Investigating the Application of TDD Practice in Large-Scale Industries

Pavan Kumar Varma Dantuluri
Rama Krishna Nethi

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden
This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Pavan Kumar Varma Dantuluri
E-mail1: pada15@student.bth.se
Rama Krishna Nethi
E-mail2: rane15@student.bth.se

University advisor:
Nasir Mehmood Minhas
Department of Software Engineering

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57
Abstract

**Context:** Developer’s within software companies work chosen choice of software development process. Choice of a particular Software Development Process impacts the work environment, end-product and might also have financial risks due to delay in delivering in the final software product.

**Objectives:** we are investigating if the TDD could impact/applicable in large-scale industries. To achieve that we first identify the strengths and challenges while using TDD in large-scale industries. Identify what modifications can enhance the impact of TDD in large-Scale industries.

**Methods:** Systematic Literature Review (SLR) has been used to investigate the application of TDD in large-scale industries. Using the info from SLR we conducted an online survey for validating the results obtained from SLR. As a last step we have conducted semi structured interviews to gather information from developers across industries practicing and practiced TDD. The data from the qualitative and quantitative methods is triangulated by identifying the strengths, challenges and modifications in applying TDD to large-scale industries.

**Results:** The findings from our SLR, results validated from Survey and responses from interview participants show that TDD have both strengths, Challenges and modifications.

**Conclusions:** Some challenges encountered when using TDD in large-scale industries are Negative test cases, linking test case modules of several developers, Communication, Overall idea about project, lack of TDD knowledge for developers practicing TDD, also lack of automation tools supporting the development of unit test-cases.

**Keywords:** Test Driven Development(TDD), Systematic Literature Review (SLR)
We would first like to extend our immeasurable appreciation to our thesis advisor Mr. Nasir Mehmood Minhas, whose guidance and understanding made it possible for us to work on this topic. It was a pleasure working with him. We would like to express our profound gratitude to whole staff of Software Engineering department who supported us all the way.

It would be incomplete of us without mentioning the motivation and strength we received from our classmates, course-mates here at Blekinge Institute of Technology during the duration of our stay here at Sweden.

Finally, we would like to express our never ending love towards our parents for achieving the success in all the aspects of our life. We thank you Mom and Dad for providing us everything we need in this beautiful world.

Pavan Kumar Varma Dantuluri
Rama Krishna Nethi
1 Introduction
  1.1 Prologue to Research Context ........................................... 1
  1.2 Aim and Layout of the Research ........................................ 3
    1.2.1 Aims of Research .................................................. 3
    1.2.2 Layout of Research Strategy .................................... 3
  1.3 Research Questions ..................................................... 4
    1.3.1 Research Question 1 ............................................... 4
    1.3.2 Research Question 2 ............................................... 4
  1.4 Phases involved in Research ............................................ 4
    1.4.1 Literature Survey ................................................ 5
    1.4.2 Data Collection through Survey Questionnaire ................. 5
    1.4.3 Interviews of Current TDD Practitioners ........................ 5
  1.5 Expected Outcomes ..................................................... 5
  1.6 Structure of Thesis .................................................... 5

2 Software Development Methodologies ............................... 8
  2.1 Software Development Methodologies - A Brief Insights ........... 8
    2.1.1 Traditional Methodologies .................................... 8
    2.1.2 Agile Methodologies ............................................. 9
  2.2 Test Driven Development (TDD) - Overview ....................... 10
    2.2.1 TDD Elaborated ................................................. 11

3 Research Methodology .................................................... 13
  3.1 Systematic Literature Review (SLR) ............................... 13
    3.1.1 Motivations behind Performing Systematic Reviews ........... 13
    3.1.2 Steps taken towards SLR ....................................... 14
    3.1.3 Implementing Systematic Literature Review .................. 14
  3.2 Survey ............................................................... 18
    3.2.1 Data collection through Survey Questionnaire .............. 18
    3.2.2 Interviews from software developers ......................... 19
    3.2.3 Sampling .................................................... 19
  3.3 Data Analysis ........................................................ 20
3.3.1 Quantitative Data analysis ........................................ 20
3.3.2 Qualitative Data analysis ........................................ 20
3.4 Reliability of Investigation: ......................................... 20

4 Insights from Systematic Literature Review (SLR) ............... 22
4.1 Strengths of Test Driven Development in Large Scale Industries 22
4.2 Critical criterion observed in TDD Practices in Large Scale Industries 23
4.3 Challenges faced by software practitioners in implementing TDD . 26
4.4 Structuring Systematic Literature Review .......................... 27
4.5 Modifications Observed through Literature Review ............... 28

5 Validation of Survey Questionnaire results and Interviews ........ 30
5.1 Survey Questionnaire Response ..................................... 30
5.2 Semi-Structured Interviews ......................................... 45
5.2.1 Interview Procedure ............................................. 45
5.3 Responses from Interviews .......................................... 46
5.3.1 Response of Interviewee 01 ...................................... 46
5.3.2 Response of Interviewee 02 ...................................... 47
5.3.3 Response of Interviewee 03 ...................................... 47
5.3.4 Response of Interviewee 04 ...................................... 48
5.3.5 Response of Interviewee 05 ...................................... 48

6 Analysis from Literature Review, Survey Questionnaire and Inter-
views ................................................................................. 51
6.1 Insights from Literature Review and Survey Questionnaire .... 51
6.2 Analysis of Benefits and Challenges of TDD in Large Scale Indus-
tries from data acquired through Survey Questionnaire and Inter-
views ................................................................................. 52
6.2.1 Respondent’s Qualifications ...................................... 52
6.2.2 Respondents experience with Test Driven Development (TDD) 52
6.2.3 Type of Projects Companies are solely focused on .......... 53
6.2.4 Complexity ............................................................ 53
6.2.5 Affect on Mean Time to Fix (MTTF) Metric ................. 53
6.2.6 Affect on Coupling Between Objects (CBO) ................. 54
6.2.7 Affect on Project Completion Time .............................. 54
6.2.8 Code Integration ....................................................... 55
6.2.9 Affect on Testing in TDD in Large Scale Industries ......... 55
6.2.10 Addition of New Functionalities ................................. 56
6.2.11 Affect on Code Maintenance ..................................... 56
6.2.12 Analysis of various other Parameter’s .......................... 57
6.3 Analysis of Challenges observed from Survey Responses .... 58
6.3.1 Affect due to Final task of Code Integration .................. 58
6.3.2 Affect on Testing in Large Scale Industries .................... 59
5.1 Representation of different category of respondents ............... 31
5.2 Representation of primary sector of developer’s company .......... 31
5.3 Representation of experience of survey respondents ............... 32
5.4 Representation of Number of years TDD in use ................... 32
5.5 Type of projects involved with TDD .............................. 33
5.6 Representation of factors to be evaluated when working with TDD 33
5.7 Representation of Nature of Complexity .......................... 34
5.8 Affect due to different geographical location’s of TDD developers 34
5.9 Projection of Code Maintenance in TDD ............................ 35
5.10 Affect on Mean time to fix (MTTF) metric ....................... 35
5.11 Affect of Coupling between Objects (CBO) factor ................ 36
5.12 Affect on project completion time ................................ 36
5.13 Affect of Initial transition to TDD methodology .................. 37
5.14 Affect on Code Integration ...................................... 37
5.15 Affect on Testing ............................................ 38
5.16 Affect on addition of newer functionalities ....................... 38
5.17 Influence of External factors ................................. 39
5.18 Affect on Software quality .................................. 39
5.19 Affect on development speed .................................. 40
5.20 Affect on Software Reliability ................................. 40
5.21 Affect on Defect Density factor ............................... 41
5.22 Affect on Productivity ...................................... 41
5.23 Size of Project ........................................... 42
5.24 Affect due to test-cases ...................................... 42
5.25 Affect on testing process in TDD in Large Scale Industries .... 43
5.26 Affect of TDD in large scale industries ........................ 43
5.27 Size of team ............................................ 44
5.28 Problem’s encountered by TDD developer’s .................... 45
## List of Tables

1.1 Organizational Size [2] .................................................. 1

4.1 Systematic Literature Review (SLR) Approaches .................. 27
4.2 Model of Concept Matrix [18] ................................. 27
4.3 Concept matrix of Observed Challenges and Benefits of TDD as found in various articles through literature review ............ 28
4.4 Concept matrix of Observed modifications of TDD as found in various articles through literature review .................... 29

5.1 Factors impacting the TDD in large scale industries from responders point of view ........................................ 44
5.2 Factors impacting the TDD in large scale industries from interviewee’s point of view ........................................ 49
5.3 Factors, frequency of responses for modifications from interviewee’s point of view ........................................ 50

6.1 Affect on Complexity in TDD ........................................ 53
6.2 Affect on MTTF metric in TDD ........................................ 54
6.3 Affect on CBO metric in TDD ........................................ 54
6.4 Affect on Project Completion Time in TDD ...................... 54
6.5 Affect of Code Integration in TDD ................................. 55
6.6 Affect of Testing in TDD ........................................ 56
6.7 Addition of new functionalities in TDD ............................ 56
6.8 Affect on Code Maintenance factor in TDD ...................... 56
6.9 Mapping of SLR and Survey ........................................ 57
6.10 Interview Response ........................................ 57
6.11 Initial Transition problems in TDD ............................... 59
6.12 Affect due to different geographical location of developers in TDD .......................... 60
Chapter 1

Introduction

1.1 Prologue to Research Context

Modern world has seen a significant upraise in the ubiquity and significance for programming advancement, which has developed immensely over the most recent 20 years. This has brought in for the utilization of various programming advancement procedures /Software development methodologies by the community of software developers around the globe. This procedure of developing or modifying the existing and new software development methodologies are persistent, wherein new techniques are developed keeping in mind the end goal to adapt up to the regularly evolving advances in the software development field and also taking into consideration of client’s inclinations [1]. Current business scene has constrained the enterprises to be on their toes, accentuating for auxiliary and vital consistence so that they could remain alive in the market. Also, programming itself is an elusive item, which changes quickly, and the later it is alterable in a venture stage, more prominent are the costs that an association ought to hold up under. This has brought in a need for spry advancement techniques [1].

Large-Scale Industries Everlasting result and potential benefits of the agile methodologies for small scale projects motivated the large-scale companies to implement the same. Kim et. al. interpret large-scale companies as those which are higher in size value. Here size is based on the number of persons or teams, project budget, code base size and project duration [2]. Based on the organizational size companies are categorized as follows.

Over the course of time, different software development methodologies have be-

<table>
<thead>
<tr>
<th>Number of Employees</th>
<th>Size of Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;10</td>
<td>Individual Groups</td>
</tr>
<tr>
<td>10-50</td>
<td>Small-Scale</td>
</tr>
<tr>
<td>50-500</td>
<td>Medium-Scale</td>
</tr>
<tr>
<td>Above 500</td>
<td>Large-Scale</td>
</tr>
</tbody>
</table>

Table 1.1: Organizational Size [2]
Chapter 1. Introduction

Software developers around the globe, initially concentrated on Traditional methodologies as the standard method which dates back to the 1970s and towards the later half of the twentieth century and the idea of Agile methodology similar to an adaptable approach made its appearance despite the fact that its underlying foundations date far back as the 1930s. The ideas pertaining to the Agile methodologies were initially and formally presented in the coordinated proclamation in 2001 with its 4 qualities and its 12 deft standards to help with giving better approaches to create software development practice particularly for enterprises and large-scale software development enterprises around the globe [7].

Over the course of years, in the field of software engineering Agile methodologies has been identified to be more efficient than the Traditional Methodologies [5]. A broad range of Software development methodologies are classified under the name of Agile methodologies. Some methodologies focuses on the practices (Ex: Extreme Programming (XP), Pragmatic programming, agile modeling), while some focus solely managing the flow of work (Ex: Scrum, Kanban). Agile practices cover a vast range of practices. Some of the notable ones include Test Driven Development (TDD), Iterative and Incremental development (IID), Acceptance test-driven development (ATDD) etc [6].

As of late, programming improvement groups utilizing coordinated procedures have begun broadly embracing Test-driven development (TDD). Regardless of its name, "test driven" or "test first" advancement isn’t generally a testing system. TDD works this way: For each little piece of functionality the software engineers, they first compose unit tests. At that point they compose the code that makes those unit tests pass. This strengthens the programmer to consider numerous parts of the component before coding it. It additionally gives a security net of tests that the developers can keep running with each refresh to the code, guaranteeing that re-factored, refreshed, or new code doesn’t break existing usefulness or functionality [1].
Chapter 1. Introduction

Test Driven Development (TDD) practices in Large Scale environment are different in various aspects compared to practising TDD in a small software company working on a small project. Some of the major things found to be obstructing when practising TDD in Large-Scale Industries are Level of integration and Testing the code [1]. The level of integration from an programming point of view refers to the integration of various classes written for the project. The level of integration can be overcome by test automation once all the classes belonging to the project are integrated as one. Nevertheless practicing TDD in Large scale industries pose risk in level of Integration of code, and another risk is the need for frequent communication if developers are geographically separated, which is very often the situation in Large Scale Industries [1] practicing TDD.

1.2 Aim and Layout of the Research

1.2.1 Aims of Research

The main goal is to identifying the benefits, and challenges faced by Implementing TDD practice in Large-Scale Industries through Systematic Literature Review (SLR) research methodology and knowing whether the challenges observed through literature review matches with the data collected through Survey Questionnaire and Interviews. This thesis concludes by answering the main Research questions raised and suggesting future work in the field of TDD methodology.

1.2.2 Layout of Research Strategy

The main objectives of this thesis work deals with identifying the challenges and strength of Test Driven Development (TDD) in Large-Scale industries. This thesis also answers how TDD can overcome those identified challenges faced by developers.

