QUALITY ASPECTS OF AGILE SOFTWARE DEVELOPMENT

A THESIS SUBMITTED TO

DEPARTMENT OF INFORMATION TECHNOLOGY

UNIVERSITY OF DELHI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

DOCTOR OF PHILOSOPHY

BY

SULABH TYAGI

Under the Supervision of

Dr. Ritu Sibal
Prof. (Dr.) Bharti Suri

DEPARTMENT OF INFORMATION TECHNOLOGY

FACULTY OF TECHNOLOGY

UNIVERSITY OF DELHI

DELHI – 110 007, INDIA

October, 2020

CERTIFICATE OF DECLARATION

This is to certify that the thesis entitled "**QUALITY ASPECTS of AGILE SOFTWARE DEVELOPMENT**" which is being submitted by Sulabh Tyagi to the Department of Information Technology, Faculty of Technology, University of Delhi for the award of the degree of Doctor of Philosophy is a record bona fide research work carried out by him under my supervision and guidance. He has fulfilled the requirements for the submission of this thesis. The contents of this thesis, in full or in parts, have not been submitted for any other degree or diploma in any University.

Dr. Ritu Sibal (Supervisor)
Associate Professor
Division of Computer Engineering
Netaji Subhas Institute of Technology,
New Delhi, India

Prof. (Dr.) Bharti Suri (Co-Supervisor)
Professor
University school of Information
Communication & Technology
Guru Gobind Singh Indraprastha
University, New Delhi, India

Head
Department of Computer Engineering
University of Delhi
Delhi, India

Dean
Faculty of Technology
University of Delhi
Delhi, India

Department of Information Technology

Faculty of Technology,

University of Delhi

Delhi-110007, India

Date : 10/10/2019

CERTIFICATE OF ORIGINALITY

The research work embodied in this thesis entitled "Quality Aspects of Agile Software Development" has been carried out by me at the Department of Information Technology, Faculty of Technology, University of Delhi, Delhi, India. The manuscript has been subjected to plagiarism check by Urkund software. The work submitted for consideration of award of Ph.D. is original.

Name and Signature of the Candidate

STUDENT APPROVAL FORM

Name of the Author	Sulabh Tyagi
Department	Information Technology
Degree	Ph.D
University	University of Delhi
Guide(s)	Dr. Ritu Sibal Prof. (Dr.) Bharti Suri
Thesis Title	Quality Aspects of Agile Software Development
Year of Award	2020

Agreement

1. I hereby certify that, if appropriate, I have obtained and attached here to a written permission/statement from the owner(s) of each third party copyrighted matter to be included in my thesis/dissertation, allowing distribution as specified below.

2. I hereby grant to the university and its agents the non-exclusive license to archive and make accessible, under the conditions specified below, my thesis/dissertation, in whole or in part in all forms of media, now or hereafter known. I retain all other ownership rights to the copyright of the thesis/dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis, dissertation, or project report.

Conditions :

1. Release the entire work for access worldwide	YES
2. Release the entire work for 'My University' only for 1 Year 2 Year 3 Year and after this time release the work for access worldwide.	2 Year
3. Release the entire work for 'My University' only while at the same time releasing the following parts of the work (e.g. because other parts relate to publications) for worldwide access.	

a) Bibliographic details and Synopsis only. b) Bibliographic details, synopsis and the following chapters only. c) Preview/Table of Contents/24 page only.	a. Bibliographic details and Synopsis only.
4. View Only (No Downloads) (worldwide)	Download worldwide

Sulabh Tyagi

Dr Ritu Sibal (Supervisor)
Associate Professor
Division of Computer Engineering
Netaji Subhas Institute of Technology,
New Delhi, India

Dr Bharti Suri (Co-Supervisor)
Professor
University school of Information
Communication & Technology
Guru Gobind Singh Indraprastha
University, New Delhi, India

Place : NSIT, New Delhi

Date :10/10/2020

ACKNOWLEDGEMENT

Firstly, I would like to thank Almighty God for giving me the opportunity, courage and wisdom for pursuing my dream in the form of this research work. This thesis is the resultant of the effort from several people including my supervisors, my family, my friends, and agile participants of this research. I am deeply grateful to my supervisor Dr. Ritu Sibal who has been a constant source of motivation and guided me on every aspect of my thesis. I feel extremely honored in expressing my profound sense of sincere gratitude to her for all her support, guidance and valuable inputs throughout the period of this work. It would have never been possible for me to take this work to completion without her relentless support and encouragement. I would also like to express my gratitude to my co-supervisor Prof. (Dr.) Bharti Suri whose words of wisdom and constant guidance always proved enlightening and enriching. She always have been a great support for me and guided me in the best possible way. My heartfelt gratitude and thanks to Prof. (Dr.) Yogesh Singh for his immense support and guidance.

I also wish to express my deepest gratitude to my grandparents for their blessings. I would like to thank my parents and dedicate this thesis to them for their unconditional support, patience and endless love. I would like to thank my younger siblings, Priyanka and Dr. Mohit for their never ending support and having faith on me. I am extremely thankful to my friends K.Suresh, Ravi and Parveen for all their support and encouragement.

Big thanks to all agile practitioners who participated in this research for providing their useful time and sharing their experiences with me.

Sulabh Tyagi

ABSTRACT

Ever since the advent of software engineering in 1968, the field has witnessed major changes in the manner in which software is engineered. A plethora of software process models have been proposed to help software engineers develop software so that projects are delivered on time and meet the expectations of the customers. Software process models mainly used to describe individual roles, the work products required, how the quality of the work product is judged, and the processes to create and manage those work products.

