



School of Technology
Department of Computer Science

Master Thesis Project 30p, Spring 2014

Merging Functional Requirements with Test Cases

By

Madhuri Kolla & Mounika Banka

Supervisors:

Annabella Loconsole

Examiner:

Ulrik Eklund

Contact information

Authors:

Madhuri Kolla

E-mail: madhu.madhuu@gmail.com

Mounika Banka

E-mail: mounikab.bank@gmail.com

Supervisors:

Annabella Loconsole

E-mail: annabella.loconsole@mah.se

Malmö University, Department of Computer Science.

Examiner:

Ulrik Eklund

E-mail: ulrik.eklund@mah.se

Malmö University, Department of Computer Science.

Abstract

A lot of research is done in requirements engineering and testing but often the extensive literature is missing on defining good methods for linking functional requirements with test cases. Most of the delays occurring in the software development projects are because of incomplete or inaccurate functional requirements. The two main goals of our project are to achieve a successful software project by First, to design a template, which will merge functional requirements with test cases and second is to find the benefits of the aligning requirements to test cases. Changing, updating and tracing the requirements during the development of the project is not an easy task. The main reason for project failure is due to possibility of not fulfilling specified project requirements, so one way to solve this problem is to merge functional requirements with test cases. Thus removes the need of creating a separate requirements document, which will improve the traceability process between requirements and testing, thus leads to high quality and efficient development. The template helps us to drive a successful project by identifying the issues at an earlier stage of the development cycle.

Keywords: Requirements engineering, Functional requirements, Testing, Test cases, Requirement templates, Test case templates, Alignment, benefits.

Popular science summary

In today's world any company independent of being present in any type of industry, all of them are highly dependent on software products or services to run their day-to-day business operations. Those companies that excel in technology innovation will sure be leading the markets of their industry. As such to excel in technology, software development has become the core of any technology innovation in the past decade. The two important teams of any software development project are requirements engineering and software testing, and these teams are often working in isolation in many organizations which is also one of the main reasons for a software product being delivered with wrong specifications. With our thesis we create a template that could be used by both the requirements and testing teams to update requirements and associated test cases in one template. Our methodology not only helps the teams to work closely but also helps the teams to identify issues at earlier stages of development cycle, thereby reducing the costs of delaying the technology products being developed by any company.

Acknowledgement

This thesis work is carried out at the Department of Computer Science, Malmö University, Malmö, Sweden under the supervision of Annabella Loconsole.

We consider it is our duty to acknowledge the people without whose assistance this work could not have been completed. First and foremost we would like to thank our supervisor Annabella Loconsole, Department of Computer Science, Malmö University, Sweden, for her invaluable guidance and kind cooperation extended by her throughout the course of this thesis work.

We would like to thank Ulrik Eklund for his continuous support and time in reviewing our thesis and providing valuable suggestions.

Our sincere appreciation to the employees of Sony, Maersk and Ericsson for providing the valuable inputs to our Survey Questionnaire.

We wish to express our gratitude to Amina Agovic, Dimitris Paraschakis, Zahra Ghaffari for their support in reviewing and providing feedback on our thesis report.

Our thanks to the almighty God for giving us the opportunity to be able to complete this project.

Finally we would like to thank our family, friends and colleagues for their moral support and inspiration.

December 2014,

Madhuri Kolla & Mounika Banka.

Table of Contents

1 Introduction	10
1.1 Project Idea.....	10
1.2 Goals	10
1.3 Research Questions	10
1.4 Motivation	10
1.6 Outline.....	11
2 Background	11
2.1 Software Requirements	11
2.1.1 Types of software requirements.....	12
2.2 Software Testing	14
2.2.1 Types of software testing.....	15
2.2.2 Test cases	16
2.3 Aligning RE-Testing in Software development life cycle	17
2.4 Conclusion.....	23
3 Research Methodology.....	24
3.1 Methods used for answering the Research Questions.....	24
3.2 Motivation for research methods used in this thesis	24
3.2.1 Motivation for Literature review	25
3.2.2 Motivation for Survey Questionnaire	25
3.3 Research Process	26
3.4 Literature Review	27
3.4.1 Searching	28
3.4.2 Obtaining and Assessing.....	29
3.4.3 Reading	29
3.4.4 Critical Evaluation	30
3.4.5 Writing a Critical Review	30
3.5 Survey	30
3.5.1 Defining questionnaire objectives	30
3.5.2 Selecting the sample	30
3.5.3 Designing the questionnaire format.....	31
3.5.4 Survey Design.....	33
3.5.5 Pretesting the questionnaire	33
3.5.6 Pre contacting the sample	34
3.5.7 Writing a cover letter	34
3.5.8 Following up with non-respondents	34
3.5.9 Threats to validity	34
4.1 Definition and importance of alignment	36
4.1.1 Benefits of aligning requirements to test cases.....	36
4.2 Alignment.....	39
4.3 Different kinds of alignment	40
4.3.1 Processes.....	40
4.3.2 People	41

4.3.3 Tools	41
4.3.4 Requirements process	41
4.3.5 Testing process	41
4.3.6 Requirement Based Testing	42
4.3.7 Traceability	44
4.3.8 Model-based testing.....	49
4.3.9 REST taxonomy.....	51
4.4 Conclusion.....	51
4.4.1 Benefits for Alignment	52
4.4.2 Alignment Methods	53
5 Designing Software Requirements and Test case template	55
5.1 Aligning Requirements and Test cases	55
5.1.1 Merging Requirements and Test cases	56
5.2 Requirement templates.....	58
5.2.1 Volere requirements specification template	58
5.2.2 Use case requirements template.....	59
5.2.3 Requirements template by Toro et al.....	60
5.2.4 Requirements template field prioritization	61
5.3 Test Case templates.....	62
5.3.1 Detailed and simple test case template	62
5.3.2 Test case specification template	62
5.3.3 Analysis of test case template 2.....	63
5.3.4 Test case template field prioritization.....	64
5.4 Test Case / Requirements Template Creation	65
5.5 Created Test Case Template.....	67
6 Survey evaluation.....	68
6.1 Answers to key research question of the questionnaire	76
6.2 Interpretation of the results	77
6.3 Improving the Test Case Template	77
7 Discussions and Conclusions	79
7.1 Thesis Summary.....	79
7.2 Answering Research Questions.....	79
7.3 Conclusion.....	81
7.4 Future Work	82
7.5 Scenario Sampling	82
8 Appendices	85
8.1 Appendices A	85
8.2 Appendices B	88
9 References	90

List of Figures

Figure 1 Different levels and types of Requirements [8].	13
Figure 2 Software Development Life Cycle (Figure is illustrated from Waterfall Model) [28].	18
Figure 3 Water Fall Model [38].	19
Figure 4 Incremental-Model [38].	19
Figure 5 V-Model [38].	20
Figure 6 Spiral Model [38].	21
Figure 7 Agile Method [38].	21
Figure 8 XP Method [38].	22
Figure 9 Test-driven development.	22
Figure 10 Research methodology process	27
Figure 11 Literature Review Method Based On Oates [11]	27
Figure 12 Requirement versus test cases traceability matrix [17].	46
Figure 13 Linking requirements with a possibility high number of test cases.	47
Figure 14 Relation between requirement process and testing [18].	48
Figure 15 SDLC when both Requirements Template and Test Case Template are used.	55
Figure 16 SDLC when the designed template is used.	56
Figure 17 Alignment decision.	68
Figure 18 Rating the idea of merging requirement with test cases.	70
Figure 19 Rating our template created.	70
Figure 20 Challenges faced during requirement alignment	71
Figure 21 Investigating about requirement decision	72
Figure 22 Challenges in requirement analysis	73
Figure 23 Rating the idea of having both ID's in single document.	74
Figure 24 Rating our template instructions.	74
Figure 25 Software models used in organisations.	75
Figure 26 Any improvements by using our template.	76

List of Tables

Table 1: Test case Example [16].	17
Table 2 Overview of Research Methods.	24
Table 3 RQ1 Searching string categories.	28
Table 4 RQ2 Searching string categories.	28
Table 5 Participants from discipline area.	31
Table 6 Likert scale	32
Table 7 Questionnaire used for the survey.	32
Table 8 Results of literature review for research question 1	36
Table 9 Results of LR for research question 2	39
Table 10 Literature review used for template	57
Table 11 Volere template by Robertson [21].	58
Table 12 Use case Requirement Template [43].	59
Table 13 Template for L-pattern for information storage of requirements	61
Table 14 R-pattern for information storage.	61
Table 15 Requirements template field prioritization	61

Table 16 Detailed test case template [3].	62
Table 17 Simple test case template [3].	62
Table 18 Test case specification template [16].	63
Table 19 Common columns in test cases that are present in all test case formats	63
Table 20 A very details Low Level Test Case Format [32]	63
Table 21 Test case template field prioritization	65
Table 22 Instructions to template.	66
Table 23 Derived Template.	67
Table 24 Improvements for alignment.	69
Table 25 Improved Test Case Template	78
Table 26 Filled in test case template for scenario 1	84

List of Acronyms

RE	Requirements Engineering
SRS	Software Requirement Specification
BRD	Business Requirements Document
URS	User Requirements Specification
MRD	Marketing Requirements Document
EIR	External Interface Requirements
SDLC	Software Development Life Cycle
XP	Extreme Programming
TDD	Test-Driven Development
RE	Requirements Engineering
ST	Software Testing
FR	Functional Requirement
NFR	Non-Functional Requirement
NDT	Navigational Development Techniques
RBT	Requirement Based Testing
JAD	Joint Application Development

1 Introduction

A functional requirement is a specific business need or behaviour as seen by an end user of the system [2]. Most of the delays occurring in the software development projects are because of incomplete or inaccurate functional requirements. A requirement specification should not be based on assumptions, it should be complete and cover every aspect of the software system [1]. Testing is a judgment to check if the system meets, exceeds or fails to meet the predefined requirements. Thus, to enable high product quality it is important that requirements and testing are aligned. Many organizations are interested in aligning requirements and testing closely together but there is a lack of literature or process on how to link the two [1]. In this thesis, we have created a template that merges the requirements with test cases and serves the purpose of both functional specifications and test cases in a software development project and also described various benefits of using the template.

1.1 Project Idea

IT organizations are currently facing significant difficulties in meeting schedules, budgets and have problems to predict the quality of the developed products. There are many factors that cause these problems, which may be related to human, technology and processes. The main goal of our thesis is to merge test cases with functional requirements.

Barmi et al. mentioned that linking RE and testing will benefit both disciplines [10]. Making a tight link between them will improve the outcome of the software development process. It also helps to discover errors early, which in turn will improve the quality of product and lead to more satisfied customers [10].

1.2 Goals

The main goal of our thesis is to merge functional requirements with test cases. This thesis will propose a template that can be used to specify functional requirements and also test cases required for a software development process. One of the reasons for considering test cases as requirements is that test cases can be more detailed than requirements, thereby enabling the programmers and also test teams to clearly understand the requirements and associated test steps.

1.3 Research Questions

The research questions considered as part of our thesis are as listed below:

RQ1: What are the benefits of aligning functional requirements with test cases?

RQ2: How to align functional requirements and test cases?

1.4 Motivation

In today's world, any industry be it Telecommunications, e-Business, Aviation, Oil & Gas, Marketing, retail, automobile, etc., are particularly keen towards the quality of software services and products they are working with in their day to day operational activities. One of the goals of our thesis is to merge functional requirements with test cases, which would achieve a better quality at the earlier stages of the development cycle itself.

1.5 Contribution

The template we introduce in this thesis will lead to less documentation, which also decreases the project lead-time and creates a single version of requirements and testing information. It also improves the quality of the requirements that would be provided to the developer, which in turn will also directly relate to the quality of the software product.

This template removes the need of creating a separate requirements document. Our template improves the traceability between requirements and test cases thereby using our template in a software development process will improve the quality of product being developed.

1.6 Outline

Chapter 1 presents a brief introduction, which summarizes the objectives and presents a brief overview of the thesis. Chapter 2 presents background information, which gives a brief overview of different types of requirements, testing and software development life cycle. Chapter 3 presents the research methods used to solve the research questions. Chapter 4 presents Literature Review, which contains information about alignment, different kinds of alignment in software development and benefits of aligning. Chapter 5 explains the creation of a test case template, which may provide substantial improvements to the current software development processes. Chapter 6 presents the survey evaluation of the template created through the thesis. Chapter 7 presents conclusion of the master thesis project together with the potential future work that can be carried on our thesis topic.

2 Background

Software development is a process of building a software product as per the user group requirements. A software product is considered usable when it behaves and yields the expected result. To develop required software, certain sub process should be aligned accordingly within the overall software development process, such as identifying & organizing requirements and evaluation of the product according to the requirements. For organizing requirements there exist different types of requirements techniques, just as for evaluating the software there exist different types of software testing methods. In this chapter, we will describe general methods available for specifying software requirements, types of software testing, and a brief description of a software development process and how it handles requirements and testing phases [10].

2.1 Software Requirements

Requirements are what the customers, users and suppliers of a software product must determine and agree on before the software can be built. Requirement may serve a dual function: a) as the basis of a bid for a contract, and b) as the basis for the contract itself [8]. A requirement defines “what” a software product is. Requirements should be written because several contractors can bid for the contract. Once the contract is approved, the contract should be written in a system-defined way, so that the client can understand and validate what the software will do.

Requirements engineering process is iterative, because requirements regularly change during the development process. Requirements often have to be refined or added during the

next stages of the development process. Many software development organizations are also facing problems in collecting adequate requirements. The insufficient requirements and changing requirements are the main reasons why many information technology products fail to deliver within their schedule and budget limit. Requirements are the base for both the software development and the project management activities [10].

2.1.1 Types of software requirements

Software requirements have many different classifications. Having looking at previous research articles by different authors, following articles were considered to be relevant to our research.

In detail Linda Westfall et al., described about software requirements, why are they used, what are they, who uses it, when are they used and how are they used [8]. Denger Christian et al described several techniques that are to be used to specify requirements [16].

Requirements are described using natural language, diagrams and tables. There are also alternatives to natural language specification i.e., structured natural language, design description language, graphical notations and mathematical/formal specifications. Structured natural language expresses requirements in standard forms or templates. Design description language contains more abstract features and it is similar to a programming language. Graphical language is used to define functional requirements, such as use-case diagrams. It is a graphical language with text annotations. Mathematical specification is based on mathematical concepts, such as sets or finite-state machines. A Formal/unambiguous notation reduces arguments between customers and contractors, although most customers do not understand formal specification [16].

In smaller projects, requirements information is documented in the single software requirements specification (SRS) document. In larger projects requirements are specified in multiple documents. For instance, business requirements are documented in Business Requirements Documents (BRD), Marketing Requirements Document (MRD), and in scope document. User requirements are documented in a set of Use Cases in a tool or in a User Requirements Specification (URS) document. The Software non-functional and functional requirements are documented in Software Requirements Specification (SRS). An external interface involves SRS or individual External Interface Requirements (EIR) documents [8]. Other way to specify requirements is to document them in a requirements database or tool. Figure 1 illustrates different levels and types of requirements [8].

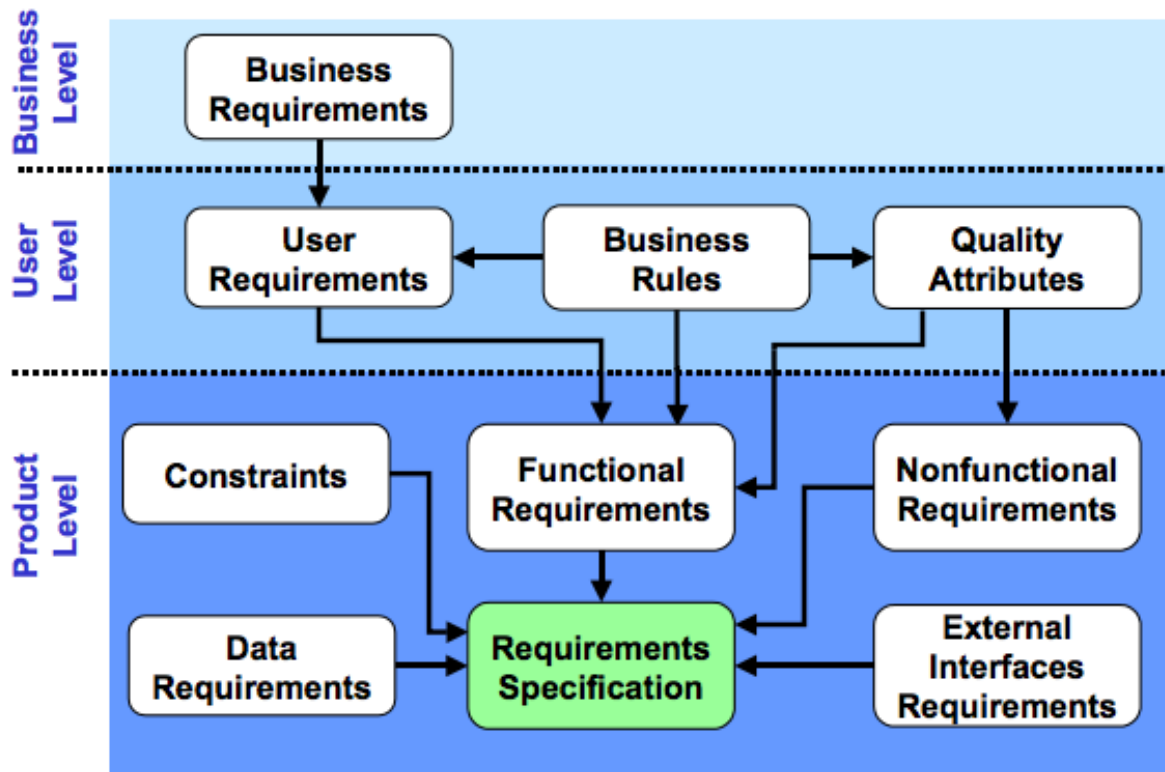


Figure 1 Different levels and types of Requirements [8].

Business level. Business requirements are described in terms of customer/stakeholders organization who is requesting the development of a software product. In general, business requirements define why the software product is being developed [8]. Business requirements state that why the organization is undertaking project/product. It makes up the project owner, stakeholders and project team that needs to be involved in the development.

User Level. User requirements contain statements in natural language with diagrams of service. User requirements must describe functional and non-functional requirements so that requirements are understandable by system users who do not have much technical knowledge. Functional requirements must describe system service in detail, i.e. how the system must react to particular situation and how system should behave in response to the particular input. For instance, user must be able to select a subset from database or should be able to search initial set of databases. Non-functional requirements are constrained on service or functions offered by the system, such as time constraints, constraints on development standards, or process [16]. Non-functional requirements consist of product requirements, organisational requirements, and external requirements. Product requirements specify that product should be delivered in a certain way, such as execution speed of system, reliability. Organisational requirements are impact of organisational procedures and policies such as process standards used in an organisation, requirement implementation. External requirements arise from factors that are external to system and its development process, such as interoperability requirements.

User requirements will give us an idea about the functionality of a software product from the standpoint of different users of that product. These user requirements define what the software should do in order to achieve their objectives. To meet a single business

requirement, there might be multiple levels of user requirements, which need to be satisfied. To fulfil a user requirement, multiple functional requirements may be needed.

Business rules in user's domain describe specific standards, regulations, guidelines, and policies, which define how the user does business. The software product must comply to the business rules in order to function appropriately within the user's domain.

User level quality attributes are non-functional characteristics that define the software product's quality. Often quality attributes encompass reliability, safety, maintainability, portability, usability, security and other properties. Quality attributes also translate into product-level functional requirements, which specify the characteristics the software must possess in order to obtain the nonfunctional attribute [8].

Product Level (Figure 1) External interface requirements define the requirements for the information flow across shared interfaces to hardware, users and other software applications outside the boundaries of the software product being developed.

The constraints define any restrictions imposed on the choices that supplier can make when designing and developing the software [8].

The data requirements define the specific data items or data structures that must be included as part of the software product. Sometimes software requirements are allocated downward from a set of system requirements, or the software requirements may be dependent on the limitations and resource availability of its system's environment [8].

Functional requirements describe the functionality of a software product. Non-functional requirements describe the characteristics, properties or qualities of software product and how well the software product performs its functions.

Westfall describe that business and user-level requirements feed into product requirements at system level as shown in figure 1 [8]. The system architecture then provides requirements from set of system requirements downward into hardware, manual operations and software components. Linda focuses on functional requirement because it has detailed information, which helps in designing a system. Functional requirement indicates what system shall do [34]. The largest part of the requirements specification deals with functional requirements that is the system input, output and the relationship between the two [35]. Some of the good characteristics of functional requirements are: 1) There is only one functional requirement per statement. 2) They are as full, definitive, complete and accurate as possible. 3) They are short and as simple as possible.

2.2 Software Testing

Testing has the ability to identify errors early and it improves customer satisfaction by providing defect-free product. Testing is a quality assurance method used throughout software development in order to discover uncovered requirements, design and coding errors in the program [25]. Robert v Binder defines software testing as "the execution of code using combination of input and state that chooses to disclose bugs". Testing can find out different types of defects, apart from design and coding errors.

2.2.1 Types of software testing

Software testing has different classifications that are explained by [9]. Some organisations have separate teams for different phases of the testing, whereas other organisations have one testing team, who works for all the testing phases. One testing team might be a better option in cases where knowledge from previous phases would be needed for the future phases.

Unit Testing. Unit testing is also known as module test because each test examines individual units of code, which comprise the application. Based on technical design document each single test is validated to perform a specific task, which will produce specific results.

Static Testing. Static testing is generally not a detailed testing where the software isn't actually used. But this testing verifies the sanity of the code, algorithm or documentation. It is basically verifying the code or manually reviewing documents to find errors.

System testing. At the system testing stage the entire system is tested towards the system specifications. These specifications are defined within the business analysis document declaring the actual purpose of the software [9]. Prior to system testing the unit testing is performed where testing is performed on individual parts of the entire system. System testing is a stage that is to prove that the software is working as expected and the system is performing well. Different stages of system testing are as follows:

- a. Functional/Functionality Testing
- b. User Interface Testing
- c. Error exit testing
- d. Help information testing
- e. Integration testing

Dynamic Testing. Dynamic testing proves that the software deliverable is functioning according to its specifications. Test scripts and test results are agreed to be completed within acceptable time frame.

Performance Testing This is the most effective way of measuring the software's capacity and scalability. The expected performance ratio of users and response times expected are measured and standardised before the testing is executed on the software [25]. A performance analysis tool can be used to measure the live behaviour of the software and the users interacting with the software. Different stages of performance testing are as follows:

- a. Stress/ Load Testing
- b. Volume Testing
- c. Limit Testing
- d. Disaster Recovery Testing

User Acceptance Testing (UAT) This is a process of obtaining confirmation that the developed software meets the expected requirements of the customer. This is usually performed at the final stages of the project and it often occurs before a customer accepts the new software. The users of the system/software carry out the UAT, which provides the development team with the required feedback to improve or change the system. Test

designers will design test cases with general use cases of the system expected behaviours and scenarios, which include severity levels for identifying the impact if the test case has failed. To identify some of the unexpected scenarios the users will attempt, the test designer must analyse the current system and identify the different possibilities. Unexpected or less obvious scenarios can be tested through Boundary Value Analysis & Equivalence class partitioning testing methods.

