ANALYTICAL REVIEW OF AGILE PRACTICES IN PAKISTAN

A THESIS PRESENTED TO

FACULTY OF BASIC & APPLIED SCIENCES

DEPARTMENT OF COMPUTER SCIENCES & SOFTWARE ENGINEERING

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE

OF

MS IN SOFTWARE ENGINEERING

BY

MEHWISH ZULFIQAR

198-FBAS/MSSE/S08

SUPERVISED BY

DR. ABDUL RAUF

DEPARTMENT OF COMPUTER SCIENCE & SOFTWARE ENGINEERING
FACULTY OF BASIC & APPLIED SCIENCES
INTERNATIONAL ISLAMIC UNIVERSITY, H-10, ISLAMABAD
(JULY, 2012)

DEPARTMENT OF COMPUTER SCIENCE & SOFTWARE ENGINEERING
INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD

Date: _____

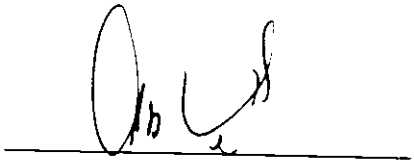
FINAL APPROVAL

This is to certify that we have read the thesis submitted by **Mehwish Zulfiqar,** Registration No. **198-FBAS/MSSE/S08.** It is our judgment that this thesis is of sufficient standard to warrant its acceptance by International Islamic University, Islamabad for the degree of **MS in Software Engineering.**

Committee:

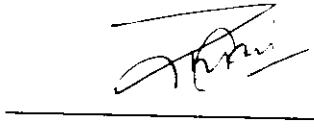
External Examiner:

Dr. Arshad Ali Shahid
Professor & HOD
Department of Computer Sciences
FAST-NUCES, H-10 Islamabad



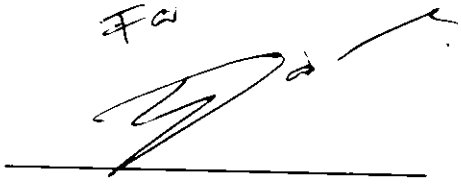
Internal Examiner:

Dr. Zunera Jalil
Assistant Professor
Department of Computer Science & Software Engineering
Faculty of Basic and Applied Sciences
International Islamic University, Islamabad



Supervisor:

Dr. Abdul Rauf
Assistant Professor
Department of Computer Science & Software Engineering
Faculty of Basic and Applied Sciences
International Islamic University, Islamabad



DEDICATION

To my family especially my **mom**

A dissertation Submitted To

Department of Computer Science & Software Engineering,

Faculty of Basic and Applied Sciences,

International Islamic University, Islamabad

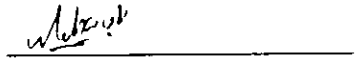
As a Partial Fulfillment of the Requirement for the Award of the

Degree of

MS in Software Engineering (MSSE)

DECLARATION

I hereby declare that this Thesis, **"Analytical Review of Agile Practices in Pakistan"** neither as a whole nor as a part has been copied out from any source. It is further declared that I have done this research with the accompanied report entirely on the basis of my personal efforts, under the proficient guidance of my teachers especially my supervisor *Dr.* Abdul Rauf. If any part of the system is proved to be copied out from any source or found to be reproduction of any project from any of the training institute or educational institutions, I shall stand by the consequences.



Mehwish Zulfiqar
198-FBAS/MSSE/S08

ACKNOWLEDGEMENT

I am obliged to **Allah Almighty** the Merciful, the Beneficent and the source of all Knowledge, for granting me the courage and knowledge to complete this Project.

There are a number of people without the support of whom this thesis might not have been completed, and to whom I am very grateful.

Dr. Abdul Rauf has been the ideal thesis supervisor. His kind guidance, wise suggestions, and enduring support helped me enormously to complete this research. I am grateful to him for his dedicated support and kind attitude throughout my work.

I would like to express my gratitude to my **parents** who have been very supporting and encouraging throughout. Most especially, my mother **Firdous Nasreen** and father **Zulfiqar Ali** are responsible for my intellectual curiosity and were always there for me in my hard times to pull me up. Without their help and erstwhile love this work might not have been completed. I am grateful to my **family** for being a source of great strength for me.

I am thankful to my friends for their advices and wishes, especially **Fareeha Qureshi** for her support at every stage of my work.

Mehwish Zulfiqar

198-FBAS/MSSE/S08

THESIS IN BRIEF

THESIS TITLE:	**"Analytical Review of Agile Practices in Pakistan"**
OBJECTIVE	To investigate the issues and benefits of agile practices from literature and compare the results with the industrial survey to generalize the results
UNDERTAKEN BY:	**Mehwish Zulfiqar** 198-FBAS/MSSE/S08 Student of MS in Software Engineering Department of Computer Science, Faculty of Basic and Applied Sciences International Islamic University, Islamabad
SUPERVISED BY:	**Dr. Abdul Rauf** Assistant professor DCS&SE International Islamic University, Islamabad
DOCUMENTATION TOOLS:	Microsoft Word 2007. Adobe Reader.
OPERATING SYSTEM USED:	Microsoft Windows Vista
START DATE:	December 15, 2010.
COMPLETION DATE:	January 23, 2012.

Abstract

Software development projects are considered to be successful if completed within time and budget while satisfying customers' needs. Software development is always associated with ambiguity due to changing requirements, change in technology or early delivery, hence plan-driven, engineering based methodologies are not suitable for such dynamic environment. Agile development provides flexibility to handle changing requirements, improved communication and coordination mechanisms, and improved quality while enhancing the time-to market speed.

The motive of this work is to accumulate the knowledge regarding benefits and challenges of the agile practices at one place. The objective of the study is to examine whether the benefits and challenges of the practices reported in Literature are experienced the same way in practice.

Systematic Literature Review is used to identify the issues and benefits of agile practices reported in literature. The data obtained as results of Systematic Literature Review is compared with industry responses by conducting a survey in software industry of Pakistan. Finally the gaps between Literature and Survey findings are analyzed and conclusions are given. XP and Scrum practices are being examined for their issues and benefits as these are reported to be the most widely used agile methods.

List of Tables

List of Figures

List of Abbreviations

XP	Extreme Programming
PM	Project Manager
CM	Configuration Management
SLR	Systematic Literature Review
TDD	Test Driven Development
CI	Continuous Integration

Table of Contents

CHAPTER: 1. INTRODUCTION

Chapter 1

1.1 Introduction

Software development is considered to be successful if it completed within time and budget while satisfying customers' needs. Software development is always associated with ambiguity due to changing requirements, change in technology or early delivery, hence plan-driven, engineering based methodologies are not suitable for such dynamic environment [8]. To deal with these circumstances a group called 'Agile Alliance' was formed in mid 1990s, proposed by software professionals practicing and sharing a common set of values for software development. [1]. Agile development provides flexibility to handle changing requirements, improved communication and coordination mechanisms, and improved quality while enhancing the time-to market speed [2].

The Agile development [17] values:

"Individuals and interactions over processes and tools, Working software over comprehensive documentation, Customer collaboration over contract negotiation, and Responding to change over following a plan."

The Agile proposal includes different agile methods such as Pragmatic programming (PP), Dynamic systems development method (DSDM), Extreme Programming (XP), Scrum, Feature Driven development (FDD), Crystal [1]

Agile software development has been under considerable attention in research and practice since 1990s. Due to these striking claims about Agile Software Development, the trend to adopt agile started greatly from 2003 and now Agile methods are being successfully implemented by either adapting them or combining them with other approaches [16]. The number of publications on agile methods indicates the market and research interest in Agile implementation. The need for rapid time to market encourages people to adopt agile as it promotes early delivery along with good quality. Many empirical studies report the successful adoption and execution of agile methods [15].

Applying Agile methods require intensive and open communication, hence suitable for collocated settings in small or medium size companies [17],[26]. Applying Agile in large-scale organization having large teams can decrease the benefits associated with agile practices due to decreased communication. Empowered teams, increased needs for communication and coordination, shared ownership, continuous integration are difficult to manage in large teams [26]. According to [18], agile has limited support for legacy systems and safety critical systems. Hence agile methods introduce challenges while adopting in large projects involving large teams and for developing safety critical systems.

Agile development projects are delivered in small incremental iterations. The cycle is Analyse, Develop, Test; Analyse, Develop, Test; and so on [10, 11]. Agile methods put a considerable emphasis on feedback. Detailed planning is done only for the current tasks. Each agile method focus on different phases of software development lifecycle. XP provide sufficient directives for requirements gathering, designing, implementation and testing. Scrum is focused on project management perspectives [2].

[4][6][7] Conducted survey on the most widely used agile methods and their ultimate results. It was found that the Scrum practices is most widely used, and gives positive results in most cases. Scrum Alliance Membership Survey in 2007 [4] reported the increasing satisfaction rate in customers and the companies using Scrum. Scrum provides a framework for building projects; it is not a model which can be applied exclusively [26]. XP does not provide sufficient support for project management viewpoint while Scrum is mainly intended for providing the project management practices for agile software development. Therefore, Scrum should be combined with other methods to provide a complete set of development and management practices [2]. A hybrid of XP and Scrum practices produce very successful results by adopting the developmental practices of XP and managerial practices of Scrum.

This study is intended to examine the issues and benefits of implementing agile practices in software development projects by evaluating the empirical studies from literature.. XP & Scrum are reported to be the most widely used agile methods; hence their practices are being examined for their issues and benefits. Systematic Literature Review is used to

identify the empirical studies of Agile development. The data obtained as results of Systematic Literature Review is compared with industry responses by conducting a survey in IT market of Pakistan for determining the gaps between benefits and issues associated with agile practices.

1.2 Related Work

There are several empirical studies reporting the benefits and issues of agile practices by describing the experiences of agile implementation:

Chen et al. [23] performed a case study at Intel to investigate how agile methods can be employed at large organizations. This study provided valuable findings by comparing different agile methods and proposing framework which facilitates the selection of agile methods according to team size, project type etc. The study is a good initiative to direct research efforts towards the challenges of applying agile methods in large organizations.

Begel and Nagappan [22] conducted a survey at Microsoft with a focus on how software organizations are adopting Agile practices. The motivation behind the study was to investigate Agile adoption in large organizations. 487 respondents contributed in the study to determine the widely adopted Agile practices. Practices of Scrum and XP were most widely used in Microsoft. The study provided valuable contributions for identifying the challenges related to Agile adoption and acceptance in large organization as Microsoft. Focus is on a broad view of overall benefits and challenges of using agile methods but each practice with both benefits and challenges is not considered.

Fruhling et al. conducted a case study [24] in a government organization to examine the adoption of XP practices in organizations practicing the plan-driven software development methodology. The results provided valuable insights for the practitioners aiming to move from traditional software development methods to agile methods. Challenges associated with adopting the practices: iterative development, time-boxed iterations, continuous integration, TDD, pair programming and onsite customer are discussed. This study

Salo and Abrahamsson [7] conducted survey in embedded software development industry and provided insights regarding the application of Agile methods in such a complex environment. But the results of the study cannot be generalized as it is limited to Embedded Industry.

Petersen and Wohlin [25] conducted a case study at Ericsson to determine the application of Agile practices in large companies. The study explored the benefits and challenges of using a model comprised of incremental methods and agile methods. The objectives for this work is to verify whether agile methods can be applied in large organizations and the benefits and challenges faced while agile adoption. This study is a good effort to generalize the literature results by comparing them with the results of case study. The comparison conclude that the benefits reported in literature are almost no different from those experienced in case study but the new challenges were identified due to applying agile and incremental practices in large organization with a different study context of telecommunication. Challenges related to the practices: pair programming, onsite customer and iterative development, short releases, regular meetings, continuous integration, unit testing are explored while the benefits of continuous integration, iterative development and open communication are discussed. The study is a good guideline for practitioners in identifying the issues associated with the use of agile methods. The limitation of the study is that the model being used for generating the results is based on both agile and incremental practices; hence it cannot be compared with the literature findings. Also, the comparison of challenges and advantages from literature with the findings of a case study cannot generalize the findings on broader context.

Overhage *et al.* [18] investigated the factors motivating and hindering the developer's motivation in adopting Scrum. They designed a framework to investigate the developers' acceptance before using Scrum and to validate the findings after developers have used Scrum. It was found that Scrum increased the morale of team, visibility of project, knowledge transfer. The benefits of practices and techniques as: daily Scrum meeting, Sprint planning meeting, short sprint, onsite customer, product backlog, team work, team

empowerment, loose management and the challenges of reduced documentation are discussed. The findings of the study need to be validated with more empirical work.

The work conducted by *Laanti et al.* [19] is an effort to fill the gap regarding fewer studies focused on agile transition at large companies. Hence they conducted a large survey comprising of 1000 respondent from different geographic locations regarding the agile adoption experiences and views. Their objective was to analyse the responses regarding the current level of Agile usage and perceptions to be used in future. Respondent were agreed with the benefits offered by Agile methods including higher quality with reduced defect rates and satisfied customer, team empowerment, and satisfaction, process flexibility to incorporate change requests. The problematic areas and challenges were discovered which is a valuable contribution. At the end of study it was concluded that Agile methods are gaining increasing popularity and the practitioners still using traditional methods are aiming to switch to agile values.

Agile practices reported in empirical studies are: iterative development, incremental design, short iterations(fixed-length sprints), sprint planning meeting, Daily scrum meeting, sprint review meeting, TDD, unit testing, refactoring, pair programming, collective code ownership. The researchers noted that agile methodologies are beneficial but have problems too and more empirical evidences are required for determining the challenges faced in large organizations. Hence more studies and evidences in large companies regarding Agile practices implementation is required in order to address the pros and cons of the practices effectively.

The purpose of this study is to systematically evaluate the empirical evidences from literature, in order to better understand the benefits and challenges of applying agile practices in projects. The Agile methods: XP and SCRUM are found to be the most widely used methods, hence the practices of XP and Scrum are considered for this study. By benefits we mean that how the application of agile practices in projects can be valuable. Similarly by challenges we mean the factors which create hurdles for implementing agile practices in projects. To the best of our knowledge, no systematic review effort has been done with the focus on accumulating benefits and challenges associated with the use of

done with the focus on accumulating benefits and challenges associated with the use of agile practices. Furthermore, we conduct a survey to determine whether the challenges and benefits stated in the literature can be confirmed by the practice. This would help us in identifying the similarities and differences between benefits and challenges enlisted in literature and survey.

1.3 Problem Statement:

From the above discussion it is concluded that there is lack of cumulative empirical knowledge on Agile practices regarding their benefits and challenges.

1.4 Research Motivation

- To accumulate the knowledge regarding the benefits and challenges of the Agile practices at one place.
- To examine whether the benefits and challenges of the practices reported in Literature are experienced the same way in Practice.

1.5 Aims and objectives

The main ideology for this study is to determine the issues and benefits of using agile practices in software development projects. Hence a list of objectives associated with the study are as follows:

- Identify the pros and cons of each agile practice by current empirical evidence.
- Provide industrial insights regarding the experiences with agile practices.
- Gap analysis of benefits and issues reported in literature and survey.
- A guide for the industry practitioners aiming to apply the agile practices by describing the issues and benefits associated with their usage.

1.6 Research questions

What are the gaps in state of art and state of practice of agile adoption considering challenges and benefits?

1.7 Expected outcomes

The expected outcomes are:

- Identification of Agile practices usage according to the survey.
- The list of issues mentioned in literature and reported by survey, regarding the use of agile practices.
- The list of benefits mentioned in literature and reported by survey, regarding the use of agile practices.
- Conclusions are drawn by a gap analysis of benefits and issues reported in literature and in survey.
- Findings based on results and discussion.
- Directions for future work are suggested.

1.8 Research Methodology

A mixed methodology approach is used for this study i.e. systematic literature review (SLR) and Survey. Systematic literature review is used to elicit the benefits and issues associated with using agile practices. The results of SLR are matched with the results of industrial survey as shown in Figure 1. The research methods are described as follows:

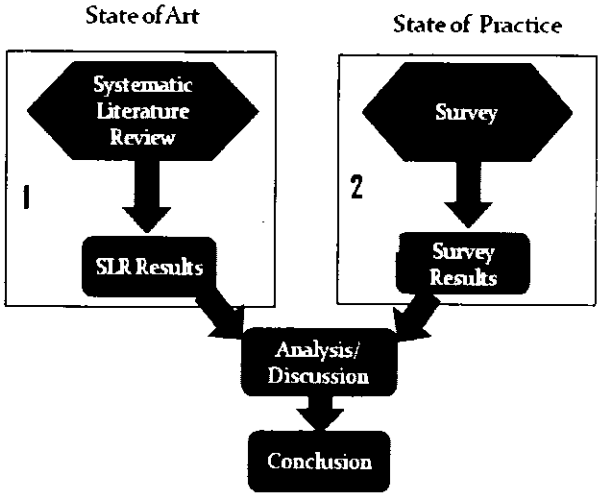


Figure 1: Research Methodology

1.8.1 Literature Review

Systematic Review is a scientific approach to identify, evaluate and interpret the available research in accordance with research questions [3]. This well-defined approach enables the unbiased way of collecting desired information from the empirical studies and facilitates in providing the background knowledge to explore the new research directions.

We have used the Systematic Literature Review (SLR) to accumulate the empirical research regarding issues and benefits of agile practices. The review process [3] for this study is given below:

"Planning the Review,

- Identification of the need for a review
- Development of a review protocol

Conducting the Review

- Identification of research
- Selection of primary studies
- Study quality assessment
- Data extraction & monitoring
- Data synthesis.

Reporting the Review

- "is a single stage phase ". In this phase results of SLR are reported effectively".

1.8.2 Survey

An *industrial survey is* conducted to identify the strength of benefits and issues reported in literature. Survey is selected for this study as it seems suitable for getting multi-disciplinary view about the benefits and issues of practices by involving different roles

from multiple companies. The purpose of the survey is to draw the generalized conclusions about the benefits and issues of agile practices [20].

The questionnaire for survey is designed based on summarized SLR results. A list of software companies in Pakistan using agile practices in their projects is organized to explore the benefits and issues faced by the companies. In this way, a current view of issues and benefits faced by industry is presented.

1.9 Thesis Structure

This thesis consists of 5 chapters. After introduction in chapter 1, the details for conducting the Systematic Literature Review and the results are described in chapter 2. Chapter 3 describes how the survey is conducted and the survey results. Then chapter 4 focuses on analysis of Survey and SLR results to analyses the similarities and variations in benefits and issues reported in SLR and Survey. Chapter 5 provides conclusions for the work; illustrate the contributions and directions for future work.

CHAPTER 2: SYSTEMATIC LITERATURE REVIEW

Chapter 2: Systematic Literature Review

A Systematic Literature Review (SLR) is conducted for identification of benefits and challenges of practices XP and Scrum.

"A literature review is defined as a way to identify, evaluate and interpret all available relevant researches related to an issue, or phenomenon of interest in a particular area". [3]

The three main phases in systematic review are: planning the review, conducting the review, and reporting the review. The first phase is about planning the review. It provides guidelines regarding the systematic way of conducting literature review and the development of review protocol. A review protocol describes the aim of research and the steps to be followed in systematic review. In order to avoid researcher bias Review protocol should be designed in well defined manner following the guidelines mention in [3]. In second phase, the systematic review is conducted. It consists of primary studies, quality assessment criteria, data extraction and data synthesis. Finally in the last stage, the results of the review are reported [3].

2.1 Planning the Review

This chapter describes the highlights of protocol defined to perform the Systematic Literature Review (SLR). SLR is conducted on the base of predefined plan (protocol) as explained below:

2.1.1 Research Questions

Q1. What are the gaps in state of art and state of practice of agile adoption considering challenges and benefits?

This question can be answered with the following three sub-questions:

Q1a. What are challenges and benefits of Agile practices in state of art?

The state of art of challenges and benefits of agile practices were answered by conducting a Systematic Literature Review of empirical studies of agile software development from 2006-2011. Dyba & Dingsoyr [14] conducted SLR on empirical studies on agile methodologies up to

2005. I extend their SLR by including studies from 2005- 2011. A list of issues and benefits of agile practices are traced from literature to identify the most beneficial and problematic practices.

Q1b. What are challenges and benefits of Agile practices in state of practice?

The state of practice of agile adoption considering challenges and benefits are addressed by conducting a survey in software industry of Pakistan including the main cities: Islamabad, Rawalpindi, Lahore and Karachi. The practitioners were asked to fill the questionnaires inquiring about the adopted agile practices and their benefits and challenges. Based on the questionnaire responses, a list of issues and benefits of agile practices are generated.

Q1c. What are the gaps in challenges and benefits of Agile practices according to state of art and state of practice?

To identify the gaps in literature and state of practice, lists of issues and benefits generated as a result of SLR and Survey are compared. In case of gaps or differences, the rationales behind the gaps are tracked and analyzed and hence results and conclusion are given.

2.1.2 Major Search Terms and Synonym

Major search terms for the Q1a regarding empirical studies of Agile are:

Agile: (Agile OR XP OR "extreme programming" OR scrum)

Method: ("software development" OR "software project development" OR "software application development")

Empirical: (Empirical OR industrial OR experiment OR "case study" OR survey)

2.1.3 Search String

(Empirical OR industrial OR experiment OR "case study" OR survey*)AND(Agile OR XP OR "extreme programming" OR scrum)AND("software development" OR "software project development" OR "software application development")

2.1.4 Search Sources

A range of data bases has been selected for rigorous search and to reduce the bias. Following data bases were searched for the retrieval of primary studies:

- IEEE Explore
- ACM Digital Library
- Science Direct

Various databases have been selected to reduce the bias of study including published Technical Reports, Journal Papers and Conference Proceedings.

2.1.5 Study selection criteria

- The initial selection was done on the basis of the TITLE and ABSTRACT of the paper.
- All data obtained from the search process was archived in database according to the journal from which it is retrieved.
- From database the duplicates originating from various search engines was removed after initial scan of results.
- Inclusion and exclusion criteria were applied on the remaining articles to sort out the accepted ones.
- Full papers of all studies that were clearly ineligible to fit in the criteria, were then discussed with supervisor for their final exclusion.
- The excluded papers and reasons for exclusion were recorded in a file, and the included papers and study type was recorded in another file.