Software Development is an evolutionary process and every new software process model was built upon the success and failures of the previous models. The latest entrant is the agile methods for software development. With the introduction of agile methods the paradigm has shifted from prescriptive process models to the adaptive models. Agile software development methods are the de facto standard for software development across the globe today because of their abilities to deliver quick and quality software. But due to the stringent timeliness and constantly changing requirements, agile teams often face challenges in meeting the requirements of the customer. Agile methods ensure maximum participation of stakeholders in the entire development process. Agile development methods serve the customers in entirely different way by producing a working piece of functionality at the end of every iteration. This thesis aims to explore quality in agile software development from two important perspectives: product quality and teamwork quality. Effective and sound software testing practices are the predictor of quality on any development project. Test driven development (TDD)/Acceptance Test Driven Development (ATDD) are the building blocks for producing quality product on agile development projects. Test automation is one of the key factors to keep development and testing in synchronization on agile projects. Improving quality of the product on agile projects involves adopting best practices of test automation and building a good test automation infrastructure so that continuous automated testing is performed at unit level, integration level and acceptance level.

Agile software development projects demands self organized and self managing teams, this increases the complexity associated with agile software development. Teamwork is found to be central in high performing agile teams. Teamwork quality

affects product quality on agile software development projects. The most crucial element on which teams gel in an agile setup is the existence of trust amongst its team members. The agile manifesto proactively emphasizes on existence of good interpersonal relationships, and continuous customer collaboration over the development period. Trust amongst team members and customers plays a significant role in improving teamwork quality and performance of global software development teams.

This study is based on Grounded Theory research method that involved 60 agile practitioners from 28 different software organizations in India, USA, Germany, Australia and New Zealand. This thesis presents the: a) A substantive theory that explains transition journey of adopting test automation on agile development projects. The transition journey of adopting test automation involves several challenges related to the technical, organizational and human factors during pre transition phase. Then, different overcoming strategies were used by agile teams during in-transition stage and finally they start getting benefits by practicing and streamlining test automation practices in post-transition stage. b) Development of Reusable Hybrid Test Automation Framework (RHTAF) using selenium 2.0 tool for scrum based web projects, the primary focus of this framework is to ensure reusability and maintainability of test scripts so as to speed up the testing process of web based agile projects. Experimental analysis of the framework is performed in a simulated scrum environment. Experimental analysis of RHTAF on two famous e-commerce websites successfully demonstrates that higher reusability, maintainability, test efficiency and reduction in test execution time is achieved by using this framework, c) Empirically developed framework for building trust amongst agile team members for improving teamwork quality. The factors are working environment, Leadership, Socio-Cultural, Personal and Organizational. These factors are critical in building trust among agile team members which further strengthen collaboration and interaction between them, d) Development of **A**ssessment of **T**rust in **A**gile Teams (**ATA**) instrument for assessment of trust levels within agile teams. It measures trust by measuring trust factors across five dimensions namely: Working Environment, Leadership, Organizational, Personal and Socio Cultural.

This thesis will provide the roadmap to the agile practitioners towards their test automation adoption journey by making them aware about different types of

impediments and ways to overcome them which is based on experiences gained from the real life agile projects. Also, this thesis will help teams, agile coaches and senior management in understanding the critical factors in building the environment of trust and providing the requisite support to the teams.

TABLE OF CONTENTS

CERTIFICATE OF DECLARATION i
CERTIFICATE OF ORIGINALITY ii
STUDENT APPROVAL FORM iii
ACKNOWLEDGEMENT v
ABSTRACT vi
CONTENTS ix
LIST OF FIGURES xiv
LIST OF TABLES xvii
LIST OF ABBREVIATIONS xviii

Chapter 1: Introduction 1-10
1.1 An Overview of Software Quality 1
1.2 Software Quality in Agile Software Development 3
1.3 Research Motivation and Objective 6
1.3.1 Product Quality Aspect in Agile Software Development 6
1.3.2 Teamwork Quality Aspect in Agile Software Development 7
1.4 Research Contributions 8
1.5 Outline of the Thesis 9

Chapter 2: Literature Review 11- 27
2.1 The Emergence of Agile Software Development 11
2.2 Agile Software Development 13
2.2.1 Scrum 15
2.2.2 Extreme Programming (XP) 18
2.3 Testing in Agile Software Development 22
2.4 Social Aspects of Agile teams 25
2.4.1 Human Dynamics in Agile teams 26

Chapter 3: Research Design 28-51
3.1 Research Approaches 28
3.2 Overview of Research Methods 29
3.2.1 Case Study 29

3.2.2 Survey 29
3.2.3 Ethnography 30
3.2.4 Grounded Theory 30
3.3 Selecting our research method 30
3.4 Research Perspectives 31
3.5 Researcher Role 32
3.6 Grounded Theory 33
3.6.1 Brief Overview of Grounded Theory 33
3.6.2 Selecting the Research Area 36
3.6.3 Minor Literature Review 37
3.6.4 Data Collection 37
3.6.5 Data Analysis 45
3.6.6 Memoing 48
3.6.7 Sorting 49
3.6.8 Major Literature Review 49
3.6.9 Theoretical Coding 50
3.6.10 Theory Write Up 50
3.6.11 Evaluating a Grounded Theory 51