Compatibility Testing/Data Migration. This testing is performed to measure the compatibility of the developed system/software and the capability of its data migration and adaptability for different subsystems or the previous versions of the same software.

Security Testing. This testing is performed to measure the security levels of the developed system by using several methods and compromising the security of the system. It measures the vulnerability and improves the security.

Blackbox Testing. Black box testing is a testing technique that neglects the internal structure of the system and concentrates on the output returned against any input and implementation of the system. In some instances it can be referred as a high level functional testing [10].

Whitebox Testing. White box testing is a testing technique that takes into account the internal structure of a system. It is also called structural testing and glass box testing.

Black box testing is much used for validation and white box testing is much used for verification.

In our thesis we have focussed on functional testing (part of system testing), which is a base line technique used to design test cases for number of reasons. Functional test case can (and should) begin as part of the requirements specification process and continue through each level of design and interface specification [10]. Finally, functional test cases are less expensive to design and execute compared to the other.

2.2.2 Test cases

Testing is a process of verifying if a required function is executed as expected through the entire software development life cycle. According to Denger et al, test cases are established from requirements documents and need to be run across the product development in order to find out whether or not the achieved results are the expected ones [16].

Test cases are a set of test inputs, execution conditions, and expected results established for a special objective, such as to study a particular program path or to verify compliance with a particular requirement. Test cases can be carried out in three main levels: unit testing, system testing and acceptance testing as described in the previous section. Moreover, anytime a unit or a system is combined with another one, integration testing is carried out. Test cases must be built based on requirements documents that conform to these levels. This is the reason why the requirements should be written in such a way that the test cases could be developed efficiently and productively. Requirements and testing play a major role in diverse stages of the development life cycle [16].

Test case is a scenario that is made up of sequence of steps with conditions or variables, where test input is given and the software product is executed using the inputs to see how it works. An expected result is outlined and the actual result is compared with the expected result. Every requirement is expected to pass the test case. Designing test cases are divided into three parts: Information, Activities and Results. “Information” consists of general information such as a case identifier, test case version, name of the test case, and brief description of test case [16]. It should also specify any software and hardware requirements, as well as the setup or configuration requirements. “Activities” consists of actual test case activities such as dependencies during testing, activities that are done at initialization of the test, activities that are done after test case is performed, step by step actions that are done while testing and input data that is to be supplied for testing. “Results” are outcomes of a test case where result consists of expected results. The criteria are necessary for the program to pass the test and actual result is recorded.

Example: As mentioned above, designing test cases is divided into three parts: Information, Activities and Results. The test case used in the example is based on a Purchase Order System. Information fields (Table 1) are Test case name, Input data, and Initial condition. Activities consist of Test steps, whereas results consist of Expected results.

Table 1: Test case Example [16].

Test case name	Select product
Input data	Product ID:123667
Initial Condition	-PO Status = “in progress” - PO #2735 is opened and displayed for vendor #976 - User ID #96 is authorised - 8 lines items have been already inserted
Test Steps	- Check satisfaction and initial conditions - Compare actual versus expected result
Expected results	- PO #2735 is till opened and displayed for vendor #976 - User ID #96 is still authorized - 9 lines appears on the PO

2.3 Aligning RE-Testing in Software development life cycle

A development process is a structure imposed on the development of any product. Similarly, a software development process or life cycle is a structure imposed on development of Software products. There are many models for software development processes. Every process has its unique way of describing approaches to different tasks involved in the development process or activities that take place during the development process. A Software development life cycle (SDLC) starts with requirement analysis and definition

phases, where the purpose of the software product should be determined, and a set of requirements should be developed. Next, software product must be designed and produced, in compliance with all the requirements, which were set in the previous stage. Next, implementation phase, the result of design phase is translated into code for the project.

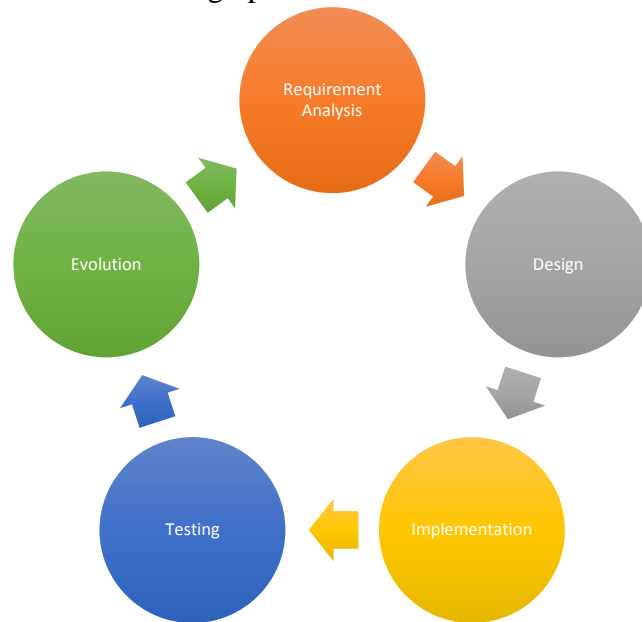


Figure 2 Software Development Life Cycle (Figure is illustrated from Waterfall Model) [28].

Next is the testing phase. Developed code should be tested using dynamic and static analysis, and also security testing to make sure that the application is not exploitable by hackers, which may result in critical security issues. Once the software product is secure enough to use then it is implemented in real world to test i.e., beta testing. Later, it enters into the maintenance/evolution phase, which allows application to be adjusted to organizational, systemic, and utilization changes.

Waterfall Model. The waterfall model was the first structured approach to system development. It is also known as linear-sequential life cycle model [30]. This model is easy to use and understand. Each stage should be completed before moving to the next stage within the cycle. At the end of each stage the project is reviewed to ensure and assure that it meets the requirements.

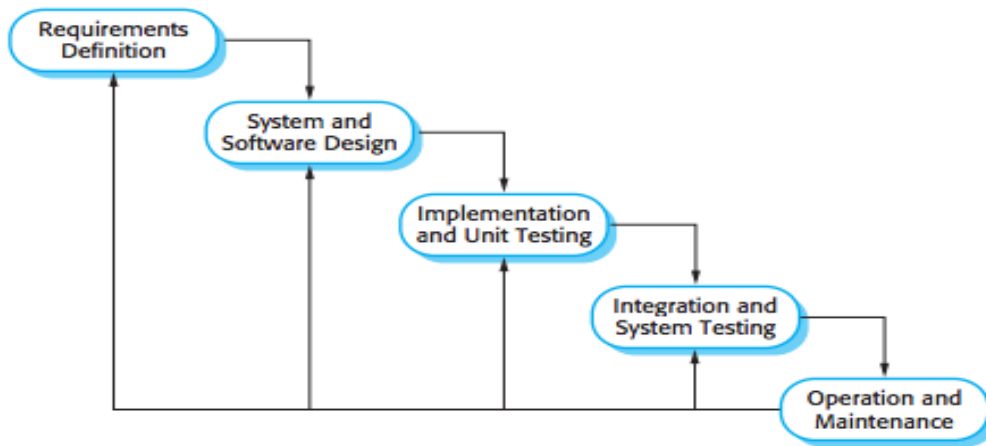


Figure 3 Water Fall Model [38].

The waterfall model has no direct connection to the requirements phase and the testing phase. Each phase should not start until the previous phase has finished. In practice, these stages overlap and feed information to each other [38]. During the system design phase, problems with requirements are found. During implementation phase, problems with system design are identified and so on.

Incremental Model. In the incremental model the requirements are divided into subsets. The model involves multiple development cycles that makes the software life cycle look like a “multiple waterfall” model. The requirements in the first increment are defined in detail and that increment is developed. During development, requirement changes for the current increment are not acceptable. Once the increment is completed and delivered, customers can experiment with early system version, which helps them clarify their requirements for later system increments [38]. As soon as new increments are completed, they are integrated with existing increments so that functionality improves with each increment version.

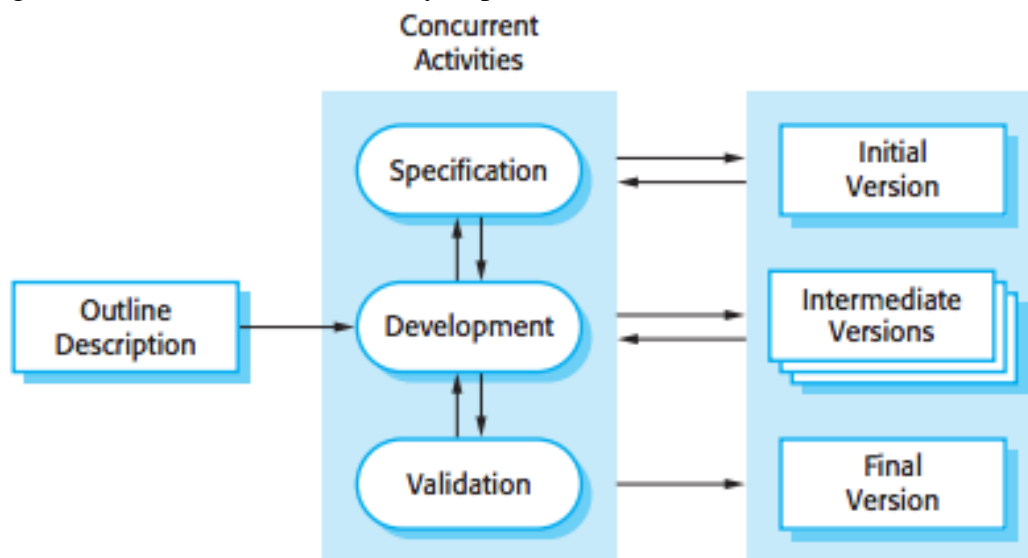


Figure 4 Incremental-Model [38].

Likewise the waterfall model, the incremental model has no direct connection to requirement phase and testing phase. The software life cycle is divided into smaller cycles, which makes module to manage easier. Each module goes through the requirement, analysis, design, implementation and testing phases. At first module stage, software is developed. Each version adds functionalities and new features to the previous stage. The process continues until the system is completed [30].

V-Model. V-model represents an extension of the waterfall model. Instead of moving down in a linear way, the process bends upwards after the coding phase. V-model connects each phase to testing phase. This model considers three dimensions of requirements and test artifacts that connect through work processes. One is the Abstract level dimension from goals down to source code (i.e., requirements to testing side). Test artifacts is used to verify the code and also requirements. The relation between abstract level and test artifacts can be bi- or uni- directional. The third dimension is time, in which the processes, the product and the project change and evolve [5]. Time dimension has effect on artifacts. There is also dimension of product lines, which is used when the development is based on product line engineering approach.

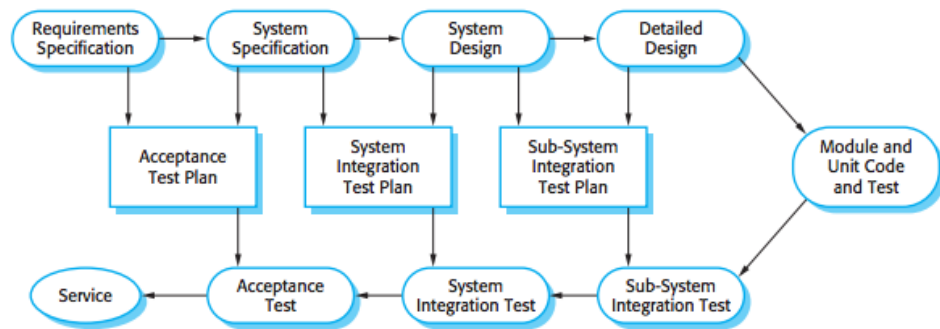


Figure 5 V-Model [38].

As shown in figure 5, each phase is connected to test plan, which means that the requirement phase and the testing phases are directly connected. Three stages of testing are done in v-model, in which system components are tested. The integrated system is tested and finally, the system is tested with customer data.

Spiral Model. The spiral model allows risk assessment feature. The development of each version is carefully designed using the waterfall model steps. In each iteration around spiral model, more complete versions of the system are built [31]. The risk assessment tasks are used to evaluate whether development should go on or not. For each of project risks, the detailed analysis is carried out. Steps are taken to reduce the risk. For example, if any risk is identified to inappropriate requirements, a prototype system is developed [38]. Thus schedules and cost are revised each time when risk assessment is performed. Based on the output, projects might be cancelled if returns cannot be expected more.

According to our analysis, the spiral model has no direct connection between requirement phase and testing phase. The software process is represented as spiral rather than a sequence of activities with some backtracking from one activity to another [38]. Each loop in the spiral

represents a phase of the software process. Innermost loop with system feasibility, the next loop with requirements definition, and the next loop with system design and so on. The main difference between the spiral model and other software models are it explicitly recognizes risk.

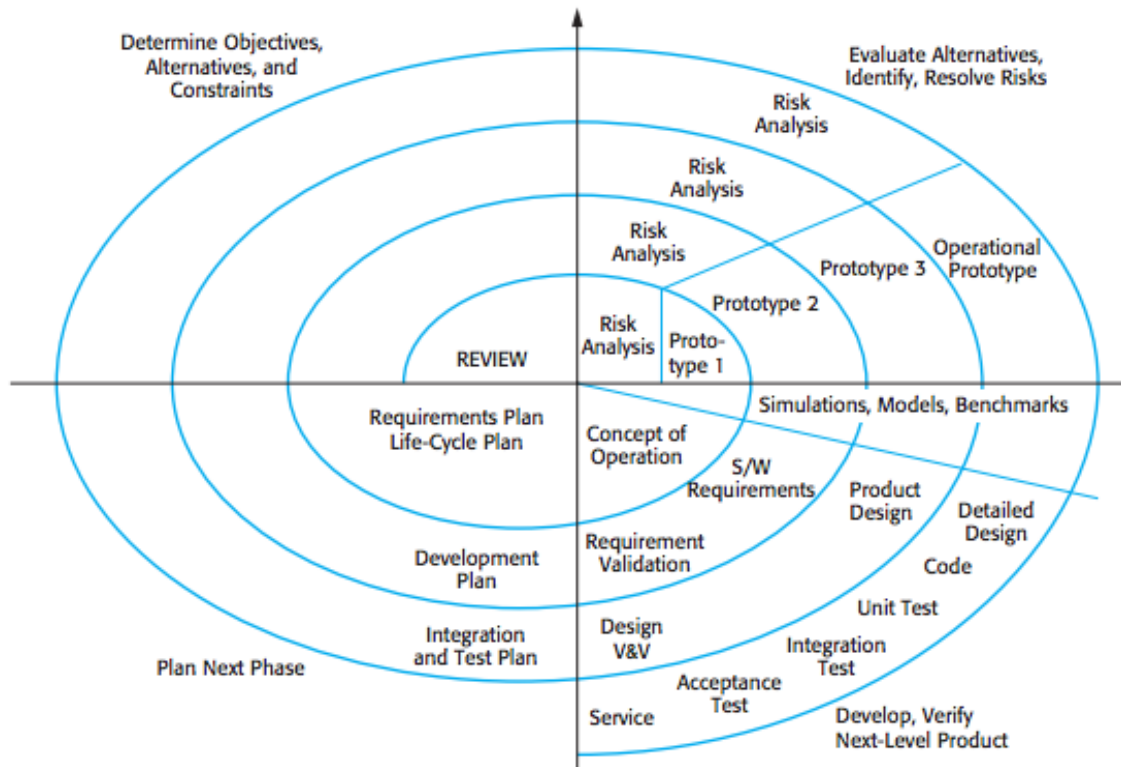


Figure 6 Spiral Model [38].

Agile Methods

Agile software development methods are based on iterative and incremental development. It promotes adaptive planning and provides flexible response to change. Agile process breaks the tasks into small increments and does not involve long-term planning. Iteration frames last for one to four weeks. Iteration frame involves planning, requirements analysis, design, coding, unit testing and acceptance testing. Agile method focuses on different aspects of software development life cycle. Some agile methods are extreme programming, pragmatic programming while other focus on managing software projects [12]. This process minimizes the risk and allows the project to change quickly.

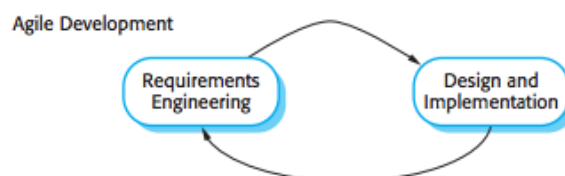


Figure 7 Agile Method [38].

Extreme programming (XP) Method. Among the agile methods, XP has obtained much attention in the past few years. Agile methods definitely have bases in the incremental, iterative, and evolutionary methods. The most important agile method, XP, is much used in combination with other agile methods such as Scrum. XP suggests using TDD as an essential factor for developing high-quality software. The extremely familiar use of “simple, lightweight quality”, and “rise up to an interesting conflict”. Despite the fact that absolute documentation is incomplete, unscientific evidence shows that TDD usage is growing.

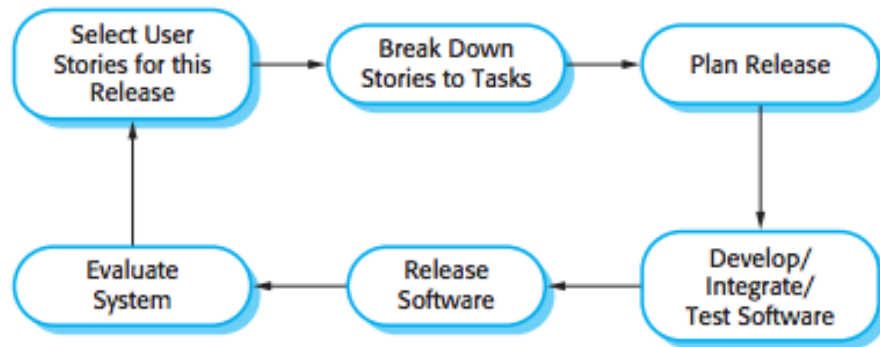


Figure 8 XP Method [38].

According to the author’s understanding, XP has connection between requirement phase and testing phase. In extreme programming, requirements are expressed as scenarios (known as stories) that are implemented directly as series of tasks. Programmers work in pairs and develop tests for each task before writing the code [38]. All the tests should be successfully executed when new code is integrated into the system.

Test-driven Method

Test-driven development is a programming practice that advises developers to write new code only when an automated test has failed. TDD is known by several names that include test-first programming, test-driven design, and test-first design. The *driven* in test-driven development concentrates on how TDD guides analysis, design and programming decisions. TDD considers that the software design is either incomplete or flexible and open to changes. Test-driven development has appeared in combination with the agile process models [6].

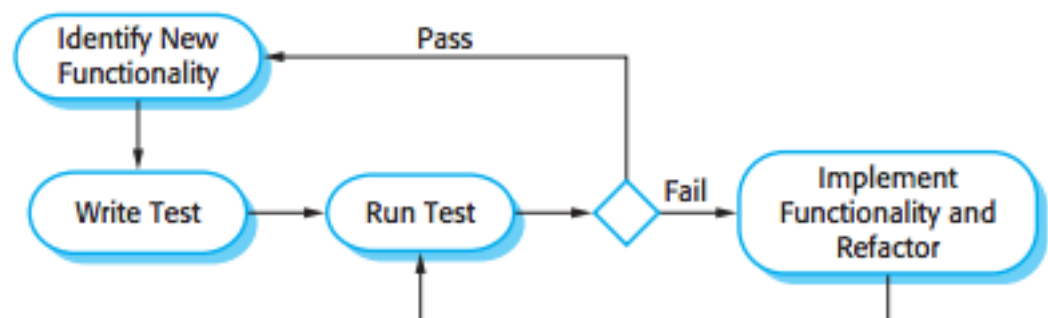


Figure 9 Test-driven development.

Test-driven development is a software development process that relies on the repetition of a very short development cycle: first, developer writes an automated test case that defines a desired improvement or new function. Then develops the minimal amount of code to pass that test, and finally restructures the new code to acceptable measures. Even though developers have been using TDD in several phases for several decades, this software development strategy has proceeded to get attention as one of the core programming practices.

As shown in the figure 9, it clearly shows that requirement and testing are directly connected. Test-driven development is an approach to program development that interleaves testing and code development. Code is developed incrementally, along with a test for increment. Until the code that was developed passes its test, it does not move to the next increment level.

2.4 Conclusion

The relation between software development process and alignment of the software requirements is important to maintain harmony in the development and verification process, as well as delivering the software to the customer within the budget and in expected time. Synchronization between requirements and verification process is crucial in order to assure that the developed software product satisfies customer's requirements [5]. This synchronization can only be achieved if the development process is aligned with the requirements of the customers.

Kukkanen et al. mentioned that requirements and testing represent complementary views of a system and have a synergistic relationship so both can benefit from each other by taking the other discipline into account. The linking of requirements and testing helps to run the project effectively and ensures that the project meets the planned schedule [18]

3 Research Methodology

This chapter provides information on the research methods used during this master thesis project. We will first describe the methods chosen for each research questions and the motivation for choosing them. After that we have described how the methods have been applied.

3.1 Methods used for answering the Research Questions

Research methodology is used to systematically solve a research problem. Research approaches include quantitative approach, qualitative approach, design science approach and mixed approach. There are many useful research approaches. The selection of research approach depends on the research questions chosen.

The following Research Questions were defined for this thesis work:

RQ1. What are the benefits of aligning test cases with functional requirements?

RQ2. How to align functional requirements with test cases?

We have chosen the literature review methodology for RQ1: The main benefits of aligning test cases with functional requirements.

The literature review performed in relation to RQ2 is to search and analyze different kinds of requirement templates and test case templates that are currently being used in the market to develop a software product or service. Further on, the results of the analysis would be used to create the template by aligning test cases with requirements. Also, the ideas gathered through our survey will help us further improve the alignment between requirements and test cases. Both research methods, i.e. literature review and survey (i.e., qualitative approach is performed) to answer RQ2.

Table 2 Overview of Research Methods.

Research Question	Literature Review	Survey
RQ1	✓	
RQ2	✓	✓

The survey was not considered for answering the RQ1, as the main goal of our survey is to gather feedback on the new methodology proposed through our work and to gather the benefits of new template created. The actual benefits of aligning test cases with functional requirements have been formulated based on the extensive literature review performed as part of our work.

3.2 Motivation for research methods used in this thesis

The methods chosen for the research are literature review and survey. The literature review has been chosen to analyse data and to collect information in order to create a template. We have also chosen a qualitative approach to validate and improve the results of our work through surveys. The survey instrument has been designed using Likert scale for collecting the data from the targeted user groups.

3.2.1 Motivation for Literature review

The relevant literature is an essential feature for any academic project. An effective review creates a foundation for advancing knowledge [44]. It facilitates theory development, close area where research exists and uncovers areas where research is needed.

In the requirement engineering and testing field, we see few published review articles. As a result the progress was implemented in our studies. The clear intention was to reduce the difficulties in meeting schedules, budgets and have problems to predict the quality of the developed products. A particular goal was to advance the state of theory within the alignment of requirement engineering and testing field.

From the initial background chapter 2, we quickly learned that no one of the software development methods is sufficient to guarantee that the testers are actually testing the user requirements and that the product delivered is what the user has asked for, which provided the motivation for our article.

In this paper, we first identified the type of articles that are appropriate. Next, we spend most of the study based on published articles, author and what we have learned from our experiences. We then discuss the reviewing of different alignment methods, three requirement and three testing templates. Finally, we conclude by providing template and summarizing our expectations for a review article.