- The excluded papers and reasons for exclusion were recorded in a file, and the included papers and study type was recorded in another file.

2.1.6 Study of Inclusion and Exclusion Criteria

Various criteria were formulated with the intension to identify those studies that provide direct evidence for the research questions. Following were the inclusion and exclusion criteria for my research questions:

2.1.6.1 Inclusion criteria:

In order to answer the stated research questions, I searched for research articles by reading the abstracts while considering the following inclusion criteria:

- Studies that focus on agile practices with some empirical work done.
- Studies including a case study, an experiment, survey or industrial experience reports were focused.
- Studies that are published between 2006-2011.
- Studies on XP and scrum methods.
- Studies with focus on any practice of XP or Scrum.

When it was confirmed after reading the abstract that article is relevant to my research, I studied the whole paper. The objective of the selection process was to identify the articles relevant to the objectives of the systematic review. The search strings, were quite broad and hence it were expected that not all studies identified would find a place in the final phase of selection process.

2.1.6.2 Exclusion criteria:

- Those studies were excluded that were based on personal expert opinion.
- Literature surveys and books were excluded.

2.1.7 Quality Assessment

After applying the stated inclusion and exclusion criteria, detailed inclusion criteria was applied i.e., a Quality Instrument for studies to sort out the accepted ones. The studies were evaluated according to 11 criteria as used by [3],[14]. Table 2.1 shows the factors related to quality:

Table 2.1 Quality Assessment Form

Quality assessment form	
1	Is the paper based on empirical study?
2	Does the paper explicitly state the aims for conducting the study?
3	Does the context of the study is sufficiently explained?
4	Is there any the justification provided for selecting the Research design in accordance with aims of study?
5	Is there any description of sampling procedure for data gathering? 5.1 Does the procedure for selecting the contributors or cases given? 5.2 Are the selected contributors or cases are justified to be appropriate to provide information regarding research aims? 5.3 Is there any description about the contributors? 5.4 –Is the number of contributors sufficient?
6	Does the study consider any negative responses to compare results? 6.1–Does the non-responding members or opposing members are part of the sample under study? 6.2 –Are the non-responding members higher in proportion? Are they different from the respondents in some way?
7	Does the data gathering procedures are defined and explain sufficiently? 7.1–Is the count for data gathering mentioned? 7.2–Are the data gathering methods explicitly mentioned?

7	Does the data gathering procedures are defined and explain sufficiently? 7.1–Is the count for data gathering mentioned? 7.2–Are the data gathering methods explicitly mentioned? 7.3–Is there any justification for appropriateness of selected method? 7.4–is there any detailed explanation collecting data by the specified data gathering method (e.g. interview guide, questionnaire)?
8	8.1 –Is the analysis process described in detail? 8.2 –If qualitative data used, is there any explanation for analysis by driving categories? 8.3 –Is the volume of data is sufficient to provide valid finding? 8.4 –Has the opposing views being considered?
9	Is there any connection between researcher and contributors affecting the validity of results? 9.1 –Has the researchers critically examined their role of un-biasness of findings? 9.2 –Has the researcher participated and responded to study occurrences?
10	10.1 Are the findings stated explicitly? *10.2 –Is the reliability of findings justified?* 10.3–Has the researcher clearly stated the limitations of the study? 10.4 –Does the findings stated in accordance with aims and research questions? 10.5 –Have the conclusions been derived from results?
11	11.1 –Has the study made any valuable contributions to literature 11.2 –Does the study identify new areas for future work? 11.3 –Are their any suggestion for continuing the study in other ways?

2.1.8 Search Process Documentation

Primary Search Documentation

The customized search strings were applied to the databases according to decided strategy. As the process go on, the results were saved by the following decided strategy.

i. A folder by name of RQ1 was created.
ii. Within the folder further sub folders by the name of specified data base or journal were created.
iii. Within each folder for different search string terms different folders were created by the name of their IDs.
iv. All records were maintained for one search string in library of reference manager software (endnotes).
v. Results of that specific search strings were placed in that folder created by the name of that search string.
vi. Same process was performed for the research questions and on all data bases.
vii. Duplicates (in papers and studies in papers) were removed from data base after scanning the records.
viii. After applying inclusion/exclusion criteria within the folder by the name of journal, I created two folders for included and excluded papers. This also give indication that which journal gave more evidence then others.
ix. Database was updated for included and rejected papers.
x. Reasons for not including were recorded in a file.
xi. All the included papers were moved to one folder.
xii. Conflicts for papers where inclusion or exclusion is ambiguous were consulted according to the decided rules, another file were created to record these activities, the decisions were recorded accordingly and papers either be accepted or rejected.

2.1.9 Data Extraction

- Data-Extraction form were applied to all the accepted papers
- The forms were entered into the Data base for results.

- Duplicate publications were identified and the paper that reports the most recent results of the study were preferred.
- Data extraction was performed only for that decided paper.
- The form first obtained general information about the paper and then data extraction procedures were applied related to the research questions.

2.1.10 General Information Required for a Single Study

Table 2.2 shows the general information extracted for each paper, providing the complete information on structure and quality of the paper.

Table 2.2 General Information Form

Study ID	
Names of Authors	
Reference	
Article Type	
Objectives	
Context	
Research Design	
Sampling	
Study Setting	
Control group	
Data gathering	
Analysis process	
Researcher Bias	
Findings and conclusions	
Value of research	

2.1.11 Extraction for Research Question

Table shows the data extraction form extracting the information according to research questions. The table also contains fields essential for the analysis of results obtained after data extraction.

Table 2.3 Data Extraction Form

P. ID	Quality Score	Research Method	Agile Method	Agile Practices	Benefits	Issues

2.2 Conducting the review

SLR is conducted to evaluate the available research in relation to research questions. The endnote software was used for managing the references obtained as a result of search.

The research articles were selected according to the criteria mentioned in the above section. Same search string was used for each database. Table2.4 shows the search strings used for each database.

2.2.1 Search String Application to Databases and Results Retrieved:

General search string is provided in section 2.1. An initial scoping study helped in identifying search terms and search sources. Google scholar was included in search sources but later removed as it gave different search results of the same search string at different times. Some databases did not allow the complete search string. Different search sources have different search string format so search string was modified and then applied on such search sources.

Customized search string applied on each database and citations are downloaded in a master library in endnote. Customized search string for each database and results retrieved are shown in table 2.4 below:

Table 2.4 Search Strings for Databases

Database Name	Search String	No of Articles
IEEE	(Empirical OR industrial OR experiment OR "case study" OR survey*)AND(Abstract:Agile OR XP OR "extreme programming" OR scrum)AND("software development" OR "software project development" OR "software application development")	56
Science Direct	FULL-TEXT(empirical or industrial or experiment or "case study" or survey*)and FULL-TEXT(agile or XP or "extreme programming" OR scrum)and FULL-TEXT("software development" or "software project development" or "software application development")	499
ACM	(Abstract:agile and Abstract:scrum) and (Abstract:empirical or Abstract:industrial or Abstract:experiment or Abstract:"case study" or Abstract:survey) and (PublishedAs:journal OR PublishedAs:proceeding)	37
	(Abstract:agile and Abstract:"extreme programming") and (Abstract:empirical or Abstract:industrial or Abstract:experiment or Abstract:"case study" or Abstract:survey) and (PublishedAs:journal OR PublishedAs:proceeding)	27

ACM allows limited no. of search terms to be executed. That's why major search string is broken down into sub strings.

Results retrieved from each database were stored in master library.

2.2.2 Level 1 Screening

Level 1 searching is performed on title, keywords and abstract. The purpose is to exclude completely irrelevant articles. Abstract level screening provides an easy way to exclude unrelated articles. Applying search string on different databases retrieved 608 studies. Inclusion/exclusion criteria defined in protocol was applied on these studies. Studies that failed to fulfill the criteria i.e., that were unrelated studies were excluded. In case of confusion co supervisor was consulted. No. of articles found after level 1 screening were 51.

2.2.3 Level 2 Screening

Inclusion/Exclusion criteria defined in protocol is applied for level 2 screening. Papers not meeting the inclusion criteria and fulfilling the exclusion criteria were rejected and the reason for rejection was recorded. In case of uncertainty about inclusion/exclusion of paper co-supervisor was consulted. No. of selected articles was 33.

Table 2.5 Search sources and results retrieved:

Search Sources	Results Retrieved	After Duplicate Discarded	Primary Selection	Final Selection
IEEE	56	42	17	8
Science Direct	499	412	120	17
ACM	53	42	19	7

2.2.4 Quality Assessment

Quality assessment was performed following the criteria defined in protocol. Quality score for each paper is presented in *Appendix A.*

Five papers obtained a full quality score of 11, the reason is that they defined a clear context, sample was described completely. The threats to validity were explored to remove researchers' bias, limitations were defined, conclusions were defined and findings were related to research questions. Also the useful directs for future work were suggested.

The papers that scored less than 5 were excluded. The lowest score in list is 5.5 and the reason behind was that the factors as: sample size, the control measures, and author biases were not addressed, while the factors: research design, data collection and data analysis were discussed partially and were not justified.

2.3 Reporting the Review

The purpose for conducting the SLR was to extract the benefits and challenges of agile practices. The agile methods addressed in the studies are shown in Figure 2.1.

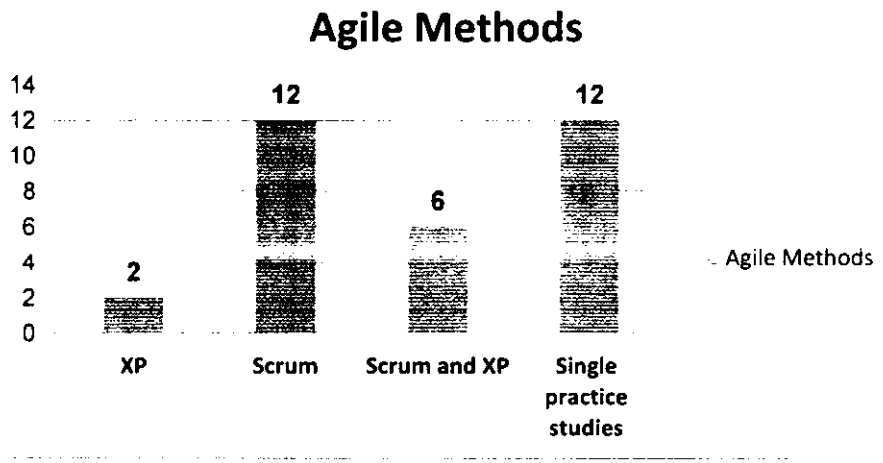


Figure 2.1 Agile methods reported in Literature

2.3.1 Reported Benefits of Agile Practices

2.3.1.1 Iterative Development

Table 2.7 Benefits of Iterative Development

ITERATIVE DEVELOPMEMT:
Early feedback Improve quality and stakeholders' satisfaction [P1,P5,P6,P7]
Development of core functionality determined by customer reduces waste [P5,P6,P7,P8]
Flexibility for adjusting change requests increases [P1,P8,P9]
Deliverable at the end of each sprint give a sense of accomplishment to team and facilitates in measuring progress [P1,P7]
Insights regarding the importance/usage of products by delivering the most wanted features first [P6]
Reduces the complexity of project by breaking it into small features/iteration [P7]
Flexibility of canceling sprints without suffering too many expenses, in case of failure or great change request [P1]
Early discovery of design errors, hence alternative solutions can be decided in start of project and reduce delays caused by late discovery of issues [P1]
Reduce defect rate and increase productivity [P7]

2.3.1.2 Incremental Design

Table 2.8 Benefits of Incremental Design

INCREMENTAL DESIGN
New features can be organized quickly, hence makes the design flexible [P3,P5]
Reduces defects and effort, in case of new features [P3]
Reduces the refactoring effort [P5]

2.3.1.3 Sprint Planning Meeting

Table 2.9 Benefits of Sprint Planning Meeting

ITERATION/SPRINT PLANNING MEETING
Iterative planning can provide quick response to business needs [P6]
Short term planning can adjust the change requests easily hence enabling flexibility [P8]
List of features to be completed in coming weeks are decided [P13]

2.3.1.4 Daily Scrum Meeting

Table 2.10 Benefits of Daily Scrum Meeting

Daily Scrum
Update the team about the project status [P1,P5,P8,P11,P13]
Daily communication solves the issues/ hurdles early and speeds the delivery of products [P1,P5,P8,P11]
Increases communication and coordination [P1,P5,P8]
Awareness of each other activities, thus in case of any help or query, members can coordinate [P1,P5,P11]
Knowledge transfer activity is improved as teams share their challenges and achievements and learn from each other, hence the inexperienced members get trained [P8,P11,P12]
Increase Job satisfaction by updating developers with current information about project and communication with other teams [P20]
Increased visibility about the tasks allocated to each team member which encourage them to provide quality work [P22]
Ambiguities get resolved hence transparency increases [P8]
Teams can discuss solution about the arising issues in lesser time [P24]

2.3.1.5 Sprint Review Meeting

Table 2.11 Benefits of Sprint Review Meeting

SPRINT REVIEW MEETING
Stakeholders get opportunity to provide a thorough feedback and clear requirement misunderstandings with developers, which reduces risk for next iterations [P5,P6,P9,P12]
The quality of features is assured [P5]
Client provide necessary reprioritization of the product backlog to create the goal of the next sprint [P12]

2.3.1.6 Sprint Backlog

Table 2.12 Benefits of Sprint Backlog

SPRINT BACKLOG
If updated during daily Scrum meetings, it becomes very useful [P11]
Enhance situational awareness- project wall is used to display the items [P11]

2.3.1.7 Burn down Chart

Table 2.13 Benefits of Burndown Chart

BURNDOWN CHARTS
Very helpful for team awareness of progress made [P11]
Automatic generation is very helpful [P11]

2.3.1.8 TDD & Unit Testing

Table 2.14 Benefits of TDD & Unit Testing

UNIT TESTING
Reduces bugs, improve quality of code, hence increase client agreement [P1,P5, P7]
Problems can be discovered early by running tests cases in parallel with code development [P26,P27]
Provide a safety net for the whole system and good maintenance due to automated testing [P26, P27]
They are written besides acceptance tests hence ensure clear understanding of system [P4]
Ensure the completion of functionality if applied successfully, feature is ready to be delivered if properly pass these tests [P4]
In case of change requests, creating again these tests can avoid bugs [P5]
Offer reasonable confidence to developers that no part of the code is broken [P26]
Creating unit tests can break down the functionality in smaller and manageable tasks, hence improving low level design [P26]
High assurance about quality of code by running the tests multiple times [P26]
Writing tests before coding results in better branch coverage [P26]
Improved productivity due to coupling of testing and coding [P26]
Error rates are decreased with an increase in productivity which reduce project cost [P28]
Using TDD developers conduct more testing as writing the test cases before the code enable the written tests to be executed before delivery [P28]

2.3.1.9 Continuous Integration

Table 2.15 Benefits of Continuous Integration

CONTINUOUS INTEGRATION
A way to maintain quality by detection of compatibility problems earlier [P1]
Results in early testing hence reducing risk and improving quality [P1]
Force develops to adopt simple design and continuously provide coding for smaller feature sets [P13]
Constant integration provides feedback which can help identify errors quickly [P13]

2.3.1.10 Refactoring

Table 2.16 Benefits of Refactoring

REFACTORING
In case of change requests, refactoring reduces the probability of errors, hence increase quality [P5,P7]
Code is continuously updated to reflect changes [P4]
Improves quality in terms of robustness, reliability and maintainability which reduces the development and maintenance costs [P13]

2.3.1.11Pair programming

Table 2.17 Benefits of Pair programming

PAIR PROGRAMMING
Code produced by PP is less complex and easier to understand, hence high quality [P15,P16,P17,P22]
Pair programmers help in training of inexperienced developers, they learn more about software development [P15,P21,P22]

Developers can sometime work together to solve complex problems, "intelligent pairing" can be more helpful in such cases [P3, P13]
Less experienced team members can be incorporated by PP [P15, P21]
No need for formal code review as testing and removing errors are generally much more costly than coding, hence cost-effective approach [P17, P18]
Error rate is decreased [P16]
Pressure of critical co-programmer results in improved code quality [P16]
Pair programming can decrease delivery time [P17]
Satisfaction and communication among developers is improved [P19]

2.3.1.12 Onsite Customer

Table 2.18 Benefits of Onsite Customer

ON-SITE CUSTOMER
Continues feedback improves quality [P2]
Client can a good chance to comment or accept the results [P3]
Identify requirements for next iterations [P3]
Improve the team knowledge about the business domain [P8]
Can be easily involved in decision-making [P24]

2.3.1.13 Collective code Ownership

Table 2.19 Benefits of Collective code Ownership

COLLECTIVE CODE OWNERSHIP
Helpful for knowledge-sharing about the system [P13]

2.3.2 Reported Challenges of Agile Practices

2.3.2.1 Iterative Development

Table 2.20 Challenges of Iterative Development

Iterative Development
L1. Change requests cause scope creep [P3]
L2. Change requests create irritation for develops [P3]
L3. Customer is not available or busy [P3]
L4. Clients' willingness to get the system in iterations.
L5. Prioritizing/ breaking requirements in iterations is a difficult task [P6]
L6. The manner for prioritizing/categorizing requirements is inconsistent between teams [P6]
L7. Frequent releases of increments increases maintenance and testing efforts for supporting different versions [P22]
L8. Coordinating iterations in large projects is an issue for management due to product scope and large team. [P22]
L9. High Configuration management effort requires to coordinate the number of releases [P22]

2.3.2.2 Incremental Design

Table 2.21 Challenges of Incremental design

Incremental Design
L1. Less experienced customer make wrong estimates while planning [P5]

L2. Lesser emphasis on architecture can result in bad design decisions [P22]

L3. Lack of focus on design due to tight schedules can result in bad design [P22]

L4. Waiting for requirements introduces delay [P22]

L5. Refining the feature set for each iteration [P6]

L6. For complex problems, detailed planning and design is required [P13]

2.3.2.3 Time-boxed Iterations/Sprints

Table 2.22 Challenges of Time-boxed Iterations/Sprints

Time-boxed Iterations/Sprints
L1. The items not completed in an iteration are moved to next iteration with a result in delayed schedule [P1]
L2. Pressure on project managers is increased due to increased need for communication and coordination [P24]
L3. Pressure and stress on developers is increases due to deadlines [P3]
L4. Time-boxing iterations is dependent on nature of project [P11]

2.3.2.4 Sprint Planning Meeting

Table 2.23 Challenges of Sprint Planning Meeting

Sprint Planning Meeting
L1. Dependencies between some modules are unable to be tracked in short term planning [P22]

L2. Pluralistic decision making for requirement prioritization can result in delays if the number of stakeholders is large [P22]

L3. Requirements are not well understood/ ambiguous / not mature [P11]

L4. During planning, the factors: tight deadlines set by upper management and scope creep cause frustration and difficulty for the team [P13]

2.3.2.5 Daily Scrum Meeting

Table 2.24 Challenges of Daily Scrum Meeting

Daily Scrum Meeting
L1. Inexperienced Scrum master cannot make the meetings useful [P1]
L2. Can divert the developers' mind [P1]
L3. These meetings place additional responsibility on developers. Developers get discouraged and focus on daily work which can lose their concern on their coding tasks [P1]
L4. Difficult to handle with large teams. Each team member has different focus area hence the points and discussion is not relevant to everyone [P13]

2.3.2.6 Sprint Review Meeting

Table 2.25 Challenges of Sprint Review Meeting

Sprint Review Meeting
L1. Team participation issues which depends on their eagerness to learn and get improved by these meetings [P11]

2.3.2.7 Product Backlog

Table 2.26 Challenges of Product Backlog

Product Backlog
L1. Initially creating the product backlog and prioritizing features is not an easy task [P6]
L2. Maintaining backlog is challenging due to unstable and dynamic requirements change [P11]
L3. Prioritizing requirements to design products according to customer current needs is time-taking [P11]

2.3.2.8 TDD & Unit Testing

Table 2.27 Challenges of TDD & Unit Testing

Unit Testing
L1. Developers assume continuous testing as a cumbersome task that needs extra effort [P1, P22,P26]
L2. Learning TDD requires effort and time [P22]
L3. It is not possible to write unit tests for legacy code and it is very expensive [P13]
L4. Unit testing is very time consuming task [P22]
L5. Developers rely on testers to do this task. Also quality department exist [P1, P26]
L6. Writing tests besides coding is cumbersome in tight deadlines; hence developers tend to discard testing [P13]
L7. Write unit tests for each class of code is a cumbersome task [P13]

2.3.2.7 Product Backlog

Table 2.26 Challenges of Product Backlog

Product Backlog
L1. Initially creating the product backlog and prioritizing features is not an easy task [P6]
L2. Maintaining backlog is challenging due to unstable and dynamic requirements change [P11]
L3. Prioritizing requirements to design products according to customer current needs is time-taking [P11]

2.3.2.8 TDD & Unit Testing

Table 2.27 Challenges of TDD & Unit Testing

Unit Testing
L1. Developers assume continuous testing as a cumbersome task that needs extra effort [P1, P22,P26]
L2. Learning TDD requires effort and time [P22]
L3. It is not possible to write unit tests for legacy code and it is very expensive [P13]
L4. Unit testing is very time consuming task [P22]
L5. Developers rely on testers to do this task. Also quality department exist [P1, P26]
L6. Writing tests besides coding is cumbersome in tight deadlines; hence developers tend to discard testing [P13]
L7. Write unit tests for each class of code is a cumbersome task [P13]

L8. Difficult to completely provide test coverage as there is lack of independent testing due to short timelines [P22,P26]

L9. As designers, developers and testers are supposed to work together; hence verification and validation can be biased [P22]

L10. To write test before own code is perceived as additional effort [P26]

L11. TDD has no effect on quality and productivity [P26,P27]

L12. External quality of the product is reduced by overlooking high level tests in tight schedule [P27]

L13. Developers has to write tests before coding, hence less time is left for coding tasks [P27]

2.3.2.9 Continuous Integration

Table 2.28 Challenges of Continuous Integration

Continuous Integration
L1. Cannot be applied for tasks having clear boundaries [P3]
L2. Continuous integration is difficult for large teams [P13]
L3. Team should be aware of agile practices in addition to having single codebase [P5]

2.3.2.10 Refactoring

Table 2.29 Challenges of Refactoring

Refactoring
L1. Risk of introducing new errors in existing codebase leads to abandon refactoring

[P30]

L2. Refactoring does not guarantee whole improvement in quality of software [P23]

2.3.2.11 Pair programming

Table 2.30 Challenges of Pair Programming

Pair programming
L1. Developers are not willing [P3]
L2. More resources are utilized as compared to working individually [P15]
L3. Arduous activity with the requirement of partners having same caliber and qualification[P22]
L4. Not feasible in case of limited resources [P26]

2.3.2.12 Onsite Customer

Table 2.31 Challenges of Onsite Customer

Onsite Customer
L1. Customer unavailability results in delayed feedback [P13]
L2. Customer cannot be assigned for availability during the whole development phase [P22]
L3. It is not reasonable to daily give conveyance to the customer for coming to the site [P5]

2.3.2.13 Collective Code Ownership

Table 2.32 Challenges of Onsite Collective Code Ownership

Collective code Ownership
L1. Lack of individual responsibility leads to decreased code quality [P13]
L2. Not suitable for large or complex systems with many modules [P13]

CHAPTER 3: SURVEY

Chapter 3: Survey

A survey is conducted to investigate the benefits and issues of Agile practices experienced in industry. Survey is chosen as a research methodology as it allows a large number of responses to be assessed and in getting responses from a pool of employees from different companies [20].