Chapter 4: A Transition Journey of Adopting Test Automation on Agile Development Projects 52-73
4.1 Introduction 52
4.2 Research Method 52
4.2.1 Data Analysis 53
4.2.1.1 Phase 1: Identifying the Core Category 53
4.2.1.2 Open Coding and Memo Writing 55
4.2.1.3 Selecting the Core Category 55
4.2.1.4 Phase 2: Refining the Core Category 56
4.2.1.5 Selective Coding and Memo Writing 56
4.2.1.6 Reaching theoretical Saturation 56
4.2.1.7 Phase 3: Theory Write Up 57
4.3 Results 59
4.3.1 Pre-Transition and In-Transition Stage 59
4.3.1.1 Pre-Transition Stage: Technical Challenges 59

4.3.1.2 Pre-Transition Stage: Organizational Challenges 62
4.3.1.3 Pre-Transition Stage: Human Challenges 64
4.3.1.4 In-Transition Stage: Overcoming Strategies 66
4.3.2 Post-Transition Stage: Benefits 68
4.4 Discussion 70

Chapter 5: Development of Reusable Hybrid Test Automation Framework for Web Based Scrum Projects 74-117
5.1 Introduction 74
5.2 Proposed Framework 76
5.2.1 Three Layer Architecture of Automation Framework 76
5.2.2 Implementation and Experimental Analysis of RHTAF 78
5.2.2.1 InSprint Test Automation Test Strategy 79
5.3 Implementation details of RHTAF using Page Object Model 81
5.4 Evaluation and Validation of RHTAF 94
5.4.1 Evaluating RHTAF 94
5.4.2 Validation w.r.t automated test execution time 109
5.5 Results 115
5.6 Discussion 116

Chapter 6: Empirically Developed Framework for Building Trust in Distributed Agile Teams 118-148
6.1 Introduction 118
6.2 Background 120
6.2.1 Trust 120
6.2.2 Significance of trust in distributed teams 120
6.3 Research Method 122
6.3.1 Data Analysis 122
6.3.1.1 Phase 1: Identifying the Core Category 123
6.3.1.2 Open Coding and Memo Writing 123
6.3.1.3 Selecting the Core Category 124
6.3.1.4 Phase 2: Refining the Core Category 126
6.3.1.5 Selective Coding and Memo Writing 126
6.3.1.6 Reaching theoretical Saturation 127

6.3.1.7 Phase 3: Theory Write Up 128
6.3 Results 129
6.3.1 Working Environment Factors 129
6.3.2 Leadership Factors 131
6.3.3 Organizational Factors 134
6.3.4 Personal Factors 136
6.3.5 Socio-Cultural Factors 139
6.4 Discussion 141

Chapter 7: A Trust Assessment Tool for Measuring and Improving Trust in Agile Teams 149-169
7.1 Introduction 149
7.2 Background and Related Work 150
7.3 Research Method 152
7.3.1 Data Collection 152
7.3.2 Questionnaire Construction and Item Generation 153
7.3.3 Questionnaire Procedure and Design 153
7.3.4 Controlled Experiment using ATA 154
7.4 Statistical Data Analysis and Validation 156
7.4.1 Factor Analysis using PCA 158
7.4.2 Correlation 159
7.4.3 Reliability 159
7.4.4 Content Validity 161
7.5 Results 161
7.6 Discussion 167

Chapter 8: Conclusion and Future Work 170-177
8.1 Conclusion 170
8.2 Limitations 173
8.3 Evaluating the Grounded Theory 174
8.4 Future Work 177
Appendix A: Initial Interview Protocol 178-179
Appendix B: Observation Protocol 180-181
Appendix C: Different Scrum Artifacts 182-183

References **184-209**
List of Publications **210**

LIST OF FIGURES

FigureNo.	Title	Page No.
Figure 2.1	The Waterfall Model	11
Figure 2.2	The Spiral Model	12
Figure 2.3	Iterations and Workflow of RUP	13
Figure 2.4	Scrum Framework	15
Figure 2.5	XP Practices	21
Figure 2.6	Agile Testing Quadrants	22
Figure 3.1	Grounded Theory Process	36
Figure 3.2	Summary of our Research Design	38
Figure 3.3	Scrum Board in Organization X showing Task Status	43
Figure 3.4	Use of Agile Project Management Tool by Distributed Team Members of Organization Z	44
Figure 3.5	Different levels of data abstraction in GT	46
Figure 3.6	Emergence of category *choosing the right tool* from underlying	48
Figure 3.7	Memo on Test Environment Management	49
Figure 4.1	Application of Grounded Theory in Our Research	54
Figure 4.2	Example of Selective Coding Process	58
Figure 4.3	Technical Challenges during Pre- Transition Stage	60
Figure 4.4	Organizational Challenges during Pre- Transition Stage	63
Figure 4.5	Human Challenges during Pre- Transition Stage	65
Figure 4.6	Overcoming Strategies used by agile teams during In-Transition Stage	67
Figure 4.7	Benefits Experienced by Agile Teams during Post-Transition Stage	69
Figure 4.8	Mapping of Transition Stages with Bridges Transition Model	71
Figure 5.1	Three Layer RHTAF Architecture	77
Figure 5.2	Input Excel File containing Keywords and Data for amazon.in	78
Figure 5.3	Implementation of RHTAF over Three Sprints in Scrum Environment	81