3.2.2 Motivation for Survey Questionnaire

The decision to choose qualitative methodologies is to yield rich information, which is not obtainable through statistical sampling techniques such as quantitative approach, design science or mixed approach. The qualitative methods can be used for better understanding of any phenomenon, which is not yet known. They can gain new perspectives on things already we knew, or gain in-depth information, which is difficult to deliver quantitatively [39]. The ability of qualitative data to more fully describe a phenomenon is important consideration not only from the researcher's perspective but also from the reader's perspective. "If you want people to understand better than they otherwise might, provide them information in the form in which they usually experience it" [40].

William [37] described that surveys are divided into two broad categories: the questionnaire and the interview. When most people think of questionnaires, they think of email surveys. There are many advantages to use email surveys. They are inexpensive to administer. A questionnaire can be sent to a wide number of people. Respondents can fill it in at their own convenience. The questionnaire provides the researcher with data that can be evaluated and can be understood. They help us to provide an effective way of gathering data from many people. Our survey is also designed to gather information around merging test cases with functional requirements. This helped us to clearly define our requirements for building the template [11].

In contrast to that, interviewer works directly with the respondent [37]. Respondents are forced to select between the alternative answers the interviewer gives them. It can be difficult

to retrieve reliable information on attitudes, opinions and values. Interviews can be time consuming in terms of data collection and data analysis. The best approach will always be based upon a combination of factors such as time, the complexity of data collection, the sample profile and budget [36].

Our survey uses questionnaire method in order to keep it short and less time consuming and efficient enough to capture all the needed information. Before designing the survey questionnaire, we have developed a set of objectives for our topic and the list of information that we are trying to capture. The research questions and the list of objectives helped us to plan the survey questionnaire.

There are two types of questions used to collect information through the survey. 1) Structured or fixed response questions and 2) non-structured or open questions.

We chose structured questions, as they help respondent to choose from a closed set of answers. Structured questions make analysis much simpler, take less time to answer and allow us to collect the required data effectively. Structured questions are well suited when participants have good understanding of the topic and when trying to capture new ideas from participants. Surveys are useful research methods where one needs to gather same kind of data from different people or events, in a consistent and efficient way.

3.3 Research Process

As explained in the above section the two main research questions we have formulated to define our thesis work are to 1) to identify the methods to align functional requirements with test cases and 2) to derive/formulate benefits that could be achieved by completing the alignment. We have identified that both the RQ's can be answered by performing some literature review so as to draw some conclusions, however, instead of just formulating the methodologies on how to align requirements and test cases and deriving benefits of that new methodology, we have also created an actual template that consists of the essential elements of both the test cases and requirements. Once receiving the suggestions from the companies on the initial template created (through surveys), an improved template has been derived in the later conclusions of our thesis work.

The idea of this newly created template not only helps the researchers who would like to further analyze/perform research works related to our topic, but also will pave them a foundation to use the template to test in an organization so as to draw some tangible quantitative conclusions.

Figure 10 describes our research process. We have chosen two research methods for our thesis: the first one is a literature review and the second one is survey. In the first step, literature review is used to analyse alignment methods, three different requirements templates and three test case templates.

And the second step is used to collect information in order to create a test case template. In the third step, we have designed a survey, which has been distributed to target group. In the fourth step, we have collected the feedback from the participants and fifth step is to improve the template created according to our analysis and survey feedback.

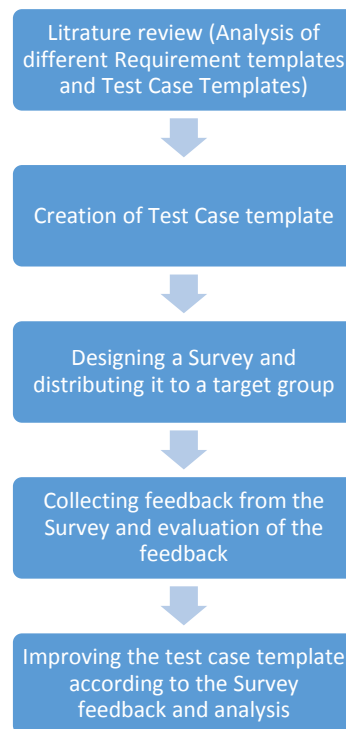


Figure 10 Research methodology process

We have found in the literature, three requirement templates and test case templates (See table 9). We have identified common items of the templates and derived one template (See table 20).

3.4 Literature Review

Literature review is used to extract information for RQ1: What are the benefits of aligning test cases with functional requirements?

For RQ2: How to align requirement with test cases? Literature review and quantitative approach such as survey are used.

During the process of our literature review, steps presented by Oates as shown in figure 11: i.e. searching, obtaining, assessing, reading, critical evaluation and writing a critical review [12] have been extensively utilized

Searching
Obtaining
Assessing
Reading
Critically evaluating
Writing Critical Review

Figure 11 Literature Review Method Based On Oates [11]

3.4.1 Searching

For the literature review, the resources we have chosen for search include Google scholar, Malmo university digital library and other digital libraries like IEEE, Science Direct, and Springer Link. A sample summary of both the search strings related to RQ1 and RQ2 and also the articles retrieved through the search phrases have been tabulated as shown below.

Searching Process for RQ1: The keywords used to find resources for RQ1 are “benefits of aligning requirements with test cases” and “challenges of aligning”, “issues of aligning test cases with requirements”, “advantages and disadvantages of aligning test cases with requirements”, “merging of requirements and test cases”, “mapping requirements to test cases”.

Table 3 RQ1 Searching string categories.

“Benefits of aligning requirements with test cases” OR “issues of aligning test cases with requirements” OR “advantages of aligning test cases with requirements” OR
--

Searching Process for RQ2: The keywords used to find resources for RQ2 are: Alignment of requirements and test cases, Problems with traceability, requirements process, testing process, test case template, different types of testing and requirements. We searched articles using keywords. We also used the references of already found articles to find other related articles on the same topic.

Table 4 RQ2 Searching string categories.

Alignment “Alignment of requirements and test cases” OR “ Mapping requirements to test cases” OR “Linking functional requirements with testing” OR “merging requirements and test cases” OR
--

The search process is based on the criteria that researchers establish before they begin the process of identifying, locating, and retrieving the research needed to address the problem. The eligibility criteria define which papers will be included and which ones will be excluded from the systematic review [31]. The criteria are also applied to make sure that the relevant papers are included in our review and no paper is excluded without the complete evaluation. The inclusion/exclusion criterion was different throughout the selection process. Papers have been eliminated from the list of the review articles, if the titles are not relating to our topic. The papers with unmatched abstracts have also been excluded from our selection.

Furthermore, the full text of the paper is needed to include or exclude papers from the bibliography. We determined whether articles in the search results lists are related to the literature review. Irrelevant articles are excluded by reviewing the titles and if the articles are not clear from the title then we review the article abstracts. When reviewing articles, we paid attention to the authors. If the same group of authors published many articles on the same study, we have considered which articles are more suitable to the literature review, and excluded the articles with duplicate information.

Final inclusion/exclusion decision was taken after retrieving the full texts of the articles. Studies identified by the electronic and hand search can be clearly excluded based on titles and abstracts. Studies have been excluded because they met one or more of the exclusion criteria, thereby making the study not relevant to the review's purpose [32]. We have also excluded some studies based on time period.

From the literature review, we have excluded list of articles titles that are not related to our research

- Combining software requirements specifications with Use case modelling
- Combining structural and functional test case generation
- Automatic generation of path tests by combining static and dynamic analysis
- Developing operational requirements
- Use case maps for the capture and validation of distributed systems requirements
- Software product line testing

3.4.2 Obtaining and Assessing

Resources for this thesis work are obtained using literature review from articles, journals, conference papers and technical reports. Several resources such as Malmo university digital library and Google scholar were used to obtain information.

Obtaining and Assessing for RQ1: Articles that were published in recent years were assessed. 15 articles were obtained by using the keywords that are mentioned in section 3.1.1. Out of 15 articles, 4 articles were considered that describe about benefits of aligning requirements with testing.

Obtaining and Assessing for RQ2: These research questions gave authors a possibility to explore and find latest improvements in Requirement Engineering and Testing domain. Structured keywords were used to find related material and check each article by first reviewing the year of publication to focus on recent work, number of citations to determine the scientific value of the article and abstracts. 30 articles were obtained using keywords, which are mentioned in section 3.1.1. Out of these 30 articles, 23 articles were considered as most suitable to RQ2. The articles discarded either provide information about tools for developing requirements, risk factors for testing and other information that is not related to RQ2. In the obtained articles, the approaches were chosen that were relevant to requirement templates, testing templates, traceability between test cases with requirements, and aligning or mapping test cases with requirements.

3.4.3 Reading

Reading for RQ1: Similar process is applied to RQ1 as mentioned above. We considered articles that provide information about benefits of aligning test cases with requirements.

Reading for RQ2: For RQ2 we considered articles that provide more information for aligning test case with requirements. We start reading the abstract of the article that helps us to find whether the article is relevant to our research. As we proceed with our reading, we get familiar with the methods that are used for their research. If we find that article is providing more information, then we continue reading introduction, result and conclusion of the article.

Thus, there is more clear understanding about what authors have researched, what is needed to implement in future and what is lacking.

3.4.4 Critical Evaluation

Critical evaluation for RQ1: For this research question we read the articles that are related to our topic. The assessed articles provide information about benefits of aligning requirements with test cases. The articles provide more theoretical factors rather than practical factors. The evaluation was done by assessing, why the content of literature is important.

Critical evaluation for RQ2: The section described about how we have evaluated collected articles. The assessed articles provide similar information about aligning process test cases with requirements. However, there is lack of literature work for aligning test cases with requirements. Most of the authors provide information regarding related subject or provide details about particular case such as requirement based test case priority. The evaluation was done based on content of the literature. We considered articles that are related to align test cases with requirements. Other methods were discarded that provide information about developing requirement, testing tools or other information.

3.4.5 Writing a Critical Review

Critical review for RQ1: To structure the collected data, articles collected were related to our topic. For RQ1, the keywords mentioned in above section 3.1.1 were considered to obtain the relevant articles. Based on obtained articles, the benefits of aligning test cases with requirement are described from each of the obtained articles.

Critical review for RQ2: The keywords used to search different requirement and test case templates mentioned in section 3.1.1. There are several templates for requirements and test cases. We considered the method, which provides details on aligning the both. We also conducted a survey to gather information on test case template that has been created.

3.5 Survey

In order to gather relevant data, survey must be structured in a precise way. The procedure to follow will be obtained from the steps proposed by Neuman [26]. (3.5.1) Defining questionnaire objectives, (3.5.2) selecting a sample (3.5.3) designing the questionnaire format (3.5.4) survey design (3.5.5) pre-testing the questionnaire (3.5.6) pre-contacting the sample (3.5.7) Writing a cover letter and distributing the questionnaire, (3.5.8) following up with non-respondents. This approach has a valuable framework to understand the whole process, which must be performed in the current research. We include (3.5.9) threats to validity.

3.5.1 Defining questionnaire objectives

Questionnaire Aim: The aim is to find out how to align test cases to functional requirements.

Questionnaire Objective: The objectives are split as follows:

- a. To find out how different companies evaluate our template (Answering RQ2).
- b. To gather more information on how we can further improve the created template.

3.5.2 Selecting the sample

The total number of participants in the sample was 9 employees from companies Sony Mobile Communication's, Ericsson and Maersk, the companies selected are also carrying out different businesses. The data is collected from different individuals working with Software

testing and requirements. We are sure that collected information from survey was reliable and provided us with the relevant information to address the current study.

People involved in this data collection are from software companies such as Sony Mobile Communication's and Ericsson, and Copenhagen based Maersk, which represents the Energy and logistics sector. In order to capture wide range of perspectives, the survey has been sent to different organizations dealing with wide range of software development projects. As all of these companies are dealing with different business models, we believe the survey data captured through these organizations will be more advantageous compared to sending survey to similar companies. The industries these companies are working with are Telecom, Mobile Phones, Oil and Gas, Supply Chain Management and Shipping.

There were 9 participants in total that answered the survey questionnaire. They belong to software testing, business analyst and manager positions. Of these 80 % are software testers, and 20 % are from management team. Table 5 shows the total amount of participants from the different areas.

Table 5 Participants from discipline area.

Area	Number
Software Tester, Business Analyst	7
Manager	2
Total	9

3.5.3 Designing the questionnaire format

Our Survey is using Likert Scale for collecting the data from the targeted user groups and below is the description of how the Questions and their options are designed.

Likert scale. Each question would be scored on a scale of 1-5. The format of five-level Likert scale

1. Strongly disagree
2. Disagree
3. Neither agree nor disagree
4. Agree
5. Strongly agree

Approach: The Survey customers shall complete Questionnaire's and from the ratings a Likert scale shall be tabulated as shown in Table 6.

Table 6 Likert scale

Survey Customer	Q.No.1 Rating	Q.No.2 Rating	Q.No.3 Rating	Q.No.4 Rating and so on...
Customer No.1	Neither agree nor disagree	Strongly disagree	Strongly agree	Strongly disagree
Customer No.2	Agree	Disagree	Agree	Strongly disagree
Customer No.3	Neither agree nor disagree	Disagree	Strongly agree	Strongly disagree

For a particular question, the average rating of all the survey customers rating is considered as the final rating. Once all the questions have been rated, the next important step is to prioritize the questions/items of the Test Case template that need further improvements. On top of selecting scores for each question we would also provide a comments field for each question, so as to capture benefits and issues of capturing that particular item in the newly created test case template [35].

The questionnaire is designed to request important information from different software companies about the created test case template because we believe that real time research information will guide us to create good test case template. The questionnaire is created to gain deep understanding on our research aim and research question.

Key Questions of our Survey: There are 4 questions that are closely linked to survey objectives. The questions are divided as 3 sections in order to create more accurate questionnaire.

Table 7 Questionnaire used for the survey

Division	Q No.	Questions	Observation
Section 1	1	What is your present requirement alignment process?	Gathers information about Alignment. [RQ2].
	2	How would you improve the alignment between requirements and testing at your organization?	
	3	How would you rate the idea of linking requirements in the test case template itself?	
	4	How would you rate the test case template we have created?	
Section 2	5	What are the challenges you face in managing requirement alignment?	Collects the information about challenges and issues that are faced due to aligning requirement [RQ2].
	6	When investigating requirements decisions, which of the following components effects decisions as per your view?	
	7	Where do you face more challenges in requirement analysis as per your view?	
	8	Would you foresee any issues with having requirement ID and Test case ID being in the same template?	
Section 3	9	Would the instructions sheet on the template	Collects suggestion

		submitted is clearly understandable?	for improving the template.
	10	Which model is used in your software development organization?	
	11	Do you believe for any substantial improvements within your company by using test case template as we created?	

3.5.4 Survey Design

This section provides information about how the survey questions are designed.

Below are the points, which were considered while designing the survey.

- Goal of the survey questions
- Content of the survey questions
- Order of the survey questions
- Length of the Survey

The initial step was to identify the goal of our survey and then design the questions. While designing the survey questions it was important for us to know what can the survey results achieve for understanding our research questions and improving the result part and how can the survey questions relate to our research topic and the respondent's visual perceptive.

The first section of questions were designed in such a way that they should be

- Applicable to all the respondents of a target group,
- Must be easy to read and understand,
- Must be interesting and should connect the respondent with the goal of the survey.

The first set of questions then follow with a second section of questions to collect suggestions to improve the thesis work.

After designing the survey questions we made sure if the questions were placed with appropriate wording and in a meaningful order. The survey had consistent font size, indentation and style sheet.

The survey is designed in a shorter way by considering the level of our respondent's responsibility, commitment and interest. These steps were considered to design the survey questions in a simple, easier and systematic way.

3.5.5 Pretesting the questionnaire

The survey questionnaire sample was first distributed to a specific Verification & Validation team of Sony Mobile Communications, in which the participants have different roles and responsibilities. The comments were taken in consideration for refining the questionnaire. These comments helped us to develop the questionnaire into a more understandable survey. The questions were rechecked for reliability and validity. The final document is a 2-pages questionnaire that is used as a survey. The questionnaire is attached as Appendices A.

3.5.6 Pre contacting the sample

The rate of response is a great issue to solve. In many cases, participants are not really interested in answering survey due to time constraint. Neuman [26] suggests that pre-contacting may increase the rate of response. In this survey pre-contact was done in verbal way before questionnaires were distributed [26].

3.5.7 Writing a cover letter

For our study, a cover letter has not been created, because the initial sample communication was done verbally. However, a short description has been appended to the email when survey questionnaires have been sent to the respondents.

3.5.8 Following up with non-respondents

Not all the survey respondents provided their feedback according to the deadlines mentioned in the first survey email. As a result we have sent some reminder emails to the non-respondents after the deadlines has been passed which finally resulted a 90% response rate for our surveys.

3.5.9 Threats to validity

Wybo Wiersma [42] clearly outlined all the threats that would need to be considered when conducting a survey research. First of there are many limits that needs to be considered while designing a survey i.e. surveys should not be too constrained in terms of users being able to provide answers, should not include too many questions and the questions should be presented in the best possible logical order. Secondly, there are two types of validity i.e. external and internal validity [42]. External validity details the validity of the survey being created beyond the study i.e. how valid is the topic of research in terms of both the selected survey population and also in terms of the selected context. Internal validity mainly details the level of precision to which the actually intended study is to be measured [42]. For our thesis survey, further to considering all the above-mentioned validity threats, we have also considered various other items around solicitation problems, delivery problems, design problems so as to make sure that the survey being sent would result in gathering the best possible feedback.

Threats to Construct Validity. The construct to validity will give us the accuracy of the approach being used in the study. In this study the construct validity threat is in regards to the design of questionnaire being used to send the surveys. The questions that need to be included in the survey questionnaire have been finalized based on many discussions held within the thesis team and also with the professor in charge of the thesis work. Other important threat in this study is that users might get confused when we talk about aligning test cases and requirements and merging test cases with requirements, so a summary of our thesis which clearly shows that we aim to merge requirements into test cases by aligning both of these two functions have been presented to the users taking our survey. The main aim of this study is to get feedback on the test case template created and on the methodology of using test cases as requirements, and as we do not aim to test our test cases template to gather some quantitative data, this study will be limited to perform qualitative analysis [5].

Threats to External Validity. The threats to external validity provide us the general view. Our study was not intended to target any specific industry but to gather benefits of using our methodology and template in any industry within their software development departments. The survey was performed with the users who are used to the terminology of the methodology proposed within our study.

Reliability. Based on the analysis done on our survey results, we believe there is high level of consistency in terms of feedback received on various topics we have highlighted in the questionnaire, so we have minimal issues with reliability threat [5].

4 Literature Review

This chapter describes how different authors have described the benefits of aligning requirements and test cases and also describes about different alignment methods.

4.1 Definition and importance of alignment

Alignment is the “adjustment of RE and ST efforts for coordinated functioning and optimized product development”. Alignment-as-activity is the “act of adjusting or arranging efforts involved in RE & ST so that they work better together”. Alignment-as-state is the “condition of RE & ST efforts having established a coordinated functioning [30]. This chapter presents literature about benefits of aligning, and also presents the summary of literature on alignment of requirements and test cases. Articles considered are shown in tables 8 and 9, where the reviews have been classified based on the goals and research questions of our study.

4.1.1 Benefits of aligning requirements to test cases

This section presents how different authors have described the benefits of aligning requirements with test cases, articles obtained and articles selected to perform literature review for RQ1 are shown in table 8.

Table 8 Results of literature review for research question 1

Search Engine	Search string	Reference	Article Obtained
Springer Link, Google Scholar, ACM, IEE	“Benefits of aligning requirements with test cases” OR “issues of aligning test cases with requirements”	[5]	Challenges in Aligning Requirements Engineering and Verification in a Large-Scale Industrial Context
		[14]	Challenges and practices in aligning requirements with verification and validation: a case study of six companies
		[18]	Approach to link requirements and testing
		[10]	Alignment of requirements specification and testing: A systematic mapping study
		[1]	Linking requirements and testing in practice
	“Advantages of aligning test cases with requirements”	[29]	Aligning Requirements and Testing, current challenges and solutions
		[27]	Linking functional requirements and software verification

	“Disadvantages of aligning test cases with requirements”		No useful articles were found
	“Merging requirements and test cases”		No useful articles were found
	“Mapping requirements to test cases”	[4]	Generation of test cases from functional requirements

Kukkanen et al. mentioned that requirements and testing represent complementary views of a system and have a synergistic relationship so both can benefit each other by taking other discipline into account. Requirements and testing were not fully connected. Testing was mostly based on the considerable experience of testers, instead of the requirements. The links between requirements and tests were disconnected, which led to difficulties in governing the quality of the system under development. Requirements are not achieving all the quality criteria such as integrity, verifiability, and accuracy, and test cases were also suffering from similar issues. No professional requirements and testing tools are used, which made errors evident in the documentation. The implementation of processes, roles and selection of supporting tools becomes easier.

Author describes that concurrent development of the requirements and testing processes makes sure the information flow between requirements and testing processes throughout the entire R&D project life cycle. Concurrent development will reduce the risk of overlapping processes and roles. It also prevents the gap formation in the processes [18]. The linking of requirements and testing helps a project manager to run a project effectively and it makes a project to meet its planned schedule. In practical point of view the R&D pilot project has succeeded to keep its schedule that has been a major improvement in comparison to result of previous project.

Barmi et al. mentioned that linking RE and testing will benefit both disciplines [10]. Making a tight link between will improve the outcome of software development process. It also helps to discover errors early, which in turn will improve the quality of product and lead to more satisfied customers [10]. From perspective of project management, linking requirements and testing will help to reach accurate plan, which in turn improve project planned schedule estimation and budget. Organizations are becoming more interested in linking requirements and testing, but the link is not provided and there is a gap between the areas.

The focus has been mainly on functional requirements than on non-functional requirements. The work which has been done on alignment a lot of attention has been given to traceability. Traceability of requirements can decide what requirement has been covered by which test and how the created test cases cover these requirements. Practices that can be

applied to create a strong link between requirements engineering and testing are involving testers during project planning and requirements reviews, which results in higher quality requirements and improves testability [10]. A systematic approach is presented by Kukkanen et al. [18] for developing requirements and testing processes together with the goal of linking requirement and testing.

Eero et al. mentioned that bringing requirements and testing closer to each other in real product development, which will improve the result of project and customer satisfaction [1]. The article also mentions that traceability helps in many ways: efficiency of change management is improved, improved test coverage where it is more apparent, which requirements the test cases cover, Error removal is more efficient because testers can change their viewpoint between requirements and tests, communication between requirements and testers will reduce assumptions made by testers, The ability to progress in lower quality requirements, and Increase in test results and subsequent products.

Testers will understand higher level requirements better and need less added data. The most important part of the link of requirements engineering and testing is to make sure the flow of data about requirements to the testing process. Most other benefits gained from the link results in higher quality requirements [1]. The two practices, test traceability to requirements, and linking testers with requirement owners, focus on the information flow.

Hendrik et al., indicates that requirements management and quality can greatly benefit from introducing software verification [27]. For parts of an industrial product the technique has proven to be extremely efficient. The connection between testers, developers and requirements managers can be simplified using technical scenarios. Requirements change rapidly during early development stages. The reason for better coordination of requirements and scenarios is that the number of scenarios is extremely lower. Formal scenarios give an assuring alternative for maintaining links between implementation analysis and component requirements. The approach has been proven to be highly capable.