3.1 Design of Questionnaire

The questionnaire is designed at the completion of SLR results for conducting survey. The SLR results formed the basis for enquiring industry practitioners against the benefits and issues of Agile practices. The objective for questionnaire based evaluation is to validate the SLR results from industry.

The questionnaire is designed using an online survey tool, Kwik Surveys. The rationale behind designing the survey in this way is to enable the respondents in different locations to fill the questionnaire by just clicking the link. http://www.kwiksurveys.com/ is the link for creating the survey Instrument i.e. questionnaire. KwikSurveys offers cost-free services; anyone can generate an account on KwikSurveys and design the questionnaire. Data analysis is also easy after collecting data using KwikSurveys, hence it is a cost-free and easy approach for data collection and analysis.

After designing the questionnaire using KwikSurveys, a link for the questionnaire was generated that was sent to the practitioner by email. This provided ease to practitioners as they can fill questionnaire by accessing it from any location.. The questionnaires consist of following type of questions:-

- Demographic information
- Close-ended questions
- Open questions

The information about the role, experiences, name and type of project and organization name is acquired in the start of questionnaire. This demographic data is used to confirm that the respondents belong to software development companies using Agile methods and are knowledgeable enough to answer questions. Close-ended questions are designed to identify the benefits of Agile practices and to confirmed that the benefits reported in literature are experienced by the industry too. Open questions are intended to get the issues of Agile practices to match them with those reported in literature. This questionnaire helped to discover the gaps between Literature and Practice. Questionnaire is presented in (*Appendix B*).

3.2 Questionnaire Evaluation

Before placing the survey online, pilot testing was performed to check either there is any need of improvement in survey instrument. Pilot testing was performed by getting peer review with the students who were well-aware of agile methodology. After minor modifications as result of peer review, the questionnaire was sent to the supervisor for suggestion and verification. Finally the questionnaire was sent to 2 experts in industry to check the validilty of instrument.

3.3 Questionnaire Distribution

In order to get the companies' contact details using Agile methods, a list of companies registered with PSEB is obtained by visiting PSEB office in Evacuee Trust. Questionnaire link was sent to respondents through following steps:

- Phone calls were made to the companies enquiring about the usage of agile practices.
- The email addresses of the participants who can give the desired information and willing to participate were acquired.
- An email describing the aims of questionnaire along with questionnaire link was sent.
- A reminder was sent by email in case of late responses

Our target population is Pakistan's Software Industry i.e. Software companies that are using agile methodologies. The software companies have their head offices in three major cities; Karachi, Lahore, and Islamabad. Reason behind the selection of target population is to get the expert opinion, from software practitioners e.g., project managers and senior software managers, developers, who are already working in an environment where agile practices are implemented.

Our sampling includes 79 responses from 45 software companies using agile practices. Non-probabilistic samples are used because the target population is specific. Convenience sampling will be used, as it takes responses from people who are willing to participate.

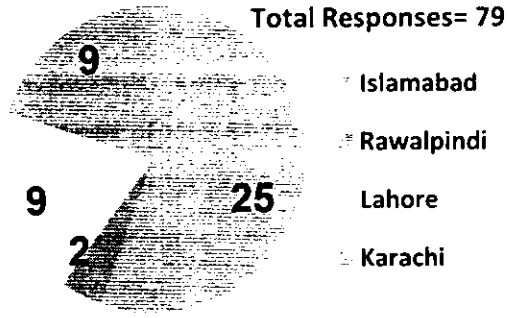


Figure 3.1: Categorization of responses

3.4 Survey Results

We conducted an online survey over a period of 2 months. An invitation was sent by email to 62 companies. We received 79 responses from 45 companies, of which 2 were invalid (due to non-agile), 4 were incomplete. We are unable to get more responses as the respondents in companies didn't have enough time due to their busy schedules. To have a multidisciplinary view, the survey considered all types of professionals involved in a team, i.e., Project Managers, Architects, Product Managers and Developers etc. as shown in Figure 3.2. More than 1 response was collected from each company. Figure 3.1 shows the count of responses according to locations.

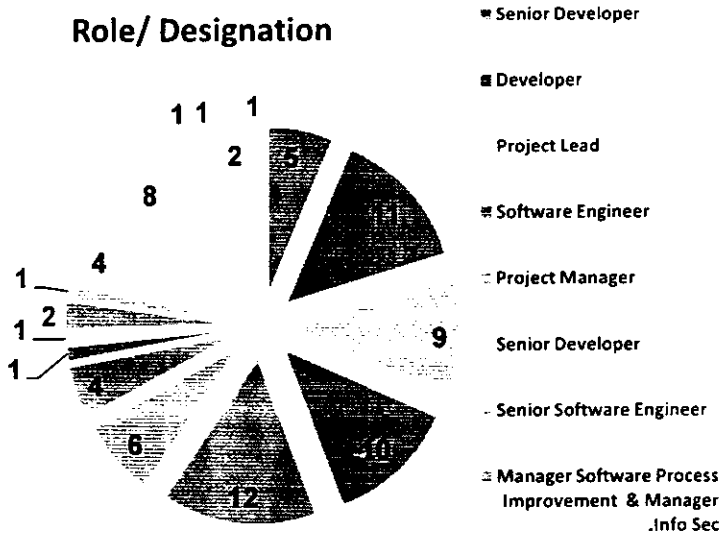


Figure 3.2: Role of Respondents

3.4.1 Demographic Information

A total of 56 questions, divided in 3 sections were asked: 1st four questions were inquiring about demographic information as represented by Figures 3.2, 3.3, 3.4. Figure 3.3 is representing the experience of respondents to have an insight regarding how long Agile is being practiced in companies. As mentioned in literature that Agile is more suitable to be used for small teams, fourth question inquires about the team size to have awareness about how many companies are using XP and Scrum practices for larger teams as shown in Figure 3.4. The fifth question inquired whether agile methods were being used in the company in order to assure that the person filling the questionnaire has experienced agile development.

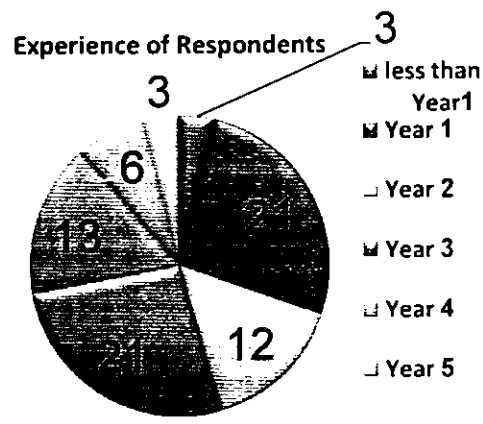


Figure 3.3: Experience of Respondents

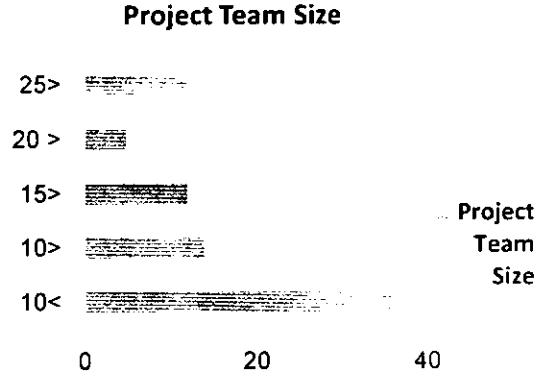


Figure 3.4: Project Team Size

3.4.2 Agile methods usage

Sixth question inquired about the name of method being used to gain insight of the most widely used agile method as shown in Figure 3.5. The respondent can select other methods in case of usage of a mixed methodology.

3.4.3 Agile practices usage

The seventh question inquired about the agile practices being used to calculate the extent of usage of the practices as shown in Figure 3.6. A list of both Scrum and XP practices is given so that the respondents using a mix of XP and Scrum can give input.

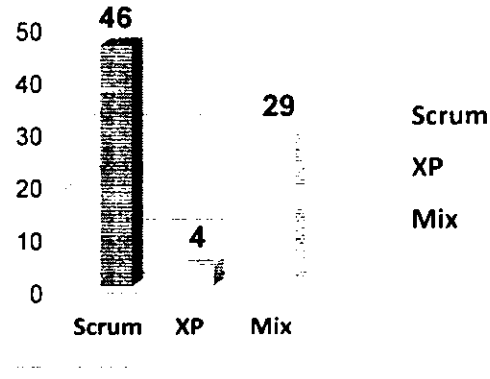


Figure 3.5: Agile methods Usage

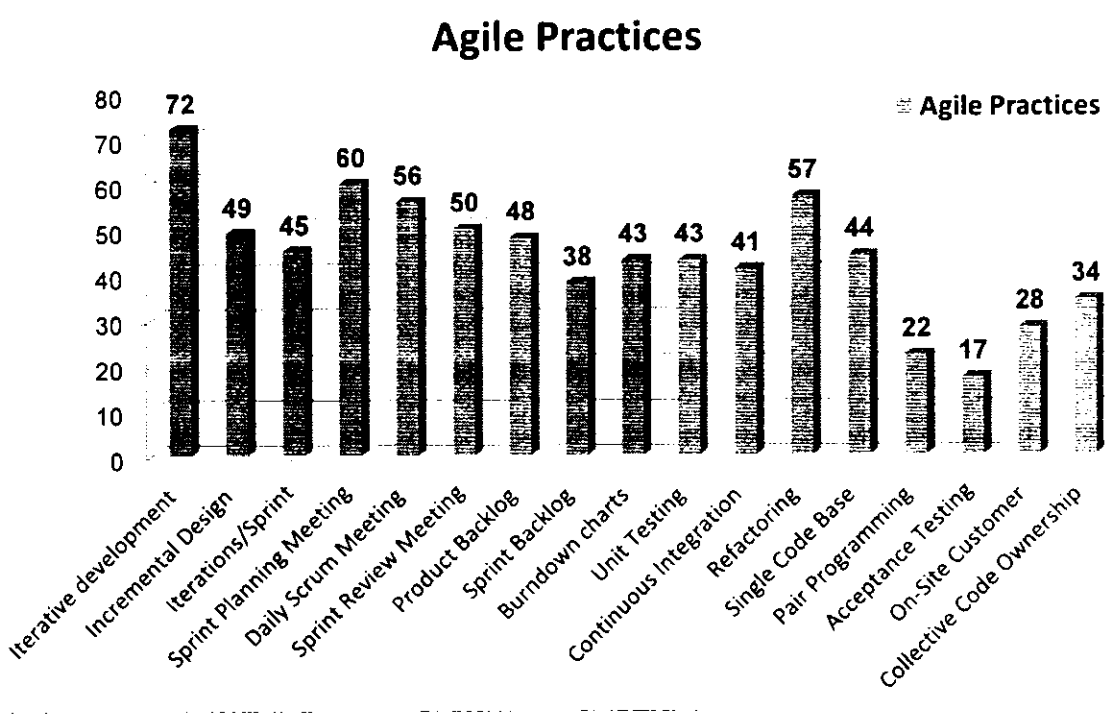


Figure 3.6 Agile Practices Usage

3.4.4 Agile organizational culture

Agile methods require some prerequisites to be successfully applying the practices to get a maximum of benefits. The following changes are required for adopting and successfully implementing Agile practices.

- Intensive and open communication to promote a culture of knowledge sharing.
- More focus on people than process. Agile advocates the empowerment of teams to increase their morale and encourage them to innovate. After assigning the tasks to the teams, it is the teams' responsibility to decide the ways to accomplish the task hence promoting freedom of development.
- Pluralistic decision making, involving teams and client, so that all stakeholders share the responsibility for project success, instead of making only project manager accountable for the results. Pluralistic decision making promotes confidence, satisfaction and trust in teams so that they can utilize their abilities in best possible way.

- Leadership-and collaboration over command-and-control so that PM can focus more on removing hurdles to knowledge transfer and promote agile culture for team empowerment. Hence the project management is light ad adaptive as opposed to traditional management where PM is more a task manager than facilitator.
- Teamwork is preferred over single assignments.
- Involving customers in activities as: prioritizing of tasks, and planning releases etc.

Questions 8-12 of the questionnaires inquires about the organizational culture to help in results analysis and drawing conclusions for the survey data as shown in

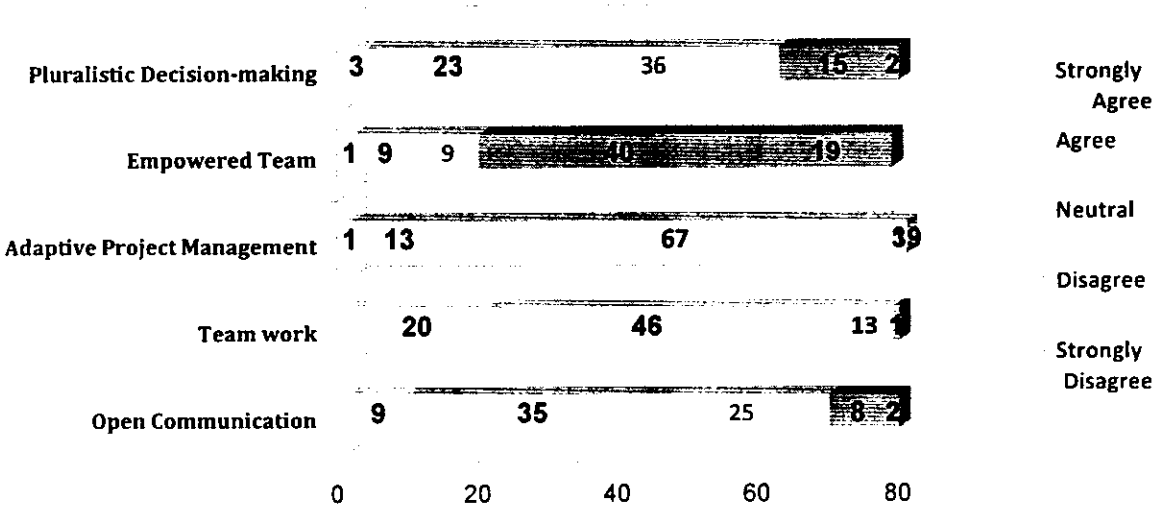


Figure 3.7: Agile Culture

3.4.5 Benefits of Agile practices

Literature has reported many benefits of Agile practices. The benefits are grouped in related themes to generate a list of the main benefits related to each agile practice. The benefits for each practice are confirmed from industry by asking the close ended questions. The benefits has received high acceptance. Here is a list of benefits of each practice according to the priority order of agreed responses.

Table 3.1: Benefits of Agile Practices

Iterative Development
Iterative and Incremental delivery enables the development of core functionality first as prioritized by Client
Iterative development helps teams adapt quickly to random &rapidly changing requirements.
Short iterations help in getting early feedback to improve quality and increase stakeholders' confidence.
Deliverable at the end of each sprint gives a sense a accomplishment to team and facilitates in measuring progress.
Iterative development is useful for measuring progress and greater control over the delivery.
Iterative development help explore alternative solutions at the beginning of project by early discovery of design errors.
Incremental design
Incremental design provides flexibility to incorporate new features with reduced defects and effort.
Daily Scrum meeting
Daily Scrum meeting increase communication and coordination and update team about project status
Daily Scrum meeting solves the issues early and results in team satisfaction.
Daily Scrum meeting increase awareness of each other activities and improve knowledge transfer.

Sprint Planning meeting

Sprint Planning meeting provide concise set of tasks for each iteration which speeds up delivery.

Sprint Review meeting

Sprint review meeting provide stakeholders an opportunity to give feedback and clear requirement misunderstandings which reduces risk for next iterations.

Sprint backlog

Sprint backlog enhance situational awareness if updated daily.

Burn down charts

Burn down charts are very helpful for getting awareness of the progress made.

Refactoring

Refactoring reduces the probability of errors during change requests and improve quality.

Test-Driven development & Unit Testing

TDD gives confidence to developers that no part of code is broken and clear their understanding of system.

TDD improve code quality and feature is ready to be delivered if properly pass these tests which increases client confidence

TDD acts a safety net and provides better branch coverage as TDD requires writing tests first and only coding the functionality that is already covered by tests.

Pair programming

Code produced by PP is less complex, easier to understand and of high quality.

PP reduces defects and eliminates the overhead cost of a more formal code review.

Less experienced team members can be incorporated and trained by PP.

Collective code ownership

Collective code ownership implies constant peer reviews and knowledge distribution.

Continuous integration

Continuous integration improves software quality and reduces risk.

Continuous integration allows detection of compatibility problems early.

3.4.5 Challenges of Agile practices

The third portion of the questionnaire contains open-ended questions to explore the challenges faced while implementing the Agile practices. The data collected from open-ended questions was in raw form; hence it was necessary to categorize the data according to relevant themes. The data was tabulated against each practice by assigning them a unique ID.

The data analysis process for open-ended questions is categorized as follow:

1. The statements describing similar problems against any practice were grouped together.
2. After grouping of relevant statements, issues were derived based on the reasons of similarity between statements. Each issue is explained in not more than one statement.
3. The issues were grouped based on their relationship or dependency in a branching way e.g, if one issue results in further two issues.

3.4.6.1 Challenges of Iterative Development:

The issues initially reported in survey are grouped in main issues and their sub-issues as shown in Figure 3.8. By applying the three-step data analysis process described in section 3.4.6, the challenges are summarized in Table 3.2. The challenges with lesser frequency and seeming highly irrelevant are not included in table.

Table 3.2: Challenges of iterative development

Iterative Development
S1. Clients' unavailability or Lack of cooperation
S2. Clients' willingness to get the system in pieces.
S3. iterative development for large projects result in scope management issues
S4. Continuous change requests can delay the project due to scope creep
S5. Continuous change requests results in developers frustration
S6. Prioritizing/categorizing/breakdown of requirements in iterations/user stories is a difficult task
S7. Active project management and control is required to manage time very properly and scoping of features
S8. Frequent releases increase CM effort
S9. Require more effort in term of testing which can increase time and cost constraints
S10. Integration with existing application architecture
S11. Deployment while system is being used
S11. Configuration of different iterations increase complexity

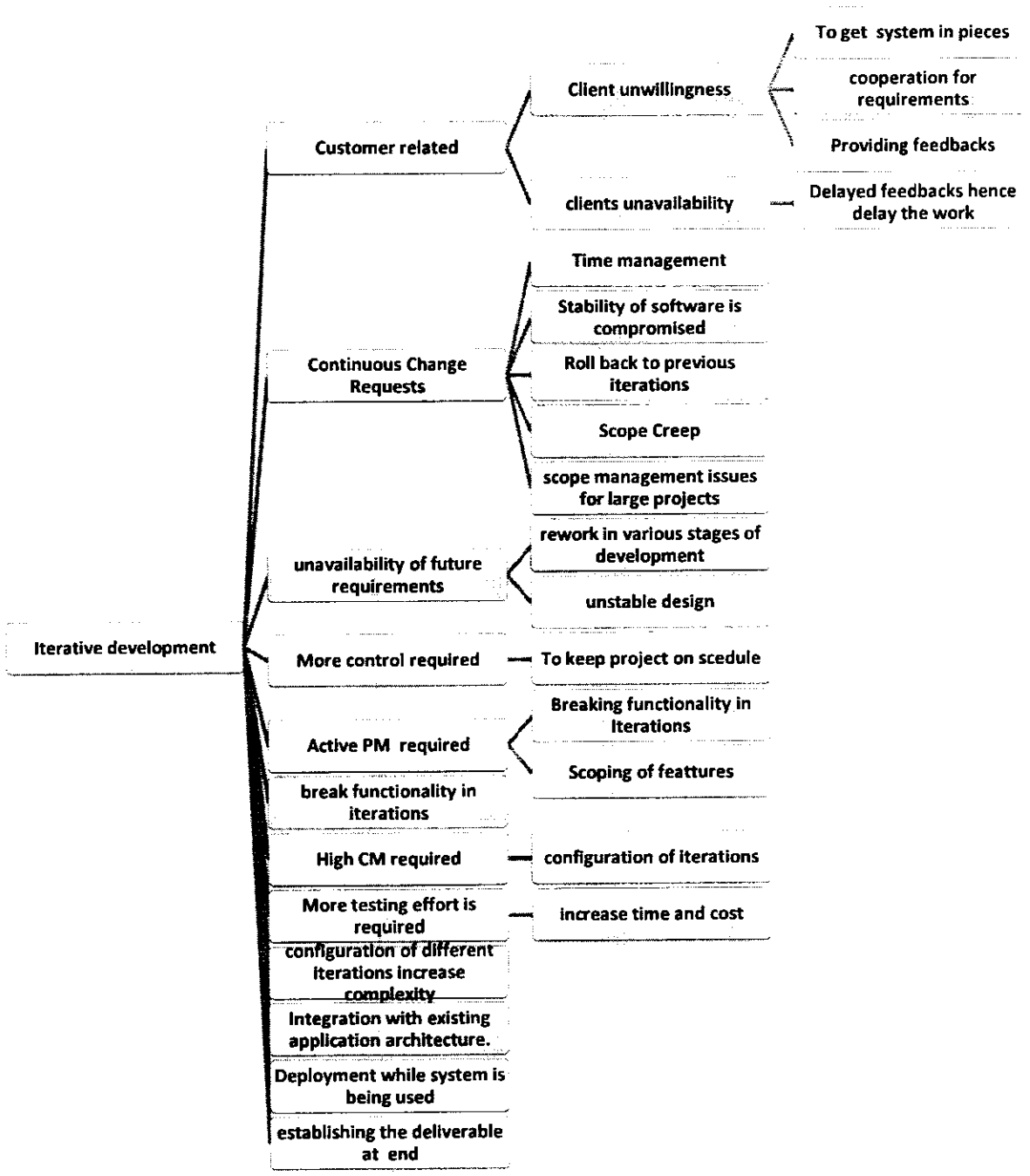


Figure 3.8: Challenges of Iterative Development

Discussion:

Clients' unavailability (S1) happens and it is mainly due to the communication gaps and delays. A more appropriate approach would be to devise a communication plan before starting the project with mutual consensus from all stakeholders. The communication plan should be communicated accordingly via email.