FigureNo.	Title	Page No.
Figure 5.4	Driver (Main) Function of amazon.in	82
Figure 5.5	Driver (Main) Function of flipkart.com	83
Figure 5.6	Sprint 1 - Automation of Login Functionality using Page Object Model for amazon.in	84
Figure 5.7	Sprint 1 - Execution of Login Test Case of amazon.in	84
Figure 5.8	Input Excel File containing Keywords and Data for flipkart.com	85
Figure 5.9	Sprint 1 - Automation of Login Functionality using Page Object Model for flipkart.com	86
Figure 5.10	Sprint 1 - Execution of login Test Case of flipkart.com	86
Figure 5.11	Sprint 2 - Automation of Search Product Functionality using Page Object Model for amazon.in	87
Figure 5.12	Sprint 2 - Execution of Search Product Test Case of amazon.in	88
Figure 5.13	Sprint 2 - Automation of Search Product Functionality using Page Object Model for flipkart.com	89
Figure 5.14	Sprint 2 - Execution of Search Product Test Case for flipkart.com	90
Figure 5.15	Sprint 3 - Automation of Choosing Payment Method Functionality using Page Object Model for amazon.in	91
Figure 5.16	Sprint 3 - Execution of Choosing Payment Method Test Case for amazon.in	92
Figure 5.17	Sprint 3- Automation of Choosing Payment Method Functionality using Page Object Model for flipkart.com	93
Figure 5.18	Sprint 3 - Execution of Choosing Payment Method Test Case for amazon.in	94
Figure 5.19	SnapdealLoginPage Class	95
Figure 5.20	SnapdealLoginTest Class	96
Figure 5.21	SnapdealAddAddress1Page Class	97
Figure 5.21	SnapdealAddAddress1Test Class	98
Figure 5.23	Sonar cube report of Snapdeal without using RHTAF	99
Figure 5.24	Sonar cube report of Snapdeal using RHTAF	99
Figure 5.25	MyntraLoginPage Class	100
Figure 5.26	MyntraLoginTest Class	100

FigureNo.	Title	Page No.
Figure 5.27	MyntraAddAddressPage Class	101
Figure 5.28	MyntraAddAddressTest Class	102
Figure 5.29	Sonar cube report of Myntra without using RHTAF	102
Figure 5.30	Sonar cube report of Myntra using RHTAF	103
Figure 5.31	ShopCluesLoginPage Class	104
Figure 5.32	ShopCluesLoginTest Class	104
Figure 5.33	ShopCluesAddAddressPage Class	105
Figure 5.34	ShopCluesAddAddressTest Class	106
Figure 5.35	Sonar cube report of ShopClues without using RHTAF	106
Figure 5.36	Sonar cube report of ShopClues using RHTAF	107
Figure 5.37	Sonar cube report of Amazon without using RHTAF	107
Figure 5.38	Sonar cube report of Amazon using RHTAF	108
Figure 5.39	Sonar cube report of Flipkart without using RHTAF	108
Figure 5.40	Sonar cube report of Amazon using RHTAF	109
Figure 5.41	Automation of Amazon test cases using Katalon	110
Figure 5.42	Automation of Amazon test cases using Katalon	111
Figure 5.43	Automation of Flipkart test cases using Katalon	112
Figure 5.44	Automation of Flipkart test cases using Katalon	113
Figure 5.45	Test execution summary for Amazon	114
Figure 5.46	Test execution summary for flipkart	114
Figure 5.47	Test result summary of Amazon and Flipkart	117
Figure 6.1	Incidents emerged from coding of interviews	124
Figure 6.2	Emergence of core category	126
Figure 6.3	Our GT model for building trust	129
Figure 7.1	Scrum framework with trust assessment	157
Figure 7.2	Mean trust score of working environment over 3 sprints	162
Figure 7.3	Mean trust score of leadership over 3 sprints	164
Figure 7.4	Mean trust score of organizational over 3 sprints	165
Figure 7.5	Mean trust score of personal over 3 sprints	166
Figure 7.6	Mean trust score of socio cultural over 3 sprints	167
Figure 7.7	Normalized sprint burn down chart	168

LIST OF TABLES

Table No.	Title	Page No.
Table 1.1	Agile practices that assure quality in agile software development	4
Table 3.1	Brief comparison among three GT versions	35
Table 3.2	Demographic details of the participants	42
Table 4.1	Example of Open Coding Process	57
Table 5.1	Different regression test cases automated over 3 sprints	80
Table 5.2	Different metrics value before and after applying RHTAF	115
Table 5.3	Performance comparison of automated test execution	116
Table 6.1	Example of open coding process	125
Table 6.2	Example of Selective Coding Process	127
Table 6.3	Emergence of concepts for working environment	131
Table 6.4	Emergence of concepts for leadership	133
Table 6.5	Emergence of concepts for organizational	134
Table 6.6	Emergence of concepts for personal	137
Table 6.7	Emergence of concepts for socio cultural	139
Table 6.8	Mapping of trust factors to mayer's model	144
Table 7.1	Teamwork Quality (TWQ) model construct	153
Table 7.2	A big five teamwork model construct	154
Table 7.3	Items in Trust Assessment Questionnaire	155
Table 7.4	Factor analysis using PCA	160
Table 7.5	Pearson correlation coefficient for three sprints	161
Table 7.6	Trust measurement framework- Reliability	161

LIST OF ABBREVIATIONS

Abbreviations	Description
ISO	International Organization for Standardization
SPI	Software Process Improvement
IEC	International Electrotechnical Commission
SDLC	Software Development Life Cycle
RAD	Rapid Application Development
RUP	Rational Unified Process
SQM	Software Quality Management
SQA	Software Quality Assurance
SQC	Software Quality Control
SQP	Software Quality Planning
CI	Continuous Integration
XP	Extreme Programming
TDD	Test Driven Development
ATDD	Acceptance Test Driven Development
IDE	Integrated Development Environment
POM	Page Object Model
FDD	Feature Driven Development
DSDM	Dynamic Software Development Method
ASD	Adaptive Software Development
PO	Product Owner
QTP	Quick Test Professional
UFT	Unified Functional Tester
AUT	Application Under Test
GSD	Global Software Development
DSD	Distributed Software Development
CASP	Critical Appraisal Skills Programme