Testing and verification have validated to show a great alliance in which both methods identify unique errors. Testing is essential on system level whereas verification offers full coverage on function level. Verification is integrated in a development process. The reason for the productivity is the higher manageability for scenarios linked to requirements.

Kukkanen et al [18] investigated that improving RE and testing process together with the dual goal of improving client satisfaction and product quality. Integrating requirements and testing processes, including clearly defining RE and testing roles for the integrated process, improves alignment by connecting processes and people from requirements and testing. Communication and interaction between requirement and testing includes early tester participation, traceability approaches, consider feature request are the alignment processes that improves the link between requirements and testing.

Traceability between requirements and other development artifacts support impact analysis, maintenance costs, increased test coverage, and quality in the final products. Tracing is also crucial to software verification for the high quality development. Model-based testing is research field with formal models and languages for representing requirements. The important benefits of model-based testing are increased test coverage, increased testing

productivity. The generation of test cases directly from the requirements completely links the two without any need for manual testing.

The alignment challenges classified through this study are aligning goals and aspects within an organization, cooperating successfully, good requirements specification quality, validation and verification quality, maintaining alignment when requirements change, requirements abstraction levels, tracing between artifacts, time and resource availability, outsourcing of testing.

Sufficient and effective software development needs coordination between people, activities and artifacts involved. Approaches and procedures for combining artifacts includes tracing and use of model-based testing. There are many challenges in achieving alignment including traceability. This implies that communication and coordination between people is important and also between requirements engineers and testers [14].

4.2 Alignment

In this section we present how different authors have described alignment methods and the results of the literature review for research question 2: “How to align functional requirements and test cases?” The articles obtained and selected while performing the literature review for RQ2 are shown in table 9.

Based on the results of table 9, below are the different methods to align requirements and testing.

- Requirement based testing.
- Traceability (Linking processes, Linking people and Applying good practices), Traceability with the help of tools, Test traceability to requirements.
- Model based testing (First Approach and Second Approach).
- Code centric approaches, model-centric approaches.
- REST taxonomy.

Table 9 Results of LR for research question 2

Search Engine	Search string	Reference	Article Obtained
Springer Link, Google Scholar, ACM, IEE	Linking functional requirements with testing	[18] [27]	A systematic approach to link requirements and testing. Linking Functional Requirements and Software Verification
	Alignment of requirements and test cases	[10] [41]	Systematic mapping study on alignment of requirements specification and testing. A taxonomy for requirements engineering and software test alignment.

	Mapping requirements to test cases	[4]	Generation of test cases from functional requirements
	Requirement based testing	[2]	Requirements-Based Testing Process in Practice.
		[3]	What is Requirements-based testing?
		[13]	Automatic test case generation from functional requirements in NDT.
	Traceability (Linking processes, Linking people, and Applying good practices)	[19]	The Role of Requirement Engineering in Software Development Life Cycle
		[5]	Challenges in aligning requirements engineering and verification in a Large-Scale Industrial Context.
		[15]	Requirements traceability in the model-based testing process
		[17]	
		[24]	Automated acceptance tests and requirements traceability
			An analysis of the requirements traceability problem.
	Model-based testing	[13]	Automatic test case generation from functional requirements in NDT
		[15]	Requirements traceability in the model-based testing

4.3 Different kinds of alignment

Below are the different alignment methods on aligning requirements and testing processes.

4.3.1 Processes

The requirements and verification process are independent processes and are not aligned as a result of them being independent. Moreover, processes can use different standards of documentation, which adversely affect the hand-over between different parts of organization. Frequent process changes negatively affect alignment and it would take time for people to learn and use the new processes. Sometimes, people hesitate to use a process knowing that it may change soon [5].

The time distance between the requirements development and test artifacts can create alignment issues. Requirements can be accepted without having test cases linked with them. This can lead to non-testable requirements. Process changes are time consuming, mainly when the changes are requiring many units. Taking these challenges into consideration, the requirements and verification processes are not aligned [5].

4.3.2 People

Requirement engineers at times lack knowledge about implementation as well as testing, while testers lack knowledge of requirements. This has a negative effect on alignment between requirements and verification processes. If there is no cooperation between requirements team, developers and testers, then it will affect the alignment. When there is a lack of communication between requirements and testing people, alignment is affected individually [5].

4.3.3 Tools

Software tools play an important role in alignment between different artifacts. The lack of correct tools affects the alignment and it is very important to have good and easy to use requirements and verification tools. If the tool is difficult to use, then people are not ready or confused to use them. It is very important for testers to have a good requirement management tool which carry information about requirements and also the flow of requirements.

Tools for maintaining quality requirements are very crucial, else there is a risk that quality requirements are not tested. Lack of tools will also result in misalignment between requirements and client needs, and therefore affects customer satisfaction with the final product [5].

4.3.4 Requirements process

Requirements sometimes are not given enough consideration and attention by other departmental units, such as development and testing units. Developers do not regularly review the requirements, and determine requirements that cannot be executed during development, even when requirements are agreed in advance. This could be due to lack of involvement of developers and testers in requirements reviews. If the requirements gathering process is not handled appropriately, then it becomes more difficult to define requirements, especially requirements at a high abstraction level.

Requirements engineers do not think about testability of requirements. Hence, requirements could become non-testable. Handling with quality requirements is a tough task. When the number of requirements reaches tens of thousands, then it is tough to maintain alignment with testing, especially in organizations working with a large set of requirements [5].

4.3.5 Testing process

Sometimes testers do not have clear guidelines on how to continue with testing. Especially, while testing high-level requirements, such as roadmaps. It is tough to test the products that support roadmaps, as testing takes a long time and is expensive.

Requirement team's involvement in analyzing test cases will lead to better alignment, as this would help to make sure that test cases satisfy more accurately with requirements. Having a good testing process at different development stages, and good coordination between testing units can help improve alignment [5].

4.3.6 Requirement Based Testing

The requirements based testing presents two primary points: first, ensuring that the requirements are accurate, integral, clear, and reasonably consistent and second, developing an essential and adequate set of test cases from those requirements, to make sure that the design and code fully meet those requirements. The Requirement engineering (RE) is the most important stage of the software development life cycle (SDLC). This stage is utilized to interpret the inaccurate, incomplete needs of software into complete, precise and formal specifications. The errors developed at this stage, if undiscovered until a later stage of software development, can be very costly. If it can define the requirement specification properly, errors of later stages will be less [19].

Generating test cases from functional requirements

Gutierrez et al. article presented result of 13 survey approaches to improve the fulfillment in system testing by generating test cases from functional requirements. System test cases might be derived from the requirements of the system under test [Javier j. Gutierrez et. Al.]. The goal of system testing is to verify that requirements are successfully implemented. There are many approaches to derive test cases from functional requirements where requirements are expressed in precise and formal notation i.e., z or algebraic. In this research article the author analyzed and compared different approaches for the generation of system test cases from requirements and generated reports for two sets using different approaches. The first report analyzed 12 approaches and second report analyzed 13 approaches.

This article has approached Denger and Gutierrez analyses and concluded that both approaches have similar conclusions. Denger report works well at theoretical level without describing how to generate executable test cases (Javier J. Gutierrez et al.). The Gutierrez report concluded that new approaches have same lack of problems than the one analyzed in Denger. There is a lack of case studies, documents and supporting tools. All these facts makes hard the application of these approaches over real projects. After analysis and comparison article has listed a table to extract which aspects are treated satisfactorily and which aspects need more development. The main conclusion is that there are many approaches to derive test cases but there is no complete and integrated approach, which describes the whole process. The article has concluded that most common technique used in software organization is Model based testing [4].

Requirement based testing process

Gary et al. (2003) gives an overview of requirement based testing process (RBT) and the two efficient phases to follow the RBT process: Ambiguity reviews and Cause-effect graphing. Ambiguity review is a procedure to identify uncertainties of functional requirements to strengthen the quality of the project requirements [3]. Cause-effect graphing is a test-case design procedure that allows the design of minimum number of test cases to test 100 percent of the functional requirements. The author also discusses about the activities involved in testing which are from test completion criteria to test cases management, providing a detailed overview of the Testing process. Author also mentions that Test Completion Criteria, Test Cases design & Test coverage verification are addressed by RBT.

The studies show the reasons why software projects fail. Requirements and specifications change too often. Requirements and specifications are incomplete. There is a lack of user

input. The RBT process addresses some issues such as it begins at the first phase of software development where the correction of errors is the lowest. In requirements phase, the largest portion of bugs have their root cause. Incomplete requirements often are the reason for failing projects. Good requirements are very critical and good testing process is needed. Testing begins when requirements are first designed. The test case design approach should provide high functional coverage of the requirements. Minimum number of test cases should be created to reduce the amount of time needed to execute tests.

The article discusses about a 12-step RBT Methodology that improves the Software Development lifecycle of a project. The methodology involves validating requirements against the objectives, applying use cases against the project requirements. An initial ambiguity review of the project requirements and eliminate the ambiguous matter and allowing domain experts to review the requirements to verify the correctness and completeness.

Creating cause-effect graph solves any problems with aliases like using different terms for the same cause or effect. It defines the precedence rules among the requirements and begins the process of integration testing. The code programs must integrate with each other. If the requirements that define these programs cannot integrate, then the code programs cannot be expected to integrate. The cause-effect graph shows the integration of the causes and effects. A tool determines any logic errors in the cause-effect graph. Requirements authors review designed test cases. If there is a problem with a test case, the requirements combined with the test case can be corrected and the test cases are redesigned [3].

Test case are also analyzed by developers. Test cases can be used to approve that the design is correct enough to satisfy the requirements and used to validate that each code module delivers what is expected.

RBT methodology delivers maximum analysis with the minimum number of test cases. RBT provides quantitative test progress metrics within the 12 steps of the RBT methodology, assuring that testing is sufficiently provided and is no longer an obstacle.

Importance of Requirement based testing process

Skokovic Skokovic p et al. (2010) addresses the effects and importance of the Requirement Based Testing (RBT) Process, similar to the above article but focusing on different aspects that should be improved and are affected due to poor requirement specifications using a specific case study [2]. This article discusses about the Measurement in RBT process and multiple measures can be used to quantify the status of deliverables and activities. This helps managers and process experts oversee quality initiatives across the IT application portfolio. The data that could be measured include percent of requirements reviewed by domain experts, designers and developers, amount of requirements that contain uncertain terms.

The article also discusses about the importance of traceability in RBT process, since developing & maintaining traceability between requirements and test-cases is important which is very helpful for monitoring project progress and test coverage and also helpful in managing the effect of changes in requirements. If there is no traceability maintained from the beginning of the project, it becomes difficult to determine which test cases are affected for specific requirement changes. The drawbacks are all the defects identified in software

projects are introduced in the requirement phase. Some of the requirements defects are result of poorly written, unclear, ambiguous, and incorrect requirements. Other requirements defects are due to incompleteness of specification.

4.3.7 Traceability

Traceability is a required link between testing and requirements. Traceability is very helpful for verification of requirements and also for planning verification process. Efficient traceability also helps the users with aspects such as analysis of software change impact, analysis of software reuse, program comprehension and regression testing. A strong connection between testing and requirements engineering can benefit both fields. This allows a set of good patterns that can be utilized to get RE and testing closer to each other in real production development plans, which enhance the likelihood of the leading products meeting client necessities. Good project management and good requirements prioritizing are a good beginning in controlling potential users never-ending appetites.

Three fundamental reasons to practice requirements traceability are as below:

1. Develop scope management: Requirements can be omitted. By utilizing traceability, the individual handling requirements can see at each need and make sure that there is a lower stage of requirements that satisfy the scope. It's much more beneficial to meet with missed requirements throughout a requirements stage than to allow the project at a later date where it generally costs more and harm your authorization with your customer.
2. Develop test reporting and test price: Entire report testing is both impossible and pricy. Traceability can assist here as well. A high-priority requirement must have a related test case or term [15]. Finding out that a little priority requirement is followed to a prominent amount of test cases or terms can point that the result is being over-tested, leading in more time and price expended than needed.
3. Develop change impact assessment: Business motives occur in the most agile of designs with extremely experienced analysts. With good traceability in place, a modification to a requirement can be mapped to lower degree requirements, to encrypt elements, and to test artefacts.

When the causes for requirements traceability are forcible and the possible profits appealing, there are some drawbacks. Traceability can be difficult to maintain without good devices. Simple traceability can be moderately handled with programs.

For example:

- For most middle-sized projects, it is generally enough to trace at the degree of use case ways adequate to characteristics and motives, and down to test cases/terms.
- A project handling with very complicated parts or functions pertained to safety may well have to trace at a lower degree of coarseness and preserve more traceability relationships.

Orlena et al., described that traceability refers to the ability to follow the life of a requirement or testing, in forward and backward direction that is from its origins through its development and specification [24]. This paper describes about the nature of requirements traceability.

Many commercial tools and research product support RT such as General-purpose tools, Special-purpose tools, Workbenches, Environments. Techniques have been used at the RT problem without any investigation. In spite of growth in specialised tools, RT remains an extensively reported problem area by industry. RT is of 2 basic types: Pre-RS traceability refers to those conditions of a requirement's life prior to inclusion in RS. Post-RS traceability refers to those conditions of a requirement's life that result from inclusion in the RS. Forward and backward RT are surely needed. RT problems are mainly due to the difference between pre-RS and post-RS. The main differences include the data they deal with and the problems they can address. The issues that pre-RS traceability are to deal with are neither well understood nor fully supported. Post-RS traceability support is not suitable.

Most of the problems for poor RT is due to the lack of pre-RS traceability. Pre-RS traceability was also needed to improve the quality as already closed issues could be made possible to re-open and possible to re-work. Solutions to pre-RS traceability are increasing awareness of data, obtaining information, organizing and maintaining information, accessing information. In conclusion, to accomplish any improvement with the RT problem, there is a need to focus on pre-RS traceability [24].

Manually creating traceability between requirements and testing represents state-of-the-practice. Maintaining the traceability links using the available tool involves significant effort by the verification team, but it can be time consuming. Modern requirement management tools provided features for manual traceability, but still it is expensive to maintain. Automating the acceptance testing indicates continuous authorization of the software product and thus continual authentication of traceability. An automatic test is a programming since business writer or customer can specify these tests, not necessarily programmers. The challenge is to link business writer and developers in perfect way. Manual and automated traceability is given in the form of matrices. An example of traceability matrix is explained in figure 12. It is requirement traceability vs. test cases traceability matrix. For each requirement there must be one or more tests defined i.e., use case requirement is labeled as UC 1.2 can be traced to test cases 1.1.2 and 1.2.1 [17].

Reqs IDs	Reqs Tested	UC 1.1	UC 1.2	UC 1.3	UC 2.1	UC 2.2	UC 2.3.1	UC 2.3.2	UC 2.3.3	UC 2.4	UC 3.1	UC 3.2	TECH 1.1	TECH 1.2	TECH 1.3
Test Cases	231	3	2	3	1	1	2	1	1	1	2	3	1	1	1
1.1.1	1	x													
1.1.2	2		x	x											
1.1.3	2	x											x		
1.1.4	1			x											
1.1.5	2	x												x	
1.2.1	1		x												
1.2.2	1			x											
1.2.3	2				x		x								
1.2.4	2					x		x							
1.2.5	2								x	x					
etc...															

Figure 12 Requirement versus test cases traceability matrix [17].

Requirements traceability notices the capability to define and adopt the life of a requirement, in both forward and backward direction. Recording and tracking traceability relationships among requirements do this. Tracking requirements by the project assures that all requirements are studied in the various life-cycle phases. When we write test cases from a requirement document, we should stay sensible for the coverage of the requirement and changing them to testable items. If there are any areas that are not covered in the test case, then it will go untested. In order to check that all the requirements are tested or not, Traceability matrix is created. It maps requirements to test cases. For each requirement, the matrix gives the list of required test cases to test it.

Generating the Traceability Matrix results in various advantages in the software testing process:

- From the viewpoint of requirements, it gives the generated test cases a clear functional coverage metrics.
- Knowing which requirements are not addressed by the model-based testing process permits one to finish the test suite with some manually planned test cases or to develop the model.
- It gives helpful comments on the requirements: some test cases may not be connected with any requirements, perhaps showing the lack of expressed requirements.

Mechanically generating the Traceability Matrix from requirements to test cases involves handling the links between the requirements specification, the test cases and the model [15]. Handling a Traceability Matrix from requirements to test cases is fundamental issue in a software validation and quality insurance. It presents various benefits in the overall software life-cycle:

- Confirms that all requirements were precisely carried out and tested.
- Helps in risk management by assessing the test coverage of each requirement.
- Helps when correcting to discover the related requirements when a test fails.

Test traceability to requirements

Eero J et al., presented a set of practices, which can be applied to link Requirements with testing. These practices propose that tester's participation in requirements reviews will set up traceability policies by taking future requests from tester into account and links testers with requirement owners. These practices are about early tester participation, Tester participation in requirements reviews, and Test traceability to requirements, Linking testers with requirement owners, and Requirement suggestions by testers. Early tester participation consists of testers involving in planning stage. Testers are encouraged to participate in requirement validation because no software project can be successful unless testers are present from the beginning [1]. Testers always need to retrieve more information about requirements, which means direct communication between requirements member and testers.

Hendrik Post et al., presented results of benefits of linking testing and requirements via test traceability. The goal of this approach is to identify early defect detection and provide consistency. The case study performed by Hendrik Post et al., formalized 50 set of requirements and verified that software releases conform to them by using an automatic technique called software bounded model checking [27].

The software bounded model checking has been performed in concurrent to standard development process. With this new model an additional verification engineer was also added to the development teams on top of the standard requirements engineers, testers and test managers that exist in a conventional software development model. The task of this engineer is to formalize requirements into scenarios [27]. All the time, requirements must be updated and results must be communicated with the requirements engineers and developers. This article provided us a detailed insight on experience of applying verification in parallel with requirements. The technical scenarios will help for the communication between requirements managers, developers, and testers. The success of the method is seen from the number of new errors found by verification approach. Figure 13 illustrates the benefit of linking scenarios rather than test cases with requirements [27].

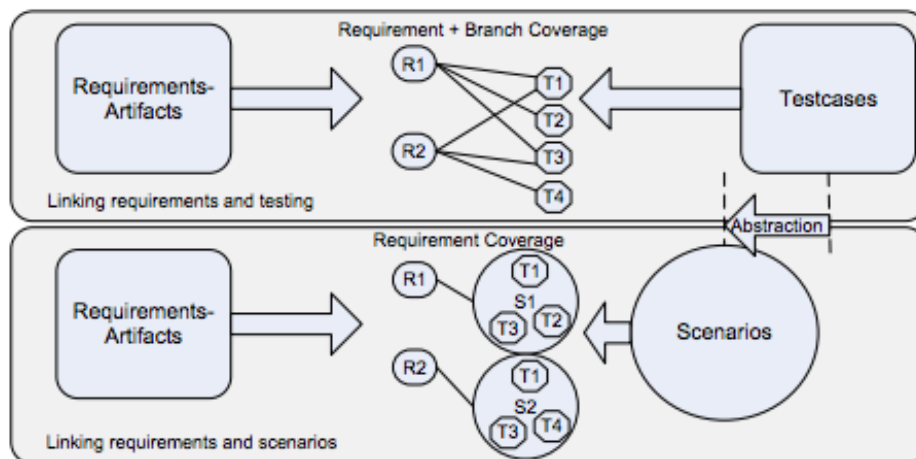


Figure 13 Linking requirements with a possibility high number of test cases

The reason to have synchronization between requirements and scenarios is that numbers of scenarios are lower compared to the number of test cases. With this case study, it is clear that testing and verification can deliver great results by having a synergy between the two. While testing works to detect the system level errors, verification offers distinct coverage on

function level. Through the model tested in this case study it is very clear that implementation errors can be detected earlier in the software development life cycle if the users start to use the technique of linking requirements and scenarios.

Traceability with the help of the tools

Kukkanen, et al., summarizes a systematic approach to link requirements and testing. Results through the case study summarizes that it is beneficial to perform linking at three different levels,

- 1) Linking of processes,
- 2) Linking of people and
- 3) Applying good practices.

The first level 1) above details that requirements and testing processes must be developed together, so that they form a solid basis for linking requirements and testing [18]. To achieve the second level 2) two critical roles of any software development life cycle i.e. requirements manager and testing manager have been identified. Linking of these roles achieved a smooth communication between relevant parties. The new requirements development, requirements management, testing development and testing management processes and data flows used in the pilot project of the case study are as presented in the figure 14.

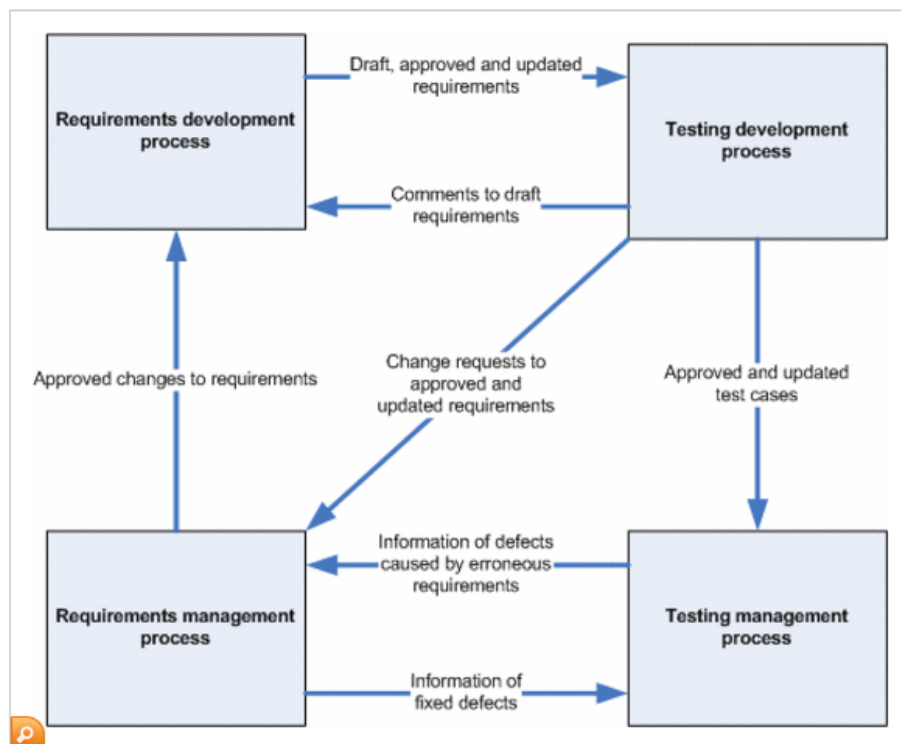


Figure 14 Relation between requirement process and testing [18].

To achieve the good practices application mentioned in 3) above, it is recommended to manage traceability with the help of tools, organized reviews and building a controlled change management processes. All these best practices were very essential to link requirements and testing according to the case study presented. Significant improvements achieved in the pilot project and the main factors associated with the improvements have

been detailed in the case study. The first improvement factor is moving the organization structure from line to matrix, which enabled effective and clear roles in the project teams. The second factor was drawing experiences from previous experiences from overdue R&D projects, which clearly stated the need for improvements, thereby reducing the change resistance within the teams. The third factor was the support and commitment from the leadership teams. The fourth factor was inviting the different business teams to the development project, which also helps further to reduce change resistance between teams working together. The final factor was highlighting the usability of the tools within requirements and testing processes, thereby making people comfortable in using the tools [2]. Disadvantages of traceability are traceability between requirements and tests were rarely maintained. This was caused for not updating traces when requirements change, due to schedules and budgets. Poor quality of requirements was a drawback to maintaining the traces. Failure to trace tests to requirements is one of the most effective ways to destroy a project.

