents

AGILE TEST PLAN

1. INTRODUCTION

Briefly introduce the project and the relation of this document to the project. Also, summarise the key functions of the project being tested.

E.g.: "This test plan is intended to be used for Grossohopper project which is a service intended for door-to-door grocery delivery. The service contains

- Login authentication
- Online Shopping
- Payments
- Ratings
……
……
……..

2. PURPOSE

Briefly explain the purpose of this test plan such as "*providing well-structured information of test strategy, testing tools, testing processes and testing execution time for the project".*

E.g.: This project will be tested using Katalon Studio for API testing the online shopping module in a period of……
……
……..

3. AUDIENCE

Mention all the targeted audience for the test plan and briefly describe their responsibilities and tasks.

E.g.:

Table 1. Project Audience.

Audience	***Description***
Customer	*This is the party that has the ownership of the product e.g.: Grossohopper*
Project Manager	*The employee that leads the team*
Development team	*The employees developing the product*
…	*…*
…	*…*

4. TEST OBJECTIVES

Information regarding the purpose of different types of testing such as what does the testing do.

E.g.:

Table 2. Test objective

Feature	***Test Goals***
Login authentication	*Verify the login credentials are from the Active Directory.*
…	*…*
…	*…*

5. TEST ASSUMPTION

State the assumptions in order to make sure that all the test activities are completed successfully and promptly. All the assumptions must be validated. Here the assumptions are divided into two parts.

- Key assumption

E.g.:
1. The application will be deployed to the production server after and only after, it has been thoroughly tested and free of bugs.
2. …

- General assumption

E.g.:
1. Developers will prepare test classes for Unit and API testing before starting to code.
2. CTO and Lead Developer will verify the testing reports.
3. Bugs will be tracked by using JIRA.
4. ….

6. TEST SCOPES

Explain the in-scope and out-of-scope of the testing. Mention all the features that will be tested. Include what system or part is being tested, priority features, features that will be tested in the next phase and features which are not going to be tested at all. It is the task of Lead QA to prepare the scope of the Product.

E.g.:

Table 3. In and Out-of-Scope of the testing.

System	*What to test?*	*When to test?*	*Device Environment Requirements*	*Need for external resources*
…	…	…	…	…

7. LEVEL OF TESTING

Give details to the different types of testing that will be conducted. Answers the why, who, what, when, where and how.

E.g.:

Table 4. Unit Testing.

UNIT TESTING	*DESCRIPTION*	*COMMENTS*
WHY?	• *To make sure every line of code is developed correctly.*	
WHO?	• *Developers/ Tech Lead*	
WHAT?		
WHEN?	• *As soon as the code is written*	
WHERE?	• *Local DEV + CI*	
HOW?	• *Automated (PHP Unit)*	

Table 5. API/Service Testing.

API/SERVICE TESTING	*DESCRIPTION*	*COMMENTS*
WHY?	• *The purpose of API testing is to test the Application Programming Interfaces (API) directly to determine if they meet the expectations for functionalities, reliabilities, performance and security.*	
WHO?	• *Developer/ Tech Lead*	
WHAT?	• *The API testing will be verified on the basis of the parameters provided by the backend developer where we verify the requested output with the response of the API.*	
WHEN?	• *As soon as new API is developed and ready*	
WHERE?	• *Local DEV and CI*	
TOOLS?	• *Soap UI, Rest Client*	

Table 6. Acceptance Testing.

ACCEPTANCE TESTING	**DESCRIPTION**	**COMMENTS**
WHY?	• *This test focuses on validating customers' requirements. End-user perform a final review of the system with this testing.*	
WHO?	• *Manual QA/ Developer/ End-User*	
WHAT?	• *Verification of features and Acceptance criteria*	
WHEN?	• *After Unit testing and features are ready to be tested.*	
WHERE?	• *CI/ Test Server or Environment*	
TOOLS?	• *Cucumber (Automated)*	

Table 7. System Testing.