Initially the Strengths, Challenges are observed through Extensive Systematic Literature Review (SLR) research method. The strengths and challenges observed by Literature Review are correspondingly validated by data obtained through Survey Questionnaire and interviews. These Survey Questionnaire and interviews are answered by current TDD practitioners of large-scale software companies.

The final phase of this thesis deals with how we can overcome the short-comes of Agile by implementing TDD, through the insights gained from Literature Review, Survey Questionnaire, and interviews.
1.3 Research Questions

The research questions pertaining to this thesis are as follows:

1.3.1 Research Question 1

- What are the strengths of TDD, is TDD capable to overcome the challenges of agile in large scale environment?

The main motive of this research question is to observe the existing challenges faced by software developers/practitioners in large scale industries, and how Test Driven Development (TDD) helps in overcoming those challenges. The method employed in answering the research question is through Systematic Literature Review (SLR) of existing literature on Agile, and TDD methodologies. The acquired through various source’s like Conference, Journal papers, Research papers.

1.3.2 Research Question 2

- What are the modifications/enhancements are necessary to help use the TDD to cope up with these challenges in large-scale environment?

The perspective of this research question, our motive is to answer the modifications/enhancements that are necessary to help use TDD overcome the challenges that were observed in Research Question 1. We have analyzed the data acquired through Systematic Literature Review (SLR) of existing Literature on TDD, and also data acquired through Survey Questionnaire, and Interviews.

The main motive of this research question is to further extend the applicability of Test Driven Development in Large-Scale industries, and how we can overcome the challenges that developers previously faced.

1.4 Phases involved in Research

This thesis work is mainly divided into there major tasks. First task corresponds to the Literature Review. Second task is an extensive Literature Survey is done on the existing literature related to the foundations of Test Driven Development (TDD), the benefits of TDD, Challenges faced by TDD practitioners around the world. Third Task is on the modifications that are needed to overcome the disadvantages faced by current TDD practitioners.
1.4.1 Literature Survey
For the purpose of a broader understanding of the literature, several articles and Journals from various sources like IEEE, Elsevier, Science-direct and textbook references are made to gain deeper insights into the project.

1.4.2 Data Collection through Survey Questionnaire
Second major tasks deals with identifying the challenges, benefits and necessary modifications faced by TDD developers through extensive literature review and Survey Questionnaire. The Questionnaire was developed making use of Google forms. The Survey responses are collected from various people in software industries. Some of the samples selected for answering the Survey Questionnaire are current software developers, project managers, newly graduated and working students, and teaching faculty experienced in the field of Software Development methodologies.

1.4.3 Interviews of Current TDD Practitioners
For acquiring the challenges faced by developers, we have also opted to conduct interviews of software developers who worked with TDD in past and also software developers who are currently working with TDD. The mode of interviews we conducted are through social platform like Skype, Face-time. We even send our interview questions through mail for those interviewees who are not available due to time constraints.

1.5 Expected Outcomes
- Advantages and disadvantages of Test Driven Development (TDD) in large scale companies. These expected outcomes are obtained as a result of the Survey questionnaire and from the interviews taken from current software practitioners.
- The modifications that are required and need to be part of the TDD methodology practice, to mitigate the disadvantages of TDD, which are obtained through Interviews. The modifications are found out through Literature Survey and taking into the consideration of current software practitioners response obtained from Interviews.

1.6 Structure of Thesis
The structure of the thesis is organized into a sequence of section’s. The overall structure of the thesis is as below:
Chapter 1. Introduction

- Related Work
  - This section deals with the explanation of theory behind various Software development methodologies, in particular about Test Driven Development (TDD).

- Research Methodology
  - The research methods implemented in this paper like Systematic Literature Review (SLR), Forward and Backward Snowballing, also the Inclusion and Exclusion criteria are discussed in detail. The motive behind conducting SLR is also being discussed. Another major part of the validation of the challenges and benefits of TDD in large-scale industries are validated by conducting Survey. The Survey was answered by current TDD practitioners of large scale Industries across several companies. We have also conducted Interviews to some senior software developers as part of qualitative analysis for acquiring the current trends and the modifications required in practicing TDD across several companies.

- Insights from Systematic Literature Review (SLR)
  - The challenges, advantages and modifications are observed after extensive Systematic review of existing literature on TDD are mentioned here. The challenges and benefits of TDD obtained from various articles are presented in the form of Concept matrix. The concept matrix briefly identifies the key concepts pointed out in each article.

- Validation of results acquired from Survey Questionnaire Quick Interviews
  - In this section, the data acquired through Survey Questionnaire, and also insights gained from Quick interviews are validated with the unidentified challenges and benefits observed in the previous section.

- Analysis
  - The analysis part plays an important role in identifying the changes and benefits by cross-checking the data acquired through Systematic Literature Review (SLR), Survey Questionnaire, and Interviews. Any new concepts identified are listed as the enhancements that contribute to the field of Test Driven Development.

- Discussions and Limitations
  - In this section we have mostly tried to address the research questions alongside with information that we have gathered from systematic literature review. From the results obtained we have tried to present
some of the limitations that are important for this study. Furthermore, the authors also have addressed the threats to validity.

- Conclusions and Future Work
  - Finally, this section identifies the key areas of Test Driven Development (TDD) in which there is a further scope for working with software products in the coming days.
Chapter 2
Software Development Methodologies

Over the course of period different software development methodologies have evolved. Initially starting with the so called Traditional Software Development Methodologies (SDM), then the so called and widely practiced Agile Methodologies and Hybrid methodologies. In this chapter we intend to briefly outline the different methodologies at a very high level.

2.1 Software Development Methodologies - A Brief Insights

2.1.1 Traditional Methodologies

Traditional methodologies follow a sequence of steps from the very initial state of implementation, like it requires proper implementation steps from the start to the end of the projects.[7]. Here we are briefly focusing in 03 main traditional methodologies and they are Waterfall, Spiral Model, and Unified Process.

**Waterfall Software Methodology**

The Waterfall model is credited to be the most earlier Software Development Life Cycle (SDLC) approach that was utilized for software development processes [7].

In Waterfall methodology it is very difficult to change the course of action i.e., going backward and changing certain implementation steps done during the initial days of the project initiation.

**Spiral Methodology**

The Spiral model consolidates the possibility of iterative improvement with also taking into the orderly, sequenced process of the waterfall methodology. This Spiral model is a blend of iterative advancement and demonstrate consecutive straight improvement show i.e. the spiral methodology is a waterfall method
with a high accentuation on hazard/risk analysis. It permits incremental arrivals of the software product or incremental refinement through every cycle around the spiral [8].

The advantages of Spiral Model are identified and presented in [7], and the challenges of Spiral Software Development Methodology are described in [8]:

**Unified Process Model**

The Unified Software Development Process or Unified Process is a prominent iterative and incremental software development methodology [4]. The best-known and widely reported refinement of the Unified Process is the Rational Unified Process (RUP).

The Unified Process is not just a procedure, yet rather an extensible system which ought to be tweaked according to the requirements Software Organizations or to specific tasks. The Rational Unified Process is, correspondingly, an adaptable system. It is frequently difficult to state whether a refinement of the procedure was obtained from UP or from RUP, thus the names have a tendency to be utilized conversely[4].

### 2.1.2 Agile Methodologies

The term Agile was initially authored for the first time in the year 2001, and is normally composed as Agile (with a capital A) [9].

Agile Software development methodology depicts an arrangement of standards for software improvement under which necessities and arrangements advance through the communitarian exertion of self-sorting out cross-useful teams [9].

Some of the important Agile Software Development methodologies are:

- **Scrum**
- **Extreme Programming(XP)**
- **Test Driven Development**
- **Lean Software development**

**Scrum**

It is the most well known framework of Agile, which focuses especially on the best way to oversee undertakings inside a group based advancement condition. Scrum utilizes iterative and incremental model of development, with shorter span of emphases. Scrum is generally easy to actualize and concentrates on fast and regular conveyances [7].
Extreme Programming (XP)

It is a kind of Agile software development methodology. It advocates visit discharges in short advancement cycles, which is expected to enhance efficiency and present checkpoints where new client prerequisites can be embraced. XP addresses the investigation, advancement, and test stages with novel methodologies that have a considerable effect to the nature of the final result [9].

Test Driven Development (TDD)

In TDD, software developer initially writes small unit test cases, and check for its validation. The functionality is then added in small incrementation of codes to unit test cases. So in this practice, unit test code is written initially and further additional functionality are added in small increments [10].

Lean Software development

It is a practice in production that considers the consumption of assets for any objective other than the production of significant worth for the end-client to be inefficient, and consequently an objective for end. Working from the point of view of the client who devours an item or administration, the term esteem is characterized as any activity or process that a client would pay for. Lean is fixated on safeguarding an incentive with less work [7].

2.2 Test Driven Development (TDD) - Overview

In one word, TDD can be referred to as "Write code only to fix a failed unit-test case" [11]. That is, in TDD the developers initially focus on implementing the functionality required to the end software product in terms of Unit-tests. Then the necessary code is written to make those tests pass. This way of building the software ensures our software to be clean and risk-less, and the project can be broke up into manageable parts, which in turn decreases the total over-all time to complete the product.

The general procedure practiced when implementing TDD are [12][11]:

- **Rewriting a test that is failed in implementing a newer functionality.**

- **Add the necessary improvements to the failed unit-test code to make it functional.**

- **The code to be re-factored.**

The main steps involved in Test Driven Development (TDD) is shown in Fig.2.3
Chapter 2. Software Development Methodologies

Fig. 2.3 Flowchart representation of crucial steps in TDD Methodology

Every cycle/iteration utilizes the 03 steps shown in the above flow-chart of TDD. In short, these steps can be described as making a new test to implement a newer functionality, add code to that specific unit-test to implement the desired and expected functionality, re-factoring the final end-code [11].

2.2.1 TDD Elaborated

The major steps involved in Test Driven Development methodology are [12]:

- **Composing a test:** In the first place we compose a test in the initial step of the TDD cycle, we’re truly accomplishing something beyond composing a test. We’re making outline decisions. By composing the test before the code it’s trying, we are compelling ourselves to ponder how we need the code to be utilized.

- **Composing simply enough code:** The next major step of the TDD cycle is to compose enough code to make the test pass. Why simply enough code? The test we’ve composed is a test that is falling flat. It’s important that when we compose simply enough code, our primary objective is to make the test go as fast as could reasonably be expected. With the tests as our security net, we can then master to enhancing the plan in the last stride of the TDD cycle: re-factoring [13].

- **Re-factoring the composed code:** The last stride of the TDD cycle of test-code-re-factor is the point at which we make a look back, take a view at our outline, and make sense of methods for improving it. The re-factoring step is the thing that makes TDD practical. We could view TDD without re-factoring as a decent method for creating appalling code. Altogether tried appalling code, yet at the same time [14].
Incremental Development

TDD moves forward in continuing in little steps where each single step results taking nearer to the final product. TDD practice ensures that when the due date comes, we have the software product that we can convey and that works. The product may not have every one of the elements the client requested, but we have something that works [14].
Chapter 3
Research Methodology

For this research we have chosen SLR and survey as our research methods. The primary reason in choosing SLR and survey as the research method for data collection is, there are many SLRs available in this area. As TDD is used in various organizations, many SLRs have been published in this domain area. There is a reason for not performing the review of existing SLRs is the objectives of this research is different from the aspects discusses in the previous SLRs which we have come across. There are no detailed SLRs which have focused completely on the results related to strengths, challenges, solutions to the challenges using TDD in large scale environment. Therefore, a Systematic Literature Review (SLR) was selected for this research as it helps in identifying the keys factors related to TDD, where the results obtained can be reliable.

Survey is chosen as the research method to validate the data obtained from the Systematic Literature Review (SLR). Even though there are many surveys on TDD, their sole focus is completely not on results related to strengths, challenges, solutions to the challenges using TDD in large scale environment. By this survey, we intend to focus completely on our objects and impose the questionnaire strictly based on the results related to strengths, challenges, solutions to the challenges using TDD in large scale environment. Therefore, a SLR and survey are conducted for this research where the results obtained from the Systematic Literature Review (SLR) are validated and categorized using surveys.

3.1 Systematic Literature Review (SLR)

SLR is a method for recognizing, assessing and translating and scrutinizing all the accessible documents to answer an formulated research question.

3.1.1 Motivations behind Performing Systematic Reviews

There are many explanations behind undertaking a SLR. The most widely recognized reasons are [15]:
• To compress the current proof concerning a treatment or innovation e.g. to abridge the exact proof of the advantages and constraints of a particular Software Development methodology like TDD in our case.

• To recognize any crevices in current research keeping in mind the end goal to recommend territories for further examination.

• To give a foundation keeping in mind the end goal to suitably position new examine exercises.

3.1.2 Steps taken towards SLR

• Initially we have focused ourselves in identifying the key resources required in taking an initial step towards SLR.
  
  – Databases like IEEE Xplore, Google Scholar were chosen.
  
  – The keywords to be more specific the search strings are chosen to identify the key papers that provide a deeper insights into the TDD. Some of the chosen keywords are "TDD", "Unit-Test", "Test Driven Development" et cetera.

• The identified key papers are filtered out using the snowballing techniques at an earlier stage, and the final selected papers are further analyzed towards the goal of thesis work.

3.1.3 Implementing Systematic Literature Review

Systematic Literature Review (SLR) research methodology was employed throughout this thesis in identifying the key objective/goal of this thesis i.e., Benefits and Challenges faced by implementing Test Driven Development (TDD) in Large-Scale Industries.

To answer the research questions of the project, SLR has been conducted based on the guidelines provided by Kitchenham et.al [18]. Considering the specifications provided by Kitchenham et. al, SLR has been performed to identify, evaluate and explain the research.