4.3.8 Model-based testing

J. Gutiérrez mentions about the techniques of automatic test case generation from functional requirements within a Model Driven Engineering (MDE) environment, a paradigm focused on creating & exploiting domain models. This article also mentions that research groups are using MDE for requirements treatment, design, development and several aspects of Web development. The article focuses on the first phases of the lifecycle and it studies how functional testing can be improved by means of early testing [13]. Thus, this paper analyses an approach that uses the MDE. This article mentioned that there is two techniques emerging for generating test cases from use cases: Round-strip strategy and Extended uses cases. The Round-strip strategy apply classic algorithm of path finding in a state machine.

The functional requirement behavior is managed as a graph. Searching a path allows finding all different paths using behavior. Each path is a scenario designed with system and each scenario is test case. The Extended use case pattern applies category-partition method to use cases. This pattern identifies categories and partitions to generate combination of such partitions. A category is a point for which functional requirement defines an alternative behavior. A partition is a subset of domain of the condition evaluated in the category that decides piece of behavior is executed or not.

The combination of categories and partitions becomes a potential test case. The functional test case defined as meta-models. Meta-models are a relationship with other concepts. This article described about four meta-models. First meta-models mentioned that key concept of meta-model is the functional requirement element. The behavior of functional requirement is modeled using step and execution order. The step element models concrete chunk of behavior of functional requirement, as calculated information or requested information. The execution order defines the order in which steps are executed. The second meta-model is result of round trip technique. The third meta- model defines the result from category-partition method.

The last meta-models are artifacts, which combine the result of two previous techniques. This article allows tracing relation between four meta-models. Tracing allow us to know which test artifacts have generated for each functional requirement. This paper presents a

continuation of NDT (Navigational Development Techniques) based on metamodels and transformations with the goal of generating test cases from functional requirements.

Automated test case and test driver generation from a UML model is a developing approach for software functional validation. This new approach for validation testing makes it possible to ensure the functional coverage of test suite. Eddy Bernard presents an approach to naturally produce the traceability matrix from requirements to test cases as part of test generation process. This paper introduces the approach to define the UML model and using it to create the traceability matrix [15].

A known way to trace from requirements to test cases is to create a traceability matrix: for each requirement, the matrix gives the list of test cases to test it. Test case can be used to test many requirements. There are many advantages in the software testing process for automatically producing the traceability matrix such as knowing which requirements are not covered by model-based testing process allows one to improve the model. It gives valuable feedback on the requirements.

Maintaining a traceability matrix from requirements to test cases is a critical issue in software validation and quality insurance. It provides many benefits in the overall software life-cycle: Verifies that all requirements were correctly implemented and helps in risk management by classifying the test coverage of each requirement [15].

Code-centric approaches

Zeinab Alizadeh Barmi et al., presents systematic mapping study on alignment of requirements specification and testing. A lot of research has been done in the field of linking requirements with testing but there are lack of summaries on how to link the two [10]. The need of testing has become more apparent in recent years in order for delivering a high quality software product or service. Bringing requirements engineering and testing closer will benefit both the disciplines. Making a strong link between the two will improve outcome of the software development process, which helps to discover the errors earlier in the development life cycle. The focus has been mainly on functional requirements rather than non-functional requirements. Non-functional requirements are more important than functional requirements for customer satisfaction. But few authors mentioned that verification of non-functional requirements are always done in later stages after finishing the implementation.

The mapping summarizes 35 relevant papers found and discussed them using the major categories, such as traceability and model-based testing. The idea of Model-based technique is derivation of executable test code from test model by analogy to Model driven architecture [10]. This technique is more interested in industry because it provides automatic deriving of test cases from behavioural model called test model. Requirements traceability in Model based testing is an important issue which is the focus of many studies. In these the focus is mainly on functional requirements

The author concludes that work on aligning requirements and testing is focused on few areas: Code-centric approaches, Formal approaches, Traceability, Test cases, Model-centric approaches, problems and set of good practices in aligning requirement and testing with

major focus on Model-based testing and traceability issues. The article concludes that the software development industry is interested in attaining a strong link between requirements and testing, as there is a significant gap in this area. This shows great possibility for future work in establishing new processes and methods with supportive tools, which we aim to achieve through our thesis work [10].

4.3.9 REST taxonomy

REST is a requirements engineering (RE) and software testing (ST) benefits from a coordinated functioning. There is a gap between requirements engineering and software test alignment (REST). To fill this gap, Gorschek et al introduces a definition of requirements engineering and software test alignment. Taxonomies are means to structure, advance the understanding, and to communicate knowledge. When the understanding in a certain area advances, concepts and relationships develop that allow for an organized representation of these concepts. Being able to communicate that knowledge provides the opportunity to further advance research. This paper forms a foundation through a taxonomy, to categorize and identify alignment research and solutions that focus between RE and ST. RE and ST are aligned for coordinated functioning and improved product development. In order to assess alignment, REST-bench is developed which acts as an assessment framework and is based on REST taxonomy [41].

Sabaliauskaite [5] established many hurdles in aligning requirements and testing. Obstacles occur in the organizational structure, processes and cooperation between people. REST taxonomy focus on the alignment and how methods for alignment are classified. The improvement of taxonomy can be approached in two different ways, top-down and bottom-up. In the top-down approach, different design and levels are established that aim to fulfil the purpose of taxonomy. Bottom-down approach is directed by the sampling of subjects from the population of interest and extraction of patterns that are civilised into a classification scheme.

The information dyad is an abstraction that supports the reasoning on RE and ST alignment methods. This paper defined RE and ST alignment both as a state and as an activity. In the context of alignment-as-activity, we developed the REST taxonomy, analysing and characterizing dyad structure properties that allow one to reason upon the phenomenon of alignment. In the context of alignment-as-stare, we developed REST-bench, an alignment assessment framework mechanized by REST taxonomy. The REST-bench framework evaluates RET alignment by mapping the information flow between requirements and testing by using an artefact map.

Taxonomy summarizes 13 REST alignment methods that allowed to reason on the overall methods. The REST taxonomy provides a new view on the aligning requirements engineering and software test, based on analysis and method. It enables the characterization of alignment methods and the assessment of alignment in a development organization [41].

4.4 Conclusion

This chapter explains literature about the definition of alignment and presents the results of the literature review for research questions. We obtained some articles for research questions 1 and 2. Articles mainly focused on requirements and testing which have an interdependent

relation and both benefit each other. It mainly describes about alignment of requirements and testing and approach is presented for developing requirements and testing processes together. Results mainly focused on traceability, which is the method to align requirements and testing. In requirements based testing method, disadvantages are all the defects identified in software projects are introduced in the requirement phase.

Some of the requirements defects are due to poorly written, unclear and incorrect requirements. In traceability alignment method, traceability between requirements and tests were rarely maintained. This was cause for not updating traces when requirements change, due to budgets and schedules. Maintaining a traceability matrix from requirements to test cases is a critical issue in software validation and quality assurance. From the literature review, we found that there is no proper method that works well to align requirements and testing.

4.4.1 Benefits for Alignment

Based on the results of table 8, below are the list of benefits for aligning requirements and testing.

- Creating a strong link between requirements and test cases will improve the result of the overall software development process [1].
- It helps to identify possible errors earlier, which can improve the product quality [1].
- Linking requirements and testing helps users to reach a correct testing plan, which would improve project cost and schedule estimation. [10]
- Benefit of a strengthened link between requirements and testing will result in an improved product quality, cost-effective testing, higher quality test cases and early discovery of incomplete requirements [5].
- Involving testers in the planning stages of a project makes sure that testing activities are properly taken into account in planning, that is budgeted for both in schedule and resources [1].
- Tester's domain and system knowledge are developed because of additional exposure to the subject [1].
- Involving testers in the planning stages and having testing group improves the quality of the requirement [1].
- Testers have a different aspect than requirements analysts or developers, which helps surface deficiencies and exclusions in requirements, and invent requirements that will be difficult to verify [1].
- Leads to higher quality requirements and therefore improved testability [1].
- Open communication between testers and requirements teams during the development and testing processes results in the ability to progress forward in spite of lower quality requirements [1].
- Increased reliability of tests carried out thereby increasing confidence on the subsequent products being developed [1].

4.4.2 Alignment Methods

Results mainly focused on traceability, which is the one of the method to align requirements and testing.

Traceability

Advantages:

Organizing a traceability matrix from requirements to test cases is an important issue in software validation and quality insurance. It provides many benefits in the overall software life cycle. Helps specification change management by identifying all application elements affected by a requirements change. It helps in risk management by evaluating the test coverage of each requirement.

Disadvantages:

In traceability alignment method, traceability between requirements and tests were rarely maintained. This was cause for not updating traces when requirements change, due to budgets and schedules. Maintaining a traceability matrix from requirements to test cases is a critical issue in software validation and quality assurance.

Model-based testing

Advantages:

The advantages of Model based development are the foundation of executable test code from test models to model-driven architecture. It provides automatic developing of test cases from behavioral model of the system called the test model.

Disadvantages:

Informal requirements of the system are the base for establishing a test model, which is a behavioral model of the system. This test model automatically creates test cases. The drawback is that the created test cases from the model cannot be performed directly against an implementation under test because they are at the same level of abstraction as the model. There are many challenges in using MBT for aligning requirements and testing. Applying formal methods for aligning requirements and testing have some advantages and disadvantages.

Requirement-based testing

Advantages:

By proposing RBT methods of testing requirements, before the implementation phase, number of ambiguous requirements found during the implementation is considerably lower. RBT process could reduce the number of valid test cases and make developers more aware of testing activities.

Disadvantages:

Errors found in requirements are the main cause of project failures, defects and rework. Tests are generally conducted at the end of the development process. Application requirements changes regularly, but there changes are not properly handled. Frequently changed requirements resulted in rework of code and design. Test cases are modified to stay current with requirements.

REST taxonomy

Advantages:

REST is a taxonomy that describes the methods linking requirements engineering and software testing. The association between RE and ST are clear and the advantages in developing the coordination between them. The improvement of taxonomies is necessary to report assumptions that accumulate knowledge on software engineering phenomena.

Disadvantages:

There is lack of research and proposal of solution on supporting the alignment between RE and ST. There are many incomplete solutions that address particular gaps. The low number of Identified RE and ST alignment methods implies that the closer association between RE and ST researchers is still in its early development. There is possibility to concentrate in these areas and that the proposed taxonomy implies gaps in research. Although there are different alignment methods that establish a strong link between requirements and test cases, there is still a significant gap between these areas. From the literature review, we found that there is no proper method that works well to align requirements and testing.

5 Designing Software Requirements and Test case template

As part of our thesis work we have created a test case template that has attributes of both the requirements and test cases and mainly the template has been derived based on our extensive literature review performed on aligning requirements and test cases.

5.1 Aligning Requirements and Test cases

As described in section 2, a typical SDLC process starts first with requirements analysis proceeded with other activities of the life cycle like testing, design, implementation, etc. For a specific development project the test cases will be separately derived based on the functional requirements defined for the project and is always done after the requirement analysis. Figure 15 shows two different processes for Development and Testing, which are using a documentation template for each process.

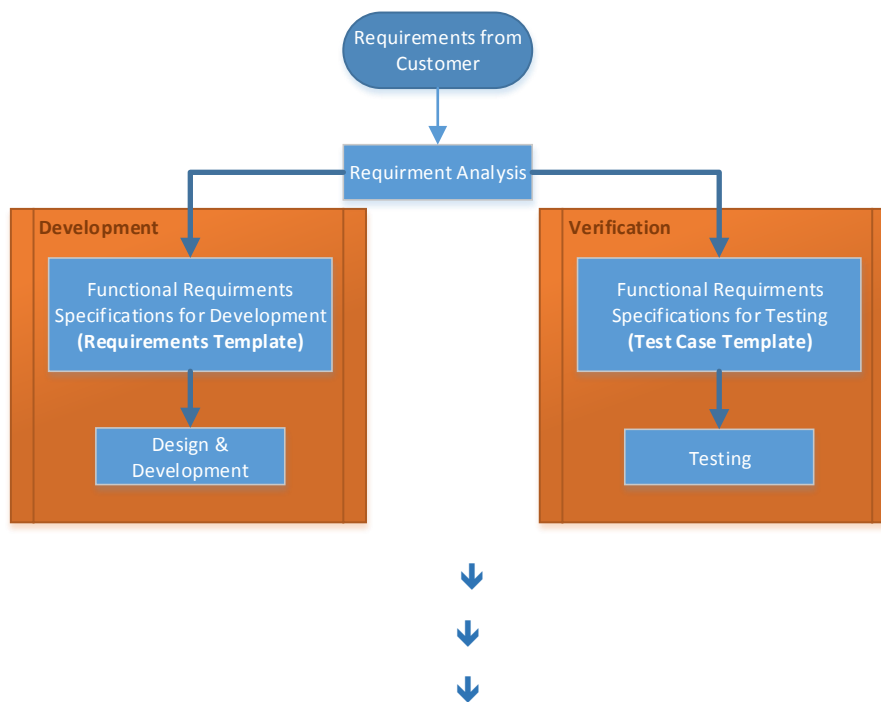


Figure 15 SDLC when both Requirements Template and Test Case Template are used.

Most of the currently available literature reviews are result of research performed on how to align requirements and test cases and the issues/challenges in aligning these two functions. However, as these two functions are often separately working with different documents i.e. requirements templates and test case templates there is always a gap in aligning these two functional groups which will be addressed with the test case template which is designed as part of this thesis work.

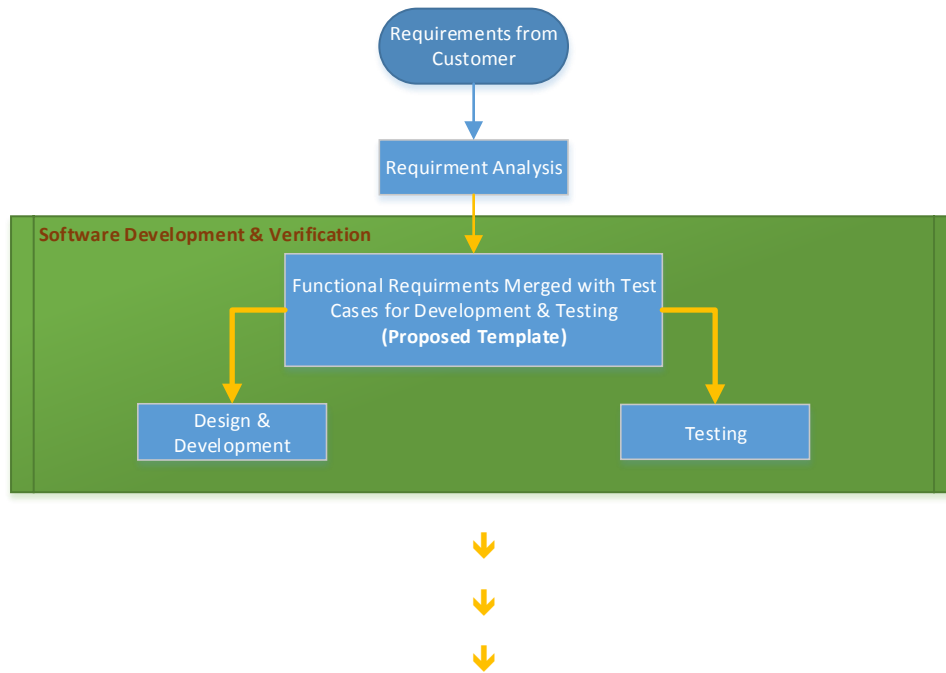


Figure 16 SDLC when the designed template is used.

During the design of template one of the key factors was to identify all the elements of requirements engineering that can be aligned with test cases in a SDLC. Aligning Requirements and test cases results not only in improving the entire software development process but also in improving the coordinating function between testers and developers and this would lead to optimized product development. Figure 16 shows a new proposed methodology by using a single template for aligning requirements and test cases with the designed template. The new template can improve the SDLC by merging functional requirement analysis for both development and verification into one sub process and then the same template is used for both development and verification of the software product.

5.1.1 Merging Requirements and Test cases

Once we have identified the aligning parameters between requirements and test cases from the above step, we have created a single template that is derived by merging all the important fields of a requirements template and a test case template as described below:

Software requirements template enlists all the information required for the development of a project. Organizations undertaking different kind of projects might need different kinds of templates to enlist project requirements information. These templates help organizations to structure and organize the project information. Similarly a test case template enlists all the information required for test teams to test certain feature in a project. Generally different test case templates are used for different types of testing. As part of our literature review, we have reviewed several requirement specification templates and test case templates to get a better understanding of what is more common and what is unique and what elements of a template can be combined between a requirements and test case templates to get a conclusion of a better test case template with only valuable information for both development teams and test teams.

Table 10 shows the analysis on different requirement and test case templates and the field, which can be prioritized according to the most common usage by different teams within a software organization.

Table 10 Literature review used for template

Field	Derived Template	Literature support
Requirement Name	*Available in all three requirement templates we analysed	Volere template by Robertson [21]/ Requirements template by Toro et al [22] /Use case Requirement Template [43].
Requirement ID	*Available in all three requirement templates we analysed	Volere template by Robertson/ Use case Requirement Template [43]/ Requirements template by Toro et al [22]
Requirement Description	*Available in two requirement templates we analysed	Volere template by Robertson [21]/ Requirements template by Toro et al [22]
Dependencies	*Available in one requirement templates we analysed	Use case Requirement Template [43]
Source & Priority	*Available in two templates we analysed	Volere template by Robertson [21]/ Requirements template by Toro et al [22]
Test Case ID	*Available in three test case templates we analysed	Test case specification template [16]/Analysis of test case template 2[32]/ Detailed test case template [3]
Tester ID	* Available in one test case template we analysed	Detailed test case template [3].
Test Scenario	*Available in one test case template we analysed	Test case specification template [16]
Test case Description	*Available in one test case template we analysed	Analysis of test case template 2[32]
Test Steps	*Available in one test case template we analysed	Analysis of test case template 2[32]
Expected Result	*Available in two test case templates we analysed	Analysis of test case template 2[32]/ Detailed test case template [3].
Actual Result	*Available in two test case templates we analysed	Analysis of test case template 2[32]/ Detailed test case template [3].
Pass/Fail	*Available in two templates we analysed	Detailed test case template [3]/Analysis of test case template 2[32]
Description of the Defect	*Available in one template we analysed	Detailed test case template [3]
Defect Severity	*Available in all templates we analysed	Test case specification template [16]/Analysis of test case template 2[32]
Role performing tests	*Available in one test case template we analysed	Test case specification template [16].

*Fields may not have exactly same name in the test case templates studied, however, the functions of the fields are observed to be same across the different templates.

5.2 Requirement templates

A requirement specification template should possess all the required information to develop a system that will satisfy the intended audience [10]. A typical requirements template should enlist the project context, user characteristics, and list of assumptions, design constraints and dependencies that will affect the requirements. Organizing and structuring the requirements information is also important for a smooth project development and a standard requirement template will help organizations to achieve it. Different organizations use different requirement specification templates depending on the purpose and scope of the system they are developing. Currently, there are many requirement specification templates available and in the next section we will describe the most commonly used templates [21].

5.2.1 Volere requirements specification template

The Volere requirements process is a result of many years research, practice and consulting in requirement engineering. Key factors related to different requirements types and key requirements related to testing have been considered to develop the Volere requirements process [21]. Different requirements types considered in Volere are Functional requirements, non- functional requirements, project constraints, project drivers and Project issues.

A requirement shell has been used as a guide to write each requirement, and the Volere shell has been shown in the figure 17. The fields in the shell that are not self-explanatory have been explained in the following section.

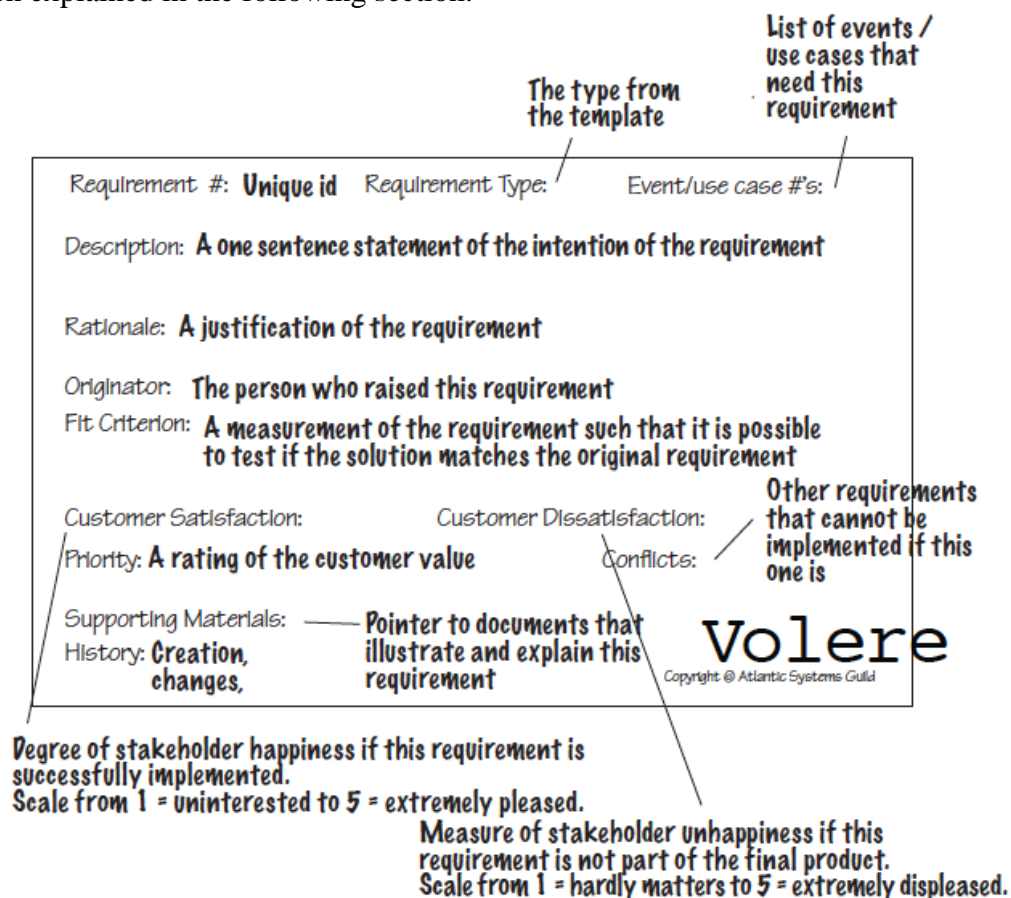


Table 11 Volere template by Robertson [21].

Requirement Numbering: Give each requirement a unique identifier to make it identifiable throughout the development process. The numbering pattern proposed in requirement shell is:

Requirement #: It is the next unique requirement number.

Requirement Type Requirement type is the section number for this type of requirement from the template. The insertion of section is not necessary because we have a unique requirement id. It indicates of what this requirement template relates to and hints why the requirement is important.