The problem related to *lack of cooperation from Customer (S1)* can be further broken down in to two sub problems:

(1) Lack of cooperation

(2) Requirements are provided later during the terminal stages of the development phase.

This problem occurs in small or low budget projects where neither the client nor the development teams i.e. Business Analysis or PM or the Dev Lead focus on requirements signoff and there is no official or legal mechanism adopted by either of the parties to put an end to the requirements. The parties should formalize a formal contract irrespective of the size of the project and there should be designated PM's from both the parties and the core requirements should be signed-off from the client. In this way the client will not only show interest but also will provide concrete requirements.

Client willingness to get the system in pieces (S2) cannot be considered as problem from clients' point of view. It should be addressed seriously, as the main reasons of adapting agile practices is to ensure efficient delivery of the core requirements to the client first. This cannot be considered as problem from clients' point of view.

Iterative development for large projects result in scope management issues especially with dynamic change in requirements (S3) *which can delay the project due to scope creep* (S4). These issues happen either due to the in competency of the Business Analyst and Development Team Lead or due to lack of interest from client side, which can result in developers frustration and de-motivation due to rework (S5). Hence, a formal contract irrespective of the size of the project should be prepared to finalize the core requirements from the client to make them show interest. This doesn't mean that the change request will not be handled but

only the main core functionality will be decided which could reduce in major scope creep to make the project run out of budget.

The requirement of active project management and control (S7) for time management, control and scoping of features is not an issue instead this is more of a corrective measure. Project management should be experienced and dynamic enough to manage in case of even poor time estimates. Prioritizing and categorizing requirements in iterations or user stories is a difficult task (S6), hence PM should be competent and self-motivated. Scope management is an issue faced only for large projects active project management and control can resolve this issue.

Increased CM effort in case of *frequent releases* (S8) is again more of a requirement than issue as it must already be in place in almost all the IT organizations. Intensive testing should be done in iteration and if planned and done as it is suggested in an agile environment it may reduce the rework.

Integration with existing application architecture (S9) and *Deployment while system is being used* (S10) are commonly occurring issue and will always be an issue. Hence precautionary measures need to be taken by the deployment team. Define a proper deployment strategy before proceeding with the live deployment and it should be done during the time when users/customers' traffic is least, live.

3.4.6.2 Challenges of Incremental Design

The issues initially reported in survey are grouped in main issues and their sub-issues as shown in Figure 3.9. By applying the three-step data analysis process described in section 3.4.6, the challenges are summarized in Table 3.3. The challenges very rarely reported and seemed highly irrelevant are not included in table.

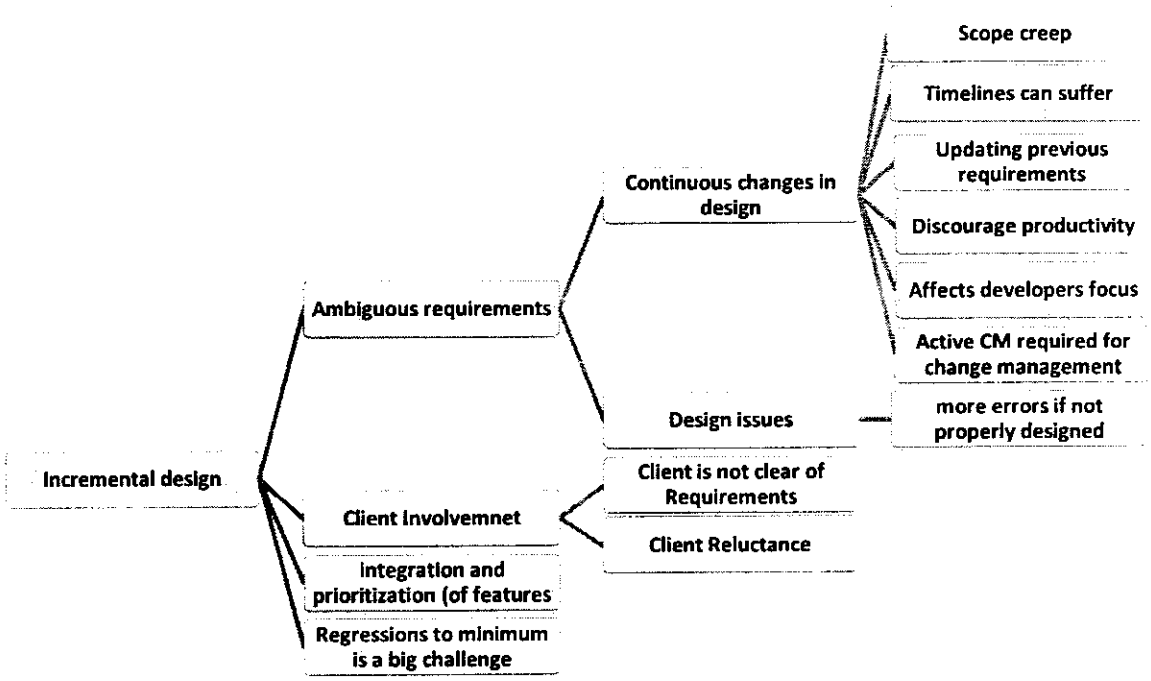


Figure 3.9: Challenges of Incremental Design

Table 3.3: Challenges of Incremental Design

INCREMENTAL DESIGN
S1 Integration and prioritization of features
S2. Difficult to manage for large projects
S3. Client involvement for prioritizing requirements, hence Timelines can suffer
S4. Client is ambiguous about requirements and estimates
S5. Keeping the Regressions to minimum is a big challenge
S6. CM required for change management.
S7. Continuous change in design disturb developers focus and reduces productivity

S8. Improper design results in more errors

S9. Sometimes work is not done and not able to demo to all stakeholders at the end of iteration

S10. Require more effort in term of testing which can increase time and cost constraints

Discussion:

Integration and prioritization of features (S1) is one of the main ideology for Agile development. It is a challenging task, hence active project management and customer support can ease this issue. The agile pre-requisite for Self-motivated and skillful team, adaptive PM and Onsite customer is required for efficient *Integration and prioritization of features.*

Incremental design is easy to manage for small projects and becomes a hectic task when the product/project is big (S2). This is a problem reported by most of the respondents; hence again competent staff is required to solve the issue.

Making the client interested and responsible for providing timely input to prioritize requirements (S3) can be challenging sometimes due to busy schedules and commitments of client. Hence a proper mechanism and schedule can be selected for meeting at the start of the project to assure client availability. The details for this issue are already covered to a certain extent against (S1) and (S2) of Iterative Development. Client is not ambiguous (S4), either the requirements are ambiguous to client or the client is not experienced and knowledgeable to properly address the requirements.

Keeping the Regressions to minimum (S5) depends upon the new development and the complexity of newly implemented requirements that are to be validated during the regression testing phase.

Continuous change in design (S7) really do disturb the focus of not only the developers but also all the other stakeholders as well, results in increased rework, late sittings, work on off days, delayed deliveries, selected user stories not completed during iteration, and above all frustration.

Improper design (S8) results in more work needs to done in a small unit of time, results in more errors. Due to short timelines, the design activity is not given much attention as required which results in delayed schedules, hence delayed projects as more errors are introduced. Therefore the team should be competent and skillful to generate a stable design in short timelines.

Sometimes work is not done and not able to demo to all stakeholders at the end of iteration (S9), hence, the user stories selected and completed in the selected iteration should be demonstrated.

Require testing is effort which can increase time and cost constraints (S10).If testing is done in iteration and if planned and done as it is suggested in an agile environment it may reduce the rework. Time and cost constraints are mainly caused due to rework while planned and timely testing avoids rework.

Establishing the deliverable product at each iteration end can be challenging (S10), hence, the issues and reasons regarding scoping of features for the iteration should be considered for next iterations.

3.4.6.3 Challenges of Time Boxed Iterations

The issues initially reported in survey are grouped in main issues and their sub-issues as shown in Figure 3.10. By applying the three-step data analysis process described in section 3.4.6, the challenges are summarized in Table 3.4. The challenges very rarely reported and seemed highly irrelevant are not included in table.

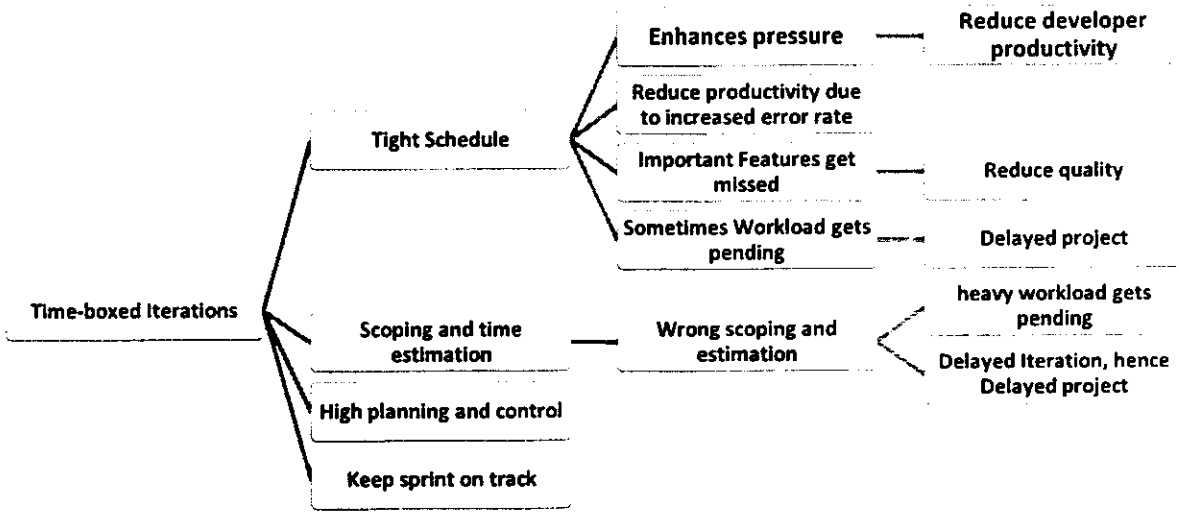


Figure 3.10: Challenges of Time Boxed Iterations

Table 3.4: Challenges of Time-boxed Iterations

TIME BOXED ITERATIONS
S1. Time limitations enhances pressure on developers which reduces productivity
S2. The items not completed in an iteration are moved to next iteration with a result in delayed schedule
S3. High planning and control is required
S4. Scoping and time estimation is tough task
S5. Due to tight schedule, higher error rate and some important features are missed thus reduces quality
S6. Testers cannot thoroughly test the iteration to mark it complete in tight deadlines
S7. Time-boxing iterations is dependent on nature of project

Discussion:

(S1) and (S2) is a challenge as the constraint to complete work in fixed and tight deadlines is a challenging task. Proper planning, scoping for the iteration and good estimation can address the challenge of deferring the incomplete tasks to next iteration and reducing pressures on developers. (S3) is a requirement not an issue which is necessary to resolve (S1) and (S2). (S4) is challenging hence require skilled staff as agile advocates.

Due to tight deadlines, developers tend to miss out some functionality (S5) and testers do not get enough time for testing (S6) which results in decreases quality. The issue is due to the incompetency of staff. If planning is done properly and proper mile stones are set, this issue would not arise. If Agile requirement of self-organizing teams is met, this issue would not arise. The teams should organize and manage their tasks to maintain quality. The review meetings should be scheduled to can create pressure in teams to meet quality requirements. The teams should be involved in Daily Scrum meetings so that they have the pressure to report their work.

Time-boxing iterations need that the project can be divided into modules (S7) and the iterations can be executed in parallel for early delivery. If the functionalities cannot be broken in iterations due to high dependencies in functionality, time-boxing is not suitable.

3.4.6.4 Challenges of Iteration/Sprint Planning Meeting

The issues initially reported in survey are grouped in main issues and their sub-issues as shown in Figure 3.11. By applying the three-step data analysis process described in section 3.4.6, the challenges are summarized in Table 3.5. The challenges very rarely reported and seemed highly irrelevant are not included in table.

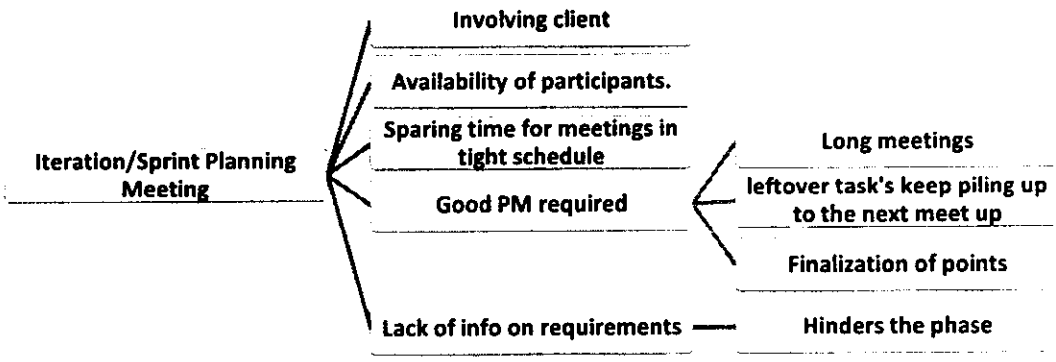


Figure 3.11: Challenges of Iteration/Sprint Planning Meeting

Table 3.5: Challenges of Sprint Planning Meeting

ITERATION/SPRINT PLANNING MEETING
S1. High responsibility on PM for involving client and Team.
S2. Team availability for sparing time for meeting in tight schedules as meetings gets long.
S3. Finalization of points due to suggestions conflicts as each member has different view
S4. Lack of information on requirements
S5. Some functional details unavailability hinders the phase
S6. If not planned properly, leftover task's keep piling up to the next meet up.
S7. Re-planning of not completed stories (in previous iteration) is complex.

Discussion:

A meeting plan should be communicated to all requesting their availability to ease (S1). Sparing time for meeting in tight schedules (S2) is issue for the team due to their tight schedules because the meetings get long. Meetings get long when the stakeholders are unable to come to a conclusion. Hence, an agenda of the meeting should be devised before the start of the meeting and preferably should be mentioned in the email. A meeting evaluation form should be created and distributed by the end of every meeting to rate the productivity level of the meetings, whether the meeting was fruitful or not. This activity can to motivate the stakeholders to be available and enhance their eagerness to learn.

Finalization of points due to Suggestions conflicts as each member has different view (S3) is an issue. All suggestions should be noted and emailed in the form of meeting minutes and it should be the responsibility of the Project Owner or the PM to bring the meeting stakeholders to a conscientious resolution.

Some requirements are ambiguous and are not mature enough to be planned (S5) which can deter the efficiency of this phase. Customers and business analyst should be involved answer the queries regarding requirements. Due to lack of requirements or misunderstandings keeps tasks piling up to the next meet up (S6) which results in re-planning of not completed stories which increase complexity (S7) and delays the project.

3.4.6.5 Challenges of Daily Scrum

The issues initially reported in survey are grouped in main issues and their sub-issues as shown in Figure 3.12. By applying the three-step data analysis process described in section 3.4.6, the challenges are summarized in Table 3.6. The challenges very rarely reported and seemed highly irrelevant are not included in table.

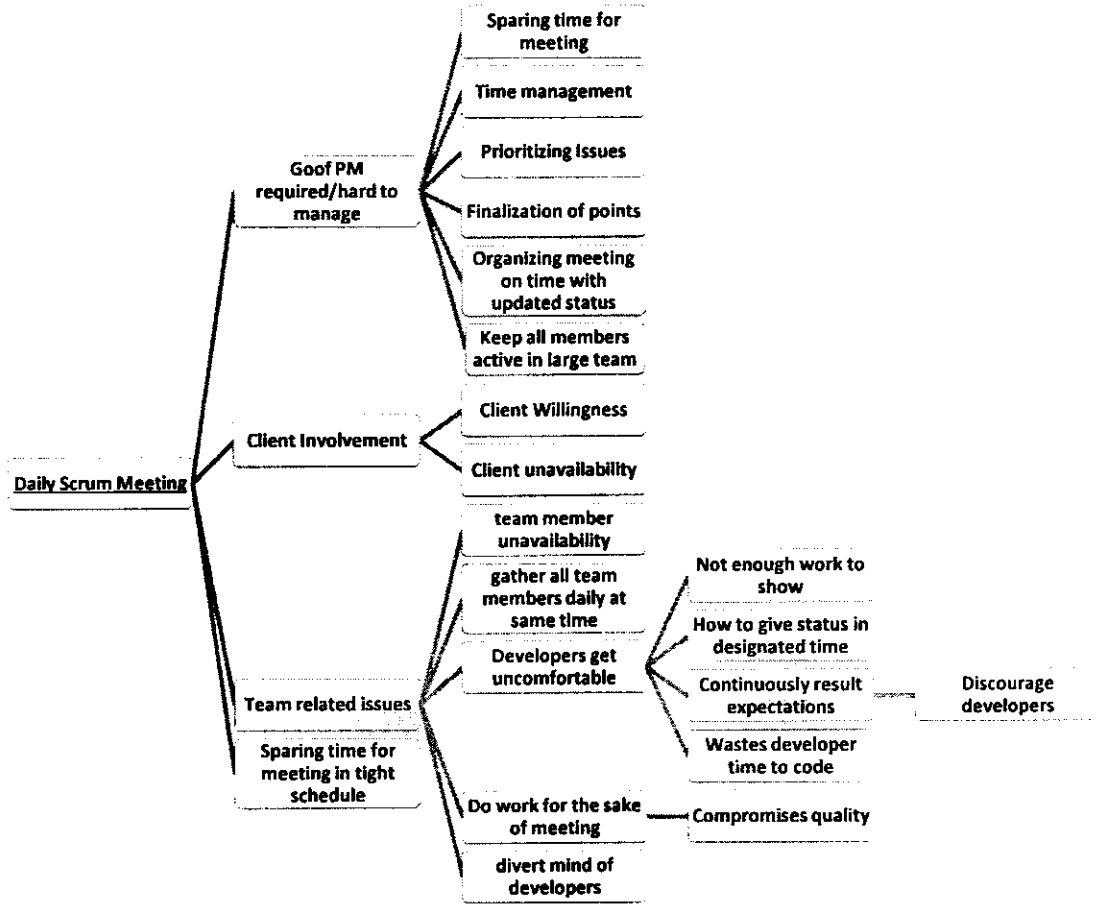


Figure 3.12: Challenges of Daily Scrum

Table 3.6: Challenges of Daily Scrum Meeting

Daily Scrum Meeting
S1. Hard to manage in terms of finalization of points, time, prioritizing issues, conducting on time with updated status.
S2. Long meetings wastes developer time and divert mind of developers from their coding tasks.
S3. Daily reporting discourages developers as there is not enough work to show daily..

S4. Hard to gather all team members daily at same time, sometimes team member are not available.

S5. Sparing time for meeting and involving client.

S6. Suggestions always waste lot of time.

S7. Due to size of team, people have tendency to become inattentive.

S8. In some projects daily scrums are not very practical

Discussion:

Experienced Scrum Master is required to manage meetings with respect to finalization of points, time, prioritizing issues, conducting on time with updated status. Issues regarding (S1) show that the Scrum master is not competent. It is a requirement of Scrum to have experienced Scrum master to implement agile practices in a successful manner.

(S2), (S4), (S5) are the issues faced by most of the people and can be better addressed as the daily scrum meetings should not be more than 10 minutes and ideally it should be done in the first hour of the day when people are more fresh and can contribute more. And for the rest of the day they can accomplish their work without any disruption. Gathering all team and ensuring teams' availability(S4) can be achieved by deciding a fixed time for meeting which seem feasible for everyone, and making it necessary for all to be available at that time.

Daily reporting discourages developers as there is not enough work to show daily (S3) as a result developers focus on daily work which can lose their concern on the purpose of team, and compromises the quality. Some people don't know how to give status in designated time, which can create pressure on their minds.

Suggestions always waste lot of time (S6). Hence it can be useful to prioritize the suggestions; if the goal and scope of the project are clear then suggestions can easily be prioritized and implemented.

Team members' tendency to become inattentive (S7) is faced normally in large teams, with different people having different focus area. Hence the points raised up can be irrelevant for the some people, e.g., points raised by developers seemed irrelevant to others on the team.

In some projects daily scrums are not very practical (S8) due to nature tasks so in those rare cases weekly scrums are preferred. Usually it is done when a project is in a critical condition and need full attention.

3.4.6.6 Challenges of Sprint Review Meeting

The issues initially reported in survey are grouped in main issues and their sub-issues as shown in Figure 3.13. By applying the three-step data analysis process described in section 3.4.6, the challenges are summarized in Table 3.7. The challenges very rarely reported and seemed highly irrelevant are not included in table.