SYSTEM TESTING	***DESCRIPTION***	*COMMENTS*
WHY?	• *System testing is the level of software testing where the complete and integrated software is tested.*	
WHO?	• *Manual QA, Product Owner*	
WHAT?	• *The web platform overall module will be covered by system testing.*	
WHEN?	• *At the end of project development. This test will be performed after User Acceptance Testing.*	
WHERE?	• *Staging Environment*	
TOOLS?	• *Automated (Webdriver)*	

8. TEST SCHEDULE

Describe the timetable for the test and update as needed. It can be customised according to the project.

E.g.:

Table 8. Testing schedule.

Schedule	*Testing components*	*Types of Testing Included*
16.12.2018	*Login Authentication*	*API Testing*
…	*…*	*…*

9. TASKS

Include information regarding the tasks such as test case writing, system setup documentation.

E.g.:

User Acceptance testing
Writing test cases
Bug Reporting
Writing test Summary

CI Server
Setting up Jenkins
Writing test class and criteria for CI server.

10. RESOURCES AND RESPONSIBILITIES

Mention the resources needed for testing.

E.g.:

Table 9. Resources and their responsibilities for testing.

Resource	*Responsibility*
Project Manager	*Maintaining the schedule of the project*
Test Lead	*Monitoring the progress* *Updating the Test Plan*
…	…
…	…

11. TEST ENVIRONMENT

List which type of testing is conducted in each environment. It is recommended to revalidate the testing while migrating from one server to another.

E.g.:

Table 10. Types of the testing environments.

Testing Types	*Local Environment*	*Development Environment*	*UAT Environment*	*Production Environment*
Unit Testing	*Yes*	*Yes*	*No*	*No*
Feature Testing	*No*	*Yes*	*Yes*	*No*
Functional Testing	*No*	*Yes*	*No*	*No*
…	…	…	…	…
…	…	…	…	…

12. TESTING TOOLS

List the tools required for a different type of testing and their purpose.

E.g.:

Table 11. Types of testing tool.

Tool	*Purpose*
Katolon Studio	*API Testing (REST and SOAP) request*
JIRA	*Bug Tracking*
Jmeter	*Load and Performance Testing*
…	*…*
…	*…*

13. TEST CRITERIA

List the procedure or judgment for the basis of the test
There are three different Criteria:

- Entry Criteria

Information regarding the things that need to be done before starting the testing.

E*.g.:*

Test plan must be prepared before starting the testing process
…

- Pause Criteria

Information regarding the conditions which will halt the testing process such as a massive amount of bugs making the whole application crash, unavailability of testers.

E.g.:

If 40% of the test case fails, test should be paused
…

- Exit Criteria

Desirable conditions that need to be met in order to proceed with the implementation, like no defects, all in-scope tests are completed.

E*.g.:*

If all the test cases pass, move the product to UAT server

14. DEFECT MANAGEMENT

Explains the process how defect will be categorised and handled from start to end after classifying it on the basis of its types.

Classifying Defects

- **Critical:** Defects which can cause a critical loss of functionality of the application and affect the server.
- **Major:** Defects that do not have a solution right away but are affecting the system.
- **Minor:** Defects that are causing minor impact, and there is an interim workaround available.
- **Trivial:** Impact level is shallow.

E.g.:

Database response time is more than 20s - Critical

Click and Drag function for the document uploading is throwing an error code - Minor

…

…

Defect Life Cycle

Address the process of handling defects in the test.

E.g.:

Identify -	*Downloaded documents are incomplete*
Prioritise –	*Defect is categorised as major*
Analyse –	*Assign it to developer and check original in the database*
Resolve –	*Developer found the issue and corrected the error in the database*
Verify –	*Download the document and check*
Close –	*Document and close the issue*

15. DELIVERABLES

Mention the deliverables which will be provided from start to the end of testing.

E.g.:

Table 12: Project deliverables

Before Testing	*Test plan, Test case, …*
During Testing	*Test Scripts, Test Data, Error Logs …*
After Testing	*Test result Report, Defect Report, Installation documentation …*

16. APPROVALS

Mention the Person who will approve the developed test Plan and their role.

E.g.: The prepared test plan document will be verified by:

Name	*Role*	*Date, Place*	*Signature*
John Doe	*Product Owner*	*16.12.2018, Turku*	*……*
…	*…*	*…*	*…*
…	*…*	*…*	*…*
…	*…*	*…*	*…*