Event/use case # It is the identifier of the use case or business event related to a particular requirement. For one requirement, there can be several Events/use case #'s because the same requirement may relate to a number of events.

Customer Satisfaction Customer value is a degree of how much the end user cares about each requirement. This field is to keep track of the customer satisfaction on each requirement, on a scale from 1 to 5 where 1 means mild interest and 5 means very happy if this particular requirement is successfully completed.

Customer Dissatisfaction The end users can also rank each requirement for customer dissatisfaction on a scale from 1 to 5 where 1 means it barely matters and 5 means that they will be dissatisfied if this requirement is not satisfactorily completed.

Priority Priority shows the rating of customer on the importance of the requirement being developed.

Conflicts This field is to keep the users informed about other requirements that are not agreed with this one. These are the conflicts that need to be addressed using some kind of negotiation techniques.

History

This field is to keep track of all the changes that a requirement went through during the development process. Also, the reason for making those changes would be recorded within this field, thereby enabling the ability to explain why the requirement has been changed or deleted [21].

5.2.2 Use case requirements template

Use cases are primarily functional or behaviour requirements, which specify how requirements will behave and what system, will do [Cockburn01]. Use case template describes about requirements that are important for a project. There are various formats used for detailed use case template. Below table shows the typical format of a use case template.

Table 12 Use case Requirement Template [43].

Use Case Section	Comment
Use case Name	Start with a verb.
Scope	The system under design

Level	“User-goal” or “subfunction”
Primary Actor	Calls on the system to deliver its services.
Stakeholders and interests	Who cares about this use case, and what do they want.
Preconditions	What must be true on start, and worth telling the reader,
Success Guarantee	What must be true on successful completion, and worth telling the reader
Main Success Scenario	A typical, unconditional happy path scenario of success.
Extensions	Alternate scenario of success or failure.
Special Requirements	Related non-functional requirements.
Technology and Data Variations List	Varying I/O methods and data formats.
Frequency of occurrence	Influences investigation, testing, and timing of implementation.
Miscellaneous	Such as open issues.

5.2.3 Requirements template by Toro et al

Toro et al presents that requirements are elicited from customers using techniques such as interviews, brainstorming, document analysis, templates and Joint Application Development (JAD [22]. This process results to system requirements known as user or customer-oriented requirements.

Customer-oriented requirements are used to analyse missing, inconsistent requirements by building structured model. Customer-oriented requirements are converted into software requirements known as Developer-oriented requirements. New requirements are picked usually after the validation of customer and user. This process will repeat until whole requirements are validated and no more requirements are elicited. Requirements are expressed using requirement templates [22]. Requirements information is structured in a form so that requirements engineer will identify missing and inconsistent information. This article has explained about L & R patterns. L-pattern is faster and easier than writing whole paragraph i.e., R-patterns. Requirement templates and patterns are explained below.

The important thing in information storage requirements is information. The template explains about important requirement information that must be stored in the system, which is shown in Table 12. The template field contains Identifier and descriptive name, Version, Author, Purpose, Description, Specific data, Time interval, Importance, and Comments.

In order to identify, each requirement should have a unique identification number and name that start with RI. Version field contains date of requirement and current version number. Author and source fields contains about name and organization of author. Purpose and Description fields contain the goal and information of requirements. Specific data field contains a list of specific data that is relevant to concept. Time interval field indicated how long requirement information is relevant for system. Urgency, Importance fields indicate the importance of requirement. Comments field is about the other information of requirement, which cannot fit in other fields.

Table 13 Template for L-pattern for information storage of requirements

RI-<id>	<descriptive name>
Version	<current version number> (<current version date>)
Author	<current version author> (<author's organization>)
Source	<current version source> (<Source's organization>)
Purpose	<purpose of requirement>
Description	The system shall store information corresponding to <relevant concept>
Specific data	<specific data about the relevant concept>
Time interval	{past and present }
Importance	<importance of requirement>
Urgency	<urgency of requirement>
Comments	<additional comments about the requirement>

After reviewing many academic research papers this article had found that there are similar requirements, which are present in most developments. R-patterns contain information about product, customers & orders. R-pattern is classified depending on different criteria and stored for future use. Table 13 shows the R-pattern information storage requirement [22].

Table 14 R-pattern for information storage.

RI-x	Information about customers
.....
Description	The system shall store the information corresponding to customers. More precisely:
Specific data	Legal identification number of customer Name of customer Address of customer
.....

R-patterns contain information about product, customers & orders. R-pattern is classified depending on different criteria and stored for future use.

5.2.4 Requirements template field prioritization

Table 14 shows the result and analysis of three requirement templates, which are prioritized by us according to the most common usage.

Table 15 Requirements template field prioritization

Field	Derived Template	Literature support
Requirement Name	*Available in all three	Volere template by Robertson [21]/

	requirement templates	Requirements template by Toro et al [22] /Use case Requirement Template [43].
Requirement ID	*Available in all three requirement templates	Volere template by Robertson/ Use case Requirement Template [43]/ Requirements template by Toro et al [22]
Requirement Description	*Available in two requirement templates	Volere template by Robertson [21]/ Requirements template by Toro et al [22]
Dependencies	*Available in one requirement templates	Use case Requirement Template [43]
Source & Priority	*Available in two templates	Volere template by Robertson [21]/ Requirements template by Toro et al [22]

*Fields may not have exactly same name in the test case templates studied, however, the functions of the fields are observed to be same across the template.

5.3 Test Case templates

Different organizations use different test case templates depending on the purpose and scope of the system. Currently, there are many test case templates available and in the next section we will describe the few templates.

5.3.1 Detailed and simple test case template

In Table 15 & Table 16 there are the two sample formats for writing test cases i.e., detailed test case template and simple test case template [3].

Table 16 Detailed test case template [3].

Test case Id	Serial no assigned to test case	
Purpose	Brief Idea about case	
Test Created By	Name of test creator	
Test Environment	Software or hardware in which the test case is executed	
Prerequisites	Conditions that should be fulfilled before the test is performed	
Test Procedure	Steps to be performed in test	
Test Data	Inputs, Variables and data	
Expected Result	What the program should do	
Actual Result	What is actually done	
Verdict:	Pass/Fail	Status of the test
Comments	Note on the procedure	

Table 17 Simple test case template [3].

Test Case ID:	Serial no. of step
Step or activity	Detailed operation or procedure
Criteria for Success	Expected result
Status	Whether the code passed the test or not

5.3.2 Test case specification template

Test cases depends on the requirements documents and need to be run against the product in order to describe whether or not the generated results are the expected ones. Test cases are

referred to the test case specification according to [IEEE829] [16]. Test case specification specifies the purpose of a particular test, recognizes required input and expected results. It also provides step-by-step procedures for executing test and determines whether the result is pass/fail. Table 17 is the example of IEEE829 Test case specification Template [16] and the description of each header of the template follows.

Table 18 Test case specification template [16].

Test case Specification Identifier	Test case Identifier should specify test case uniquely
Test Items	Describe features and conditions tested. Test item contain input data, information that is suitable for the test, execution environment.
Input Specifications	Input specification must specify the inputs that are required to execute the test case. Eg: Data Names, Ordering, Values (with tolerances or generation procedures), States, Timing.
Output Specifications	Output specification must specify expected output from the test case execution to decide pass or failure. Eg: Data Names, Ordering, Values, States, Timing
Environmental Needs	Environmental needs cover how the testing environment is set up, what configuration is required for the system. Eg: Hardware, Software, Other.
Special procedural requirements	Special constraints on the test procedures that execute this test case.
Intercase dependencies	Inter case dependencies must list the identifier of test cases, which should execute before this test case.

5.3.3 Analysis of test case template 2

To prepare these test cases each organization uses their own standard template. An ideal template is provided in Table 18 and Table 19 to prepare test cases.

Table 19 Common columns in test cases that are present in all test case formats

Test Case ID	Test Case Description	Test Dependency/Setup	Input Data Requirements/Steps	Expected Results	Pass/Fail
--------------	-----------------------	-----------------------	-------------------------------	------------------	-----------

Table 20 A very details Low Level Test Case Format [32]

Test Case ID	Req No	Version Number	Type of Test Case	Test Case Name	Action	Expected results	Actual	Status	Bug ID	Remarks
--------------	--------	----------------	-------------------	----------------	--------	------------------	--------	--------	--------	---------

On the top left corner of the template, we will fill the details like Project ID, Project Name, Author of test cases, Version Number, Date of creation and Date of release. And the fields

Test Case ID, Requirement Number, Version Number, Type of Test Case, Test Case Name, Action, and Expected Result for each test case holds the actual test function specific information [30].

Test Case ID:

If a test case belongs to application not related to a particular module then the test case is identified as TC001, if it is expected more than one expected result for the same test case then we will name it as TC001.1. If a test case is related to module, then we will name it as M01TC001, and if module is having a sub module then we name that as M01SM01TC001.

Requirement Number: It gives the reference of Requirement number for a specific test case. For each test case user define to which requirement it belongs. If a requirement changes then user can estimate how many test cases will effect due to the change [30].

Version Number: The version number is defined in which the appropriate test case is introduced.

Type of Test Case: It consists of different kinds of test cases like functionality, security, regression, User acceptance, system, Load performance, which are involved in the test plan. One test plan is selected while designing test cases.

Test Case Name: The object name is defined for which it belongs to. For example, OK button, login form.

Action (Input): Depending on the steps written here, the operations are performed on the actual application.

Expected Result: It defines what the application or user expects from that particular action. The test case document is prepared based on system requirement specification and use cases.

Actual: The actual test application against each test case. If it matches to the expected result then marks “as expected”.

Status: It implies Pass or Fail status of that particular test case. If the actual result and expected result mismatch then the status is Fail otherwise it is Pass.

Bug ID: This gives the reference of Bug Number in Bug report.

Remarks: This part is optional. This is used for extra information [30].

5.3.4 Test case template field prioritization

Table 20 shows the result and analysis of three test case templates, which are prioritized by us according to the most common usage.

Table 21 Test case template field prioritization

Field	Derived Template	Literature support
Test Case ID	*Available in one test case template	Test case specification template [16]/Analysis of test case template 2[32]/Detailed test case template [3]
Tester ID	* Available in one test case template	Detailed test case template [3].
Test Scenario	*Available in one test case template	Test case specification template [16]
Test case Description	*Available in one test case template	Analysis of test case template 2[32]
Test Steps	*Available in one test case template	Analysis of test case template 2[32]
Expected Result	*Available in two test case templates	Analysis of test case template 2[32]/Detailed test case template [3].
Actual Result	*Available in two test case templates	Analysis of test case template 2[32]/Detailed test case template [3].
Pass/Fail	*Available in two test case templates	Detailed test case template [3]/Analysis of test case template 2[32]
Description of the Defect	*Available in one template	Detailed test case template [3]
Defect Severity	*Available in all templates	Test case specification template [16]/Analysis of test case template 2[32]
Role performing tests	*Available in one test case template	Test case specification template [16].

*Fields may not have exactly same name in the test case templates studied, however, the functions of the fields are observed to be same across the template.

5.4 Test Case / Requirements Template Creation

The purpose of this section is to describe a test case template by combining the results from our analysis of both requirements templates and test case templates carried out in the previous sections. This section of the document represents a template, which has aligned a single requirement or set of requirements to one test case. A test case or a group of test cases represent a basic end-to-end system function i.e. they explain the details of requirement(s).

Creation of Test Case Statement

Currently, a test case can be written in many ways. This document aims to introduce one way of representing a functional requirement, which can be aligned to a test case. Utilizing the conclusions from our review of different templates and considering the most common elements of a typical requirement template and a test case template, our template has been created. We have also considered few other non-functional items as detailed below in order to derive the test case template:

- Easy to use and replicate
- Clear roles and responsibilities highlighted in the document
- Flexible to change if needed

Table 22 Instructions to template.

Test Scenario Definition	
<ul style="list-style-type: none"> A set of test cases to ensure that the requirement process flow is tested from end to end. They may be independent tests or series of tests that follow each other, each dependent on the output of the previous one. The terms “test scenario” and test case are often used synonymously in our template 	
Sections of the Template	
Requirement Name	Requirement Name which needs to be aligned with the Test Case
Requirement ID	Requirement ID of a particular Requirement for traceability
Source	Origin of requirement such as department, person.
Title	Explains about what requirement is about
Description	Describes about requirements in detail
Dependencies	Describes the relation between present requirement and other requirements.
Author	Person who wrote the requirement in tool
Status	Indicates where requirements stand currently.
Priority	Indicates how important the requirement is.
Test Case ID	Test Case ID of a particular Test Case for traceability
Tester ID	Identity of the Test Engineer who is responsible to execute the test case
Test Scenario	Scenario of performing the test case
Test Case Description	Describe the test case in detail
Role Performing Test Case	State the role to perform the test
Test Steps	The steps followed in order to execute the test case
Expected Results	The result that the users expects
Actual Result	The actual result of the test
Pass/Fail	Test result
Description of the Defect	If the test case fails then details of the defect are provided
Test step where defect occurs	State at which step the defect occurred
Defect Severity	Is the defect P1, P2, P3 or P4, P1 being less severe and P4 being the most severe defect.
Color Key	
	Provided by Requirements team prior to User Acceptance Testing

	Provided by the Testing Team prior to the User Acceptance Testing
	Provided by the Testing Team during the User Acceptance Testing

The main theme of our thesis is to create a template that can consist of all the essential elements of a requirement template and a test case template in one single document. Core elements of both the requirement and test case templates were already identified in our literature review (chapter 5.2 and chapter 5.3) of our thesis work and we have aligned the elements that are in common between requirement and test case templates. Merging all the elements identified into one single template then creates the test case template. The actual template information has been provided in Table 21. Once all such information is extracted, the user can manipulate the elements to generate test scenarios. Table 22 shows the list of instructions and descriptions for the different elements chosen in the created template.

5.5 Created Test Case Template

Based on Table 21 shows the different sections in the derived template. Table 22 shows the derived template, which has different sections merged from a requirement template and a test case template. We have used Color key to differentiate the sections, which are provided by the requirements team and the Test teams.

Table 23 Derived Template.

Requirement ID	Requirement Name	Source		
Title	Description	Dependencies		
Author	Status	Priority		
Test Case ID				
Tester ID				
Test Scenario	Test Case Description	Role performing Test Case		
Test Steps	Expected Result	Actual Result	Pass/Fail	Description of the Defect
Test Step where defect occurs	Defect Severity			

6 Survey evaluation

This chapter presents the results of the survey and how the collected data was evaluated to improve the thesis work and answer the research questions. The main goal of the survey was to collect feedback from professional software test organization about the template we have created. The plan is to send the template to target users together with our survey questions. The survey is used to identify advantages and disadvantages of the template created, and also to gather more information on how we can further improve the created template. The survey answers will provide us the information on requirements management and testing.

Survey questionnaire is divided into three sections and are explained below. The section1 and section 2 is used to collect information about alignment processes and challenges (for answering research question 2). The section 3 is used to collect suggestions regarding our template created (for improving our template accordingly). Each questionnaire contains explanation and figures. The result in the figures sum up more than 100 because respondent can choose multiple options for each question.

Section1

This section gathers the information about the aim of the survey, which supports the second research question

Q1. What is your present requirement alignment process?

In general, there are different types of requirement alignment processes. The questionnaire invited employees to prioritize their decisions. Maximum of the respondents made their decision based on pre-project requirement decision under organizational level. Next, the decision takes place with in-project requirement decision under product level and the last chose was pre-project requirement decision under product level.

The result shows that the decisions at organizational level will mostly occur during the pre-project requirements discussions

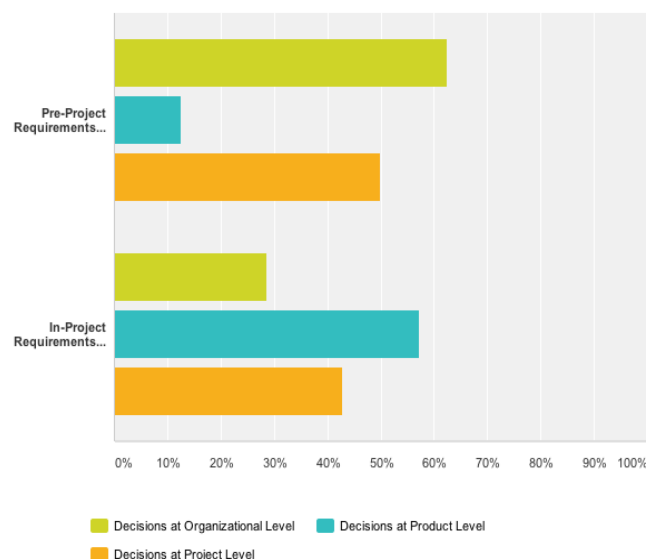


Figure 17 Alignment decision.

Q2: How would you improve the alignment between requirements and testing at your organization?

The most common initial step when merging requirement with testing templates is to know about alignment process and their improvements at organization. Most participants have given valuable input. It means that participants have a clear idea about the topic.

The result shows that better alignment between requirements and testing can be achieved by improving various aspects like

1. Easier access to report issues.
2. Good traceability matrix.
3. Less time on mapping test cases and requirements.
4. Clear role and responsibilities.
5. Make the development and testing easier
6. Avoids redundancy while testing
7. Increase in availability of requirement document in common collaboration area.

Table 24 Improvements for alignment.

By making sure the requirements are not only mapped to appropriate valid test cases; but also, those requirements are elaborated properly in the respective test case thereby it can be covered accordingly.

12/4/2013 1:38 PM [View respondent's answers](#)

Increase transparency of what requirements are sources for the test cases to have easier access to them when filing issues

11/25/2013 5:35 PM [View respondent's answers](#)

Simplifying the requirements and analyzing them more clearly so as to bring the complexity of the requirements and direct them into a technical development detail to make the development and testing easier and to avoid redundancy while testing and avoiding issue leakages.

11/25/2013 5:11 PM [View respondent's answers](#)

Spend more time on mapping TCs towards requirements

1/31/2014 3:55 PM [View respondent's answers](#)

Clear Roles and Responsibilities with easy and efficient tools/methodologies in place.

1/24/2014 5:04 PM [View respondent's answers](#)

The approach we've used so far to improve the alignment is use cases and having functional requirements identified by these, then specifying additional requirements and reviewing the use cases based on these either mapping or creating new ones. The use cases end up as the templates for test cases.

1/10/2014 3:18 PM [View respondent's answers](#)

Increase availability of requirements documentation in common collaboration areas (internal web site where everything is published). Also to have continuous meetings with the people working on requirements that are applicable for my particular area.

12/20/2013 11:21 AM [View respondent's answers](#)

By having a good tractability matrix

12/14/2013 10:43 AM [View respondent's answers](#)

Q3: How would you rate the idea of merging requirements in the test case template itself?

The question is related to the idea of merging requirements with test case. Maximum number of participants found this idea excellent. Few responded as very good. The average rating shows that 4.56/5.

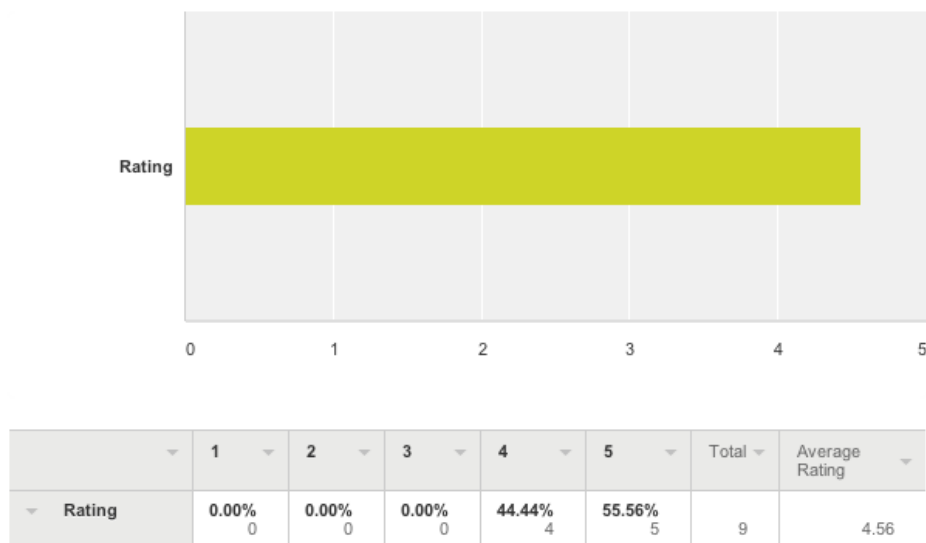


Figure 18 Rating the idea of merging requirement with test cases.

Q4: How would you rate the template we have created?

Maximum (6) of the respondents say that the template is very good, One of the respondents said that the template is excellent and one said that it's a good template.

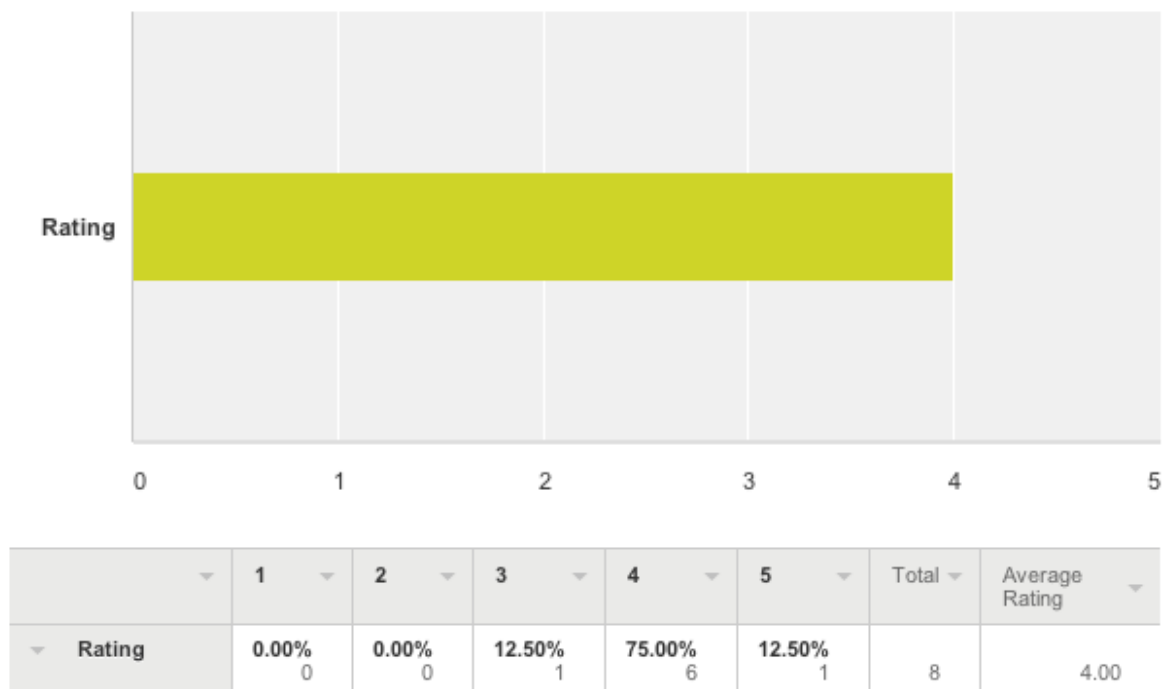


Figure 19 Rating our template created.