CHAPTER 1

INTRODUCTION

The pervasiveness of software in all aspects of our lives is an acceptable truth. This beckons for improving the quality of software products in order to ensure flawless execution of software products. The role of software development process cannot be neglected in achieving the goal of high quality software. It is a well-known fact that a lousy process leads to an inferior quality product. Last decade has witnessed the shift in paradigm from prescriptive process models to agile methods for software development. Agile methods are lightweight methods that avoid long term planning and documentation to deliver superior quality software at shorter intervals. The widespread popularity of agile development methods is primarily owing to their ability to incorporate constantly changing requirements of the customer, improving communication between development team and customers and faster time to market as compared to traditional development methods [1].

1.1 An Overview of Software Quality

The pursuit of quality in software products has generated numerous views of software quality. Software quality cannot be described in one general definition. It can be seen as an abstract term which consists of people's expectations and experience of the system [2]. Quality has several working definitions. According to Juran et al. [3] quality is fitness for purpose which means: 1) a product that meets the customer requirements and satisfies user needs. 2) a product that is free from any defects. As per the IEEE glossary [4] of software engineering terminology quality is defined as "A set of activities developed to evaluate the process by which product is engineered". Hendrickson et al. [5] associated quality with human assessment, cost and benefits. ISO 9000 defines quality as "the totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs" [6]. Where "stated needs" means those needs that are specified as requirements by the customer in a contract, and "implied needs" are those needs that are identified and defined by the company providing the product.

Kitchenham et al. [7] identified that in order to define software quality, quality needs to be considered from both ways i.e. quality of the product and quality of the process that results in the product. Characteristics or attributes of the product and the process are used to characterize and evaluate quality. Even though software engineering has undergone radical changes since its inception, the attributes that characterize quality have remained same throughout these years. McCall and Cavanos [8] were the first to propose a software quality model based on quality attributes. A software quality model is a framework comprising of a defined set of characteristics, and of relationships between them, which provides a framework for specifying quality requirements and evaluating quality [9]. McCall's quality model [10] adopts three major perspectives for defining software quality: product revision, product transition, and product operations. Product revision relates to the ability to undergo changes, product transition to the ability to adapt to new environments, and product operations to the operation characteristics of the software. Barry W. Boehm [11] suggested a quality model that focuses on the general utility of the software and decomposed it further into three high level characteristics including as-is utility, maintainability, and portability. Another variation of this model is FRUPS [12] that defines the software quality across five parameters: functionality, usability, reliability, performance and supportability. The ISO/IEC Standard 9126 [13] and its successor ISO/IEC Standard 25000 [9] decompose software quality into process quality, product quality, and quality in use. In the '80s and '90s emphasis was put on software process improvement (SPI) with the appearance of quality standards and guidelines, such as ISO 9001[14]. Software process modeling endeavor to improve the efficacy of software development process. A well-documented iterative and incremental process with regular design walkthroughs, code inspections and process quality assurance techniques is preferred over product quality corrective measures for ensuring product quality. Moreover, there is evidence to show that use of standards and process assessment models has a positive impact on the quality of the final software product [7].The various Software Development Life Cycle (SDLC) models like Waterfall, Spiral, Rapid Application Development (RAD) and Rational Unified Process (RUP) are the manifestation of the efforts to model the software process to streamline the activities of software development. Software Quality Management (SQM) fosters an overall culture of quality in software organizations. A Software Quality Management (SQM) system has three main components namely: Software Quality Assurance

(SQA), Software Quality Control (SQC) and Software Quality Planning (SQP). SQA is typically considered as a quality standard guide from the organization perspective and independent to the project. SQA consist of standards, regulations, tools for verification and evaluation of work products during the software development lifecycle [15]. SQA is of extreme importance for both internal and external purposes [16]. Internal purpose for performing SQA activities is to provide the confidence to the management. While external purpose for conducting SQA is to provide confidence to the customers. SQC activities are important to examine different artifacts during SDLC phases. SQC ensures the quality of the deliverable before final sign off by conducting code inspections, technical reviews and testing [15]. SQP is mainly defined at project level. All project planning related activities like quality goals to be achieved, risks and contingency, development schedule and effort estimation are defined under SQP.

1.2 Software Quality in Agile Software Development

Agile methods have changed the way software quality has been perceived in traditional process centric methods. Agile software development has emerged from the prevailing best practices for software development. It can be said that these practices ensure software quality assurance (SQA) during software development. Promoters of agile methods therefore claim that because of the very nature of these methods quality in agile projects is a natural outcome of these methods. Agile quality assurance has been defined as the development of software that can respond to change, as the customer requires it to change [17]. Agile software development methods integrate quality assurance and control process throughout the development lifecycle. According to [18] quality in agile development projects is due to the best practices like test driven development, code refactoring, effective collaboration and communication between agile team members. Table 1.1 provides a brief overview of the agile practices that assure quality in agile software development.