SLR is conducted in a systematic procedure, where initial step is identifying the required literature from the primary studies, providing a defined set of instructions for data extraction and quality assessment so that no studies are missed out and high quality studies are covered. By this set of guidelines, more reliable and accurate results are obtained and it helps in minimizing the selection bias [16]. By using a Systematic Literature Review (SLR), part of the research questions RQ1, RQ2 are answered.
The different steps are briefly described below:

**Planning**

- Recognizing the need to survey.
- Determining the Research questions (Standalone review).
- Recognizing the themes to be secured from the Research questions (supporting review).
- Creating and approving the review protocol [17].

The planning level determines how deliberate and unequivocal our Systematic Literature Review (SLR) will be.

**Searching**

Searching for literature resources during the course of literature review plays a crucial role in final outcome of the project. Searching can be classified into two types. They are:

- Database Searching
- Snowballing

**Database Searching**

Database searching is based on selecting the right keywords and filtering out the papers using the right strategies, so in this way lot of unnecessary papers will be eliminated at an early stage.

**Snowballing**

Snowballing alludes to utilizing the reference rundown of a paper or the references to the paper to recognize extra papers. It expands on thoughts introduced by for instance Webster and Watson [18] and also sketched out by Wohlin and Prikladnicki [19]. The snowballing system is briefly explained over in the accompanying subsections.
Chapter 3. Research Methodology

- **Keywords** The keywords used for the search are mentioned in a detailed manner in this section. The keywords have been formulated by taking a closer look at the research questions. The keywords used are listed below:
  
  **Set1**: test-driven development, TDD, test driven development, test-driven design, Test driven*
  
  **Set2**: software development methodologies, software development*, software methodologies, software*
  
  **Set3**: strengths, strength*, pros
  
  **Set4**: Challenges, cons
  
  **Set5**: Unit test

- **Search String** The search string that has been used for this research is presented below. The same has been used in the repositories and the database searches implemented.
  
  The search string used for this study is:
  
  `((("test-driven development" OR "TDD" OR "test driven development" OR "test-driven design" OR "Test driven*") AND (software development methodologies OR software development* OR software methodologies OR software*)) AND (( strengths OR strength* OR pros) AND (challenges OR cons)) AND (unit test)))`

- **Beginning Set**
  
  - In database seeks, the initial step is to recognize catchphrases and plan seek/search strings. While applying a snowballing approach, the primary test is to recognize a begin set of papers to use for the snowballing technique. The begin set is appeared in the highest point of Fig.3.3. Any scan for papers to incorporate into the begin set creates a speculative begin set. The real begin set is just those papers in the speculative begin set that toward the end will be incorporated into the orderly writing study [18][20].

- **Iterations**
  
  - Once the begin set is chosen, including just papers that will be incorporated into the last examination, the time has come to begin the main cycle directing in backward and forward snowballing. To at long last choose to incorporate a paper implies that the full paper ought to be analyzed before choosing to utilize it as a paper in the snowballing. If not doing this, a rollback is required if different papers are incorporated in view of a paper that later is barred. In this manner, it is imperative to be sure on incorporation before utilizing the paper for snowballing by any means [20].
• **Backward Snowballing**

The initial step is to make the reference list and avoid papers that don’t satisfy the essential criteria such as, dialect, year of publication and sort of distribution. Once these are evacuated, the rest of the papers are genuine possibility for inclusion [20].

Once the paper is found, the abstract is analyzed first and after that different parts of the paper are analyzed until a complete choice can be taken to either incorporate or remove the paper [18][20].

• **Forward Snowballing**

Forward snowballing alludes to distinguishing new papers in light of those papers referring to the paper being analyzed. The references to the paper being analyzed are considered utilizing Google-scholar. To start with, the dynamic is considered, and if this is lacking, the place referring to the paper effectively included is inspected.

• **Inclusion and Exclusion**

The papers found through included papers are ought to be utilized as a part of the examination. The following steps are taken once the papers are found.

• **Inclusion Criteria**
  - IC1: Articles that are published only in English are considered.
  - IC2: Full-text availability of the articles
  - IC3: Peer reviewed articles
  - IC4: Studies that directly link to the contextual factor of challenges encountered and dealing with them while using TDD in the large scale industries.
  - IC5: Articles that are published in the last 10 years are included.

• **Exclusion Criteria**
  - EC1: Articles that do not focusing on mentioning the challenges and solutions to overcome by using TDD in large scale industries.
  - EC2: Studies that are an earlier version of the latest one already included in the research.
  - EC3: Review Papers. Only references of the review papers were examined in order to eliminate the duplicate studies available.
Initially we after applying the search string in the IEEE Xplore database, we found 1252 articles. Then we apply Inclusion criteria to these articles, which resulted in 17 articles. We further applied our Exclusion criteria, then the articles are reduced to 7. Finally after thoroughly studying the full text of those articles, we selected these 7 articles as our start set.

In the iteration 1, we go through the forward and backward snowballing with the start set, where we search the literature presented in the references and citations in the google scholar, after applying the inclusion and exclusion criteria, we found 5 articles.

In the iteration 2, we used the 5 articles that found in iteration 1 for performing the forward and backward snowballing. After applying the inclusion and exclusion criteria we found 2 articles.

Finally we selected the 14 resulted articles and performed the systematic literature review.

3.2 Survey

3.2.1 Data collection through Survey Questionnaire

To validate the information obtained from SLR, survey was selected as a research method. According to [21], a survey method provides a “quantitative description of trends, attitudes or opinions by studying a sample of population”. The survey method is performed by a web-based questionnaire.

Survey was opted as one of the research method as it is a convenient method of collecting response from the respondents all over the world. As the objective of this thesis is to provide the information regarding the strengths, challenges, enhancements that are required to overcome the challenges in using TDD in large scale industries, this method can help in acquiring the data from various large scale industries. Moreover, this method is easy, low in cost and saves time. By this method, the data obtained can be analyzed quickly and helps in faster completion of the research. Our thesis is to provide the generalizable list of results of strengths, challenges and solutions to those challenges that occur in TDD, which we can apply for large scale environment instead of limited cases. So, a survey can help in this case as it provides the information from all parts of the world, which helps by providing generalizable results.

Survey method is considered for validating the results related to strengths, challenges, solutions to the challenges using TDD in large scale environment which are obtained after performing a Systematic Literature Review (SLR). Th survey is conducted by developing a web questionnaire and sharing it through the web.
Survey method has several benefits when compared with remaining methods [22]. On-line option of distributing the questionnaire was selected, which is very easy, low cost option, takes less time for development and analyzing the data and it takes less time for the respondents to respond.

SLR and Survey method is used to answer the research question RQ1, while SLR and Interviews are used to answer RQ2.

### 3.2.2 Interviews from software developers

Interviews is also one type of research method, it is primarily used to validate the gathered data by asking the participants a series of questions.

Motivation for selection of interviews:

We selected semi structured interviews as unstructured interviews when selected it may sometimes lead to situation where the results are biased towards a single direction. The structured interviews have close ended questions which restrict the participants to express their knowledge. As the semi structure interview has both the close and open ended questions if designed properly then it leads to a better information gathering tools so we preferred the semi structured interviews [23].

The interview participants we have selected are 5 participants. Initially a list of 8 participants were selected but all of them did not wish to participate as they had other busy project schedules. The interviews subjects were selected in such a way that they had various level of work experience in the Test driven development. The interview participant’s all of them had the work experience in the software field, from our interviews we have collected their knowledge and experience in the test driven development field.

### 3.2.3 Sampling

It is very important to select the sample population for survey. A sample represents the larger group, which is a subset of target population [24]. There are different sampling methods that are used to select the sample. Probabilistic sampling methods and non probabilistic sampling methods are the two sampling methods. The sampling method that has chosen for the surveys are the convenience sampling method. The respondents selected using convenience sampling method is used for the data collection using interviews and online questionnaire. The convenience sampling method is non probabilistic sampling method, where we select the respondents who are the practitioners convenient or available readily and suitable to our study.
Using convenience sampling method, we selected the respondents who are aware of TDD. The respondents are actually the employees in various software organizations, playing various roles like developer, manager, analysts and designers. Through the emails we sent the request and link of survey questionnaire to the respondents. Then a total of 52 people had responded to the survey, but finally 49 responses were completed and collected.

In our study, using the convenience sampling method we have selected the interviewees using personal contacts. The interviewees that are selected have high experience in practicing TDD. We have selected five interviewees finally, where the roles of each interviewees are senior software developers, Test engineer, project manager and analyst. The respondents of online questionnaire were not selected for the sample of the interviews.

3.3 Data Analysis

3.3.1 Quantitative Data analysis

We have used the online questionnaire for the survey, then to analyze the data that is obtained from the online survey questionnaire is done using the techniques namely frequency distribution and also the graphical display of results. Frequency distribution on our study that we have applied is useful to refer to tabular representation of responses for each option from the online survey questionnaire [25]. The data that is obtained from this frequency distribution is then used to represent them in a graphical form for better reader understandability [25]. This graphical notation of data takes advantage of bar graphs and pie charts. Further more we have used mean value to understand how TDD is affected in large scale industries.

3.3.2 Qualitative Data analysis

We gathered the qualitative data from the SLR and interviews. The data that is obtained from the articles and interviews are analyzed using the qualitative analysis method such as coding, the coding is done based on interview transcripts. There on the data is analyzed using the descriptive statistics method called frequency distribution to arrive on a conclusion [25].

3.4 Reliability of Investigation:

The following measures have been taken to make sure that the reliability of the findings are in tact.
• Cross Comparison: After each interview the identified information regarding the strengths, challenges, modification of TDD implementation have been discussed with the supervisor and amongst the authors of the research.

• Triangulation: Norman K Denzin [26] states that triangulation of data is an alternative approach used to validate the findings and also gives an indepth understanding of the phenomenon. We made use of this technique to validate the findings of implementing one method with the findings of another method.

• Constant Checking: The various factors associated with the implementation of TDD in large scale industries obtained has been used as an input to the upcoming interview. This has been carried out for the five interviews conducted until a concrete data set has been gathered. The data obtained from the larger population has been validated from the a reliable source of asking the interviewees face to face.
Chapter 4

Insights from Systematic Literature Review (SLR)

The key objectives of this thesis work is for identifying the Application of TDD practice in Large Scale Industries which has led to a set of benefits and challenges faced by current TDD practitioners across the globe.

4.1 Strengths of Test Driven Development in Large Scale Industries

Based on the key parameters or variables used in several research papers [Ref B] for identifying the benefits and limitations of TDD are grouped and summarized below. Some of the key variables are Productivity, Code quality, Development time et cetera.

- **Viable, Adaptable, Effortlessly Extensible**: In TDD, testing is co-ordinated, it is ensured that each independent bit of rationale can be tried, this has clear advantages for organizations looking for faster development process[27].

- **Unparalleled Test-Scope and Streamlined Codebase**: In TDD, code is never composed without first written completed test. This outcomes in uncommon test scope. Also, the refactoring procedure guarantees composed code is as prudent as would be prudent, streamlining the codebase [27].

- **Including new functionality**: TDD gives software engineers the certainty to change the bigger design of an application while including new usefulness.

- **Specialized Changes in Infrastructure**: TDD gives them the software developer/practitioner the certainty to roll out absolutely specialized improvements that expansion code can polish to a further extent – and in this manner make applications that are both more viable and more extensible [27].
• **Usefulness of Refactoring Code**: The refactoring procedure fundamental to TDD guarantees that designers continually shore up the quality of the codebase. This keeps applications from becoming dated and solid [14].

• **No mis-guided feeling of early advance**: Since TDD makes use of shorter life-cycle period for each unit-test case, the developer as a single and all developers associated with the project as a whole will have the current progress of the product without any misguided feeling of early advance.

• **Reduces External Interruptions**: TDD enormously lessens the effects of interferences, as the developer is only concentrating on one particular capacity at any given moment, and frequently even one part of that capacity at a given time. Yes there are things you need to stack into the RAM, however there’s significantly less [28].

• **Shorter development life cycles**: Since each developer in a project associated with TDD, is concerned only with developing particular unit-test case assigned to him/her, and since each test case is associated with incrementing only a specific functionality associated with that particular unit test case. Hence all this process leads to a shorter development cycle [11].

• **Executable Documentation**: The test cases in TDD, are composed as tests, different software engineers can see the tests as utilization cases of how the code is expected to function [28].

### 4.2 Critical criterion observed in TDD Practices in Large Scale Industries

Some of the critical factors that are observed through Systematic Literature Review (SLR) where TDD performs well compared to the Traditional software development methodologies, and over some Agile practices are briefly outlined as below:

• **Amount of time taken in Pre-development planning**: Considerably, low amount of time is taken in pre-development planning stage of Test Driven Development (TDD) scenarios [29].

• **Knowledge on TDD**: The knowledge of developers practicing TDD plays a crucial role in applying TDD to their software development of the product. And in many cases for both experienced and novice developers practicing TDD it has been observed as a severe limiting factor[30],[Ref A].

• **Complexity**: The impact on complexity is another crucial factor to be considered in the process of Software Development cycle. Smaller unit test
cases, and gradually incrementing the functionality of the test module decreases the number of unnecessary redundant classes and methods. Also, there is a gradual decrease in the degree of nesting (nested block depth) in TDD when compared with other software development methodologies. Hence the impact of complexity is reduced [27]. Below all parameters can be categorized to determine the overall Internal quality of the code.

- **Complexity of Weighted-methods**: Weighted-techniques multifaceted nature measures the total of cyclomatic complexities for all methods in a class. TDD software engineers reliably created classes with lower unpredictability as far as the quantity of branches and the quantity of methods. The reliably more straightforward classes by test-first developers isn’t amazing considering the prior report of less techniques per class [11][27].