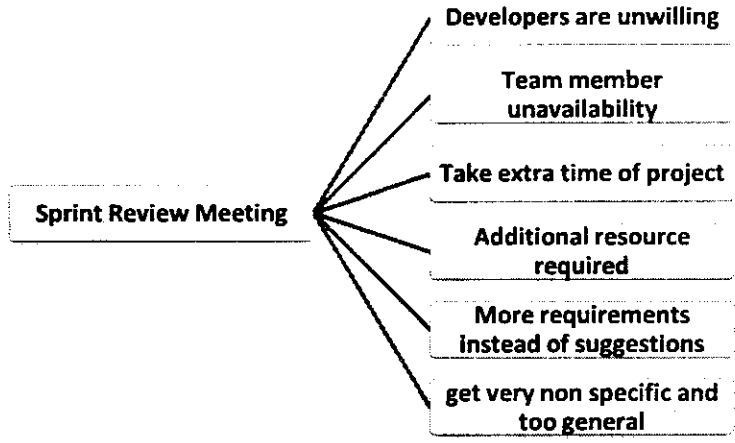


Figure 3.13: Challenges of Sprint Review Meeting

Table 3.7: Challenges of Sprint Review Meeting

SPRINT REVIEW MEETING
S1. Team member unavailability
S2. Developers are unwilling
S3. Take extra time of project
S4. Hard to Meet expectations as it get very non specific and too general
S5. More requirements instead of suggestions
S6. Additional resource required to conduct this meeting, i.e., one person is designated for this task.

Discussion:

Team members are unavailable (S1) and developers are unwilling (S2) due to their tasks or any other commitment. Meeting schedule should be communicated a day before the review meeting is to be conducted to avoid lame excuses for not attending the meeting. These meetings are very useful for the team to clear any misunderstanding as the clients give feedback to the developed product. The meetings are a good learning point for developers hence they should be involved instead of their unwilling behavior.

Review meetings taking extra time of project (S3) is a normal practice, and there are number of reasons behind this problem, few main reasons are:

1. Poor time estimation
2. Incompetent resource(s)
3. Poor change management

These hurdles should be removed to make the meetings effective.

Meetings get very non specific and too general (S4). Hence it gets hard to meet the too much needs of the client exceeding functionality; therefore the project team doesn't get much time to meet the exceeding expectations of the client(s).

More requirements instead of suggestions (S5) should be seen in this way that the Agile advocate to welcome customer requirements. Requirements are for the sake of improvement, not scope creep. The client requirements that are outside the scope of product decided at the start and increasing budget are negotiated.

Requirement of additional resource to conduct this meeting (S6) is not a issue, in fact it is the role of Scrum master and Project managers to conduct these meetings for team learning and get input from the client to make it confirmed that the right product is being build and to improve product quality.

3.4.6.7 Challenges of Product Backlog

The issues initially reported in survey are grouped in main issues and their sub-issues as shown in Figure 3.15. By applying the three-step data analysis process described in section 3.4.6, the challenges are summarized in Table 3.8. The challenges very rarely reported and seemed highly irrelevant are not included in table.

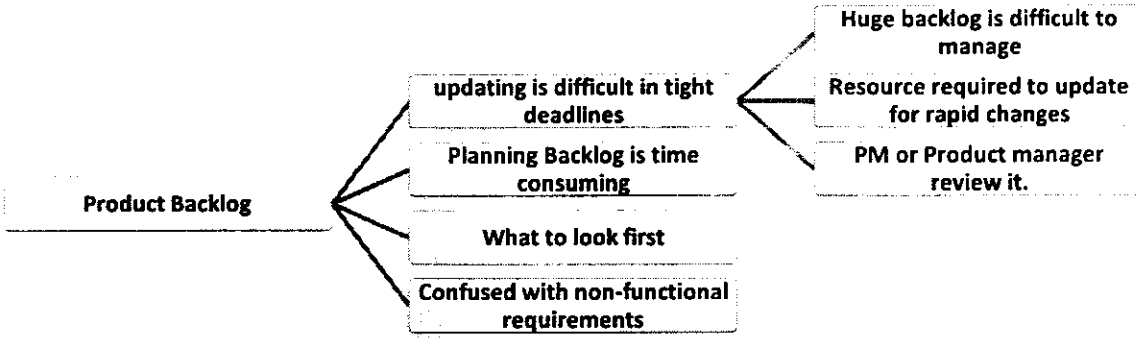


Figure 3.14: Challenges of Product Backlog

Table 3.8: Challenges of Product Backlog

PRODUCT BACKLOG
S1. Difficult to manage/update the product backlog for rapid changes.
S2. Time consumption in planning for product backlog in tight deadlines.

Discussion:

Breaking the requirements in user stories and making the right prioritization according to real and core business needs is not an easy task. Change requests add difficulty for maintain the backlog. (S1) Creating and maintaining backlog is time consuming (S2) to be conducted in tight deadlines. This issues related to product backlog can also be resolved with active customer participation, as advocated by agile development.

3.4.6.8 Challenges of Sprint Backlog

The issues initially reported in survey are grouped in main issues and their sub-issues as shown in Figure 3.16. By applying the three-step data analysis process described in section 3.4.6, the challenges are summarized in Table 3.9. The challenges very rarely reported and seemed highly irrelevant are not included in table.

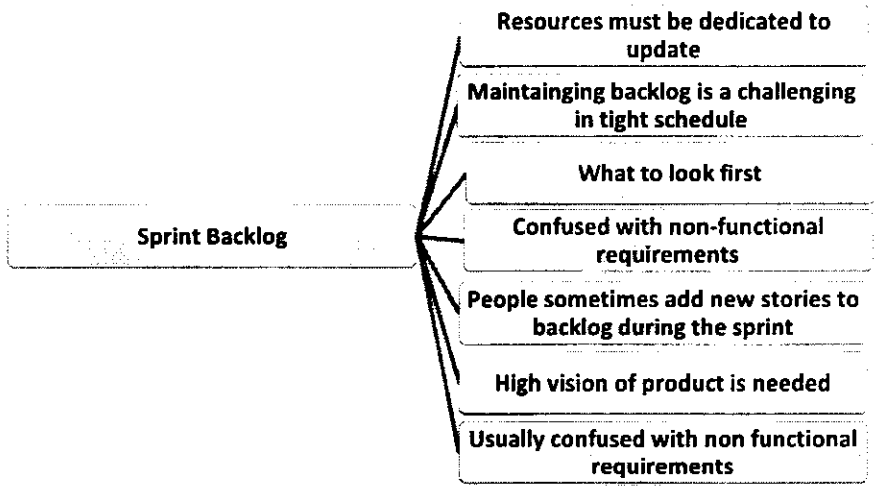


Figure 3.15: Challenges of Sprint Backlog

Table 3.9: Challenges of Sprint Backlog

SPRINT BACKLOG
S1. Continuous updating requires extra resources and time
S2. Backlog take time of other tasks due to tight deadline and this cycle continues.
S3. People sometimes add new stories to backlog during the sprint
S4. What to look first
S5. Usually backlogs get clear by the end of sprint meetings or these are marked as the future work.

Discussion:

Extra resources are required in case of Continuous updating (S1). This should not be an issue as continuous updating is required due to the following reasons:

1. Scope creep
2. Poor requirements management
3. Poor requirements gathering
4. Lack of experience as experience matters a great deal

Backlog should not take time of other tasks (S2) as the responsibilities should be shared, team is doing work and Pm should update it. Hence due to backlog, development cycles should not exceed the defined end time ideally.

People sometimes add new stories to backlog during the sprint (S3) is not a good practice, but does happen. New user stories should be planned in the next iteration. It depends on product that what type of functions it have and what changes come in feature.

Team gets confused for what to look first in backlog (S4). It can be helped by developing the requirements or modules that impact the client business process the most and are of priority in the backlog. Usually backlogs are marked as the future work (S5) is more of an approach instead of issue.

3.4.6.9 Challenges of Burn down Charts

The issues initially reported in survey are grouped in main issues and their sub-issues as shown in Figure 3.17. By applying the three-step data analysis process described in section 3.4.6, the challenges are summarized in Table 3.10. The challenges very rarely reported and seemed highly irrelevant are not included in table.

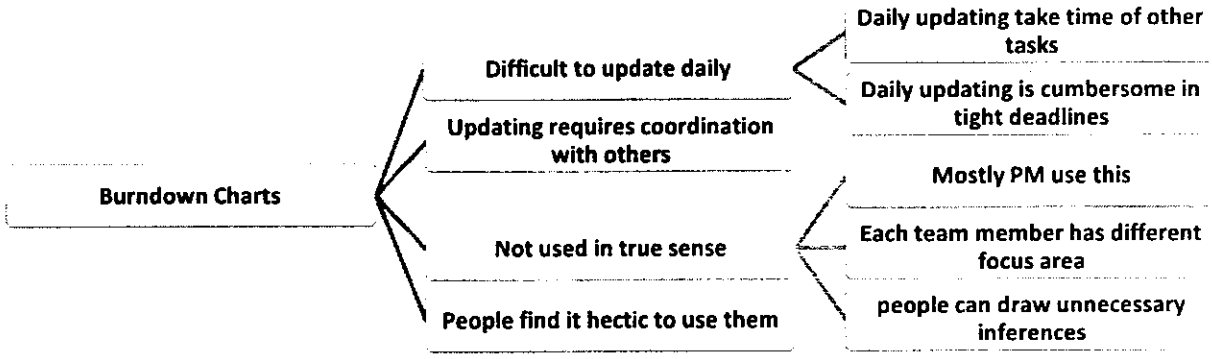


Figure 3.16: Challenges of Burn down Charts

Table 3.10: Challenges of Burn down Chart

BURNDOWN CHARTS
S1. Difficult to update daily as some tasks are partially done/dependent tasks
S2. In tight deadlines, it is cumbersome to update daily
S3. Daily updation can take time of other tasks
S4. Mostly PM use those charts we can't asked to every programmer to follow
S5. Different focus area in the team so burn down is not used in true sense.

S6. People find it hectic to make use of these charts and sometimes draw unnecessary inferences due to lack of understanding of the utility

S7. It's very summarized version of monitoring. Doesn't always allow getting the root cause of the issues.

Discussion:

Updating daily (S2) is not an issue as tools are available which can generate updated charts on a button click. These charts can be generated easily by using Microsoft Team Foundation Server.

The issues reported are not actually issues, this technique is being followed by using tools but the naming conventions create the issue of confusion. For addressing (S7) people should be trained about how to effectively make use of these charts for getting awareness about the project status. (S8) is not an issue as the charts are for the purpose of getting status update.

3.4.6.10 Challenges of TDD & Unit Testing

The issues initially reported in survey are grouped in main issues and their sub-issues as shown in Figure 3.18. By applying the three-step data analysis process described in section 3.4.6, the challenges are summarized in Table 3.11. The challenges very rarely reported and seemed highly irrelevant are not included in table.

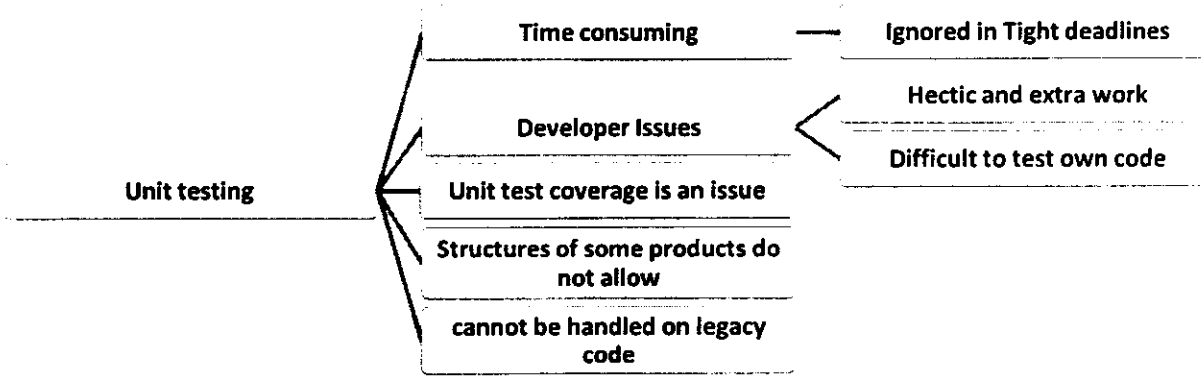


Figure 3.17: Challenges of TDD & Unit Testing

Table 3.11: Challenges of TDD & Unit Testing

TDD & UNIT TESTING
S1. Time consuming. Developer nature is that he/she only tests some common errors for their tasks so there should be a QA testing team
S2. To meet tight deadlines, this is often ignored
S3. Developers think it as hectic and extra work
S4. Developers find it difficult to test own code
S5. Unit test coverage is an issue
S6. Unable to write unit tests with legacy code
S7. Structures of some products do not allow writing unit test with each class.

Discussion:

Developers are used to the traditional way of writing the code and checking it for the common error regarding the desired output for the piece of code. In tight schedules, developers tend to ignore unit testing and rely on testers and QA teams as it is a time consuming activity which requires more effort (S1). Writing the tests prior to coding and writing tests for each class is perceived as hectic and extra effort (S2) by developers. The TDD activity is observed as time consuming (S3) as they write tests for each class before coding; hence less time is left for their primary responsibility of coding. Also, writing detailed tests for own code is perceived as difficult by the programmers (S4). Unit testing is conducted in TDD where tests are written for each class of code. Unit test coverage is an issue (S5) as the tests has to cover each class. If the same developers are writing tests for their code there can be issue of test coverage due to inadequate independent testing. Writing unit tests is not always possible for all products (S7) e.g. developers are unable to write the tests for legacy code as it is not cost effective (S6).

3.4.6.11 Challenges of Continuous Integration

The issues initially reported in survey are grouped in main issues and their sub-issues as shown in Figure 3.19. By applying the three-step data analysis process described in section 3.4.6, the challenges are summarized in Table 3.12. The challenges very rarely reported and seemed highly irrelevant are not included in table.

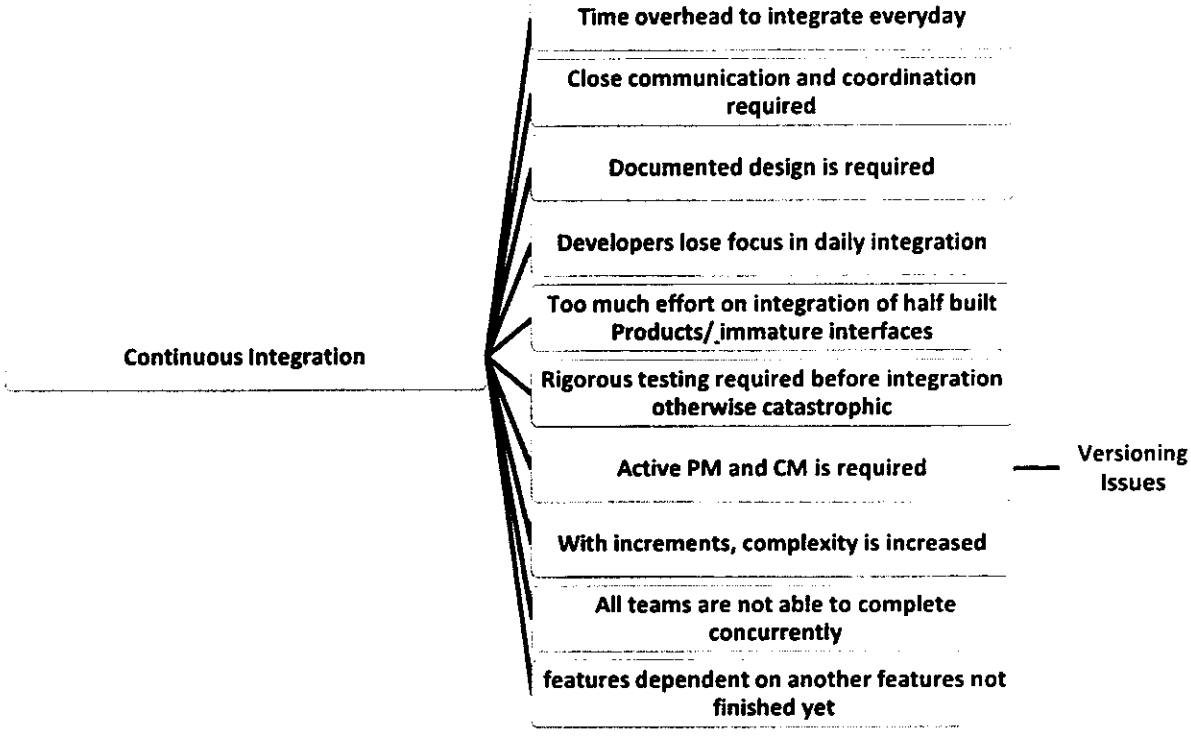


Figure 3.18: Challenges of Continuous Integration

Table 3.12: Challenges of Continuous Integration

CONTINUOUS INTEGRATION
S1. Time overhead to integrate everyday
S2. Close communication and coordination is required

S3. Documented design is required

S4. Developers lose focus in daily integration

S5. Too much effort on integration of half built Products/ unstable and immature interfaces

S6. Integration should be done between stable interfaces

S8. Units not properly tested had integration problems

S9. Difficult to update all iterations in large teams as with multiple code streams complexity is increased

S10. PM and CM is required due to Versioning issues

S11. Sometimes it poses challenges if your project is dependent on another project which is not finished yet so it may not be done at the expected schedule.

Discussion:

Continuous integration is a technique for checking system stability after integrating changes. Continuous Daily integration in order to incorporate the daily completed tasks is reported to be time consuming (S1) in tight deadlines. Developers complaints to loose focus (S4) in their primary tasks if integration has to be done on daily basis. Hence teams use to do integration on weekly basis or at the completion of some important functionality. Integration of not completed features/ half built products requires too much effort as they have not been tested thoroughly which can introduce errors. It is recommended to integrate stable interfaces (S6) after fully testing them (S8) because the errors introduced as a result of integrating unstable interfaces are difficult to track and introduce delays and frustration. The main reason of performing testing during every iteration is to ensure the delivery of stable release by the end of every iteration, which can be integrated without any issues.

Some teams claim to have documented design (S3) for CI, but documentation should be minimal while following Agile development, so it cannot be considered as an issue and teams has to managing it with minimal documentation. Similarly (S10) cannot be treated as issue as in case of change requests and early releases different versions have to be managed by Configuration Manager and Project Manager. (S10) is requirement rather than an issue.

if the modules are highly dependent (S11) on the input from other modules, then CI faces issues as team has to wait for the completion of the related modules to declare a iteration in working form. In case of large teams, CI increase complexity (S9) due to inputs from multiple teams.

3.4.6.12 Challenges of Refactoring

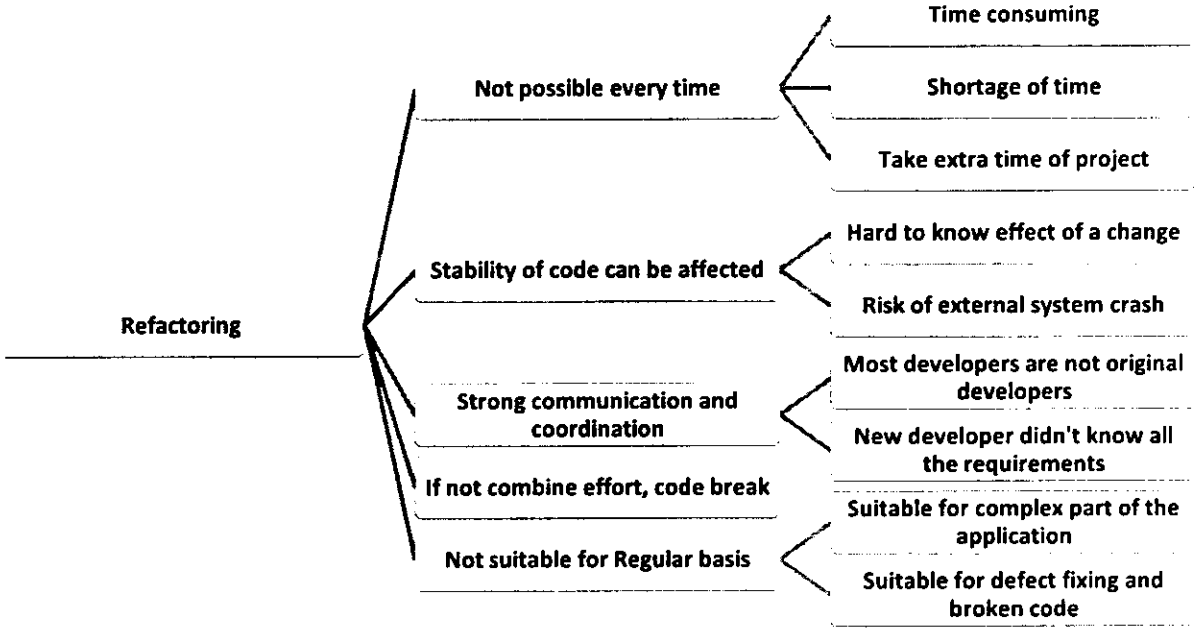


Figure 3.19: Challenges of Refactoring

Table 3.13: Challenges of Refactoring

REFACTORING
S1. Not possible every time because it is time consuming and take extra time of project
S2. Stability of code can be affected
S3. Risk of external system crash
S4. Hard to know effect of a change
S5. Strong communication and coordination is required
S6. If not combine effort, code break

Discussion:

Refactoring is used to improve code quality without changing its functionality. Refactoring is concerned with changing internal structuring without affecting external behavior. The code is refactor to make it simple and readable by anyone and also easy to adapt in case of change requests. The practice is very useful but people tend to abandon it due to lack of experience and expertise and assume it to be time consuming to be conducted in tight deadlines (S1). It should be done carefully because sometimes it is hard to know the effects of change (S4) in case of highly dependent modules or complex systems.

(S3) is not an issue; it can be due to the incompetency and lack of knowledge of the person doing refactoring. As code is developed by multiple programmers hence (S5) is a requirement, not an issue. Lack of combine effort in refactoring can result in code break (S6) is an issue; hence these issues and required level of communication and coordination should be taken into account while adopting refactoring.