Table1.1 Agile practices that assure quality in agile software development

Agile Practice	Description
Test Driven Development (TDD)	First create the test, then run the test, allow it to fail, then make changes in the code until test passes [18].
Refactoring	Make small changes in the code, code behavior must not be affected.Resulting code is of higher quality [18].
Continuous integration	Any implemented functionality and code change must be tested and check in to the main branch on a continuous basis. This ensures quality.
Pair programming	Two programmers work together on the same machine on turn by turn basic, Bugs are reported as they occur [19].
Face to face communication	Preference to face to face communication between team members over phone and emails. Daily stand up meeting is a daily 15-20 min meeting to ascertain that entire team take part and should be on the same page on the project development status.
Constant customer collaboration	Product owner usually a customer representative is part of the development team and clarifies the requirements to the development team.
Frequent customer feedback	Customer gives the feedback at the end of every sprint / release. Continuous feedback helps in improving the quality of the product. Meeting customer requirements is the key quality attribute.

TDD is a combination of test first development and refactoring where existing design is used to implement new feature or it may be refactored locally to modify the portion of the design affected by the new feature. Furthermore, addition of new features also results in the growth of the regression test suite which needs to be run frequently to ensure that new functionality has not impacted the previously built functionality. This is realized through automated tests that also ensure rapid feedback loops and dynamic interaction between team members. Automated set of tests also act as a mode of documentation and communication across the team [18].

Continuous Integration (CI) and refactoring are important practices that infuse quality in the product with every subsequent sprint. CI is a practice in which developers

integrate their code into a central repository several times a day. Every code check in is then verified by automated tests to ensure early detection of faults. Refactoring improves the structure of the code for better understandability [20]. Basically it is a technique to review the code for keeping the design as simple as possible by making changes to the code. Agile software development lays importance on simple design that yield potentially shippable product at the end of the every time boxed sprint. Collective code ownership is another practice which implies collective ownership of code by the entire team. Automation of builds and tests play a very significant role while executing different agile practices 18]. Automated tests facilitate the team members to have quick and early feedback on the respective builds that are integrated into the main system branch [18].

Agile methods are people centric and not process centric. They give more emphasis on people and their interactions rather than process and tools. In agile methods the teams are 'self-organizing'. In traditional teams responsibility is assigned in a hierarchical structure and they are built on basis of skills and tasks. In agile teams the 'self-organizing' principle assigns autonomy to teams to communicate and coordinate amongst themselves to complete the work in a best possible manner. Agile teams interact and communicate amongst themselves on a daily basis through daily standup and sprint retrospective meetings. In traditional methods the user participation is limited to two ends of the development cycle either in the initial phase of feasibility study or at the final acceptance stage. On the contrary, the agile methodologies are highly customer driven which demands onsite customer representative for providing the requirements, assigning the priorities and steering up the project [21]. Agile methods ensure maximum participation of stakeholders in the entire development process. Agile development methods serve the customers in entirely different way by producing a working piece of functionality at the end of every sprint. Customers then evaluate the product themselves and provide their valuable feedback to the development team [22]. According to [23] courage and respect are two fundamental values of XP teams in addition to simplicity, feedback and communication. Agility comes with the cultural shift and change in mindset which strives on the individual interactions within agile environment [24]. Another set of values that agile teams have are common focus, mutual trust and respect. These values act as a converging force towards quality.

1.3 Research Motivation and Objective

According to [25] there is an obvious need for further research in order to understand the requirements of quality and the requirements of agile quality in particular, within different cultural and social contexts. There is dearth of research work on quality issues of agile software development [26]. A number of studies point out the need to undertake the research that should be 1) empirical in nature 2) focusing on scrum and XP 3) based on industry people 4) address real concern of practitioners during their agile adoption journey [27-29]. Our research addresses all the above points. For the purpose of this study, we explore quality in agile software development from two important perspectives: product quality and teamwork quality. The motivation for choosing these two aspects of quality in agile software development for the purpose of this study is elaborated below.

1.3.1 Product Quality Aspect in Agile Software Development

As discussed in the earlier section, agile practices are instrumental in assuring quality in software development and in the final product. For the purpose of this research work, we explore test automation which has a major role in assuring quality of final product in agile development projects. In a study by Bhasin et al [30], it was found that Test Driven Development (TDD)/Acceptance Test Driven Development (ATDD) are the building blocks for producing quality product in every subsequent sprint on agile development projects. Test automation played crucial role in keeping development and testing in synchronization on agile projects [31]. To ensure quality on agile projects, practicing test automation is crucial. As stated in [32] continuous integration would have been impossible without test automation.

Test Automation is considered as the key contributor to successful agile development projects and is the core of agile testing [33]. In world quality report [34], around sixty one percent of the practitioners consider lack of good testing approach as one of the challenge while practicing agile testing and fifty five percent have mentioned that practicing automation on agile projects is difficult [34].

The above research studies bring forth the importance of test automation on agile development project. Hence our research objectives from the product quality perspectives are:

Research Objective 1: To investigate different challenges agile practitioners face while practicing test on agile development projects?

Research Objective 2: To investigate different strategies to effectively practicing test automation on agile development projects?

Research Objective 3: To develop effective test automation framework for agile development projects?