- **Cyclomatic Complexity per method**: Cyclomatic complexity is a software metric (estimation), used to demonstrate the complexity nature of a program. It is a quantitative measure of the number of straightly free ways through a program’s source code. It was created by Thomas J. McCabe, Sr. in 1976. It has been identified that a lower Cyclomatic complexity has been observed per method in TDD [27].

- **Nested Block Depth (NBD)** The Nested Block Depth (NBD) is a metric which specifies the level of depth of nested scopes in a method’s body. Higher NBD value makes the software code more difficult to understand and also difficult for identifying and correcting the bugs. Any Software methodology aim is to minimize this value as low as possible. It has been found that TDD developer’s reported a lower observed NBD value compared to others [31][29].

- **Efficiency and Flexibility**: It has been observed that the flexibility of code in TDD is more than other methods, as modifying a single test case is enough to improvise that particular functionality provided by that unit test case, when compared to traditional methodologies where the code change at particular point impacts the overall code, and effects the code very badly. This single modification pertaining to only single unit test cases improvises the overall efficiency and flexibility of the individual test cases and overall productivity of the code [14].

- **Code Maintenance**: The code written in TDD style is substantially less demanding to keep up, prompting life-cycle cost investment funds contrasted with non-TDD advancement. It’s too simple to overlook the experiments in test-last improvement, and the experiments make troubleshooting the framework a considerably less demanding errand. By its temperament, TDD helps you all the more rapidly pinpoint code issues, and the relapse
test suite helps you make a decent quality settle a great deal more rapidly [11].

- **Mean time to Fix (MTTF) Metric**: TDD’s belongings appear most grounded in this metric. It has been observed/reported by developers that TDD practice issues are less demanding to analyze and investigate. The accessibility of the TDD relapse test suite additionally helps massively in such manner [32].

- **Impact on Coupling Between Objects (CBO)**: The tendency of TDD developers to implement solutions with more and smaller classes and methods which generate more connections between the classes. Hence the coupling between objects (CBO), which measures the number of connections between objects, is reported to be higher in TDD [32].

- **Project Completion / Development time**: Project completion time is a crucial factor for bigger companies and sometimes when a new practice is implemented like TDD proper care has to be taken. An over-all increase in project completion time has been observed in the case of TDD [29].

- **Rate of Success**: TDD helps more quickly pinpoint code problems, and the regression test suite helps one to make a good-quality fix much more quickly. As a consequence of this the rate of success observed in projects implemented using the TDD practice are very high compared to others [9].

- **Total lines of composed code**: On the off chance of the metric is "add up to lines of code composed," TDD engineers performed superior to any other software development methodologies many would anticipate. TDD groups composed less code, not with standing including the code to cover all the experiments [9].

- **Final Software Quality**: It has been found that the Supports of TDD contend that it conveys higher-quality programming. The quality results from chipping away at very much determined errands, utilizing continuous relapse testing, and discovering blunders prior in the fast and shorter development cycle [32].

- **Internal Quality**: It is also found that with TDD, the internal quality improved – both in subjective assessments of how clean the code is and objectively in the number of bug reports. Also, developers practicing TDD are driven to think about how the system be organized in smaller steps[33].

- **Impact on code size**: In TDD practices, since the developer is motivated to write smaller unit tests and test them before further incrementing the
functionalities of the unit test. An important observation made after reviewing all studies based on Test Driven Development are, developers practicing TDD tend to write smaller unit test-cases on average [33].

4.3 Challenges faced by software practitioners in implementing TDD

Some of the challenges faced by TDD developers observed after extensive literature review are presented below:

- **Productivity**: It has been observed that many numerous potential adopters stress over an efficiency hit when focusing on TDD. Like any new practice, TDD will include an expectation to absorb information. Be that as it may, past that, the expansion of experiments related with TDD must be overseen and kept up and could subsequently require more exertion than a traditional software development approach [30].

- **Testing**: Since in TDD, each module of the software is written as smaller unit-test cases by different developers across the company. Also, because a feature can be implemented across multiple modules/ unit test modules and each of these modules might have dependencies on other feature modules. Hence a great deal of informal communication is needed among the developers of individual code units. Thus, testing product features as overall is cumbersome and time taking process in TDD [8].

- **Geographical difficulties**: The main challenge for geographically distributed development is continuous integration and testing. These activities are difficult enough with traditional integration testing techniques, but they pose a serious problem [12].

- **Difficulty in maintaining code**: Another difficulty observed with TDD is, maintaining the test code. TDD results in a high ratio of test to production code, so a system change is highly likely to affect the test code as well. The challenge is how to change a software feature knowing that such a change will affect the test code [28].

- **Code Integration**: Improving the quality of unit tests combined with continuous testing and automated testing are still a tough task for TDD developers. Also, the effort spent in integrating code from disparate teams spread across geographically (in some instances) and achieving a successful build was significantly low in the case of TDD [12].

- **Difficulty in Initial Transition**: The transition from unit to system-level testing is also a challenging for TDD. For, example low-level unit tests that
pass trivially can provide significant code coverage, giving a false sense of security in the code’s quality. Also, the proliferation of test cases associated with TDD must be managed and maintained and requires more effort than a traditional approach [12].

Just like every Software development methodology[26] has both pro’s and con’s, some of the major advantages and challenges faced by implementing Test Driven Development (TDD) are briefly stated above. With our still ongoing literature review on TDD, we are confident that our thesis and study would provide further insights into the many aspects of Implementing Test Driven Development (TDD) methodology in Large Scale Software Industries.

4.4 Structuring Systematic Literature Review

In general, Systematic Literature Review (SLR) is Concept-Centric (CC). In this manner, concepts decide the arranging structure of a review. Conversely, a few authors adopt a Author-Centric strategy and basically display a rundown of the pertinent articles. This technique neglects to incorporate the writing. The two methodologies are effortlessly perceived, as shown in Table 4.1 [18].

<table>
<thead>
<tr>
<th>Concept Centric</th>
<th>Author Centric</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConceptX [authorA, authorB, \cdots]</td>
<td>Author A \cdots Concept X, Concept Y</td>
</tr>
<tr>
<td>ConceptY [authorA, authorC, \cdots]</td>
<td>Author B \cdots Concept X, Concept W</td>
</tr>
</tbody>
</table>

Table 4.1: Systematic Literature Review (SLR) Approaches

The concept matrix has been compile according to (Table 4.2), a thought we have implemented from Salipante et al. (1982) [18]. At the point when our study is finished, the learned concepts has been synthesized by talking about each distinguished idea. Before starting this progression, we set aside some opportunity to build up an intelligent way to deal with gathering and exhibiting the key ideas that have been revealed during extensive literature review.

<table>
<thead>
<tr>
<th>Concepts</th>
<th>Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6</td>
</tr>
<tr>
<td>A</td>
<td>* * * *</td>
</tr>
<tr>
<td>B</td>
<td>* * *</td>
</tr>
<tr>
<td>C</td>
<td>* * *</td>
</tr>
<tr>
<td>\vdots</td>
<td>* *</td>
</tr>
</tbody>
</table>

Table 4.2: Model of Concept Matrix [18]
Chapter 4. Insights from Systematic Literature Review (SLR)

Using the above concept of Concept Matrix practice from SLR, the challenges and benefits of Test Driven Development (TDD) observed mentioned in the previous sections 4.1, 4.2, 4.3 are summarized in concept matrix form as shown below:

Similar tables of other parameter’s taken into consideration are shown below:

<table>
<thead>
<tr>
<th>Affect on Parameter</th>
<th>[11]</th>
<th>[12]</th>
<th>[31]</th>
<th>[32]</th>
<th>[13]</th>
<th>[14]</th>
<th>[27]</th>
<th>[28]</th>
<th>[29]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Productivity</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Quality</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complexity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Internal Quality</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD life cycle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Development Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>Rate of Success</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>Transition to TDD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>Defect density</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTTF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-Development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
</tr>
<tr>
<td>Total LOC Written</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Re-work</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cyclomatic Complexity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NBD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend**
- **others** ‘LOC’ : Lines of Code, ’NBD’ : Nested Block Depth,
  ‘MTTF’ : Mean Time To Fix

Table 4.3: Concept matrix of Observed Challenges and Benefits of TDD as found in various articles through literature review

### 4.5 Modifications Observed through Literature Review

- It is recommended as a modification that product developers practicing TDD must break down the nature of the experiments being created, through measuring and monitoring measurements, for example, code scope and imperfection proportion, with a specific end goal to keep TDD proficient and simple [34].
• It has been said as an modification that TDD will be best practiced if the developers are geographically not separated [1].

• It has been identified that frequent refactoring of unit test-cases for better product quality could be a better way for current TDD practice modification [1].

• It has been clearly emphasized that the practice of TDD should be modified such that the developer can get into the many other not so functional requirements, hence generating less test cases that lead to fewer negative results at the end of the software development cycle [35].

• Refactoring for viability should work to streamline the code wherever conceivable. The refactoring procedure may comprise of shortening strategies, streamlining control structures, and giving illustrative identifier names, (for example: capacity, variable, and class names) [36].

• For programming to develop with new requests, the programming should bolster new expanded functionalities with little cost. The capacity to help such changes is critical, particularly in a viable way [36].

• Designers don’t have to concentrate on a part of an untested element. The essential objective when composing code with TDD is just to finish the tests [37].

<table>
<thead>
<tr>
<th>Article</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>[34]</td>
<td>Break down the nature of experiments being created, through measuring and monitoring measurements.</td>
</tr>
<tr>
<td>[1]</td>
<td>Refactoring of unit test cases.</td>
</tr>
<tr>
<td>[35]</td>
<td>Modification of TDD such that generating less test cases for functional requirements</td>
</tr>
<tr>
<td>[36]</td>
<td>Refactoring strategies like shortening, streamlining and giving identifier names can be helpful.</td>
</tr>
<tr>
<td>[36]</td>
<td>New functionality implementation with low cost.</td>
</tr>
<tr>
<td>[37]</td>
<td>Finishing the test is essential objective of TDD, not to focus on untested areas.</td>
</tr>
</tbody>
</table>

Table 4.4: Concept matrix of Observed modifications of TDD as found in various articles through literature review
Chapter 5

Validation of Survey Questionnaire results and Interviews

The main focus of this section is presenting the data acquired through Survey Questionnaire answered by current TDD developers/Practitioners working in Large Scale Industries. The Survey Questionnaire is designed properly to avoid the bias to a maximum extent. Regarding the confidentiality of the answers of the survey respondents we are utmost careful that the responses will be used for the study of thesis. Also the primary reason of gathering e-mail is to mail the results of the survey to the respondents. Also the survey respondent are carefully chosen in way that the respondents are currently practicing TDD or practiced TDD in the past.

5.1 Survey Questionnaire Response

The questions and their corresponding data acquired are shown below:

1. This response deals with the different types of people answered the Survey Questionnaire belong to like Developers, Project managers, Software testers etc.

   The major portion of the survey participants are comprised of TDD developers (54.2%), Designer’s (16.7%), Project Manager’s (10.4%).

2. The primary sector company focused on:

3. Number of years of Experience in Test Driven Development (TDD) methodologies:

   From the above response we can clearly identify the experience possessed by TDD developers. It conveys that almost 48% of developers have an experience of 1-2 years in working with TDD.
4. Since how many years TDD has been in use by the company:

From the above response acquired from the Survey Questionnaire, we can clearly identify that 46% of survey has been answered by developers working with TDD and having 3-5 years experience. 20.8% developers are possessed with an experience of 1-2 years, and 29.2% developers are those who have worked with TDD only in certain projects.

5. Type of Projects handled:

The response indicates that majority (64.46%) of the developers have worked with
Chapter 5. Validation of Survey Questionnaire results and Interviews

Test Driven Development (TDD) projects for a client, and 18.8% TDD developers worked with adaptation of commercial software. And also 14.6% of developers worked with TDD with the Internal development projects within the company.

6. Typical Factors to be evaluated when working with TDD:
From the response, the view of majority of TDD developers/practitioners is that when working with Test Driven Development (TDD) the development team is taken into consideration (35.4%), and then the requirements of the project (33.3%), then project time limit (27.1%) is taken into consideration.

7. Nature of Complexity with TDD:
From the view of TDD developers, it is clearly projected that 70.8% of TDD developer’s felt that the nature of complexity of implementing a project with TDD
has been decreased, while 16.7% reported an increase in complexity when working with TDD projects, and 12.6% developer’s observed no change in complexity factor.

8. Geographical Locations:
From the above responses, it clearly states that 58.3% of developers faced problem’s due to different geographical location of other developer of same team.

9. Code Maintenance:
From the responses, we can say that 83.3% of TDD developers of who answered the survey questionnaire felt that the maintenance of code is much easier in TDD, while the rest if the survey respondents (18.7%) of developers felt difficulty in maintaining code in TDD practices.
10. Mean Time to Fix Metric (MTTF):

From the above results, we can briefly say that 68.8% of TDD developers observed a low MTTF factor compared to that of the MTTF metric in traditional methodologies, while 25% felt an increase in MTTF.

11. Coupling Between Objects:

From the above results, 66% of TDD developers who attempted the survey questionnaire felt that a LOW coupling between objects (CBO) has been observed, and 19.1% developers reported an increase in CBO, while 14.9% developers observed no significant change in CBO.
12. Project Completion Time:

It has been identified that 70.8% of the developers felt an faster project completion time, while 20.8% of developers felt an increase in over-all project completion time, while 8.3% developers felt that project completion time is not affected.

13. Initial Transition problems:

58.3% percent of developers faced problems on their initial transition to TDD methodology to their previous methodology, while 35.4% of developers smoothly transitioned to TDD from their old software development methodology with out facing any problems.

14. Code Integration:
It has been answered by 70.2% of TDD developers that code integration factor has been clearly increased in TDD, while 14.9% developers felt a significantly low code integration.