3.4.6.13 Challenges of Pair Programming

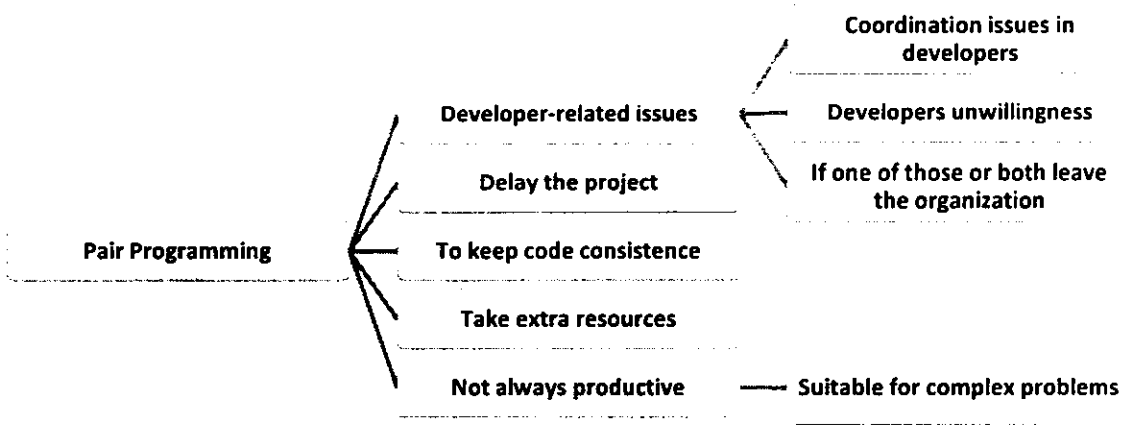


Figure 3.20: Challenges of Pair Programming

Table 3.14: Challenges of Pair Programming

PAIR PROGRAMMING
S1. Coordination issues in developers
S2. Problem occurs if one of those or both leave the organization
S3. Takes more time, if two developers work on same machine
S4. Take extra resources
S5. Developers unwillingness to work in pair
S6. To keep code consistence
S7. Not always productive
S8. Suitable for complex problems only

Discussion:

Pair programming promotes two developers work together to produce same code on the same machine at the same time. Developer's willingness (S5) is an issue in an attempt to adopt PP practice. Developers can be due to the coordination issues (S1) as the experienced developers prefer to work in isolation except for complex problems (S8). Code consistency issues can also be raised in pairs (S6) as each developer has his own way of working. It is perceived that PP tends to slow down development process (S3) as two developers work on the same machine, if they work on their individual systems; the quantity of work is more than that conducted in pair. The code developed by two developers is not always productive (S7) in terms of understanding, testing, reusability and maintenance.

If the development team has limited human resources for completing the project in allocated time than PP cannot be adopted as it is perceived that the two developers working on the same machine can slow the development. Working in PP settings can be risky if any developer leave the company (S2); hence the new developer has to be trained for the system, which can affect the overall productivity and speed of development.

3.4.6.13 Challenges of On-Site Customer

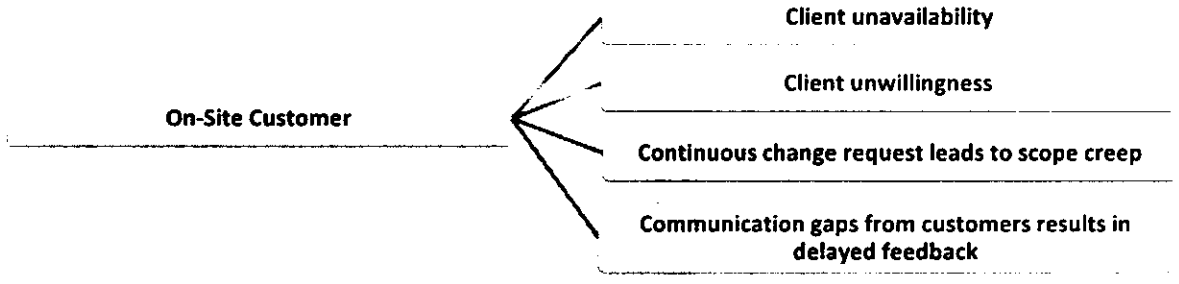


Figure 3.21: Challenges of On-Site Customer

Table 3.15: Challenges of Onsite Customer

ON-SITE CUSTOMER
S1. Client unwillingness /client unavailability
S2. Communication gaps from customers results in delayed feedback.
S3. Continuous change request leads to scope creep
S4. Delayed feedback leads to delayed project

Discussion:

Client unavailability (S1) and communication gaps (S2) results in delayed projects (S4) and developers' frustration as it tends to slow down developers activities in waiting for feedback and requirement prioritization for next iterations. In contradiction, if client is available onsite, another issue can be continuous change requests which can increase scope (S3).These issues can be resolved by committing with the client availability at fixed milestones at the start of the project and client involvement in important decisions so that they got interest in the outputs of iterations.

3.4.6.15 Challenges of Collective Code Ownership

COLLECTIVE CODE OWNERSHIP
S1. Code is not well-designed as no one is responsible individually

Table 3.16: Challenges of Collective Code Ownership

Discussion

Lack of *individual responsibility* is (S1) results in a code with lower quality and design. Without the pressure of individual accountability developers seem less motivated to develop code according to standards.

CHAPTER 4: RESULTS AND DISCUSSION

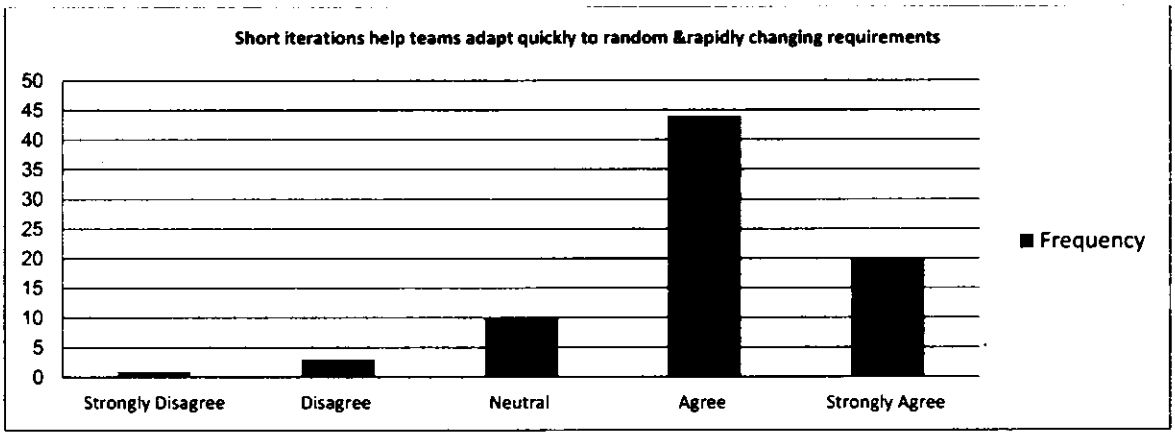
Chapter 4: Results and Discussion

This chapter is focused on the analysis of findings from SLR and Survey. The analysis is performed to discover the similarities and differences between SLR findings and experiences of practitioners. The main objective of this work is to identify the similarities and variations between the findings of SLR in comparison with survey responses.

4.1 Gaps between Benefits reported in Literature and Industrial Survey

The benefits of adopting agile practices for software development projects identified by from industrial survey are quite similar with that of literature studies. The respondents were inquired about the benefits using close ended questions and providing a likert scale to access the level of their consent with the benefits experienced by Agile practices. The scale include five levels to access the strength of their opinion. The scale levels are: (1) Strongly Disagree, (2) disagree, (3) neutral, (4) Agree and (5) Strongly Agree. The last two levels show the positive experiences with Agile practices. Some practices are discussed with graphical presentation of results. It is noted that although most of the practices are adopted by less than 60% but still the respondents are agreeing with the benefits offered by their usage.

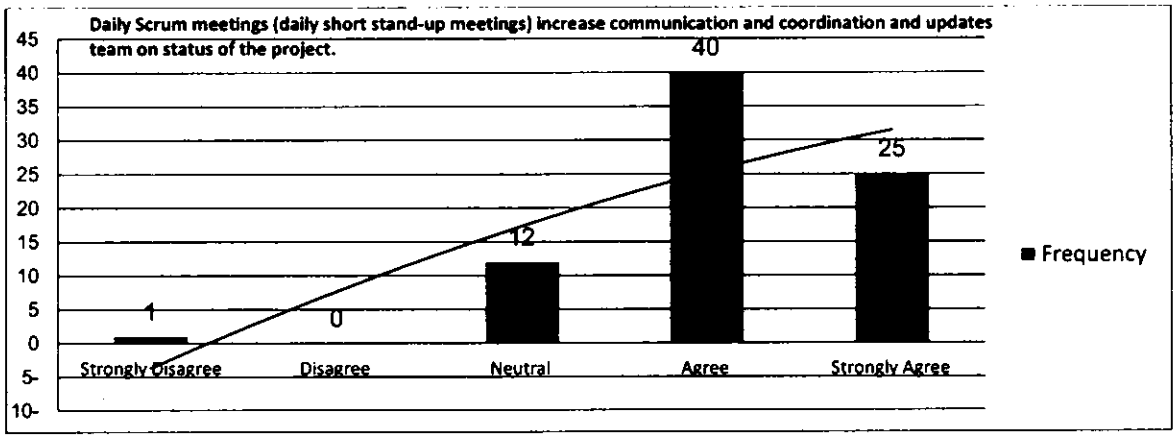
Figure 4.1: Benefits of Iterative Development



More respondents have shown an increased response towards agreeing with the benefits offered by Short iterations in welcoming change and providing flexibility to cope with the changing requirements. Ninety percent of respondents are using iterative development for all projects hence the level of acceptance of benefits associated with Iterative development is

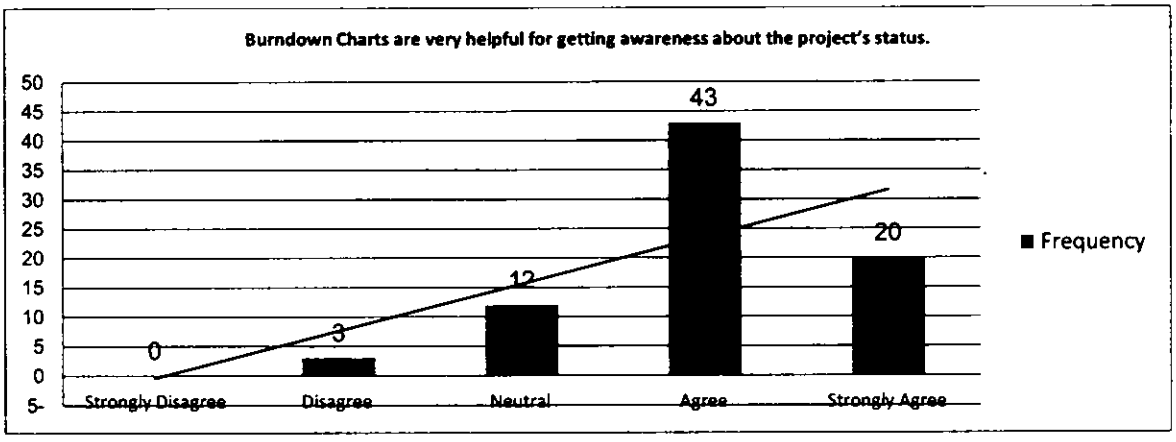
greater than any other practice. It is the highly used practice, also ranked higher in accordance with the acceptance level.

Figure 4.2: Benefits of Daily Scrum Meetings



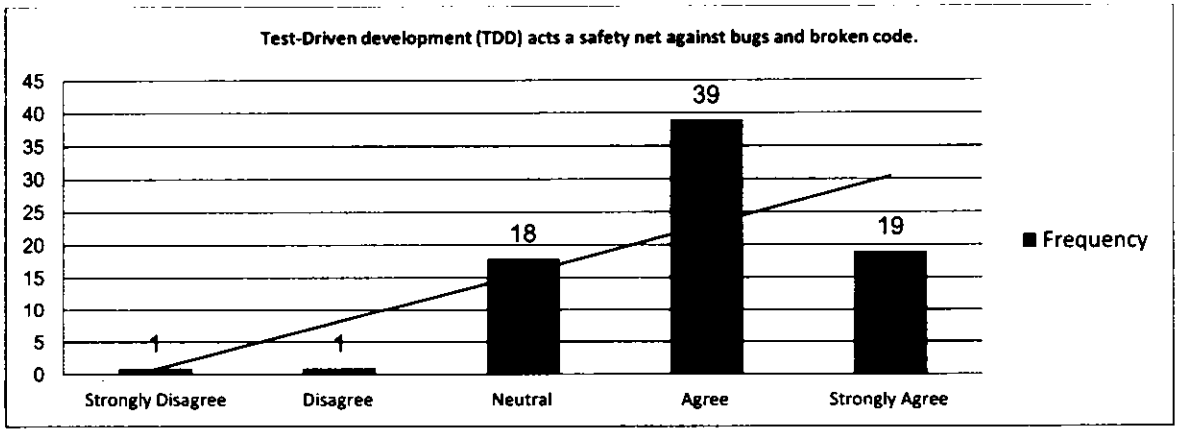
The responses for accepting the benefits of daily Scrum meetings have shown positive results in experiencing Scrum meetings. The neutral responses are due to the fact that some respondents are not using Daily Scrum meetings as they perceive the daily conduct of meetings depends on the complexity of projects, otherwise meetings should be conducted twice a week or weekly. The number of challenges reported against Daily Scrum is the reason for avoiding daily meetings. Seventy percent of survey respondents are conducting the meetings daily. Similarly Iteration planning meeting is used by 56% of the respondents and 59 out of 80 repondents agree with the benefits of Sprint planning meeting. Sprint Review meeting is used by 63% of respondents but still the number of agreed responses are more than neutral responses.

Figure 4.3: Benefits of Burndown Charts



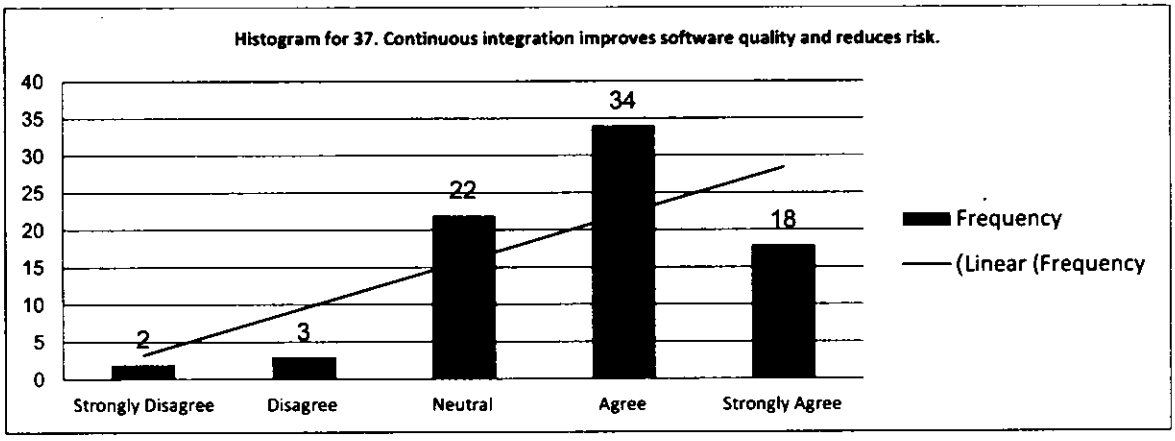
Burndown charts are used by 54% of industry but still the strength of positive responses is larger. The reason is that the people are using other tools for automatic generation of progress reports and there are differences in naming conventions. Agile practices and techniques are used in industry but people are unaware of the terminology names. The neutral responses indicate the reason that some respondents have reported the issues with using burndown charts. The analysis of issues indicates that respondents are unaware of the tools used for automatic generation of these chart. The tools facilitate the easy generation and updating of charts in minutes without taking extra effort and time, hence very beneficial for getting project status information.

Figure 4.4: Benefits of TDD



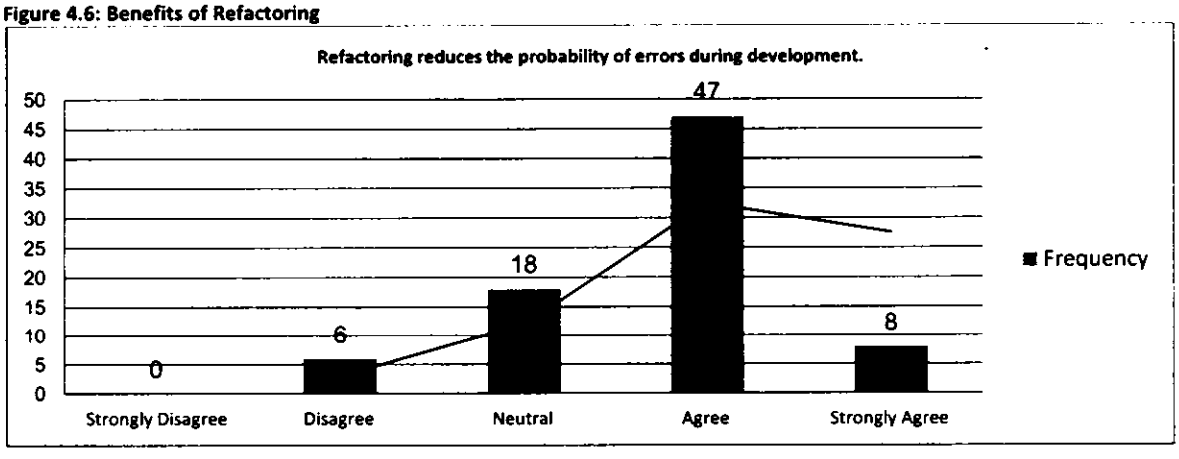
TDD is used by 27% of respondents and unit tests are used by 54% of respondents. Unit testing is used in TDD. Hence the benefits stated are due to the usage of unit testing. More responses are agreeing with the benefits offered with usage of TDD.

Figure 4.5: Benefits of Continuous Integration



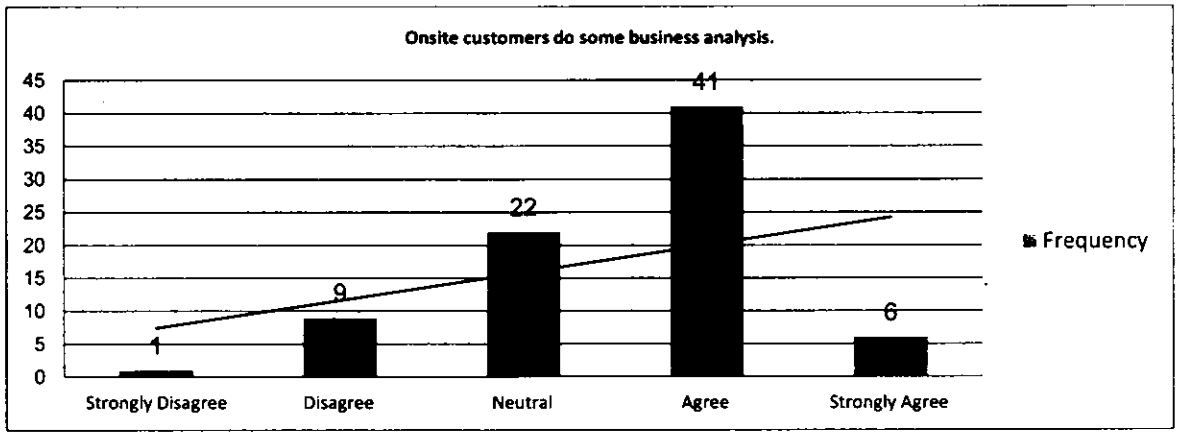
52% of participants use Continuous Integration, still more inclined to accept the benefits. 5 participants are reporting negative experiences with this practice as identified in the challenges section.. The negative findings are due to lesser emphasis communication and coordination mechanisms as recommended for Agile culture. The adoption rate is lesser due to the extensive care and control required for using the practice. Although 52% respondents use it but majority accepts the benefits associated with the use of CI as it is facilitates in detection of problems early which can introduce delays and cost overruns on later discovery. It is also beneficial for keeping the product in releasable form.

Figure 4.6: Benefits of Refactoring



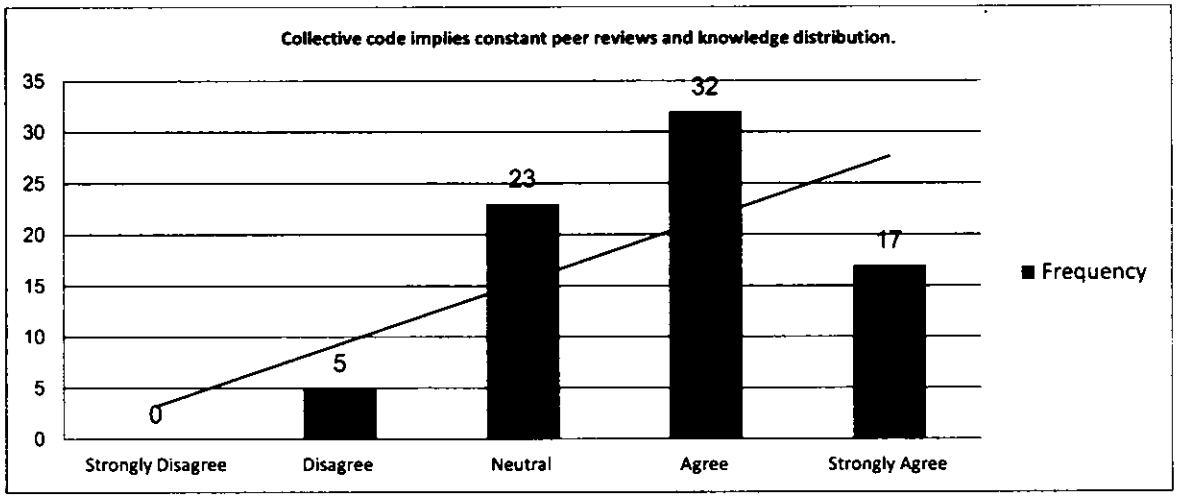
Refactoring is used by 90% of respondents. Hence a large number of respondents agree with its benefits. The respondents giving neutral and negative findings avoid the practice as it needs care and coordination with core developers otherwise it can be catastrophic. The practice is most widely used as Iterative development as it facilitates in maintaining quality of code and reduces overhead cost of maintenance.