1.3.2 Teamwork Quality Aspect in Agile Software Development

The proponents of agile firmly believed that the best way to improve software quality is to focus on people who develop the software. In agile development environment, teams are self organizing, this increases the complexity associated with agile software development. Teamwork is found to be central in high performing agile teams [35, 36]. Lindsjørn et al. [35] reported that teamwork quality affects product quality on agile software development projects. Team work is crucial in software development; team performance is influenced by the team engagement during software development [37]. In a team that lacks cohesion, team members are less likely to help out when work load gets heavy or when team member experience conflict with another member [38]. Team members must gel with each other and work in a cohesive environment to make software projects successful [39]. A highly cohesive team provides value to customer and meets project schedule, budget, and quality. Asproni et al. [40] reported motivation and teamwork as integral factors of agile software development. The most crucial element on which teams gel in an agile setup is the existence of trust amongst its team members [41]. The agile manifesto proactively emphasizes on existence of good interpersonal relationships, and continuous customer collaboration over the development period. Trust amongst team members and customers plays a significant role in performance of global software development teams [42].

The above discussion so far highlights the importance of trust in agile teams. The concept of team work quality with respect to trust is important in agile software development however it has received very less attention. Our research objectives from the teamwork quality perspectives are:

Research Objective 4: To investigate the factors that build trust in distributed agile teams.

Research Objective 5: To develop a framework to measure and improve trust levels in agile teams.

Hence, the specific topic of this thesis is stated as:

"Quality Aspects of Agile Software Development"

1.4 Research Contributions

This thesis investigates two important phenomenon's of agile development projects with respect to quality namely: (1) How agile teams adopt and practice test automation in order to improve the product quality? (2) How to build trust amongst agile teams in order to improve the teamwork quality? Grounded Theory (GT) research method is used in this study that involved 60 agile practitioners at different agile positions and diverse domains from 28 different software organizations in India, USA, Germany, Australia and New Zealand.

The main contributions of the thesis are:

- This thesis presents a Grounded Theory that explains the transition journey of adopting test automation on agile development projects.
- This thesis presents the development and application of a reusable and hybrid test automation framework for agile development projects.
- This thesis presents an empirically developed framework for building trust in distributed agile teams.
- This thesis presents a trust assessment tool for measuring and improving trust in agile teams

1.5 Outline of the Thesis

This thesis is structured as follows:

Chapter 1 Introduction presents brief overview on quality, quality in agile software development, motivation and objective behind this research, contributions from this research, and the outline of the thesis.

Chapter 2 Literature Review presents an overview of related work. According to the guidelines of the research method (described in chapter 3) followed in this thesis, minor literature review was conducted in the initial stage of this research. However, detailed literature review is presented and discussed in light of the results in the discussion section of chapters 4, 5, and 6

Chapter 3 Research Design explains types of research methods and perspectives, detailed description of Grounded Theory with the help of examples, discusses the application of grounded theory in this study with the help of examples.

Chapter 4 A Transition Journey of Adopting Test Automation on Agile Projects

A substantive theory about the transition journey of adopting test automation on agile development projects is presented in this chapter. It explains different challenges faced by agile teams while practicing test automation and different strategies used by them in order to overcome those challenges. *Adopting Test Automation* is a transition journey on agile development projects comprising of three stages: *Pre-Transition*, *In-Transition* and *Post-Transition* stage. *Pre-Transition* stage reflects the different types of impediments faced by the participants while practicing test automation. The strategies adopted to overcome these challenges are the focus of attention in the *In-Transition* stage. Finally, in the *Post-Transition* stage agile teams reap the benefits by adopting test automation. The transition stages of test automation are mapped with the Bridges transition model to explain the transition process.

Chapter 5 Development of Reusable Hybrid Test Automation Framework for Web based Scrum Projects. This chapter presents the development and application of a reusable and hybrid test automation framework that is reusable, maintainable and improves testing efficiency on web based scrum projects. The Framework is implemented using selenium, webdriver 2.0 in eclipse IDE using Page Object Model (POM). This is followed by the discussion in the light of related literature.

Chapter 6 Empirically Developed Framework For Building Trust in Distributed Agile Teams This chapter provides a comprehensive framework to build trust in

distributed agile teams. This framework has been empirically developed by the Grounded Theory Research method. This framework consists of factors divided across five categories which denote the multidimensional perspective of trust.

Chapter 7 A Trust Assessment Tool for measuring and improving trust in agile teams

This chapter presents a trust assessment instrument based on our empirically developed framework for building trust. **A**ssessment of **T**rust in **A**gile Teams (**ATA**) is an instrument to measure trust indirectly. It takes into consideration trust measuring factors across five dimensions namely: Working Environment, Leadership, Organizational, Personal and Socio Cultural. ATA is used for calculating composite trust scores within any agile team and then provide feedback on the basis of these scores to improve trust. Relation between trust levels and performance of distributed agile teams is established using ATA. ATA comprises a questionnaire consisting of 34 items across five constructs (dimensions) of trust.

Chapter 8 Conclusion and Future Work This chapter summarizes the thesis and describes the contributions and limitations of the study. Finally, some directions for future research are presented.

CHAPTER 2

LITERATURE REVIEW

Since its inception in 1960s, the field of software engineering has witnessed the emergence of a number of process models to structure, control and manage the work of software development. In this chapter, we first of all trace the emergence of agile software development as an outcome of application of various process models. The emergence of every new process model was built upon the success and failures of the previous models. This chapter also presents at thorough review of two popular agile software development methods: Scrum and XP. This is followed by an overview of agile testing and social aspects with respect to agile teams.

2.1 The Emergence of Agile Software Development

Waterfall model was one of the first process models to be proposed. This model was proposed by Winston W. Royce in 1970 [43]. Waterfall model follows a sequential approach to software development. The model has five distinct phases as shown in Figure 2.1 namely– Requirements Specification, Design, Implementation, Verification and Maintenance.