15. Testing:

It has been answered that 52.1% of developers felt the testing process to be quite simple in Test Driven development (TDD) methodology, while 47.9% developers responded as the testing process is cumbersome/difficult in TDD.

16. Addition of new Functionalities:

76.6% of developers felt that addition of new functionalities to the existing code
in TDD was easier, while 14.9% of developers felt that modifying the existing TDD test cases for adding new functionalities are difficult.

17. Influence of External factors:

It has been answered by 66.7% developers that influence of external factors on project development is very low in TDD, while 18.8% of developers felt that the development of projects in TDD environment is slightly influenced by external factors.

18. Software Quality:

Increased quality of software has been identified by 31.9% of software developers, while increase in efficiency of software has been observed by 25.5% of developers, while 34% developers has identified observed an increased effectiveness of soft-
ware, and finally 8.5% developers observed an increase in productivity of software.

19. Speed of Development:

It has been answered by 38.3% of developers that the speed of development is good in TDD, while 34% of developers answered that the speed of development is very good in TDD, while 23.4% developers has reported that the speed of development in TDD as average.

20. Software Reliability:

An increase in software reliability factor has been reported by 73.9% of developers, while 13.9% reported decreased reliability, and 10.9% of developers reported no impact on software reliability.
21. Defect Density:

67.4% of TDD developers who answered the question has felt an decrease of Defect Density factor, while 15.2% developers felt an increase in the defect density factor.

22. Affect on Productivity:

An increase in productivity of project implemented by TDD has been reported by 70.2% of developers, while 17% of developers felt an decrease in productivity, and 12.8% of developers felt there was no significant affect on productivity of projects when implemented using TDD.
23. Size of Project:

It has been responded by 76.2% of developers that the size of Project is a crucial factor to be considered when working with Test Driven Development (TDD), while 14.3% developer’s felt the size of project not as a crucial factor in TDD.

24. Test Cases:

It has been reported as severe (Scale 5 on above histogram) by 52.4% of developer’s that the final task of Code Integration of several Test case modules affected the incorporation of TDD for large scale Industrial setting, while 28.6% reported less than severe (Normal) on final code integration.

25. Testing Process:
66.7% of developers has reported that the testing process became quite difficult due to low code integration of test cases in large scale industries, while 28.6% of developers reported that testing became quite easy due to continuous integration from the very initial stage of the project to the end.

26. Difficulty of TDD:

42.9% of developers answered that there implementing TDD in large scale industrial set-up as difficult, while 28.6% of developers felt the implementation of TDD in large scale industries to be drastic.

27. Size of team:
52.4% of developer’s responded that there TDD development team in their company comprised of 100-150 developer’s in Large scale setup, while 38.1% of developers has responded that their development team has a developers of around 50-100, while 9.5% TDD developers worked with team of varying size like 30, 40 developers.

28. Problems encountered by TDD developers in Large Scale Industries:

Above we have shown the strengths or challenges that are identified by TDD developers when using in large scale industries. Now in the below table we have discussed how much or what percentage of responses do support whether each factor is affected based on several measuring parameters like easy, difficult, in-
Chapter 5. Validation of Survey Questionnaire results and Interviews

Figure 5.25: Affect on testing process in TDD in Large Scale Industries

Figure 5.26: Affect of TDD in large scale industries

creases, decreases, no change/ no impact.
Chapter 5. Validation of Survey Questionnaire results and Interviews

Figure 5.27: Size of team

<table>
<thead>
<tr>
<th>Factor</th>
<th>Percentage(%) of Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>16.7% Increase, 70.8% Decrease, 12.5% No change observed</td>
</tr>
<tr>
<td>Code Maintenance</td>
<td>83.3% Easy, 16.7% Difficult</td>
</tr>
<tr>
<td>Geographical Locations</td>
<td>58.3% Affected, 29.2% Not affected, 12.5% Negligible</td>
</tr>
<tr>
<td>(MTTF) Metric</td>
<td>25% Increase, 68.8% Decrease, 6.2% Not considered as evaluating factor</td>
</tr>
<tr>
<td>Coupling between Objects</td>
<td>19.1% Increase, 66% Decrease, 14.9% No impact</td>
</tr>
<tr>
<td>Project Completion Time</td>
<td>20.8% Increase, 70.8% Decrease, 8.3% No impact</td>
</tr>
<tr>
<td>Code Integration</td>
<td>70.2% Easy, 14.9% Difficult, 14.9% No impact</td>
</tr>
<tr>
<td>Testing</td>
<td>47.9% Easy, 52.1% Difficult</td>
</tr>
<tr>
<td>Addition of New Functionalities</td>
<td>76.6% Easy, 14.9% Difficult</td>
</tr>
<tr>
<td>Influence of External Factors</td>
<td>66.7% Low, 18.8% little influence, 14.6% Highly influenced</td>
</tr>
<tr>
<td>Software Quality</td>
<td>31.9% Increase quality, 34% Increase effectiveness, 25.5% Increase efficiency, 8.5% Increase productivity of software</td>
</tr>
<tr>
<td>Initial Transition Problems</td>
<td>58.3% Increase, 35.4% Decrease</td>
</tr>
<tr>
<td>Development Speed</td>
<td>38.3% Good, 34% Very good, 23.4% Average</td>
</tr>
<tr>
<td>Software Reliability</td>
<td>73.9% Increase, 13% Decrease 10.9% No impact</td>
</tr>
<tr>
<td>Defect Density</td>
<td>17.2% Increase, 82.8% Decrease</td>
</tr>
<tr>
<td>Productivity</td>
<td>70.2% Increase, 17% Decrease, 12.8% No impact</td>
</tr>
<tr>
<td>Size of Project</td>
<td>76.2% Crucial, 14.3% Not crucial, 9.5% No impact</td>
</tr>
</tbody>
</table>

Table 5.1: Factors impacting the TDD in large scale industries from responders point of view
5.2 Semi-Structured Interviews

We have conducted semi-structured interviews of experienced software developers/practitioners in the field of Test Driven Development (TDD) to acquire the challenges and benefits they have observed in course of time while implementing TDD in large scale industries. The interviews were conducted through Skype and an average interview went on for 25-30 minutes.

5.2.1 Interview Procedure

- Selection of candidates for interview.

We have selected senior software developers from Large Scale companies like Microsoft, Tata Consultancy Services (TCS), Capgemini who have previously and currently working with Test Driven Development (TDD). Due to their busy schedule, we have to limit the session of interview to only 25-30 minutes.

- List of Questions to be communicated during the course of interview.

  - The list of questions to be asked are being prepared before the start of interview session, so there would be ample time to communicate with the interviewee rather than focusing on framing next question during the on-going interview.
Chapter 5. Validation of Survey Questionnaire results and Interviews

• Channel of Interview.
  – Due to the geographical locations, and the time constraints, we have interviewed the software developers via Skype, which kept the interview simple and quick.

5.3 Responses from Interviews

Since the main motive behind the interview is to find out the challenges and advantages of implementing Test Driven Development (TDD) through interviewee working experience. Since the interview was held for an average duration of 25-30 minutes, only brief outline of the interviewee perspective is presented below in the individual response’s. The responses below are written from the interviewer perspective after understanding the perspective of Interviewee.

5.3.1 Response of Interviewee 01

  • Name of Interviewee Organization: Microsoft
  • Type of Company: Multi-National Technology Company
  • Focused sector’s of Company: IT Services, Health-care, Research and Development
  • Number of Employees: 120,849
  • Current Role: Senior Agile Software Developer
  • Number of years of Experience with TDD: +05 years

It has been found that Test Driven Development (TDD) from Interviewee 1 that the ideology of Test-cases the foundation’s of TDD, make it easier to implement. When working with TDD, each developer of Interviewee team is concerned only with the objective implementing their assigned test-cases, and refactoring them and making the bug free. Since user don’t need to know the entire big picture of the project, it kept the developers less influenced from external factors. It has been also found that the productivity, code quality of the code are improved significantly. While regarding the challenges of implementing TDD in Large Scale Industries, the Interviewee has responded that though the TDD provided clear developmental goals to individual developers, the final software-product delivery became difficult due to integration of several test-cases written by several developers using various developmental tools. Also it has been found from the response of the interviewee that the testing process became quite difficult. The interviewee has also suggested certain modification like unit test cases needs to be designed perfectly at the initial stage, so that the percentage of failed test cases will be regulated to a very low extent.
5.3.2 Response of Interviewee 02

- Name of Interviewee Organization: Capgemini
- Type of Company: Multi-National Information Technology consulting corporation
- Focused sector's of Company: IT Services, IT Consulting
- Number of Employees: 190,000
- Current Role: Software Testing Engineer
- Number of years of Experience with TDD: 04 years

From the perspective of Interviewee, it has been found that implementing TDD in large scale industries like Interviewee’s company Capgemini, has improved the many aspects of software developing. The benefits/advantages of TDD mentioned by Interviewee are that due to Test-First principle of TDD helped the developers with working continuously working on code, refactoring the code. Due to the shorter software development life cycle, the overall project completion time decreased, and also development speed has improved significantly. The Interviewee also responded that due to continuous code integration of test cases from several developer’s made the testing process quite simple when working with TDD in large scale. When asked about the change in different metrics, it has been found a decrease in Mean time to fix (MTTF) metric, Coupling between Objects (CBO) and Cyclomatic complexity. Regarding the modifications, the response we got from interviewee is that developers practicing TDD should limit their geographical separation to a lower extent, so that there will be a better understanding between developers.

5.3.3 Response of Interviewee 03

- Name of Interviewee Organization: TATA Consultancy Services
- Type of Company: Information technology consulting company
- Number of Employees: 371,519
- Current Role: Junior Project Manager
- Number of years of Experience with TDD: +4 years

From the perspective of Interviewee, it has been found that implementing TDD in Large Scale industries has both gain and losses. The gain/benefits reported by Interviewee are implementing TDD in large Scale Industries have significantly
improved the productivity, improved quality of the code, improved code maintenance. The interviewee has specifically pointed out that during their working with TDD, their team developers has faced problems due to different geographical localities of the rest of the developers of the team. It has also been reported that there is an overall improvement in certain factors like increased development speed, decreases total project completion times. The response regarding the modifications are that, the developers need to re-factor their code in a much simpler and easy processed way, so that revision of test-cases in future will not be a burden.

5.3.4 Response of Interviewee 04

- Name of Interviewee Organization: Microsoft
- Type of Company: Multi-National Technology Company
- Focused sector’s of Company: IT Services, Health-care, Research and Development
- Number of Employees: 120,849
- Current Role: Junior Software Analyst
- Number of years of Experience with TDD: +02 years

It has been found that with TDD, the developers sole purpose is to focus on his/her test-cases and refactoring the code. Due to shorter development cycle of test-cases in TDD, the developer can re-work the entire test-case if in future there is a necessity for change. This less re-work and more efficiency, quality of the final code are the major benefits of TDD in Large Scale Industries. The challenges pointed out by interviewee are that the different level of knowledge regarding the implementation details of TDD in large scale projects are limited, this concerned the gradually increased the overall project completion time. The modification acquired from this interviewee is that TDD should be limited to projects only unit test-cases are given a much priority.

5.3.5 Response of Interviewee 05

- Name of Interviewee Organization: TATA Consultancy Services
- Type of Company: Information technology consulting company
- Number of Employees: 371,519
- Current Role: Senior Software Developer
Chapter 5. Validation of Survey Questionnaire results and Interviews

- Number of years of Experience with TDD: +4 years

In the perspective of interviewee it has been found that implementing TDD in Large Scale Industries requires various factors to be taken into concern. While some factors like Knowledge of TDD, writing test-cases can be mastered in due time, but certain factors like testing the TDD software especially in large scale requires more working effort than that of the traditional software development methodologies. The interviewee has not suggested or felt that any modification to their current TDD practice in their internal organization.

We have put classified each factor based on corresponding responses from participants using the measuring parameters which is either the impact is increases, decreases or easy, difficult when using the TDD in large scale industries are shown in below table 5.2 . Modifications that the interview responses believe might have impact on the large scale industries are classified based on the number of participants supporting the factor and not mentioned any view on that factor are shown in below table 5.3.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Percentage(%) of Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>20% Increase, 80% Decrease</td>
</tr>
<tr>
<td>Mean Time To Fix (MTTF)</td>
<td>20% Increase, 80% Decrease</td>
</tr>
<tr>
<td>Coupling Between Objects</td>
<td>100% Decrease</td>
</tr>
<tr>
<td>Project Completion Time</td>
<td>20% Increase, 80% Decrease</td>
</tr>
<tr>
<td>Code Integration</td>
<td>60% Increase, 40% Decrease</td>
</tr>
<tr>
<td>Testing</td>
<td>60% Easy, 40% Difficult</td>
</tr>
<tr>
<td>Addition of New Functionalities</td>
<td>100% Easy</td>
</tr>
<tr>
<td>Initial Transition Problems</td>
<td>80% Increase, 20% Decrease</td>
</tr>
<tr>
<td>Influence of External Factors</td>
<td>80% Low, 20% High</td>
</tr>
<tr>
<td>Software Quality</td>
<td>100% Increase</td>
</tr>
<tr>
<td>Development Time</td>
<td>20% Increase, 80% Decrease</td>
</tr>
<tr>
<td>Software Reliability</td>
<td>60% Increase, 40% Decrease</td>
</tr>
<tr>
<td>Defect Density</td>
<td>100% Decrease</td>
</tr>
<tr>
<td>Affect on Productivity</td>
<td>80% Increase, 20% Decrease</td>
</tr>
<tr>
<td>Size Of Project</td>
<td>80% Crucial, 20% No impact</td>
</tr>
</tbody>
</table>

Table 5.2: Factors impacting the TDD in large scale industries from interviewee’s point of view
Chapter 5. Validation of Survey Questionnaire results and Interviews

<table>
<thead>
<tr>
<th>Factor</th>
<th>Frequency of Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Break down the nature of experiments being created, through measuring and monitoring measurements.</td>
<td>4 Supporting, 1 Not Mentioned</td>
</tr>
<tr>
<td>Geographical distance</td>
<td>5 Supporting</td>
</tr>
<tr>
<td>Refactoring of unit test cases</td>
<td>5 Supporting</td>
</tr>
<tr>
<td>Modification of TDD such that generating less test cases for functional requirements</td>
<td>3 Supporting, 2 Not Mentioned</td>
</tr>
<tr>
<td>Refactoring strategies like shortening, streamlining and giving identifier names can be helpful.</td>
<td>3 Supporting, 2 Not Mentioned</td>
</tr>
<tr>
<td>New functionality implementation with low cost</td>
<td>2 Supporting, 3 Not Mentioned</td>
</tr>
<tr>
<td>Finishing the test is essential objective of TDD, not to focus on untested areas.</td>
<td>2 Supporting, 3 Not Mentioned</td>
</tr>
</tbody>
</table>

Table 5.3: Factors, frequency of responses for modifications from interviewee’s point of view
Chapter 6

Analysis from Literature Review, Survey Questionnaire and Interviews

6.1 Insights from Literature Review and Survey Questionnaire

Every Software development methodology has its own advantages/pro’s and disadvantages/con’s when coming to the part of the implementation, Making suitable choice of software development methodology plays a key role in every company.