Figure 4.7: Benefits of Onsite Customer



35% of respondents claim to use this practice but only for complex projects. The usage is hindered by issues associated with the practice. Although usage is limited but respondents agree with the benefits of using the practice. Onsite customer can help in taking business decisions and can transfer the business knowledge to developers. The customer representative is the preferred alternative, as any team member having good business domain knowledge or business analyst can take decisions on customer behalf. They can visit the customer twice a week. This practice is adapted to take customer input for accepting the results, prioritizing requirements and taking decisions.

Figure 4.8: Benefits of Collective Code Ownership



42% of industry uses this practice but still the respondents agree with the benefits. The issues reported against the practice hinder its usage and level of adoption.

The gap analysis concludes that although the practices are not adopted widely, but the respondents agree with their benefits which show that they are willing to adopt the practices if there guided and trained to overcome the reported.

4.2 Gaps between Issues reported in Literature and Industrial Survey

The issues of adopting agile practices for software development projects have considerable variations, which need to be considered. There are some issues which are reported in literature but not identified by industry practitioners. Similarly some issues reported in survey are not addressed in literature. This variation can be due to the extent of usage of the agile practices, the level of experience and training required regarding the usage of these practices, the required organizational culture for using Agile. The issues for each agile practice reported in survey are mapped with literature and the individual issues not mapped are listed for discussion as shown below:

4.2.1 Comparison of Issues of Iterative Development reported in SLR and Survey

Table 4.1: Comparison of Issues of Iterative Development

	Literature	Survey
Clients' unavailability or Lack of cooperation	4%	7%
Clients' willingness to get the system in pieces	6%	11%
Iterative development for large projects result in scope management issues	4%	5%
Continuous change requests can delay the project due to scope creep	4%	11%
Continuous change requests results in developers frustration	3%	5%
Prioritizing/categorizing/breakdown of requirements in iterations/user stories is a difficult task	8%	7%
Active project management and control is required to manage time very properly and scoping of features	7%	14%
Frequent releases increase CM effort	6%	5%
Integration with existing application architecture	-	2%
Deployment while system is being used	-	5%
Requires more effort in terms of testing and maintenance	19%	11%
Sometimes code integration becomes an issue due to delay from other teams	4%	5 %
Establishing a deliverable product at each iteration end is challenging	-	7%
Configuration of different iterations increase complexity	-	5%

Ninety percent of the respondents in Survey claim to use iterative development, but instead the issues reported are more than those addressed in literature, which is an interesting finding. Issues concerning Clients' *unavailability (S1) are reported in both literature and survey. This problem is* mainly due to the communication gaps and delays. A more appropriate approach would be to devise a communication plan at the start of the project with mutual censes with all stakeholders and should be communicated accordingly via email.

Client willingness (S2) to get the system in pieces is reported in both studies, hence it should be addressed seriously, as the main reasons of adapting agile practices is to ensure efficient delivery of the core requirements to the client first.

Iterative development for large projects result in scope management issues especially with dynamic change in requirements (S3) *which can delay the project due to scope creep* (S4). These issues are reported in both studies, which shows that it is a commonly recurring issue, which can result in developers frustration and de-motivation due to rework (S5). The parties should formalize a formal contract irrespective of the size of the project and there should be designated PM's from both the parties and the core requirements should be signed-off from the client. In this way the client will not only show interest but also will provide concrete requirements. This doesn't mean that the change request will not be handled but only the main core functionality will be decided which could reduce major scope creep to make the project run out of budget.

The requirement of *active project management and control* (S7) for time management, control and scoping of features is reported in both studies which show that it iterative development and is not an easy task for management. Hence, Project management should be experienced and dynamic enough to manage in case of even poor time estimates. Prioritizing and categorizing requirements in iterations or user stories is a difficult task (S6), hence PM should be competent and self-motivated. Scope management is an issue faced only for large and active project management and control can resolve this issue.

Increased CM effort in case of *frequent releases* (S8) is not reported in Literature. It is more of a requirement than issue as it must already be in place in almost all the IT organizations. Intensive testing should be done in iteration and if planned and done as it is suggested in an agile environment it may reduce the rework.

Although the issues regarding *Integration with existing application architecture* (S9) and *Deployment while system is being used* (S10) are not reported in literature but these are the commonly occurring issues and will always be an issue. Hence precautionary measures need to be taken by the deployment team. Define a proper deployment strategy before proceeding

with the live deployment and it should be done during the time when customers' traffic is least.

4.2.2 Comparison of Issues of Incremental Design reported in SLR and Survey

Table 4.2: Comparison of Issues of Incremental Design

INCREMENTAL DESIGN	Literature	Survey
Integration and prioritization of features	20%	14%
Difficult to manage for large projects	-	3%
Client involvement for prioritizing requirements, hence Timelines can suffer	12%	6%
Client is ambiguous about requirements and estimates	18%	17%
Keeping the Regressions to minimum is a big challenge	-	12%
CM required for change management.	-	6%
Continuous change in design disturb developers focus and reduces productivity	-	14%
Improper design results in more errors	25%	12%
Require more effort in term of testing which can increase time and cost constraints	7%	3%
For complex problems, detailed planning and design is required	15%	-

Integration and prioritization of features (S1) is the prerequisite for working in an agile way. It is reported as a challenging task in both studies. Hence active project management and customer support should be provided. The agile pre-requisite for Self-motivated and skillful team, adaptive PM and Onsite customer is required for efficient *Integration and prioritization of features.*

Incremental design is easy to manage for small projects and becomes a hectic task when the product/project is big (S2). This is a problem reported by most of the respondents as 54% of the projects reported in Survey use teams greater than 10, therefore, Considerable attention is required for this issue.

Making the client interested and responsible for providing timely input to prioritize requirements (S3) can be challenging sometimes due to busy schedules and commitments of client. As this issue is confirmed from literature hence it should be taken into account. The solution can be a proper mechanism and schedule can be selected for meeting at the start of the project to assure client availability. Client is unaware of clear requirement is a commonly recurring issue both in Literature and Survey findings. The requirements are often ambiguous to client as the client is not experienced and knowledgeable to properly address the requirements.

Keeping the Regressions to minimum (S5) depends upon the new development and the complexity of newly implemented requirements. This issue is solely reported in survey.

Continuous changes in design (S7) really do disturb the focus the developers is not reported in literature but we agree with this challenge. Change request results in increased rework, late sittings, work on off days, delayed deliveries, selected user stories not completed during iteration, and above all frustration.

The issue of *Improper design* (S8) and (S9) due to short timelines is not reported in literature. It might be due to the inexperienced staff. Therefore the team should be competent and skillful to generate a stable design in short timelines.

Require testing is effort which can increase time and cost constraints (S10) is reported by both studies. If testing is done in iteration and if planned and done as it is suggested in an agile environment it may reduce the rework. Time and cost constraints are mainly caused due to rework while planned and timely testing avoids rework.

Requirement of detailed design and planning for complex projects is (L6) is reported in literature only. This issue is indirectly reported in Survey studies regarding the insufficient time for detailed design and planning in tight deadlines. Large projects often become complex due to their size, hence the designed should be detailed and some design documents should be maintained due to reduced communication and coordination in large teams.

4.2.3 Comparison of issues of Time Boxed Iterations reported in SLR and Survey

Table 4.3: Comparison of Issues of Time-boxed Iterations

	Literature	Survey
Time limitations enhances pressure on developers which reduces productivity	25%	26%
The items not completed in an iteration are moved to next iteration with a result in delayed schedule	27%	22%
High planning and control is required	-	17%
Scoping and time estimation is tough task	-	26%
Due to tight schedule, higher error rate and some important features are missed thus reduces quality	-	13%
Testers cannot thoroughly test the iteration to mark it complete in tight deadlines	-	9%
Pressure on project managers is increased due to increased need for communication and coordination	25%	-

(S1) and (S2) are also addressed in literature. Proper planning, scoping for the iteration and good estimation can address the challenges. (S3) and (S4) are not reported in literature. We are not agreed with these issues as these are required to be accomplished for using agile.

(S5) (S6) are caused due to the incompetency of staff nor they are experienced in Literature studies. Self-organizing teams and motivated teams are required to avoid such issues. The teams should be involved in Daily Scrum meetings so that they have the pressure to report their work.

For time-boxing iterations (S7) the product should be of modular nature. This is a concrete issue and covered in both studies.

Literature has posed an additional challenge (not reported in Survey) of increased pressure on PM (L2). In Agile environment PM role is to facilitate communication and coordination, hence this practice increases the PM's responsibility

4.2.4 Comparison of issues of Iteration/Sprint Planning Meeting

Table 4.4: Comparison of Issues of Sprint Planning Meeting

ITERATION PLANNING	Literature	Survey
High responsibility on PM for involving client and team	-	7%
Team availability for sparing time for meeting in tight schedules as meetings gets long	31%	18%
Finalization of points due to suggestions conflicts as each member has different view	20%	18%
Lack of information on requirements	10%	4%
Some functional details unavailability hinders the phase	12%	4%
If not planned properly, leftover task's keep piling up to the next meet up.	-	7%
Re-planning of not completed stories (in previous iteration) is complex.	-	7%
During planning, the factors: tight deadlines set by upper management and scope creep cause frustration and difficulty for the team	21%	-

We do not consider (S1) as an issue, as it is PM's responsibility to involve the team and customer. This issue can be addressed by devising a meeting plan and delegated to stakeholder to ensure their availability.

Sparing time for meeting (S2) is issue for the team due to their tight schedules because the meetings get long. (S2) is also identified in Literature studies. Therefore, an agenda of the meeting should be devised before the start of the meeting and preferably should be mentioned in the email. A meeting evaluation form should be created and distributed by the end of every meeting to rate the productivity level of the meetings, whether the meeting was fruitful or not. This activity, to motivate the stakeholders to be available and enhance their eagerness to learn.

(S3) is an issue commonly faced in using this practice but this issue is not stated in literature. The reason can be that Agile advices that the meetings facilitator (PM or Scrum Master) should be experienced enough to properly run the meetings. An ideal approach to handle this

issue can be that all suggestions should be noted and emailed in the form of meeting minutes, if finalization of points becomes issue sometimes.

Issue of ambiguous and immature requirements (S4) (S5) are also addressed in literature. Hence the availability of client should be a constraint for these meetings. Due to lack of requirements, tasks may pile up to the next iterations (S6). Re-planning of not completed stories increase complexity (S7) and delays the project. (S7) is a real issue addressed in both studies. Some planning issues as scope creep and fixed timelines cause irritation and difficulty for the team (L4). This issue is also addressed in survey too.

4.2.5 Comparison of issues of Daily Scrum Meeting reported in SLR and Survey

Table 4.5: Comparison of Issues of Daily Scrum Meeting

Daily Scrum Meeting		
Hard to manage in terms of finalization of points, time, prioritizing issues, conducting on time with updated status.	-	26%
Long meetings wastes developer time and divert mind of developers from their coding tasks.	11%	9%
Daily reporting discourages developers as there is not enough work to show daily.	25%	18%
Hard to gather all team members daily at same time, sometimes team member are not available.	-	21%
Sparing time for meeting and involving client	-	6%
Suggestions always waste lot of time.	-	6%
Due to size of team, people have tendency to become inattentive.	-	6%
In some projects daily scrums are not very practical	-	12%
Get ineffective if Scrum master is not much experienced.	-	30%

Experienced Scrum Master is required to manage meetings with respect to finalization of points, time, prioritizing issues, conducting on time with updated status. Issues regarding (S1) show that the Scrum master is not competent. It is a requirement of Scrum to have experienced Scrum master to implement agile practices in a successful manner.

(S4), (S5) are the issues discussed in Survey only. This results due to wrong implementation of this practice because it should not take more than 10 minutes and done in the first hour of the day when people are mostly available. Waste of time in suggestion (S6) is reported in Survey only. If the goal and scope of the project are clear to everyone, this issue would not arise. The Issue of *Daily reporting frustrates developers* (S3) is addressed in both studies which shows the reality and seriousness of this issue. A pressure on developers is created which results in losing focus from their tasks that ultimately reduces the quality of their work.

Team members' tendency to become inattentive (S7) is also verified from literature study focusing on application of Agile methods in large teams. It is due to the reason that different people having different focus area. Hence this practice should be applied carefully to raise the interest of team members and an effective and trained Scrum Master is required for such purpose (L1). Getting a competent Scrum master is challenging according to Literature. The same issue is reported for other practices in Survey.

Survey findings report that for some projects daily scrums are not very practical (S8). This is possible if the project context and scope is clear. Agile methods are not intended for projects with uncertain requirements. Waterfall is preferable approach for clear cut requirements. Still some people think that Scrum meeting are not necessary to be conducted daily.

4.2.6 Comparison of issues of Sprint Review Meeting reported in SLR and Survey

Table 4.6: Comparison of Issues of Sprint Review Meeting

SPRINT REVIEW MEETING	Literature	Survey
Team member unavailability	-	18%
Developers are unwilling	6%	18%
Take extra time of project	-	23%
Hard to Meet expectations as it get very non specific and too general	-	18%
More requirements instead of suggestions	-	5%

Only one issue is reported both in Literature and Survey. i.e. (S2) developers' unwillingness. Since these meetings are very useful as the requirement issues and customer feedback are the core concerns of these meetings, hence some strategies should be devised to ensure availability of team. A suggestion is to communicate a meeting schedule a day before the review meeting to ensure their availability.

(S1) is not reported in Literature but the above mentioned strategy can solve this issue. (S3), (S4), (S5) and (S6) are not verified by literature but are discussed in previous section 4.4.5.6.

4.2.7 Comparison of issues of product backlog reported in SLR and Survey

Table 4.7: Comparison of Issues of Product Backlog

PRODUCT BACKLOG	Literature	Survey
Difficult to manage/update the product backlog for rapid changes.	27%	39%
Managing and updating takes much time in tight deadlines.	10%	23%
Product backlogs creation and prioritization is difficult.	17%	-

Product backlog is a list of requirements arranged in an order of highest priority requirements at the top. It is the responsibility of customer or customer representative to

provide a list of requirements which is prioritized according to the high value business needs. Breaking the requirements in user stories and making the right prioritization according to real and core business needs is not an easy task. Hence, the primary prioritization of product backlog is a tough task (L1). Change requests from the client is a usual issue, hence the dynamic updating and maintaining of backlog is challenging (S1) and time consuming (S2) in tight deadlines.

Client involvement and active project team participation should be made to prioritize the requirements This issues related to product backlog can also be resolved with active customer participation, which indeed is a major aspect of agile development. Product managers should do continuous work on updating and maintaining backlog to reflect continuous changes.

4.2.8 Comparison of issues of Sprint Backlog reported in SLR and Survey

Table 4.8: Comparison of Issues of Sprint Backlog

SPRINT BACKLOG	Literature	Survey
Continuous updating requires extra resources and time	-	13%
Backlog take time of other tasks due to tight deadline and this cycle continues.	-	17%
People sometimes add new stories to backlog during the sprint	-	22%
What to look first	-	4%
Usually backlogs get clear by the end of sprint meetings or these are marked as the future work.	-	4%

Challenges of Sprint backlog are not discovered in the literature studies considered in this work. *Required of extra resources for continuous updating* (S1) is not n issue as at happens due to inexperienced team. Main reasons for updating backlog multiple times are poor requirements management, poor requirements gathering and lack of experience.

It is the responsibility of product manager or project manager to maintain backlogs, hence the deadlines cannot exceed by maintaining it (S2). *Adding new stories to backlog during the sprint* (S3) is not a good practice, but does happen. It is suggested by agile advocates to add new

user stories in the next iteration. (S4) can be helped by developing the requirements or modules that impact the client business process the most and are of priority in the backlog. Usually backlogs are marked as the future work (S5) is more of an approach instead of issue.

Comparison of issues of Burn down Charts reported in SLR and Survey

Table 4.9: Comparison of Issues of Burn down Charts

BURNDOWN CHART	Literature	Survey
Difficult to update daily as some tasks are partially done/dependent tasks	-	15%
In tight deadlines, it is cumbersome to update daily	-	35%
Daily updation can take time of other tasks	-	8%
Mostly PM use those charts we can't asked to every programmer to follow	-	9%
Different focus area in the team so burn down is not used in true sense.	-	12%
People find it hectic to make use of these charts and sometimes draw unnecessary inferences due to lack of understanding of the utility	-	4%
It's very summarized version of monitoring. Doesn't always allow getting the root cause of the issues.	-	

The reported literature does not contain the issues about burn down charts but survey study has identified the issues. The issues reported are not actually issues as tools are available which can easily generate and update charts in minutes. These charts can be generated easily by using Microsoft Team Foundation Server.

Comparison of issues of TDD & Unit Testing Reported in SLR and Survey

Table 4.10: Comparison of Issues of Unit Testing

	TDD	Literature	Survey
Developer nature is that he/she only test some common errors for their tasks so there should be a QA testing team.		25%	3%
Time consuming. To meet tight deadlines, this is often ignored		L6	39%
Developers think it as hectic and extra work		30%	21%
Developers find it difficult to test own code		24%	15%
Unit test coverage is an issue		25%	21%
Unable to write unit tests with legacy code		35%	26%
As designers, developers and testers are supposed to work together; hence verification and validation can be biased.		21%	-
TDD has no effect on quality and productivity.		24%	-
External quality of the product is reduced by overlooking high level tests in tight schedule.		18%	-
Developers has to write tests before coding, hence less time is left for coding tasks.		30%	-

The issues reported in literature about test-driven development and unit testing are the almost the same as reported in Survey. Following TDD developers are forced to write the test for designated functionality prior to writing code. Unit testing is conducted in TDD where tests are written for each class of code. Unit test coverage is an issue (S5) as the tests has to cover each class. If the same developers are writing tests for their code there can be issue of test overage due to inadequate independent testing. Writing unit tests is not always possible for all products (S6), it depends on the structure of product. Writing unit tests for legacy code is expensive to be conducted in cost constraints; hence developers are unable to write the tests for legacy code (S7). Developers perceive it as extra work (S3) as they have to do two tasks at the same time i.e. writing tests and code. In tight deadlines this may produce pressure

on developers as they have to complete the functionality in designated time. Hence in tight deadlines developers seem to abandon writing tests (S2) before code. They only test the common errors and expect the testers and the Quality Assurance teams to test the code (S1). Also developers find it hectic to test their own code (S4) and prefer to dedicate their efforts for coding. Due to extreme emphasis on developers for writing tests for each class, sometimes less time is left for coding (L13), which can have negative impacts on productivity and quality. Agile development advocates that developers, designers and testers should work together which can have negative impact on productivity as the testing efforts can be biased. This concept diminishes the advantages of independence testing where Testers and QA independently test the code (L9). As compared to traditional testing at the end of coding, teams following TDD spend more time on testing each class with a result in less time left for high level tests which can impact external quality that affects the maintainability of product (L12). Hence in some cases, TDD has no effect on quality and productivity (L11) especially if the high level tests are abandoned due to lack of time and if independent testing is totally ignored by just relying in developers.

Comparison of issues of Continuous Integration reported in SLR and Survey

Table 4.11: Comparison of Issues of Continuous Integration

CONTINUOUS INTEGRATION	Literature	Survey
Time overhead to integrate everyday		31%
Close communication and coordination is required		11%
Documented design is required		3%
Developers lose focus in daily integration		3%
Too much effort on integration of half built Products/ unstable and immature interfaces		8%
Difficult to update all iterations in large teams as with multiple code streams complexity is increased	33%	17%
PM and CM is required due to Versioning issues		14%
If the project is dependent on another project which is not finished yet		6%
Cannot be applied for tasks having clear boundaries	12%	-
Team should be aware of agile practices in addition to having single codebase	29%	-

Continuous integration is used to ensure system stability after introducing changes. More issues are reported against continuous integration in relation to those reported in literature. Daily integration in order to incorporate the daily completed tasks is reported to be time consuming (S1) in tight deadlines. Developers complaints that they lose focus (S4) in their primary tasks if integration has to be done on daily basis. Hence teams use to do integration on weekly basis or at the completion of some important functionality. Integration of not completed features/ half built products requires too much effort as they have not been tested thoroughly which can introduce errors. It is recommended to integrate stable interfaces (S6) after fully testing them (S8) because the errors introduced as a result of integrating unstable interfaces are difficult to track and introduce delays and frustration. The main reason of

performing testing ,during every iteration is to ensure the delivery of stable release by the end of every iteration, which can be integrated without any issues.
Some teams claim to have documented design (S3) for CI, but documentation should be minimal while following Agile development, so it cannot be considered as an issue and teams has to managing it with minimal documentation. Similarly (S10) cannot be treated as issue as in case of change requests and early releases different versions have to be managed by Configuration Manager and Project Manager. (S10) is requirement rather than an issue.

If the modules have clearly defined boundaries (L1) and there are no change requests then continuous Integration is not much applicable. In addition if the modules are highly dependent (S11) on the input from other modules, then CI faces issues as team has to wait for the completion of the related modules to declare a iteration in working form.

CI faces challenges for large teams (S9) due to multiple code streams. CI should be done with care and coordination with the responsible team leads and developers as the large number inputs from multiple teams increase complexity. Active CM is required to control versioning issues. Single code base is adopted for continuous integration but still the teams need to be trained (L3) to effectively apply CI. (L3) is more of an agile requirement than an issue.

Comparison of issues of Refactoring reported in SLR and Survey

Table 4.12: Comparison of Issues of Refactoring

Not possible every time because it is time consuming and take extra time of project		25%
Stability of code can be affected	27%	21%
Risk of external system crash		4%
Hard to know effect of a change		17%
Strong communication and coordination is required		13%
If not combine effort, code break		8%
Refactoring does not guarantee whole improvement in quality of software	30%	-

Refactoring is very useful but time consuming activity and difficult to practice in tight deadlines (S1). But it should be adopted instead of tight schedules as it tends to positively optimize code. Sometimes refactoring can introduce errors because it is hard to know the effects of change (S4) in case of highly dependent modules or complex systems. Hence it should be conducted with care and preferably it should be done for fixing defects and treating broken code. To avoid the risks of code break (S6) refactoring should be conducted in coordination (S6) of the most relevant developers; therefore, it should be a combined effort. Strong communication and coordination is a pre-requisite for this agile practice.