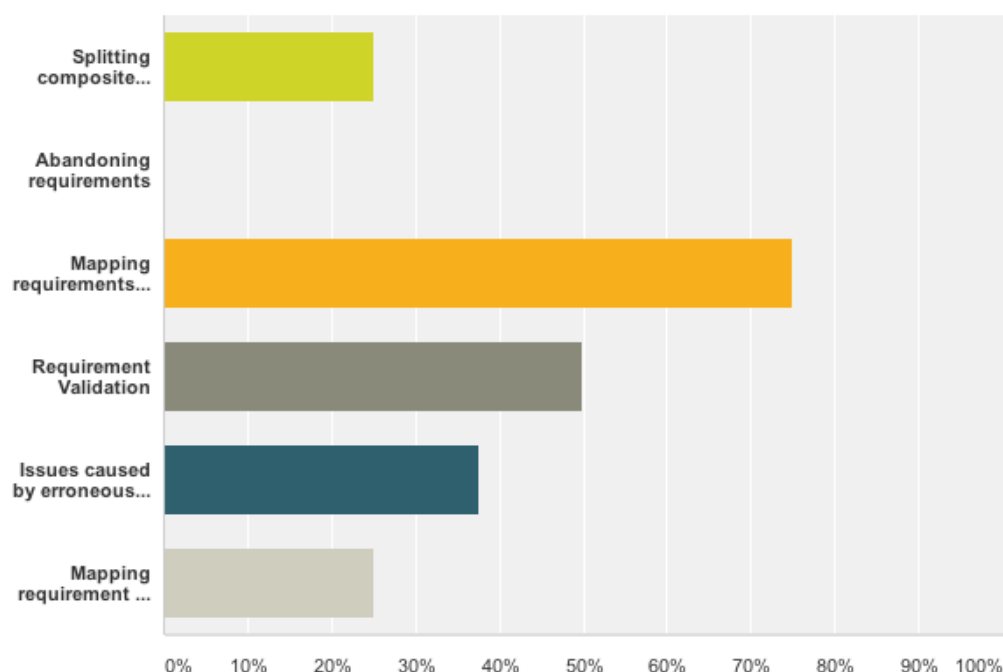
Section 2

Section 2 is created in order to collect the information regarding the issues faced in the field of requirement alignment and test cases.

Q5: What are the challenges you face in managing requirement alignment?

The result shows that challenges faced in managing requirement alignment are due to lack of clear mapping between requirements and test cases.

Most of the respondent says that challenges are faced due to mapping requirement with test cases. Next challenges faced are due to requirement validation. Few challenges are due to erroneous test case issues, mapping requirement to erroneous test cases and also splitting composite requirements into simpler once.



Answer Choices	Responses	
Splitting composite requirements into simpler ones	25.00%	2
Abandoning requirements	0.00%	0
Mapping requirements to test cases	75.00%	6
Requirement Validation	50.00%	4
Issues caused by erroneous test cases	37.50%	3
Mapping requirement to erroneous test cases	25.00%	2
Total Respondents: 8		

Figure 20 Challenges faced during requirement alignment

Q6: When investigating requirements decisions, which of the following components affects decisions as per your view?

The question is related to the components that effects decision while investigating requirements. Maximum of the result shows that most of the requirements decisions will be more affected with integration between various stakeholders. Few respondents say that effect is due to business rules. Very few effects are due to goals of organization, information regarding requirements and process of organization.

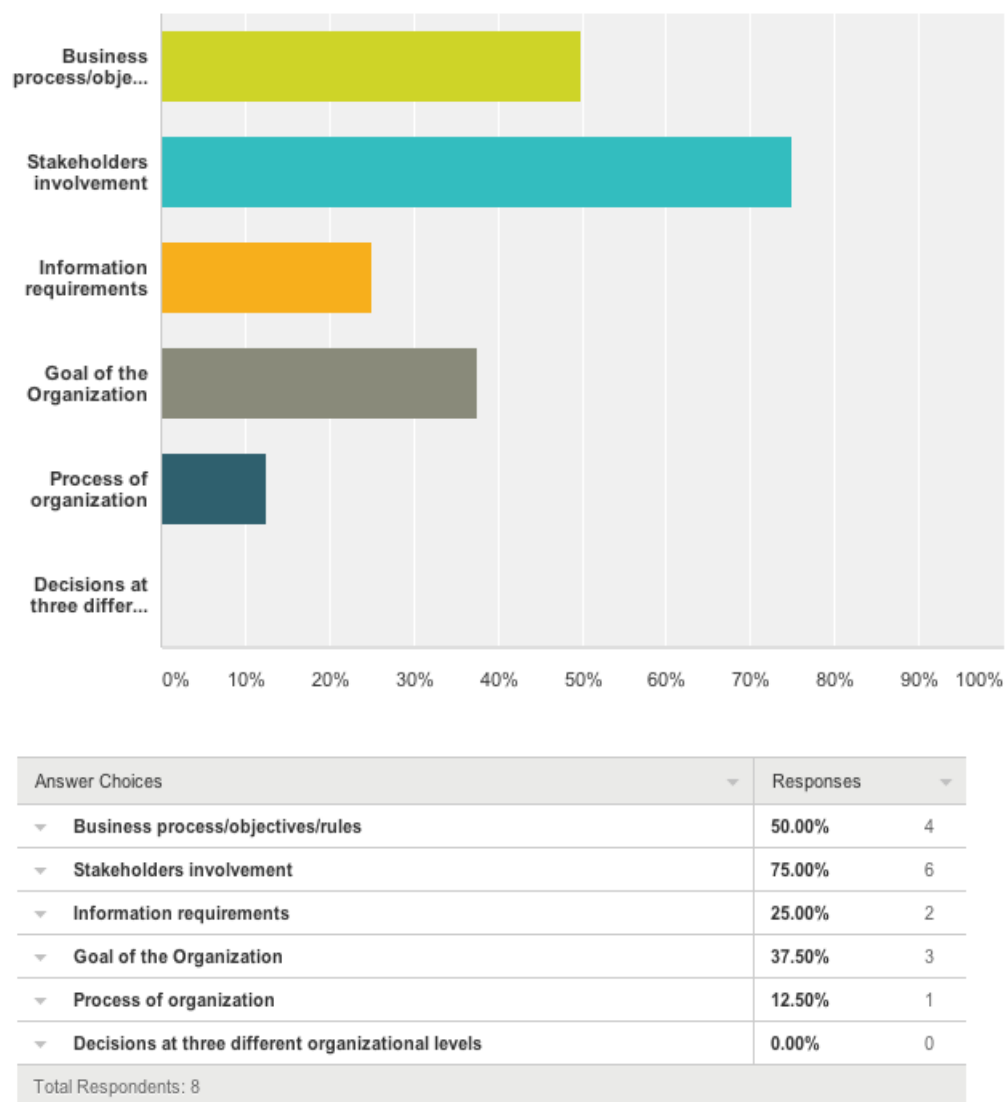


Figure 21 Investigating about requirement decision

Q7: Where do you face more challenges in requirement analysis as per your view?

The question is related to the challenges faced while analysing requirements. Maximum of respondent stated that challenges are while collecting the requirements. Few chose documentation of requirements and due to software product requirements. Very few respondent stated that due to business process and due to business rules.

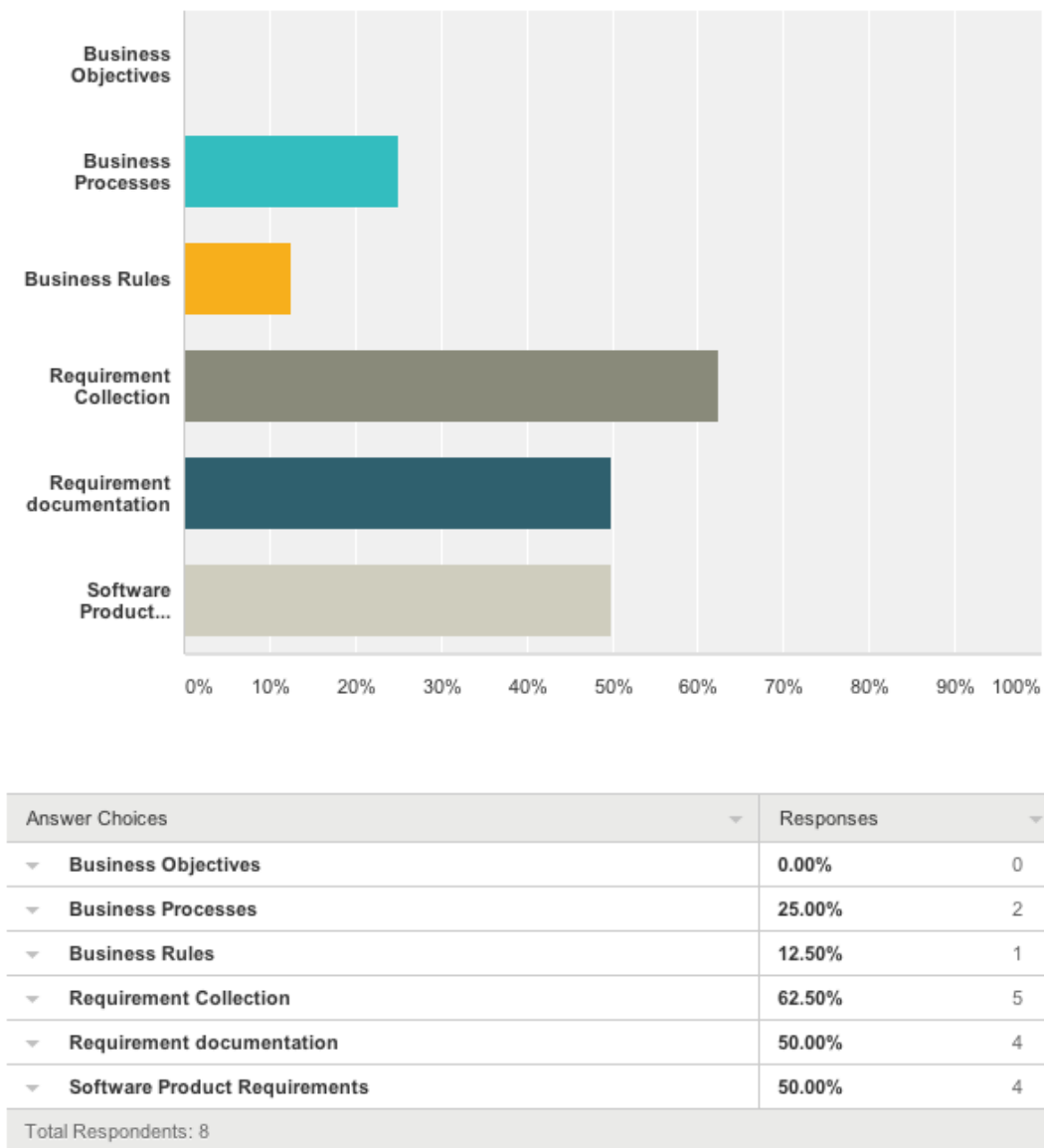


Figure 22 Challenges in requirement analysis

Q8: Would you foresee any issues with having requirement ID and Test case ID being in the same template?

It is extremely important to know if there will be any issues caused due to requirement ID and test case being in the same template. Two of respondents thought that it's an excellent idea to use both ID's. 4 thought that it's a very good idea and 2 respondents says that it's a good idea.

	1	2	3	4	5	Total	Average Rating
Rating	0.00% 0	0.00% 0	25.00% 2	50.00% 4	25.00% 2	8	4.00

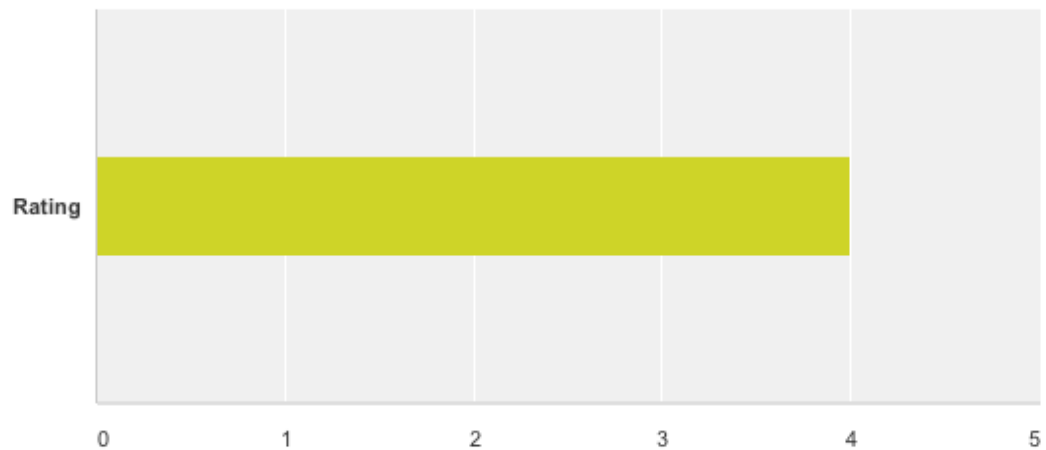


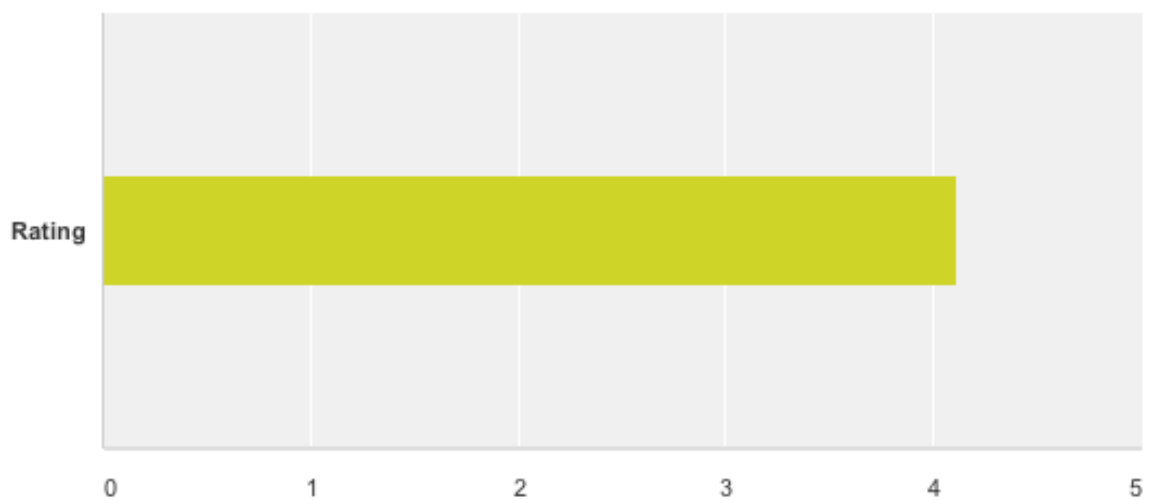
Figure 23 Rating the idea of having both ID's in single document.

Section 3

This section is designed for collecting suggestions from other people for the test case template.

Q9: Does the instructions provided on the template is clearly understandable?

Maximum number of respondents thought that the template is clearly understandable. The result shows that respondents are sure to use this kind of template, which is clear and understandable.

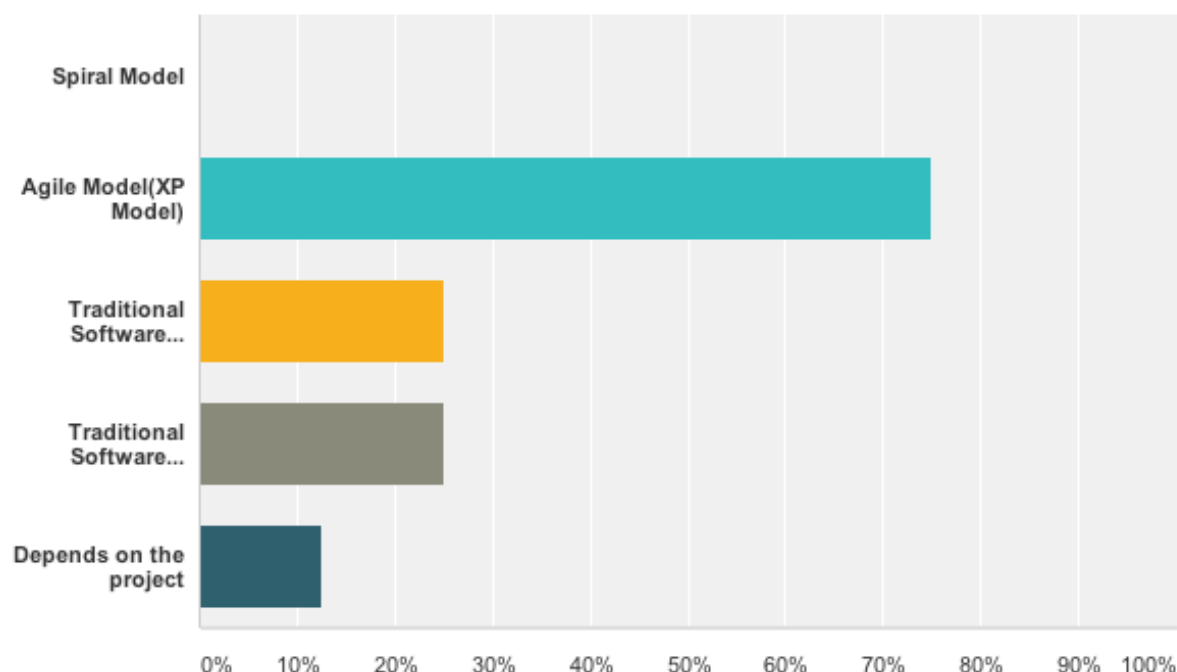


	1	2	3	4	5	Total	Average Rating
Rating	11.11% 1	0.00% 0	0.00% 0	44.44% 4	44.44% 4	9	4.11

Figure 24 Rating our template instructions.

Q10: Which model is used in your software development organization?

The common approach while creating a test case template is to gather general information about software development model used in an organization. Most of the respondents use Agile Model (XP model). Few chose v-model and waterfall model and very few thinks that it depends on the project.



Answer Choices	Responses	
▼ Spiral Model	0.00%	0
▼ Agile Model(XP Model)	75.00%	6
▼ Traditional Software Model(V-Model)	25.00%	2
▼ Traditional Software Model(Waterfall Model)	25.00%	2
▼ Depends on the project	12.50%	1
Total Respondents: 8		

Figure 25 Software models used in organisations.

Q11: Does your company will have any substantial improvements towards using test case template as we created?

In order to emphasize the use of test case template it is essential to find out whether substantial improvement within the company will be noticed towards using the template we created. Five respondents thought that template created is very good. Three respondents said that it's a good template. It means substantial improvements can happen due to the test case template we have created.

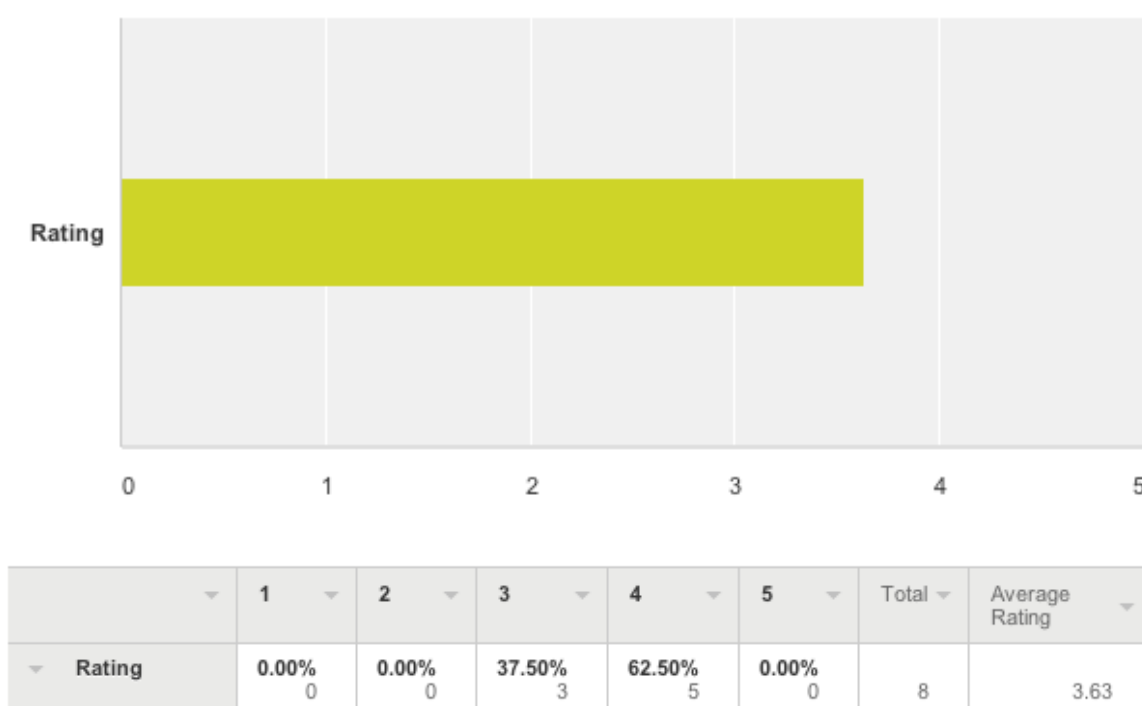


Figure 26 Any improvements by using our template.

There are three relevant aspects that were taken into account in order to evaluate the created test case template

- Quality of the template
- Applicability of the Template
- Intention in Software organization.

The overall results show that respondents are interested in using the test case template created.

Survey question 4 shows the summarized graph that the survey rating is obtained for the test case template created through this research work. Average rating is displayed as 4/5. Considering 1 being poor and 5 being excellent. Allowing the authors to conclude that the test case template is of Good standards.

6.1 Answers to key research question of the questionnaire

Below are the brief explanations for the evaluation of survey questions, which were linked to the research questions. The survey questions were related to alignment processes at organization, which is our key research question.

- **What is your present requirement alignment process?**

Participants belong to different teams and organizations. Most of them are software testers from different software organizations. The result shows that decisions at organization level will mostly occur during the pre-project requirement discussions.

- **How would you improve the alignment between requirements and testing at your organization?**

Results shows that better alignment can be achieved by merging requirements and test cases. Below is the list of benefits when aligning requirements and testing.

1. Easier access to report issues.
2. Good traceability matrix.
3. Less time on mapping test cases and requirements.
4. Clear roles and responsibilities

- **How would you rate the idea of linking requirements in the test case template itself?**

Most of the participants select that it's a very good idea to link requirement with test cases. The average rating given by participants is 4.56 out of 5.

- **What are the challenges you face in managing requirement alignment?**

Most of the challenges the organizations face in managing requirement alignment are due to lack of clear mapping between requirements and test cases, followed by requirements validation issues.

6.2 Interpretation of the results

The survey results show interesting aspects. There are different requirement templates and test case templates that have been largely accepted by different Software organizations. However, people prefer those that are aligned/merged with each other. The created test case template in this thesis work is better when requirements are elaborated properly in the respective test case. This template can help organizations to have good traceability between requirements and test case templates, increases transparency while reporting issues, less time on mapping test cases towards requirements, and availability of documents in common. An interesting issue was analysed in evaluation results of the test case template, that is majority of the participants are interested in aligning requirements with test cases but few of them also thinks ambiguity might arise in case of multiple requirements assigned to a test case.