Initially, the Benefits and challenges of implementing Test Driven Development (TDD) in Large Scale industries are found out using extensive Systematic Literature Review (SLR) using various existing techniques like Database searching using various search-strings, Snowballing. In case of Snowballing, the inclusion of exclusion of papers for final consideration also plays a very important role. Un-necessary initial selection of papers at exclusion at later stage is very time consuming and also tedious.

The main objective of this section is to find out the rate of correlation among the data observed through Systematic Literature Review (SLR) from existing literature on Test Driven Development (TDD), and data acquired through Online Survey Questionnaire. The main strengths, challenges and modifications observed through SLR are briefly stated by article number against the observed challenges, in Tables [4.3,4.4]. It has been also identified the affect of each parameter like increase/decrease on the factor mentioned in Table[4.3].
6.2 Analysis of Benefits and Challenges of TDD in Large Scale Industries from data acquired through Survey Questionnaire and Interviews

6.2.1 Respondent’s Qualifications

It has been chosen such the interviewee candidates, and survey respondents are experienced developer/practitioners with past, current experience in the domain of TDD. The respondent’s of the survey are categorized into sub-categories within their company, depending on their role. The respondents who answered this are such and are observed as:

- It has been found that 52.2% of the total respondent’s are TDD developers.
- While, 18.8% of them are Analyzers, and 16.7% are designers.
- The project managers comprised a 10.4% of total respondent’s who answered the survey.

6.2.2 Respondents experience with Test Driven Development (TDD)

It has been crucial in taking the experience of TDD developer as an important consideration. The more experience the developer possess, the developer possess more practical knowledge of implementing TDD in real time projects, and all this experience add us to benefit in identifying the benefits and challenges of Test Driven Development (TDD) when such respondents answered our survey questionnaire. Based on experience, the following categories are observed from Survey results:

- It has been identified that 67.2% of TDD developer’s possessed an experience of 1-2 years in working with TDD.
- And 12.5% of total respondents possessed an experience of 3-5 years in working with TDD.
- 19.1% of respondents have less than 1 year of experience in working with Test Driven Development (TDD).
- While 2.1% of total respondents have started working recently in the field of TDD.
6.2.3 Type of Projects Companies are solely focused on

- It has been found that 36.2% of the companies of the respondents focused on delivering IT services to other large scale and small scale companies.

- While, 14.9% of respondents companies focused on delivering software services to government sectors, and 12.8% of respondent’s working companies focused on Manufacturing sector.

6.2.4 Complexity

<table>
<thead>
<tr>
<th>Factor</th>
<th>Literature Review</th>
<th>Survey Response</th>
<th>Interview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>Decrease</td>
<td>Decrease</td>
<td>Decrease</td>
</tr>
</tbody>
</table>

Table 6.1: Affect on Complexity in TDD

It has been identified from Literature Review that there is an over-all decrease in the nature of complexity of project. From Figure 5.7 we can find from the responses of survey questionnaire that:

- 70.8% of total respondents have reported reduce in overall complexity of the project when working with TDD.

- While, 16.7% of total respondents have responded that an increase in nature of complexity.

- And 12.5% respondents have reported that not considerable amount is observed in change of complexity factor.

- 4 out of 5 interviewee respondent’s responded that they have observed an decrease in complexity while implementing projects with TDD in Large Scale industries.

6.2.5 Affect on Mean Time to Fix (MTTF) Metric

Mean Time to Fix (MTTF) metric value tells how quick a developer/practitioner can identify and fix the Bug/problem occurred in the path of software development. Hence, the lower the value of MTTF, better is the efficiency of the employed software development methodology. It has been identified through Literature Review, Survey responses, and Interviews that a lower MTTF metric has been identified when working with TDD.

This response can be very ideal when working with TDD, because in TDD each developer is responsible for implementing a particular feature through a single test case or multiple test cases, this makes bug identification much easier because is developer is responsible for making his/her test case from bug-free and clean.
Chapter 6. Analysis from Literature Review, Survey Questionnaire and Interviews

54

<table>
<thead>
<tr>
<th>Factor</th>
<th>Literature Review</th>
<th>Survey Response</th>
<th>Interview</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeanTimeToFix(MTTF)</td>
<td>Decrease</td>
<td>Decrease</td>
<td>Decrease</td>
</tr>
</tbody>
</table>

Table 6.2: Affect on MTTF metric in TDD

6.2.6 Affect on Coupling Between Objects (CBO)

IT has been found through Literature review, that a low CBO has been observed in TDD when compared to software development methodologies. The same aspect of CBO when asked in survey questionnaire the following response is been observed. The responses acquired through survey are presented briefly below:

<table>
<thead>
<tr>
<th>Factor</th>
<th>Literature Review</th>
<th>Survey Response</th>
<th>Interview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coupling between Objects (CBO)</td>
<td>Decrease</td>
<td>Decrease</td>
<td>Decrease</td>
</tr>
</tbody>
</table>

Table 6.3: Affect on CBO metric in TDD

- 66.6% respondents of total survey respondent’s reported that they have have observed a low CBO when working with Test Driven Development (TDD).
- While 19.1% of respondents reported an high CBO value, and 14.9% of respondent’s answered that they there has been no significant change in CBO metric.
- 5 out of 5 interviewee candidate’s has reported that a significant decrease is observed in the Coupling between Objects (CBO).

6.2.7 Affect on Project Completion Time

<table>
<thead>
<tr>
<th>Factor</th>
<th>Literature Review</th>
<th>Survey Response</th>
<th>Interview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Completion Time</td>
<td>Decrease</td>
<td>Decrease</td>
<td>Decrease</td>
</tr>
</tbody>
</table>

Table 6.4: Affect on Project Completion Time in TDD

It has been identified through literature review that 60% of study based on TDD implementation in Large Scale Industries has reported a decrease in project completion time, while 40% of the studies revealed that there is an overall increase in projection completion time in TDD. It has been found through the responses of the survey responses that:

- 70.8% of total respondents reported that there is an decrease or Faster project completion time when working with TDD in Large Scale Industries.
- While, 20.8% of respondents found that the overall project completion time increased.
Chapter 6. Analysis from Literature Review, Survey Questionnaire and Interviews

- And 8.3% found no change in project completion time factor.
- 4 out of 5 interviewee candidates responded that there is a decrease in overall project completion time.

6.2.8 Code Integration

The practice of continuous integration of each developer’s code to mainline software is referred to as Continuous Code Integration. This concept was first formalized by Grady Booch. Also Continuous code Integration infers that we fabricate our framework at whatever point it changes. We may run our test suite by then, which is a smart thought, yet we are not required to. In TDD due to continuous refactoring of the test cases for each functionality, there would be no need of continuous code integration, since all the test cases can be integrated at one final step. It has been found through Literature Review that many studies found that

<table>
<thead>
<tr>
<th>Factor</th>
<th>Literature Review</th>
<th>Survey Response</th>
<th>Interview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Integration</td>
<td>Decreased</td>
<td>Decreased</td>
<td>60% Increased:40% Decreased</td>
</tr>
</tbody>
</table>

Table 6.5: Affect of Code Integration in TDD

there is low overhead due to Code Integration in TDD in Large scale Industries. From the survey response Figure5.14 we can say that :

- 70.2% respondents reported similar results to that of SLR that there has been little overhead due to Code Integration.
- While 14.9% respondents reported an slightly decreased overhead due to Code Integration in TDD.
- And 14.9% reported that there has been no reduce in overhead due to code integration.
- 3 out of 5 interviewee respondents reported that the final code integration is difficult in the case of TDD.

6.2.9 Affect on Testing in TDD in Large Scale Industries

Testing is a crucial factor in determine/pin-pointing the problems efficiently. When coming to testing process in TDD process in TDD it is not just enough how efficient the testing process is, while it is also important to point out how easy the Testing process. It has been observed through Literature review that some studies pointed out that Testing is quite easy in TDD, while some studies reported the Testing process as quite cumbersome and time-taking.
Chapter 6. Analysis from Literature Review, Survey Questionnaire and Interviews

<table>
<thead>
<tr>
<th>Factor</th>
<th>Literature Review</th>
<th>Survey Response</th>
<th>Interview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing</td>
<td>50% Easy, 50% Difficult</td>
<td>Easy</td>
<td>Easy</td>
</tr>
</tbody>
</table>

Table 6.6: Affect of Testing in TDD

From the responses obtained from Survey Response, Figure 5.15 following results can be summarized:

- 52.1% of total respondent’s reported the testing process to be quite cumbersome.
- While 47.9% of total respondents reported that testing became quite easier when working with TDD.
- It has been reported through interview that the task of testing is easier in TDD.

6.2.10 Addition of New Functionalities

<table>
<thead>
<tr>
<th>Factor</th>
<th>Literature Review</th>
<th>Survey Response</th>
<th>Interview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition of new Functionalities</td>
<td>Easy</td>
<td>Easy</td>
<td>Easy</td>
</tr>
</tbody>
</table>

Table 6.7: Addition of new functionalities in TDD

The following responses are being observed from survey responses regarding addition of newer functionalities to existing code produced by TDD.

- 76.6% of total survey respondent’s reported that addition of newer functionalities to existing TDD code is easier.
- While 14.9% respondent’s reported that there was an increased difficulty in addition of newer functionalities to existing TDD code.

6.2.11 Affect on Code Maintenance

<table>
<thead>
<tr>
<th>Factor</th>
<th>Literature Review</th>
<th>Survey Response</th>
<th>Interview</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeMaintenance</td>
<td>50% Easy, 50% Difficult</td>
<td>Easy</td>
<td>Easy</td>
</tr>
</tbody>
</table>

Table 6.8: Affect on Code Maintenance factor in TDD

As the size of project increases, naturally the Number of Lines of Code (NLOC) will also increase. An increase in number of lines of code, indicates an increase in maintaining the code. It has been found through extensive Systematic Literature Review (SLR) that both in some cases the task of maintaining code is difficult, while some case reported an easier code maintenance. It has been found through Survey that:
Chapter 6. Analysis from Literature Review, Survey Questionnaire and Interviews

- 83.3% of total respondents felt that the task of maintaining code in Large Scale TDD development project’s are easy.

- While the rest 16.7% of respondents felt an increased difficulty in maintaining code.

- It has been noted that in the interview session with TDD developer that every developer felt an decreased difficulty in maintaining code when working with large scale TDD projects.

6.2.12 Analysis of various other Parameter’s

In summary, the affect on various other parameters like Influence of external factors, Software quality, Development Speed, Software reliability, Defect density are compared with the results obtained from Literature review to that of the data collected through Survey questionaire.

<table>
<thead>
<tr>
<th>S.No</th>
<th>Factor</th>
<th>Literature Review</th>
<th>Survey Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Influence of External Factors</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>2</td>
<td>Software Quality</td>
<td>Increase</td>
<td>Increase</td>
</tr>
<tr>
<td>3</td>
<td>Development Speed</td>
<td>60% Decreased : 40% Increased</td>
<td>Good</td>
</tr>
<tr>
<td>4</td>
<td>Software Reliability</td>
<td>Increase</td>
<td>Increase</td>
</tr>
<tr>
<td>5</td>
<td>Defect Density</td>
<td>Decreased</td>
<td>Decreased</td>
</tr>
<tr>
<td>6</td>
<td>Affect on Productivity</td>
<td>80% Increase : 20% Decrease</td>
<td>Increase</td>
</tr>
<tr>
<td>7</td>
<td>Size of Project</td>
<td>Crucial</td>
<td>Crucial</td>
</tr>
</tbody>
</table>

Table 6.9: Mapping of SLR and Survey

<table>
<thead>
<tr>
<th>S.No</th>
<th>Factor</th>
<th>Interview Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Influence of External Factors</td>
<td>Low</td>
</tr>
<tr>
<td>2</td>
<td>Software Quality</td>
<td>Increase</td>
</tr>
<tr>
<td>3</td>
<td>Development Speed</td>
<td>80% Decreased : 20% Increased</td>
</tr>
<tr>
<td>4</td>
<td>Software Reliability</td>
<td>Increase</td>
</tr>
<tr>
<td>5</td>
<td>Defect Density</td>
<td>Decreased</td>
</tr>
<tr>
<td>6</td>
<td>Affect on Productivity</td>
<td>Increased</td>
</tr>
<tr>
<td>7</td>
<td>Size of Project</td>
<td>Crucial</td>
</tr>
</tbody>
</table>

Table 6.10: Interview Response

- It has been identified from both concept matrix derived from Systematic Literature Review (SLR) Tables[4.3,4.4,4.5] and the results acquired from survey response Figure 5.17 that 66.7% of total respondents answered that the influence of external factors on working TDD developer’s is very low,
Chapter 6. Analysis from Literature Review, Survey Questionnaire and Interviews

since because in TDD each developer is only concerned with his/her test cases and refactoring their test cases. While 18.8% of developers felt that there is a little influence due to external factors. As a whole, we can say that TDD developers are less concerned with influence from external qualities.