System crash cannot happen due to refactoring (S3), it can be due to the incompetency and lack of knowledge of the team. The main motive behind refactoring is that it ease the process of introducing changes in code as the code is simple non-repeating. Hence it does not mean that refactoring tend to improve the external quality (L2) of software regarding its desired output.

Comparison of issues of Pair Programming reported in SLR and Survey

Table 4.13: Comparison of Issues of Pair Programming

PAIR PROGRAMMING	...ure	...iterat...	Sur...
Coordination issues in developers	24%	16%	
Problem occurs if one of those or both leave the organization	-	4%	
Takes more time, if two developers work on same machine, it may be delay the project, whereas client requires more output	-	24%	
Take extra resources	35%	28%	
Developers unwillingness to work in pair	33%	8%	
To keep code consistence	-	4%	
Not always productive	-	8%	
Suitable for complex problems	-	8%	

PP depends on the developer's willingness (S5) to work in pair. Some experienced developers tend to work in isolation and feel disturbed while working with others. This can be due to the coordination issues (S1) in developers or code consistency issues (S6) as each developer has his own way of working. It is perceived that PP tends to slow down development process (S3) as two developers work on the same machine, if they work on their individual systems; the quantity of work is more than that conducted in pair. The code developed by two developers is not always productive (S7) because PP is sometimes adapted to pair experienced and novice developers for the purpose of training and to utilize the inexperienced staff. It is necessary that PP can improve the overall productivity (S7) of code in terms of understanding, testing, reusability and maintenance. To produce the high quality code and resolve the coordination issues both developers should be of same caliber and qualification. But again, if the experienced developers work in isolation, the development can be faster. Hence PP is conducted in complex situations (S8) where competent programmers can work to explore the best solutions.

Working in PP settings can be risky if any developer leave the company (S2); hence the new developer has to be trained for the system, which can affect the overall productivity and speed of development. Code should be properly commented so that the new resources should not have problems in understanding the code. PP is not feasible in case of limited resources (S4), as the client requires more output in less time. By limited resources, we mean, number of staff as compared to project size and time limits; hence it is not possible to make two developers work on same machine.

It is suggested to do PP for complex situations in case of expert pair and for training in case of expert-novice pair.

Comparison of issues of On-Site Customer reported in SLR and Survey

Table 4.14: Comparison of Issues of Onsite Customer

ON-SITE CUS...	...ure	Survey
Client unwillingness /client unavailability		37%
Communication gaps from customers results in delayed feedback.	30%	15%
Continuous change request leads to scope creep		30%
Customer cannot be assigned for availability during the whole development phase	29%	-
It is not reasonable to daily give conveyance to the customer for coming to the site	12%	-

Client unavailability (S1) and communication gaps (S2) with clients are commonly recurring issues also reported in survey. These issues results in delayed projects (S4) and developers frustration as it tends to slow down developers activities in waiting for feedback and requirement prioritization for next iterations. This issue ultimately affects the productivity. If developers do not wait for feedback and the client is involved later, it results in change requests which can result in scope creep (3) and developers' disappointment due to rework. This issue can be resolved by committing with the client availability at fixed milestones at the

start of the project and client involvement in important decisions so that they got interest in the outputs of iterations.

Literature has identified two more issues. According to L2 and L3 it is not affordable to bring clients daily, also client cannot commit to be available on such a regular basis. Hence there should be a customers' representative from the team having good knowledge of application domain. The representative should visit the client twice a week to discuss the issues requiring client input.

Comparison of issues of Collective Code Ownership reported in SLR and Survey

Table 4.15: Comparison of Issues of Collective Code Ownership

COLLECTIVE CODE [illegible]	Literature	Survey
Extra resources required	-	32%
Extra burden	-	8%
Lack of individual responsibility leads to decreased code quality	20%	-
Not suitable for large or complex systems with many modules	35%	-

The challenges of collective code ownership are also reported in survey. The issue regarding lack of *individual responsibility* is (S1) confirmed from survey too which shows the severity of the issue. As the individual developer is not responsible for the quality of final code, the motivation and pressure to develop code according to standards is reduced. (S2). For *large and complex projects* (2), CI is not possible as there is much iteration. All team members are not aware and experienced with the development of complex systems; hence collective code ownership is not possible in this case.

Chapter 5: Conclusion

Chapter 5

5.1 Conclusion

This study focuses on identifying the issues and benefits faced by applying agile practices in software development projects. The aim is to investigate the gaps in State-of art and State-of-practice regarding the implementation of agile practices. Systematic Literature Review is conducted to determine the issues and benefits of agile practices reported in Literature and a survey in Pakistan IT market is conducted to capture practitioners' experiences regarding the issues and benefits of Agile practices. Finally, the results of SLR and Survey were analyzed and mapped to identify the similarities and differences in the findings.

By mapping the SLR and Survey results we come to know that majority of the benefits identified by the survey participants are already reported in Literature. Survey has reported new challenges in addition to those reported in Literature. One reason for this might be that the studies considered in Literature are mostly on small teams [P5, P11, P15, P25, P27]. Agile practices are more suitable for small teams, in order to adopt them for larger teams require considerable effort and adaption and results in new challenges. Agile practices require intense communication and coordination, which is a challenge in large teams. Among the questionnaires received from industry, 54% of the projects used the team size greater than 10, therefore the challenges reported are greater than in survey.

Challenges of the practices: *Time-boxed Iterations, daily Scrum Meetings, Sprint Review Meetings* are mostly reported from developers' unwillingness and unavailability perspectives due to tight deadlines. This might be due to the reason that according to the survey results, only 33% of the respondents agree that the decision-making is pluralistic. This means that if the teams are not empowered enough as Agile culture advocates, issues arise regarding the true implementation of practices. Another reason can be that due to large team sizes, as 54% of the projects reported in Survey has team size greater than 10, hence more control is required in large teams which can affect the team empowerment.

Issues are reported against the technique of maintaining *Sprint backlog* and *burndown chart* in Survey study only. Literature has not highlighted these issues. The reason is that as only

47% respondents use this practice, it means either people have no knowledge about these techniques or they are not aware about the tools used for automatic generation, e.g. *burndown charts* can be generated by using Microsoft Team Foundation Server in a minute. Hence no additional effort or resources are required for it. Hence experience and awareness of correctly using these techniques matter a lot. This shows that training the team before using agile and experienced Scrum master are the core for successful application of these practices.

Only 1 issue against *Continuous Integration* and *Refactoring* map with the one reported in Literature regarding the issue that Continuous Integration is difficult to be implemented in large teams. The reason is that CI and refactoring requires coordination and good knowledge transfer in teams. Only 16% respondents integrate changes daily. For refactoring, if the original developer is not known or leave the team or is unavailable, refactoring should be done very carefully. In case of large team, there are always issues for coordination and communication which inhibit the easy implementation of these agile practices in large team. Hence the more challenges reported can be justified by the fact that the 54% of projects reported in Survey has team size greater than 10.

For pair programming 50% of the issues map with literature and some additional issues are reported. There is a contradiction that 76% percent on respondents support team work but only 27% report to use PP. this is mainly due to the reason that in order to complete work in tight deadlines, PP introduces delays, and hence it should be preferred for complex problems.

For onsite customer only 1 issue reported in survey is mapped with literature findings. There are in fact 2 main issues for using this practice. First, either the client is not supportive enough to be available or the client cannot spare time due to his busy schedule. Second, developers don't like the customer at the development site, as the continuous change request divert the mind of developers and cause scope creep (sometimes due to unnecessary requirements) which can cause developers frustration.

Hence the study has identified many new issues by using survey as compared to the issues identified from literature; therefore there is an increased need to critically examine the practices for their issues regarding their implementation in large companies and an increased

need for training the team before using agile practices in company. It is also suggested that the teams should be empowered and pluralistic decision-making should be followed to increase the team's interest to coordinate and communicate which will automatically enhance the benefits of agile practices wit a result in decreased issues faced.

5.2 Contribution

- More issues associated with each agile practice are explored in addition to that reported in literature.
- Cumulative empirical knowledge on agile practices regarding their benefits and challenges.
- The studies focusing on challenges are lesser than that the studies reporting benefits, hence, this work is an effort to emphasize more research attention on this area.
- New insights are gained by comparison to what has been reported in Literature.
- Variations and similarities of survey results with literature findings have generalized some of the benefits and issues.
- A guide for the software teams aiming to apply the agile practices.
- They can choose the agile practices by evaluating their reported issues, characteristics or benefits
- Practitioners can benefit from the suggestions and experiences of this work for implementing of agile practices.
- There are very fewer studies addressing the state-of-practice of agile practices in Pakistan IT market.

5.3 Limitations

We cannot give any final conclusion using this survey results. The sample size for the study is small; hence there might be a possibility for missing the some issues and benefits regarding implementation of agile practices in literature.

5.4 Future work

The issues and benefits of agile practices identified by investigating State-of-art and State-of-Practice are very useful for guiding the software practitioners about the benefits and challenges agile methods before applying in software projects. The main focus of this work is to determine the benefits and challenges of agile practices. Some suggestions are given to mitigate the issues associated with the practices. This study can be replicated for a different and larger sample to validate the results.

The future work should be targeted for the development of focus on identifying the improvement strategies to lessen the challenges. Most of the challenges arise as a result of applying agile practices in large teams. Hence more empirical evidences are needed regarding the application of agile methods in large teams.

There is a need for a framework reporting challenges of agile practices in different project contexts. The common challenges faced in all contexts and the strategies to overcome the challenges.

References A:

[1] *S*erena, An Introduction to Agile Software Development, [online] June 2007. [Cited: January 18,2011.] www.serena.com/docs/repository/solutions/intro-to-agile-devel.pdf

[2] P. Abrahamsson , J. Warsta, M.T. Siponen , J. Ronkainen, "New Directions on Agile Methods: A Comparative Analysis," in *Proc. 25th Int. Conf. Software Engineering*, pp. 244 -254, 2003.

[3] B. Kitchenham, "Guidelines for performing Systematic Literature Reviews in Software Engineering," Version 2.3, Keele University and University of Durham, *EBSE Technical Report*, 2007.

[4] Z. Mushtaq and M.R. Jameel Qureshi, "Novel Hybrid Model: Integrating Scrum and XP," in *IJITCS*, vol.4, no.6, pp.39-44, 2012.

[5] Versionone, "State of Agile Survey: 2011," The State of Agile Development, [online] June 2011, [Cited: January 20, 2012.] http://www.versionone.com/state of agile development survey/10/

[6] Versionone, "State of Agile Survey:2011," The State of Agile Development, [online] June 2012, [Cited: January 20, 2012.] http://www.versionone.com/state of agile development survey/11/

[7] O. Salo and P. Abrahamsson, "Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum," in *IET Software*, Vol. 2, No. 1, pp. 58-64, 2008.

[8] J. Cho, "Issues and Challenges of Agile Software Development With Scrum," *Issues in Information Systems* , VOL IX, No. 2, pp. 188-195, 2008.

[9] P. Abrahamsson, 0. Salo, J. Ronkainen, J. Warsta, "Agile Software Development Methods: Review and Analysis," *VVT Publications*, Vol.478, pp. 7-94, 2002.

[10] Kelly Waters, "10 Key Principles of Agile Development," [online] 30 April 2007, [Cited: February 16, 2011.]
http://www.allaboutagile.com/category/10-key-principles-of-agile/

[11] A. Dagnino, "An Evolutionary Lifecycle Model with Agile Practices for Software Development at ABB," in *Proc. 8th IEEE Int. Conf. Engineering of Complex Computer Systems (ICECCS'02)*, 2002.

[12] E. S. F. Cardozo, J. Benito, F. A. Neto et al. "SCRUM and Productivity in Software Projects: A Systematic Literature Review," in *14th Int. Conf. Evaluation and Assessment in Software Engineering (EASE)*, Keele University, UK, 2010

[13] P. Sfetsos and I. Stamelos, "Empirical Studies on Quality in Agile Practices: A Systematic Literature Review," in *7th Int. Conf. Quality of Information and Communications Technology*, pp. 44-53, 2008

[14] T.Dyba and T. Dingsoyr, "Empirical studies of agile software development: A systematic review," *Information and Software Technology*, pp.833–859, 2008.

[15] P. Abrahamsson, O. Salo and J. Ronkainen, "Agile Software Development Methods - Review and Analysis," *VTT Publications* 478, 2002. [online] June 2011, [Cited: January 20, 2012.]
http://eljabiri1.tripod.com/sitebuildercontent/sitebuilderfiles/Comp-agile.pdf

[16] R. Baskerville, J. Pries-Heje, and S. Madsen, "Post-agility: What follows a decade of agility?," *Information and Software Technology*, vol. 53, pp. 543-555, 2011.

[17] J. Highsmith, "History: The Agile Manifesto||." [online] June 2011, [Cited: December 30, 2011.]
http://agilemanifesto.org/history.html

[18] S. Overhage, S. Schlauderer, D. Birkmeier, and J. Miller, "What Makes IT Personnel Adopt Scrum? A Framework of Drivers and Inhibitors to Developer Acceptance," in *44th Hawaii Int. Conf. System Sciences (HICSS)*, 2011.

[19] K. Petersen and C. Wohlin, "A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case," *Journal of Systems and Software*, vol. 82, pp. 1479-1490, 2009.

[20] P. S. Lawrence and A. K. Barbara, "Principles of survey research: part 1: turning lemons into lemonade," *SIGSOFT Softw. Eng. Notes*, vol. 26, pp. 16-18, 2001.

[21] M. Staples and M. Niazi, "Experiences using systematic review guidelines," *Journal of Systems and Software*, vol. 80, no. 9, 2007, pp. 1425–1437

[22] A. Begel and N. Nagappan, "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study," in *1st Int. Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2007.

[23] J. Q. Chen, P. Dien, B. Wang, and D. R. Vogel, "Light-Weight Development Method: a Case Study," in *Int. Conf. Service Systems and Service Management*, 2007.

[24] A. Fruhling, P. McDonald, and C. Dunbar, "A Case Study: Introducing eXtreme Programming in a US Government System Development Project," in *Proc. 41st Annual Hawaii International Conf. System Sciences*, 2008.

[25] K. Petersen and C. Wohlin, "A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case," *Journal of Systems and Software*, vol. 82, pp. 1479-1490, 2009.

[26] F. J. Pino, O. Pedreira, and F. I. Garcia etal. "Using Scrum to guide the execution of software process improvement in small organizations," *Journal of Systems and Software*, vol. 83, pp. 1662-1677, 2010.

References B:

ID	Studies
1	A. Begel and N. Nagappan, "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study," in *1st Int. Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2007.
2	J. Q. Chen, P. Dien, B. Wang, and D. R. Vogel, "Light-Weight Development Method: a Case Study," in *Int. Conf. Service Systems and Service Management*, 2007.
3	A. Fruhling, P. McDonald, and C. Dunbar, "A Case Study: Introducing eXtreme Programming in a US Government System Development Project," in *Proc. 41st Annual Hawaii International Conf. System Sciences*, 2008.
4	Y. Ghanam and F. Maurer, "Extreme Product Line Engineering: Managing Variability and Traceability via Executable Specifications," presented at *Conf. AGILE '09*, 2009.
5	S. J. Khalaf and K. A. Maria, "An Empirical Study of XP: The Case of Jordan," presented at *Int. Conf. Information and Multimedia Technology (ICIMT '09)*, 2009.
6	R. Mencke, "A Product Manager's Guide to Surviving the Big Bang Approach to Agile Transitions," presented at *Conf. AGILE '08*, 2008.
7	A. Moore and W. T. Flannery, "Use of Extreme Programming Methodologies in IT Application Design Processes: An Empirical Analysis," presented *at Management of Engineering and Technology*, Portland International Center for, 2007.
8	S. Overhage, S. Schlauderer, D. Birkmeier, and J. Miller, "What Makes IT Personnel Adopt Scrum? A Framework of Drivers and Inhibitors to Developer Acceptance," in *44th Hawaii Int. Conf. System Sciences (HICSS)*, 2011.
9	O. Salo and P. Abrahamsson, "Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme

	Programming and Scrum," *Software, IET*, vol. 2, pp. 58-64, 2008.
10	M. J. Akhtar, A. Ahsan, and W. Z. Sadiq, "Scrum adoption, acceptance and implementation (a case study of barriers in Pakistan's IT industry and mandatory improvements)," in *IEEE 17Th Int. Conf. Industrial Engineering and Engineering Management (IE&EM)*, 2010.
11	J. Srinivasan and K. Lundqvist, "Using Agile Methods in Software Product Development: A Case Study," in *6th Int. Conf. on Information Technology: New Generations (ITNG '09)*, 2009.
12	K. Vlaanderen, S. Brinkkemper, S. Jansen, and E. Jaspers, "The Agile Requirements Refinery: Applying SCRUM Principles to Software Product Management," in *3rd Int. Workshop Software Product Management (IWSPM)*, 2009.
13	L. Layman, L. Williams, and L. Cunningham, "Motivations and measurements in an agile case study," *Journal of Systems Architecture*, vol. 52, pp. 654-667, 2006.
14	S. T. Acuna, M. Gomez, and N. Juristo, "How do personality, team processes and task characteristics relate to job satisfaction and software quality?," *Information and Software Technology*, vol. 51, pp. 627-639, 2009.
15	T. Bipp, A. Lepper, and D. Schmedding, "Pair programming in software development teams : An empirical study of its benefits," *Information and Software Technology*, vol. 50, pp. 231-240, 2008.
16	S. Bryant, P. Romero, and B. du Boulay, "Pair programming and the mysterious role of the navigator," *International Journal of Human-Computer Studies*, vol. 66, pp. 519-529, 2008
17	G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. A. Visaggio, "Evaluating performances of pair designing in industry," *Journal of Systems and Software*, vol. 80, pp. 1317-1327, 2007.

18	K. S. Choi, F. P. Deek, and I. Im, "Exploring the underlying aspects of pair programming: The impact of personality," *Information and Software Technology*, vol. 50, pp. 1114-1126, 2008.
19	K. S. Choi, F. P. Deek, and I. Im, "Pair dynamics in team collaboration," *Computers in Human Behavior*, vol. 25, pp. 844-852, 2009.
20	W. Pedrycz, B. Russo, and G. Succi, "A model of job satisfaction for collaborative development processes," *Journal of Systems and Software*, vol. 84, pp. 739-752,2011.
21	K. M. Lui and K. C. C. Chan, "Pair programming productivity: Novice-novice vs. expert-expert," *International Journal of Human-Computer Studies*, vol. 64, pp. 915-925, 2006.
22	K. Petersen and C. Wohlin, "A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case," *Journal of Systems and Software*, vol. 82, pp. 1479-1490, 2009.
23	N. B. Moe, T. Dingsoyr, and T. Dyba "A teamwork model for understanding an agile team: A case study of a Scrum project," *Information and Software Technology*, vol. 52, pp. 480-491, 2010
24	K. Conboy and L. Morgan, "Beyond the customer: Opening the agile systems development process," *Information and Software Technology*, vol. 53, pp. 535-542, 2011.
25	R. Hoda, J. Noble, and S. Marshall, "The impact of inadequate customer collaboration on self-organizing Agile teams," *Information and Software Technology*, vol. 53, pp. 521-534, 2011.
26	M. Panaour and M. Ciglaric, "Impact of test-driven development on productivity, code and tests: A controlled experiment," *Information and Software Technology*, vol. 53, pp. 557-573, 2011.

27	L. Huang and M. Holcombe, "Empirical investigation towards the effectiveness of Test First programming," *Information and Software Technology*, vol. 51, pp. 182-194, 2009.
28	L.-O. Damm and L. Lundberg, "Results from introducing component-level test automation and Test-Driven Development," *Journal of Systems and Software*, vol. 79, pp. 1001-1014, 2006.
29	M. Laanti, O. Salo, and P. Abrahamsson, "Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation," *Information and Software Technology*, vol. 53, pp. 276-290, 2011.
30	A. Mohammad, "Empirical investigation of refactoring effect on software quality," *Information and Software Technology*, vol. 51, pp. 1319-1326, 2009.
31	H.F. Landim, A.B. Albuquerque, T.C. Macedo, "Procedures and conditions that influence on the efficiency of some agile practices," in *7th Int. Conf. Quality of Information and Communications Technology*, pp. 385-391, 2010.
32	M. Korkala, P. Abrahamsson and P. Kyllönen, "A Case Study on the Impact of Customer Communication on Defects in Agile Software Development.," in Conf. *AGILE '06*, 2006.

Appendix A: Quality Assessment Score

Study	1	2	3	4	5	6	7	8	9	10	11	Total
S2	1	1	1	0.5	0	0	0.5	0	0.5	0.5	0.5	5.5
S4	1	0.5	1	1	0	0	0	0	0	1	1	5.5
S6	1	1	1	0.5	0	0	1	0	0	1	1	6.5
S8	1	1	1	1	1	0	1	1	0	1	1	9
S10	1	1	0	0.5	0	0	0.5	0.5	0	1	1	5.5
S12	1	1	1	1	0.5	0	1	1	1	1	1	9.5
S14	1	1	1	1	1	1	1	1	1	1	1	11
S16	1	1	0.5	0.5	0.5		1	1	1	1	1	8.5
S18	1	1	1	1	1	0.5	0.5	1	1	1	0.5	9.5

S20	1	1	1	0.5	1	1	1	1	1	0.5	**9**
S22	1	1	1	1	1	1	1	1	1	1	**11**
S24	1	1	1	1	0	0	0.5	0	0.5	1	1	**7**
S26	1	1	1	1	0.5	1	0.5	1	1	1	1	**10**
S28	1	1	1	1	0	1	0.5	1	1	1	0	**8.5**
S30	1	1	1	0.5	0.5	0	1	1	1	1	0.5	**8.5**
S32	1	1	1	0.5	0.5	0	1	1	1	1	1	**9**

Appendix B: Survey Questionnaire

Analytical Review of Agile Practices in Pakistan

The purpose of this survey is to validate the issues and benefits of agile practices in software development organizations of Pakistan. Based on the frequency of occurrence, the benefits and issues of agile practices will be prioritized to highlight the most problematic and most useful practices. The results of survey will be shared with industry to help practitioners in selection of agile practices by evaluating the issues and benefits associated with their use. The survey is focused on the software development industry of Pakistan.