6.3 Improving the Test Case Template

Considering the feedback from the Survey, the requirement and test case templates have been improved accordingly, as shown in Table 24. The template is divided into four sections for easy understanding and future improvements. The first section describes the required information for multiple requirements, which can be merged into single test case. Merging multiple requirements into a single test case will improve the test efficiency and is also helpful for requirement traceability. The second section describes information about the requirement dependencies and test environment pre conditions. These dependencies and pre conditions will help the test teams to maintain a practice of creating an appropriate environment to test software for accurate expected results. Third section describes the Test case itself and the test steps for verifying requirements aligned to this test case, where in each test case is aligned to one or many requirements, allowing the test teams to know which requirement is failing if the test step fails. The fourth section describes the defect prioritization and issue description. It provides easy handling of issues and allows the development teams to find where exactly the issue occurs and what requirements are being

affected and the priority of the requirement for the customer. This information clearly allows the development team to prioritize and fix the issue before the requirements can be retested and released. Bold letter in Table 25 shows that those are improved (new/changed) fields when compare to Table 23.

Table 25 Improved Test Case Template

Multiple requirements merged into single test case	Requirement ID (R1)	Requirement Name	Description	Source	Priority
	Requirement ID (R2)	Requirement Name	Description	Source	Priority
	Requirement ID (Rn..)	Requirement Name	Description	Source	Priority
Dependencies and pre conditions	Dependencies				
	Test Environment Pre Conditions				
Test case & test steps for verifying requirements aligned to this test case	Test Case ID	Test Case Name	Test Case Author		
	Tester ID				
	Test Scenerio	Test Case Description	Role Performing Test Case		
	Test Steps	Aligned Requirements	Expected Result	Actual Result	Pass / Fail
	Step1	R1, R2			
	Step2	R2,...Rn			
	Step3	R1,R2,...Rn			
	Step4	R2,...Rn			
	...Step(n)	Rn			
Defect prioritization and issue description	Test Step where defect occurs	Defect Severity	Description of the Defect		
	Step1	P1			
	Step2	P1			
	Step3	P2			
	Step4	P3			
Step(n)	P2			

7 Discussions and Conclusions

7.1 Thesis Summary

Analysis from the literature review convey that most of the delays occurred during a software development project are because of lack of traceability between requirements and test cases. In this thesis we have described a literature review performed with the goal of aligning functional requirements with test cases and the results of the review were that there is a possibility to align both requirements and test cases in a single template. As a result, a new template is created, which will set test cases as functional requirements for development and testing of a software product. The template has been created by selecting items on requirements and test cases templates found in the literature. After that we have validated the template by performing a survey and distributing the template to Software Organizations. The result of this survey is an improved template, shown in Table 22, and also a list of alignment methods. In the following sections we will summarize and discuss how the research questions have been answered.

Our thesis work results in a single test case template that is derived by aligning functional requirements and test cases. The single template we created removes the need of creating a separate requirements document. In order to create the template that contains both requirements and test cases we have chosen the two important research questions as explained below. Implementing the survey feedback has eventually resulted in the creation of test case template with aligned requirements. What we have covered in this thesis work are only the benefits achieved by using the test case template. It may have some disadvantages as well, for example, the health care industry has more regulated laws, which may require having a separate requirements template and test case template, in which case our template may not serve as a standard document. However, the test case template with aligned requirements could easily serve as a starting point to further perform improved literature review related to any other complicated software development processes.

7.2 Answering Research Questions

RQ1. What are the benefits of aligning test cases with functional requirements?

Based on the literature review described in Chapter 4, and having a measurable process, clear ownership, and templates to support the process the benefits of aligning requirements to test cases have been detailed below.

- Visibility of the status on specific stages of the requirements – Traceability
- Minimizing issue leakage since all requirements and test cases are aligned during the software product life cycle.
- Early signals assessing if the software product is on track or not
- Allow follow-up and proactive action if a task is lagging behind
- Timely delivery of an accurate guestimate of a functional requirement.
- Manage expectations from stakeholders

- Documentation efforts will be minimized
- Involving testers earlier in the development life cycle and assuring that the test cases are aligned with the requirements of the business at an early stage will lead to fewer development errors

From the survey that was conducted supporting this thesis work we have analysed the benefits which were mentioned in Chapter 6 (RQ2), considering the analysed benefits from the research and the analysis of comments from the survey respondents the below benefits are concluded

- Efficient requirements analysis, which is the foundation of any Software development and verification.
- Aligning requirements and test cases into a single document will efficiently improve the software development lifecycle.
- Efficient traceability of requirements and its implementation in the software product.
- Minimizing the resource efforts on mapping Test cases with requirements
- Easier maintenance of requirements and test cases due to minimized documentation efforts.
- Increased transparency of the requirements aligned to a test case, which is useful during verification and reporting issues by pinpointing what requirements are failing during test execution.
- Avoid redundancy while verification since the test teams have a clear visibility of what requirements are linked to a test case.

To achieve all the benefits mentioned above we define the requirements and their associated test cases at the same time by using a template as derived in this thesis.

RQ2. How to align requirements to test cases?

Based on the literature review on aligning requirements to test cases, we have retrieved few important alignment methods as listed below

- Requirements based testing
- Traceability, Traceability with the help of the tools, Test traceability to requirements
- Model-based Testing (First Approach and Second Approach), Code centric approaches, model-centric approaches
- REST taxonomy

Also, the survey feedback on the test case template we created clearly reiterates that better alignment is obtained by having a good traceability matrix. Based on the survey feedback, below is the list of aligning methods retrieved

- Traceability Matrix
- Mapping roles and responsibilities

The aligning methods retrieved both from literature review and survey concludes that there are quite a few methodologies already well defined as part of various research projects, which followed would definitely achieve a better alignment between test cases and requirements.

However, there is lack of research done on how to create the actual tool (single template) that will help the users achieve all the benefits of above-mentioned alignment methods in the most efficient way. With our thesis we have drawn conclusions that alignment could be achieved most efficiently by using our proposed methodology i.e. by creating a single template that consists of both requirements and testing elements, both aligned to each other. The results gathered through the survey feedback on the test case template we have created as part of our thesis work also evaluates that the use of same template for both processes is highly valuable and can reduce lot of traceability issues in the early stages of development & testing life cycles. It indirectly reduces the lead-time of a project and also removes the redundant documentation needed separately for requirements and testing, if the practitioners are using separate templates for requirements and testing. Below we have provided details on how we aligned requirements to test cases before deriving a single test case template.

Our analysis indicates that a good template must store different kind of information, which is implemented during requirement development. If requirements templates are unclear, developers will understand the requirements in a different way, so it is important to have a good requirements template. Different templates are required in different phases of requirement management. Below are the few fields, which cover three key fields i.e., who should do it, why they must be done and what should be done.

Considering all the above key elements of both the requirements template and test case template, our template has been derived. In our template, few items like Requirement ID and Test case ID are aligned so that users are informed if one of them changes the other needs to be updated accordingly.

7.3 Conclusion

Requirements are the base for any project to deliver a successful product to the end user. Aligning requirements to test cases helps us to derive a successful project by identifying the issues at an earlier stage of the development process. There is lot of research performed on defining various aligning methods between requirements engineering and testing processes, however, there is lack of literature review on creating the actual tool(s) that will help organization's achieve the alignment between these two critical software development processes. Through our research we have delivered a template and the detailed process that describes the step-by-step procedure used for creating that template, which can be used both for the requirements and testing processes. To quantify the benefits of using our template and to further append more improvements to the template, we are recommending the software development organizations or researchers to use our template to manage both the requirements engineering and testing processes i.e. to adapt the template as Requirements and Test Cases template in an agile software development life cycle. The benefits of implementing our methodology described in this document are many. Above all it will give the developing teams control over what is being developed, built and tested. This ultimately will ensure that the delivered application is of a very high standard.

Based on our analysis on requirements templates, it is identified that good requirements must contain few sub-headings to make sure everything is included in the requirement such as Title, Description, ID Number, Source, Author, status, priority Motivation and

Dependencies. Title must clearly explain what the requirement is about. Description field must write about the requirement in detailed manner. ID number field contains a unique number, which helps in cross-reference. Source field is origin of requirement such as department, person. Author is the person who writes the requirement. Motivation field explains about why the requirement exists. Status field indicates where the requirement stands currently. Priority field indicates how important the requirement is, i.e. if it needs to be included in the initial stages of the development or can be included in the later stages of development. Dependencies field describes the relation between present requirement and other requirements. Test steps, Expected Results, Actual result, Pass/Fail, Description of the defect, Test step where defect occurs and Defect Severity. Test steps indicates steps followed in order to execute the test case, Expected Results explain the result that the users expects, Actual result field describes the actual result of the test, Pass/Fail indicates test result.

7.4 Future Work

Next step is to validate the test case template of our thesis by running a pilot in one of the organizations or research institutes. It helps us understand the detailed benefits/issues of the newly created template and it would enable us to quantify the benefits of the newly created test case template. Another topic of high interest in the research community of software development is finding methodologies/tools to improve the cross collaboration between developers and testers in a software development project, which is also one of the key factors to achieve a successful software product. There is scope for further research on checking if implementing our template would also increase the cross collaboration between the two important requirements engineering and testing teams, as the core aim of our template is to get these two processes aligned as much as possible.

7.5 Scenario Sampling

In this section we aim to detail how our template could be used in a real time project by illustrating through a simple business scenario. For the purpose of this thesis, we have considered a scenario with simple business requirements that lead to few functional requirements. Also, assuming that these functional requirements would be used by the developers to develop an IT application using an agile software development methodology.

Business Scenario 1: An online web solution that can be used by multiple users based on signup functionality and the first page would be logon screen with two key fields User Name and Password. Username should not be case sensitive, however, Passwords needs to be case sensitive and should be at least containing 6 characters long, and also password cannot be typed without a username.

Functional Requirements:

From the above scenario below listed three functional requirements would be derived

- 1/ First page of website is to be the logon screen with fields to input Username and Password
- 2/ UserName field that takes any character
- 3/ Password field that needs be at least 6 characters long and case sensitive

Considering the above functional specifications, both requirements and testing teams of this particular project would need to fill in the requirements and testing templates in order to manage the project during the software development life cycle. Using our template for this particular project would look as the below.

Multiple requirements	Requirement ID (R1)	Requirement Name Logon Screen	Description First page of the web application should be the logon page	Source BS1	Priority 5
	Requirement ID (R2)	Requirement Name Username field	Description Username field that takes any character and	Source BS1	Priority 5
	Requirement ID (R3)	Requirement Name Password field	Description Password field that can only take 6 characters long text and case sensitive	Source BS1	Priority 5
Dependencies and pre conditions	Dependencies				
	Test Environment Pre Conditions				
Test case & test steps for verifying requirements aligned to this test case	Test Case ID Password	Test Case Name Password field functionality test	Test Case Author		
	Tester ID				

	Test Scenerio	Test Case Description	Role Performing Test Case		
	Test Steps	Aligned Requirements	Expected Result	Actual Result	Pass / Fail
	Password field available in the logon page	R1	Available password field		
	Password cannot be typed without username	R2	Password can be typed only after typing username		
	Password length test, it should consists of 6 characters minimum	R3	Password is accepted only if it is atleast 6 characters long.		
Defect prioritization and issue description	Test Step where defect occurs	Defect Severity	Description of the Defect		
	Step1	P1			
	Step2	P1			
	Step3	P2			
	Step4	P3			
Step(n)	P2			

Table 26 Filled in test case template for scenario 1

Let us also assume that after performing few detailed reviews, business users or the end customers came up with a slightly different requirement for the password field i.e. passwords should be at least 8 characters long as per the below scenario.

Business Scenario 2: An online web solution that can be used by multiple users based on signup functionality and the first page would be logon screen with two key fields User Name and Password. Username should not be case sensitive, however, Passwords needs to be case sensitive and should be at least containing **8** characters long, and also password cannot be typed without a username.

As a result of the updated requirements, the requirements team would need to update the functional requirement related to the password field and since the test cases are also in the same template users can simply update the related test cases as well at the same time or easily communicate the changes to the testing teams when they pass on the updated requirements list. Based on our survey it is clear that most of the companies are still using two different templates for managing requirements and test cases and in this particular scenario where the change in the requirement is very minor to be noticeable, there is high probability that the users might over look the change in the requirements, which might provide wrong test results, further delaying the project.

8 Appendices

This survey collects the data via web link (www.surveymonkey.com) that is focused towards software companies who undertakes different roles and responsibilities in testing field. The survey result will be part of master thesis project. There are 12 questions, which might take 5-10 minutes to complete. Each contribution is appreciated and we would like to thank in advance for taking part in survey.

8.1 Appendices A

1. Which model is used in your software development organization?

- ☐ Spiral Model
- ☐ Agile Model(XP Model)
- ☐ Traditional Software Model(V-Model)
- ☐ Traditional Software Model(Waterfall Model)
- ☐ Depends on the project
- Other (please specify)

Comments

2. What is your present requirement alignment process?

		Decisions Organizational Level	atDecisions Level	at ProductDecisions Level	at Project Level
Pre-Project Decisions	Requirements	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
In-Project Decisions	Requirements	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>

Comments

3. When investigating requirements decisions, which of the following components effect decisions as per your view.

- ☐ Business process/objectives/rules
- ☐ Stakeholders involvement
- ☐ Information requirements
- ☐ Goal of the Organization
- ☐ Process of organization
- ☐ Decisions at three different organizational levels

Comments

4. Where do you face more challenges in requirement analysis as per your view?

- ☐ Business Objectives
- ☐ Business Processes
- ☐ Business Rules
- ☐ Requirement Collection
- ☐ Requirement documentation
- ☐ Software Product Requirements

Comments

5. What are the challenges you face in managing requirement alignment?

- ☐ Splitting composite requirements into simpler ones
- ☐ Abandoning requirements

- ☐ Mapping requirements to test cases
- ☐ Requirement Validation
- ☐ Issues caused by erroneous test cases
- ☐ Mapping requirement to erroneous test cases

Comments

6. How would you improve the alignment between requirements and testing at your Organization?

Comments

**7. What do you suggest for future work?
How would you improve our test case template?**

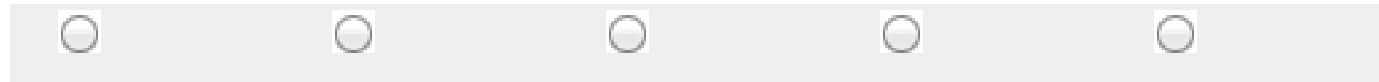
- ☐ Improved Test Case template
- ☐ Improved Requirement Analysis
- ☐ Improved Requirement Alignment process

Comments

8.2 Appendices B

**1. How would you rate the test case template we have created as part of our thesis?
(1 being poor, 5 being excellent)**

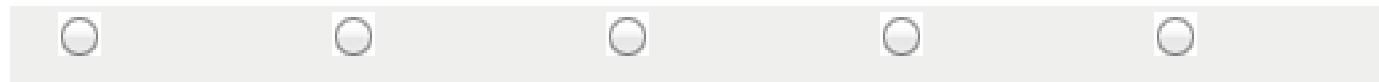
1 2 3 4 5



Other (please specify)

**2. How would rate the idea of linking requirements in the test case template itself?
(1 being poor, 5 being excellent)**

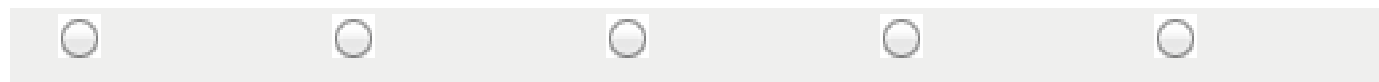
1 2 3 4 5



Other (please specify)

**3. Would the instructions sheet on the template submitted is clearly understandable?
(1 being poor, 5 being excellent)**

1 2 3 4 5



Other (please specify)

4. Would you foresee any issues with having requirement ID and Test case ID being in the same template? (1 being poor, 5 being excellent)

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Other		(please		specify)

5. Do you believe for any substantial improvements within your company by using test case template as we created? (1 being poor, 5 being excellent)

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Other (please specify)				

9 References

1. Uusitalo, E. J., Komssi, M., Kauppinen, M., & Davis, A.M. "Linking requirements and testing in practice." *International Requirements Engineering, 2008. RE'08. 16th IEEE*. IEEE, 2008
2. Skoković, Predrag, and Marija Rakić-Skoković. "Requirements-Based Testing Process in Practice." *International Journal of Industrial Engineering and Management (IJIEM)*, Vol.1 No 4, 2010, pp. 155 – 161
3. Gary E. Mogyorodi, "What Is Requirements-Based Testing?" *CROSSTALK The Journal of Defense Software Engineering*, March 2003.
4. Gutiérrez, J. J., Escalona, M.J., Mejías, M., & Torres, J. "Generation of test cases from functional requirements. A survey." *4^o Workshop on System Testing and Validation. Potsdam. Germany*. 2006.
5. Sabaliauskaite, G., Loconsole, A., Engström, E., Unterkalmsteiner, M., Regnell, B., Runeson, G., Gorshek T., & Feldt. R. "Challenges in aligning requirements engineering and verification in a large-scale industrial context." *requirements engineering: foundation for software quality*. Springer Berlin Heidelberg, 2010. 128-142.
6. Janzen, David, and Hossein Saiedian. "Test-driven development concepts, taxonomy, and future direction." *Computer* 38.9 (2005): 43-50.
7. Subramaniam, Venkat. "Test Driven Development–Part III: Continuous Integration."
8. Westfall, Linda. "The what, why, who, when and how of software requirements." *ASQ World Conference on Quality and Improvement Proceedings*. Vol. 59. 2005.
9. British Columbia. "System Testing Process Version 1.0." *IM/IT Standards & Guidelines*. Aug 10, 2007.
10. Barmi, Zeinab Alizadeh, Amir Hossein Ebrahimi, and Robert Feldt. "Alignment of requirements specification and testing: A systematic mapping study." *Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on*. IEEE, 2011.
11. Oates, Briony J. *Researching information systems and computing*. Sage, 2005.
12. "Agile Software development." Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 22 June 2013. < http://en.wikipedia.org/wiki/Agile_software_development >
13. Gutiérrez, J., Aragón, G., Mejías, M., Mayo, F. J. D., & Cutilla, C.M.R. "Automatic test case generation from functional requirements in NDT." *Current Trends in Web Engineering*. Springer Berlin Heidelberg, 2012. 176-185.
14. Bjarnason, E., Runeson, P., Borg, M., Unterkalmsteiner, M., Engström, E., Regnell. B., Sabaliauskaite, G., Loconsole, A., Gorshek, T., Feldt. R. "Challenges and practices in aligning requirements with verification and validation: a case study of six companies." *Empirical Software Engineering*. Springer Berlin Heidelberg, 2013.
15. Bernard, Eddy, and Bruno Legeard. "Requirements traceability in the model-based testing process." *Software Engineering (Workshops)*. 2007.
16. Denger, C. Medina M., and M. Medina Mora. "Test case derived from requirement specifications." *Fraunhofer IESE Report. Germany* (2003).
17. Popovic Tomo. "Automated Acceptance Tests and Requirements Traceability." *Software Development Magazine- Programming, Software Testing, Project Management, Jobs*. 2011.

18. Kukkanen, J., Vakevainen, K., Kauppinen, M., & Uusitalo, E. (2009, December). Applying a systematic approach to link requirements and testing: a case study. In *Software Engineering Conference, 2009. APSEC'09. Asia-Pacific* (pp. 482-488). IEEE.
19. Chakraborty, A., Baowaly, M. K., Arefin, A., & Bahar, A. N. (2012). The Role of Requirement Engineering in Software Development Life Cycle. *Journal of Emerging Trends in Computing and Information Sciences*, 3(5).
20. Erickson, John, Kalle Lyytinen, and Keng Siau. "Agile modeling, agile software development, and extreme programming: the state of research." *Journal of Database Management (JDM)* 16.4 (2005): 88-100.
21. Robertson, James, and Suzanne Robertson. *Volere: Requirements specification template*. Technical Report Edition 6.1, Atlantic Systems Guild, 2000.
22. Toro, Amador Durán, et al. "A Requirements Elicitation Approach Based in Templates and Patterns." *WER*. 1999.
23. John E. Bentley, Wachovia Bank, Charlotte NC, —Software Testing Fundamentals—Concepts, Roles, and Terminology, SUGI 30.
24. Gotel, Orlena CZ, and C. W. Finkelstein. "An analysis of the requirements traceability problem." *Requirements Engineering, 1994., Proceedings of the First International Conference on*. IEEE, 1994.
25. Boehm, Barry W. "A spiral model of software development and enhancement." *Computer* 21.5 (1988): 61-72.
26. Neuman, WL 2004, Basics of Social Research. Qualitative and Quantitative Approach First. J Lasser (ed), Pearson Education, Boston.
27. Post, H., Sinz, C., Merz, F., Gorges, T., & Kropf, T. Linking functional requirements and software verification. In *Requirements Engineering Conference, RE'09. 17th IEEE International* (pp. 295-302). IEEE, 2009.
28. Robertson, Suzanne, and James Robertson. *Mastering the Requirements Process: Getting Requirements Right*. Pearson Education, 2012.
29. Robert Feldt. "Aligning Requirements and Testing - Current Challenges and Solutions". Chalmers and Blekinge Tekniska Hogskola, 2013.
30. Sumit, Brijesh . P, Braidy . H, and Amy . W. Software Testing Times.
31. Mange. J, Martin. S. "A Systematic review of Software Development Cost Estimation Studies". IEEE transactions on software engineering, Vol .33, no. 1, January 2007.
- 32 Kitchenham. B. "Procedures for Performing Systematic Reviews". Keele, UK, Keele University, 33, 2004.
- 33 .Lauesen, Soren. "Task descriptions as functional requirements." *Software, IEEE* 20, no. 2 (2003): 58-65.
34. Lauesen, Søren, and Houman Younessi. "Six Styles for Usability Requirements." In *REFSQ*, vol. 98, pp. 155-166. 1998.

35. Harry N. Boone, Deborah A. Boone. "Analyzing Likert data". West Virginia University, Vol.50 No 2, April 2012.
36. PHELLAS, Constantinos N.; BLOCH, Alice; SEALE, Clive. Structured methods: interviews, questionnaires and observation. *Researching Society and Culture*. London: SAGE Publications Ltd, 2011, 181-205.
37. TROCHIM, William MK. *Research methods: The concise knowledge base*. Atomic Dog Publishing,
38. SOMMERVILLE, Ian; SAWYER, Pete. *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc., 1997.
39. HOEPFL, Marie C. Choosing qualitative research: A primer for technology education researchers. 1997.
40. Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry*. Beverly Hills, CA: Sage Publications, Inc.
41. Unterkalmsteiner, M, Feldt, R, & Gorschek, T. A taxonomy for requirements engineering and software test alignment. *ACM transactions on Software Engineering and Methodology (TOSEM)*, 2014, 23(2), 16.
42. Wybo Wiersma. *The Validity of Surveys: Online and Offline*, Oxford Internet Institute
43. Larman, Craig. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 3/e. Pearson Education India, 2012.
44. Webster, Jane, and Richard T. Watson. "Analyzing the past to prepare for the future: Writing a literature review." *Management Information Systems Quarterly* 26.2 (2002): 3.