- From responses of Survey questionnaire Figure 5.18, we can say that 25.5% of respondents answered that an increased efficiency of Software is observed when working with TDD, while 31.9% of total respondents felt that there is an substantial improvement in quality of software in TDD. And 34% of respondent’s felt an increased effectiveness of software.

- It has been identified through literature review that Defect density has been decreased in TDD. It has been observed from survey responses that similar sign has been observed, i.e., 67.4% of respondents responded that there is a significant decrease in defect density. As an overall, it is a better indication that an decrease of Defect Density in TDD when working with Large Scale Industries.

- 70.2% of total respondent’s felt an increase in Productivity when working with TDD, which is coinciding with the result acquired from Literature Review. While 17.8% of respondents felt that there is a decrease in productivity while working with TDD. And 12.8% of people felt no impact due to productivity factor.

- 76.2% of total respondents felt that Size of project plays a crucial role in working with TDD in large scale industry. While 14.3% felt that size is not a crucial factor.

6.3 Analysis of Challenges observed from Survey Responses

6.3.1 Affect due to Final task of Code Integration

In Large Scale Industries working with Test Driven Development(TDD), the final task of Code Integration of several Test case modules will pose a serious problem. The responses collected from survey regarding this problem are that:

- It has been identified that 52.4% of total respondents felt that there is an severe/adverse affect due to the final task of Code Integration of several Test case modules in large scale industrial set up.

- While 28.6% of total respondent’s felt that the task of final code integration to be normal.
6.3.2 Affect on Testing in Large Scale Industries

- 66.7% of total respondents felt that the testing process became quite difficult due to low code integration of test cases in large scale industries using TDD.

- While 28.6% of respondents felt that testing became quite easy due to continuous code integration from the very initial stage of the project to the end.

As a result of conclusion regarding the testing of TDD projects in Large Scale industries, we can say that testing is quite cumbersome and hectic.

Also the various problems faced by TDD developers working in Large Scale industries are pointed out in Figure 5.28.

6.3.3 Affect on Initial Transition Problems

<table>
<thead>
<tr>
<th>Factor</th>
<th>Literature Review</th>
<th>Survey Response</th>
<th>Interview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Transition Problem's</td>
<td>67% : Increased 33% Decreased</td>
<td>Increased</td>
<td>Increased</td>
</tr>
</tbody>
</table>

Table 6.11: Initial Transition problems in TDD

Initial transition to any new Software Development Methodology from the existing implementations create many problems. The same question when asked to the current TDD developers/practitioners the following responses are observed:

- 58.3% of total respondents reported that they have faced problem during their Initial transition to TDD from their past working methodology.

- While 35.4% of respondents reported that they transitioned smoothly without any problems.

- It has been found through interview that practically every Large Scale Industry developers faced initial transition problems from moving through their old Software Development methodologies to their current implementing TDD practice.

As a result of analyzing both the info from survey questionnaire and Literature review, we can say that majority of developers faced problems during their initial transition to TDD.
Chapter 6. Analysis from Literature Review, Survey Questionnaire and Interviews

6.3.4 Difficulty in geographical locations of various developer’s in TDD

It has been identified in Large Scale Industries working with Large projects face problems due to collaboration keeps become difficult due to different geographical locations, different time-zones of the TDD practitioner’s across the globe. This has been one of the major hurdle when working with large scale projects with TDD in Large Scale Industries. It has been identified from survey responses that:

<table>
<thead>
<tr>
<th>Factor</th>
<th>Literature Review</th>
<th>Survey Response</th>
<th>Interview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impact of geographical location</td>
<td>Affected</td>
<td>Affected</td>
<td>60% Affected</td>
</tr>
</tbody>
</table>

Table 6.12: Affect due to different geographical location of developers in TDD

- 58.3% of developers working with large scale TDD projects in their respective companies has faced problems due to different geographical localities of their co-developers among their developers.
- While 29.2% of responded felt that they felt no difficult due to geographical locations of their co-developers.
- 3 out of 2 interviewee respondents felt difficulty that while working with large scale industrial projects with TDD, due to the different geographical locations of co-developers.

As a result we can say together from Literature Review and survey responses that majority of faced this different geographical locality factor as a major hurdle.

6.4 Overall Analysis of Modifications acquired through Literature Review and Interviewee Responses

- It has been found that through Literature Review and responses of Interviewee candidates, majority of authors have suggested that when a team is practicing TDD oriented towards a continuous modification of final product, then the team could produce best results when the team is not geographically separated.
- The another major modification suggested is that the unit test-cases should be designed carefully, and the unit test-cases should be re-factored frequently.
- The productivity with in the TDD development team can be gradually improved if all the team members have an equal understanding and knowledge of TDD.
Chapter 7

Discussion and Limitations

7.1 Discussions

The goal of our thesis is to identifying the benefits, and challenges faced by Implementing TDD practice in Large Scale Industries through Systematic Literature Review (SLR) research methodology. The challenges observed through literature review matches with the data collected through Survey Questionnaire and Interviews. We have selected the Systematic Literature Review (SLR) as it offers a background in a domain so that newer research activities can be performed in a systematic manner. We have selected two types of searching database and snowballing.

7.1.1 Answering the Research Questions

The research questions pertaining to this thesis

1. What are the strengths of TDD, is TDD capable to overcome the challenges of agile in large scale environment?

Findings from Systematic Literature Review (SLR):

The strengths of TDD in large Scale Industries that were noticed when the SLR is performed on the current set of research articles are Unparalleled Test Scope and Streamlined Codebase, Specialized Changes in Infrastructure, Usefulness of Refactoring Code, Reduces External Interruptions, Shorter development life cycles, Executable Documentation and Viable, Adaptable, Effortlessly Extensible.

Some of the critical factors that are observed through Systematic Literature Review (SLR) where TDD performs well compared to the Traditional software development methodologies, and over some Agile practices: Understanding of basic requirements, Efficiency and Flexibility, Mean time to Fix (MTTF) Metric, Total lines of composed code, Final Software Quality and Impact on Coupling
Chapter 7. Discussion and Limitations

Between Objects (CBO).

An important observation made after reviewing all studies based on Test Driven Development are, code size which is considered as a major software metric in analysing any software development project, TDD developers tend to write smaller unit tests and test them before further incrementing the functionalities of the unit test.

Some challenges faced by the developers when implementing the TDD are Difficulty in maintaining code, Geographical difficulties, Productivity, Difficulty in Initial Transition, Code Integration.

Using the Concept Matrix practice from SLR, the challenges and benefits of Test Driven Development (TDD) an effect on Parameter (which in our case is challenges and success factors) was assessed for the selected articles to understand if it is increased, decreased, easy, difficulty for all the find outs are addressed in figure 4.3.

From the survey response questionnaire, we justify that the respondents were relevant to the study scope. 54.2% developers and they were from IT service sector. 62% of them were experienced between 1-5 years. Some projects these participants have worked were client based or internal development projects. The factors to be evaluated when working with TDD were project size, time limit, team and requirements.

- 58% of the respondents agree that Geographical Location where developers work is important.
- 70.8% of TDD developer’s felt that the nature of complexity. 68.8% of TDD developers observed a low MTTF factor.
- 70.2% of TDD developers that code integration factor has been clearly increased in TDD.
- 52.1% of developers felt the testing process to be quite simple in Test Driven development (TDD) methodology.
- 76.6% of developers felt that addition of new functionalities to the existing code in TDD was easier.
- 66.7% developers that influence of external factors on project development is very low in TDD.
- 31.9% of software developers, while increase in efficiency of software has been observed by 25.5% of developers, while 34% developers have identified observed an increased effectiveness of software.
An increase in software reliability factor has been reported by 73.9% of developers.

67.4% of TDD developers who answered the question has felt and decrease of Defect Density factor. An increase in productivity of project implemented by TDD has been 70.2% reported by of developers.

52.4% of developer’s that the final task of Code Integration of several Test case modules affected the incorporation of TDD for large scale Industrial setting.

66.7% of developers has reported that the testing process became quite difficult due to low code integration of test cases in large scale industries.

Problems encountered by developers were linking test case modules of several developers, negative test cases. Unable to get the overall idea of the project, different developers use different tools to write the test cases. Test case integration is a very big concern. Communication among the developers also has some impact.

We have conducted semi-structured interviews of experienced software developer’s in the field of Test Driven Development (TDD). Some important views from developer’s perspective are addressed as follows:

• When working with TDD, each developer of Interviewee team is concerned only with the objective implementing their assigned test-cases, and refactoring them and making the bug free. The productivity, code quality of the code are improved significantly and also TDD provided clear developmental goals to individual developers.

• Test-First principle of TDD helped the developers with working continuously working on code, refactoring the code.

• When asked about the change in different metrics, it has been found a decrease in Mean time to fix (MTTF) metric, Coupling between Objects (CBO).

• There is an overall improvement in certain factors like increased development speed, decreases total project completion times.

• The interviewee has specifically pointed out that during their working with TDD, their team developers has faced problems due to different geographical localities of the rest of the developers of the team.

• The developer can re-work the entire test-case if in future there is a necessity for change. This less re-work and more efficiency, quality of the final code are the major benefits of TDD in Large Scale Industries.
• Writing test-cases can be mastered in due time, but certain factors like testing the TDD software especially in large scale requires more working effort than that of the traditional software development methodologies.

2. What are the modifications/enhancements are necessary to help use the TDD to cope up with these challenges in large scale environment?

Modifications suggested by the interview respondents are addressed below:

• Some modifications observed from Literature are that product developers practicing TDD must break down the Complex functionality or feature, through measuring and monitoring measurements, for example, code scope and imperfection proportion, with a specific end goal to keep TDD proficient. Refactoring of unit test-cases for better product quality. Refactoring procedure may comprise of shortening strategies, streamlining control structures, and giving illustrative identifier names, variable, and class names.

• Unit test cases needs to be designed perfectly at the initial stage, so that the percentage of failed test cases will be regulated to a very low extent.

• TDD should limit their geographical separation to a lower extent, so that there will be a better understanding between developers.

• The developers need to re-factor their code in a much simpler and easy processed way, so that revision of test-cases in future will not be a burden.

• TDD should be limited to projects only unit test-cases are given a much priority.

The current research topic is discussed in the previous studies. the papers are reviews, that are emphasized on Test driven development. The results from [38] article show that there is increase in development time, TDD knowledge and practical experience, insufficient design. Insufficient developer testing skills. In our findings as well after observations from literature and the interviews feedback any small amount of up-front design, also refactoring can help in keeping the architecture from disruption from initial plan.

The developers need to have some experience in testing to be able to write efficient and effective automated test cases that are run during testing process. Some important aspects to be kept in mind when working with TDD is that it is important to have TDD based design techniques, testers have prior experience and knowledge to be able to adopt to TDD process [38]. From our study we understand that there must be a synergy between development and testing effort in a more productive manner and TDD perspective of testing should be understood
and deciphered in a proper way to get a clear understanding of how we can take advantage of TDD practice.

Munir et al [39] mentioned that there are 8 factors that affect the TDD like Time effort, internal quality of the code, conformance, robustness, size, productivity, developer opinion. External quality has positive or is in favor of TDD from the study. However, the performance productivity is negatively for TDD and also time and effort that needs to be put is very high when following TDD process. In our research study as well we identified that the time taken has significant amount of difference when it comes to applying traditional vs TDD process. The quality of a system is detected by the number of pre and post release defects, TDD is associated with tasks that are easy to comprehend, external quality can be detected based on test cases passed, defect density, defects per test, change density, preventive change percentage, effort required to fix defects. In order to assess the internal quality of the system TDD trials can be based on object-oriented metrics like depth of inheritance tree, code size per feature, code density, cyclomatic complexity [40]. In our study research as well we have involved these metrics or factors based on which the TDD trials and its outcomes to measure the quality of a system can be deduced are addressed. When it comes to productivity the Author support that in long term the TDD has positive effect on productivity his experiment results do support the increase in productivity. However, it is not true in the pilot study cases where some cases are against the TDD trial increase the productivity vote. We believed that there are lot of parameters and measures that are taken into account and different projects have different scope, time budget restraints and we still believe that from interview resist that TDD might have a positive impact in the quality and increaser the productivity as it requires steep learning curve [40]. When it comes to Test quality the TDD trial is often better than other approaches which is also supported by the arguments from the interview responses.

7.2 Limitations and Threats to validity

7.2.1 Limitations

Every research design has some unaddressed areas or limitations and in our study we would like to discuss from our view the limitations that are accounted while working with the reported findings.

- The Literature is mostly composed of challenges and strengths of Test Driven Development (TDD). For modifications that could help TDD to cope with the corresponding challenges are not addressed or given importance in the literature

- The study did not conduct survey for second research question as the liter-
nature is very low to understand the modifications that could help TDD to cope from current challenges, so instead we used the interviews to gather more relevant information in this case.

### 7.2.2 Threats to validity

For the empirical studies such as ours, there exists four major validity threats which we would like to discuss as part of our thesis project, these validity threats are namely Conclusion validity, construct validity, internal validity and external validity [41]. This below discussion also includes the mitigation strategies that are included for the study. Another attempt to reduce the validity threat we made use of data triangulation [26] since this thesis had made use of both qualitative and quantitative data collection source.

#### Internal validity

This threat is primarily concerned when irrelevant responses. One reason for this irrelevant responses is improper design of the survey instrument and survey design. The selection of participants also might show significant influence over the results.

- This threat regarding irrelevant responses are reduced by asking the questionnaire respondents whether they had any previous experience in the relevant field. As the authors used online questionnaire form to design the questionnaire they were able to use question logic. The question logic and clearly stating what it actually means allowed the respondents to answer the remaining questions if they specified that they had previous experience. The online questionnaire is designed such that if the participant had no previous experience in the TDD or industrial experience in the TDD the participants cannot be able to answer the remaining questions.

- The survey instrument used is online questionnaire. The authors have mitigated the design issues of the questionnaires’ by conducting pilots in group of 3 who had experience in TDD within student environment who were doing intern in industry. Based on feedback from the intern participants who have little relevant experience and also supervisor suggestions on the questionnaire had helped us to constructively satisfy and update for all the feedback accordingly after proper discussions within authors and also with supervisor and then hosted online.

- The selection of participants was mitigated by sending the questionnaire to organizations whose personal have experience in TDD. We initially gathered the list of organizations that are well established and who have good number of personal who would be involved in TDD and have experience in it are only
involved with in the online questionnaire loop and only such organization with TDD profile fit personal were sent the emails.

**External validity**

The threat is concerned when the results from the online questionnaire, interview the participants have a view and the authors have tried to generalize their view to fit the thesis context. The sampling method used for selection of interview participants is convenience sampling. since the authors used the convenience sampling the results could only be generalized in those as where the TDD teams and organizations that possess similar characteristics to the interview/s teams and organizations. As he questionaries’ is hosted online and mails are sent to participants, sample of interviews is heterogeneous with respect to the work experience and interviews role in the organization.

**Construct validity**

It is concerned about the study that has been conduct measures things that authors proclaimed. In this cases even though from the literature we have observed that there is abundant relevant literature that describes the challenges and the success factors for the TDD in large scale industries. There is no proper documentation or research in the TDD that specifically identifies the modifications that can be done to specific challenges vice versa. This study in our case helps to fill this gap, TDD should be limited to projects only unit test-cases are given a much priority. The interview structure or the interview questions are prepared in such a way that threat the participant is clear on what answer we want from them and the questions are clearly focusing on how these different challenges could be mitigated if they face them and how were they able to handle or resolve or come out of such challenging situations, did they use TDD how it helped is the mitigation relevant for that challenge can it be scaled or generalized such that for this challenge this mitigation could be useful to apply.

**Conclusion validity**

This threat arises when there is possibility of drawing incorrect conclusions about the observations that have been made aligning to the thesis goals. This issues is reduced by documenting the interviews and recording the interviews from the observations that are made. The authors used only one of the descriptive statistics method and utilize the statistics and percentage to generalize how many of the people from same background feel same about the question. The interview results aware carefully analyzed and we tried to reduce the miss conceptions and made sure that the participants view was clearly conveyed.
Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this study the authors have tried to explore into Test Driven Development to understand what are the Strengths of TDD and is TDD capable to overcome the challenges of agile in large scale environment. From the Systematic Literature review that we have conducted, critical factors that are observed through Systematic Literature Review (SLR) where TDD performs well compared to the Traditional software development methodologies, and over some Agile practices: Understanding of basic requirements, Efficiency and Flexibility, Mean time to Fix (MTTF) Metric, Total lines of composed code, Final Software Quality and Impact on Coupling Between Objects (CBO). TDD developers tend to write smaller unit tests and test them before further incrementing the functionalities of the unit test. Some challenges faced by the developers when implementing the TDD are Difficulty in maintaining code, Geographical difficulties, Productivity, Difficulty in Initial Transition, Code Integration.

We have conducted surveys and interviews to understand the people’s view in industry about the findings that we have made in the systematic literature review. Interestingly some notable responses from the survey results were 58% of the respondents agree that Geographical Location where developers work is important. 52.4% of developer’s that the final task of Code Integration of several Test case modules affected the incorporation of TDD for large scale Industrial setting. 31.9% of software developers, while increase in efficiency of software has been observed by 25.5% of developers, while 34% developers have identified observed an increased effectiveness of software.

An increase in software reliability factor has been reported by 73.9% of developers. 67.4% TDD developers who answered the question has felt and decrease of Defect Density factor. An increase in productivity of project implemented by TDD has been reported by 70.2% of developers.

We have conducted semi-structured interviews of experienced software de-
Chapter 8. Conclusions and Future Work

Developer’s in the field of Test Driven Development (TDD) some important views from interview participant perspective is that Test-First principle of TDD helped the developers with working continuously working on code, refactoring the code. The developer can re-work the entire test-case if in future there is a necessity for change. This less re-work and more efficiency, quality of the final code are the major benefits of TDD in Large Scale Industries. The interviewee has specifically pointed out that during their working with TDD, their team developers has faced problems due to different geographical localities of the rest of the developers of the team.

The Modifications that could help to overcome the challenges of using TDD in large scale industry from Interview results are Unit test cases needs to be designed perfectly at the initial stage, so that the percentage of failed test cases will be regulated to a very low extent. The developers need to re-factor their code in a much simpler and easy processed way, so that revision of test-cases in future will not be a burden. TDD should limit their geographical separation to a lower extent, so that there will be a better understanding between developers. TDD should be limited to projects only unit test-cases are given a much priority.

8.2 Future Work

When any product is concerned the main success story is always who fast is the product delivered without compromise in the quality standards. Using TDD in large scale industries has both strengths and challenges as we have discussed in our study. The literature relevant to modifications that could enhance the challenges that TDD can face in large scale industry scenario is pretty low and we tried our best to add few more modifications that could help enhance the TDD in large scale industries appliance. It is important that this research study continues and more modifications are discovered from various real time experiences and documented so that it could be useful for next generation to utilize them and rediscover or perhaps identify new findings to reduce the challenges and improve the percentage of usage for TDD in large scale industries.
Bibliography


Appendices
Appendix A

Survey Questionnaire

Survey Invitation:

Hello,

We (Ramakrishna and Pavan), are students of software engineering at Blekinge institute of technology, Karlskrona, Sweden. Currently we are working on our thesis (Investigating the Application of TDD Practice in Large-Scale Industries) to complete masters degree. We are conducting this survey to collect data for my thesis from people who have been working in Test driven development.

This survey is very important for my research. It will be take about 15-20 minutes. Thanks in advance for taking your time and please do forward this survey to your friends and colleagues.
Please go through the google form below to answer this survey.
We appreciate if you could complete as soon as possible. Your data will be confidential. If you have any further questions about these survey, please contact us at: dpkvarma9@gmail.com, rakhi743@gmail.com

Regards
Pavan And Rama krishna
A Short Survey on Identifying
Strengths & Challenges of TDD
Framework in Large Scale Software
Industries

This a small survey questionnaire Regarding the collection of data from Large Scale Software
industries practicing different software development methodologies within Their companies. The
Acquired Data will be used for the study of the Master Thesis titled "Investigating the Application of
Test Driven Development (TDD) in Large Scale Industries". The thesis is supervised by Nasir
Mehmood Minhas, Blekinge Institute of Technology (BTH), Sweden.

* Required

Email address *

Your email

Which best describes your current position?

- [ ] Developer
- [ ] Designer
- [ ] Analyser
- [ ] Project Manager
- [ ] Other:  


Appendix A. Survey Questionnaire

Which of the below sector’s is your company primarily focused on?

☐ Government
☐ IT Services
☐ Manufacturing
☐ Retail

How many years of experience do you possess in working with TDD methodologies?

☐ 1 - 2 Years
☐ 3 - 5 Years
☐ Less than 1 Year
☐ Started working recently

Since how many years has TDD been in use by your company?

☐ 1 - 2 Years
☐ 3 - 5 Years
☐ Less than 1 Year
☐ TDD limited only to certain projects
Choose the type of project that is most common in your company

- Internal development projects
- Development Projects for a client
- Adaptation of Commercial Software
- Other: 

Could you express which factor must be evaluated first in when working with TDD?

- Project Size
- Project time limit
- Development Team
- Requirements
- Other: 

How could you express the improvement in nature of the complexity of implementing a project in TDD environment?

- Reduced Complexity
- Increased Complexity
- No change observed
- Other: 

Appendix A. Survey Questionnaire

At times developers across different geographical locations has a need to collaborate for a project, does you ever felt this affected the workflow of project implemented using TDD?

- [ ] Affected
- [ ] Not Affected
- [ ] Negligible
- [ ] Other: 

Maintaining code is extremely useful for future references. How does the code maintenance factor influenced by TDD practices?

- [ ] Difficult to maintain
- [ ] Easier to maintain
- [ ] Other: 

How is the Mean time to Fix (MTTF) metric affected in TDD during your experience with TDD?

- [ ] MTTF is low in TDD compared to Traditional methodologies
- [ ] MTTF metric value increased in TDD compared to other methodologies
- [ ] MTTF metric not considered as an Evaluating factor
- [ ] Other: 

Appendix A. Survey Questionnaire

How was the Coupling Between Objects (CBO) factor affected in TDD?

- High CBO observed in TDD
- Low CBO observed in TDD
- No change in CBO
- Metric not considered

How was the project completion time factor impacted by TDD?

- Faster project completion time
- Low project completion time
- Project completion time not affected
- Other: __________________

Does your company faced any initial transition problems changing to TDD?

- Yes
- No
- Maybe
Appendix A. Survey Questionnaire

How was the Code Integration affected by using TDD practices?

- Improved Code Integration
- Significantly low code integration
- Not improved
- Other: ______________________

How was the testing process affected by making use of TDD?

- Testing became quite simple
- Cumbersome
- Other: ______________________

Does adding new functionalities to existing code is easier in TDD?

- Yes
- No
- Maybe
Appendix A. Survey Questionnaire

Upto which extent external factors influenced the project in case of TDD?

- Less
- Highly influenced
- Not influenced
- Other: __________________________

Choose the rate of improvement of Software Quality achieved by implementing Test Driven SDM practice?

- Increased quality of software
- Increased efficiency of Software
- Increased effectiveness of software
- Increased productivity of Software

Does the development speed increased by adopting TDD?

- Little Bit
- Average
- Good
- Very Good
Could you express how the software reliability factor has been affected by using TDD Methodology your company is practising?

- Increased Reliability
- Decreased Reliability
- No Impact
- Other: ____________

What is affect on Defect Density while using Test Driven Development as compared to traditional techniques?

- Decrease
- Decrease Slightly
- Increase
- Increase Slightly

Choose which of these software metrics have you have used personally?

- Order Growth ("Big O")
- Cyclomatic Complexity ("McCabe")
- Bugs per Line of Code
- Function Point Analysis
- Other: ____________
What is affect on productivity while using Test Driven Development (TDD)?

☐ No impact

☐ Decrease

☐ Increase

☐ Other: ____________________
Appendix A. Survey Questionnaire

Does the size of the project affect the development process when using TDD?

☐ Yes
☐ No
☐ Maybe

How did the final task of Code Integration of several Test case modules affect the incorporation of TDD for large scale Industrial setting?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Could you express the major difficulties faced by developers in your team working with TDD projects in Large scale projects?

Your answer

How did the testing process affected when working with TDD in large scale set up?

☐ Testing came quite difficult due to low code integration of test cases

☐ Testing because quite easy due to continuous code integration from start of the project

☐ Other:  
Appendix A. Survey Questionnaire

How many years of experience do you have in implementing TDD in large scale software projects?

- [ ] 1-2 Years
- [ ] 3-5 Years
- [ ] Less than 1 Year
- [ ] Other: 

How could you express the difficulty of TDD usage in large software development projects from your experience?

1 2 3 4 5
Low: [ ] [ ] [ ] [ ] [ ] High

In general what is the size of your TDD development team when working with large scale projects?

- [ ] 50-100
- [ ] 100-150
- [ ] Other: 

SUBMIT
Appendix B

Interview Questionnaire

Project: Investigating the Application of TDD practice in Large Scale Industries

Date:
Location:

Note: Thanks for your participation. I consider your input will be treasured to this studies and in supporting grow all of our professional exercises. Confidentiality of responses is assured. Approximate duration of interview is 25mins,five foremost questions.
1. Can you please specify your background in field of software industry?
   a. What is the name of your organization?
   b. Which type of sector your company solely focused on?
   c. How many number of current employees in your industry?
   d. Please specify your role in the company?
   e. Please specify your number of years of experience with TDD?

2. How would you explain the current usage of TDD in your organization?
   a. Are you developing more unit test-cases with incremental functionality from test-case to test-case?
   b. How could you explain the effect of TDD when some of your colleague is working from different geographical area?
   c. How could explain briefly the shift of your current TDD to your past software development methodologies?
   d. How could you comment on the question like do everyone practicing TDD requires same level of knowledge or the level of knowledge can differ?

3. How would you comment on the following factors when working with TDD?
   a. How would you comment on the Quality of the code?
   b. How would you comment on the major metrics like over-all productivity, speed of development, identifying the bugs, and also number of lines per code?
   c. How would you comment on the frequent refactoring of the code, the main objective behind the practicing of TDD?
   d. How would you comment on the factor that “TDD is better than the traditional software development methodologies”?

4. What are your suggestions to make TDD much better in improving the software development process?
   a. Do you think there should be some changes in designing, testing and refactoring the unit test-cases?
   b. How could you comment on identifying bad quality unit test-cases at early stage so the development process needs to focus on developing/refactoring the code at later point of time?

5. What are all the other factors with-in TDD that can be developed in the future practicing of TDD?