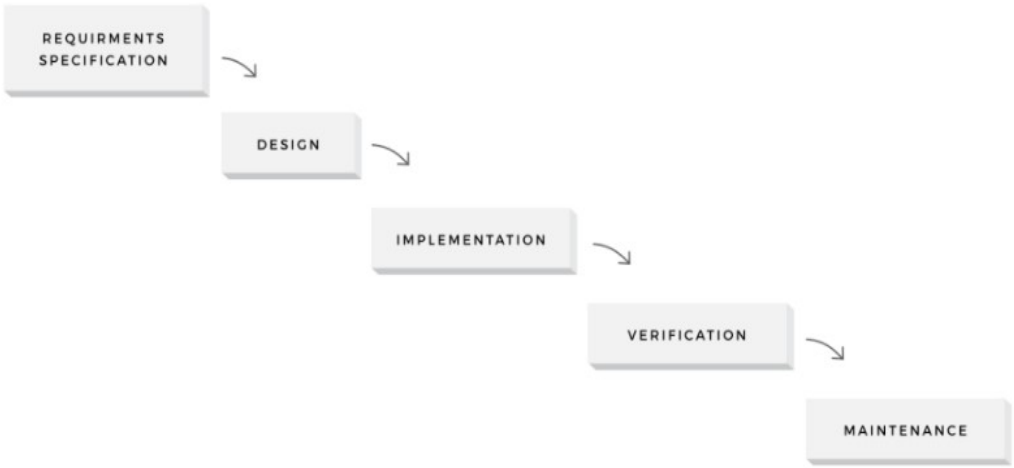


Figure 2.1 The Waterfall Model [43]

All the phases occur in this sequence and do not overlap. Furthermore, each phase must be completed before proceeding to the next phase. Even though largely popular even today, the model has a serious flaw of "Fixed Requirements Scope". Numerous studies conducted to ascertain the root cause of project failures indicated that the

Waterfall approach to full requirements definition prior to development is not a realistic approach in the face of changing business and customer requirements.The application of the waterfall model is therefore limited to the projects where requirements are well understood beforehand [44].

In the years to follow, industry pundits fiddled with more innovative ideas to accommodate "change" in the software development processes. The failure to handle larger projects drove the need for discovery based models [45]. 1980s and 1990s saw the continuum of a number of such models. The Spiral model, Rapid Application Development (RAD), and Rational Unified Process (RUP) were the popular ones. In these models requirement gathering is not a "fixed stage" but rather a continuous process along with stakeholder's involvement in requirements definition. The spiral model [46] that was developed by Barry Boehm in 1986 was the first published model in this series. This model was the starting point for the iterative and incremental approach to software development as shown in Figure 2.2. Software development took the shape of a "spiral" comprising of sequential steps of designing, coding, implementation and testing with a strong pace for feedback amongst these stages.

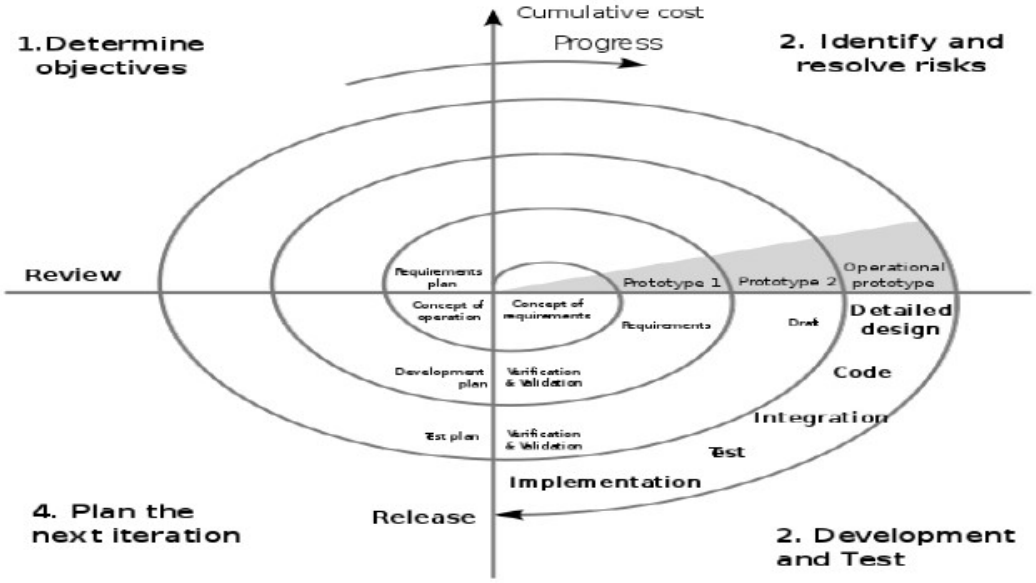


Figure 2.2: The Spiral Model [46]

Spiral model was followed by Rapid Application Development (RAD) model [47] developed by James Martin in 1990s. The idea was to develop fast and deliver before requirements changed. And even if requirements changed the tools were fast enough to get it right faster than traditional paper based requirements discovery models. In

some ways this model was the harbinger for the successive models. Rational Unified Process (RUP) [44] that followed RAD was largely meant to address the needs of large scale application development. This model was the first of its kind that recognized the necessary overlap amongst the various stages of software development life cycle phases of *inception, elaboration, construction, and transition* as shown in Figure 2.3.

In the discovery based iterative and incremental models there was a meaningful concern to shift from large scale documentations like requirements specification (SRS), Design specifications to lighter weight documentation, vision documents, use case models and so on.

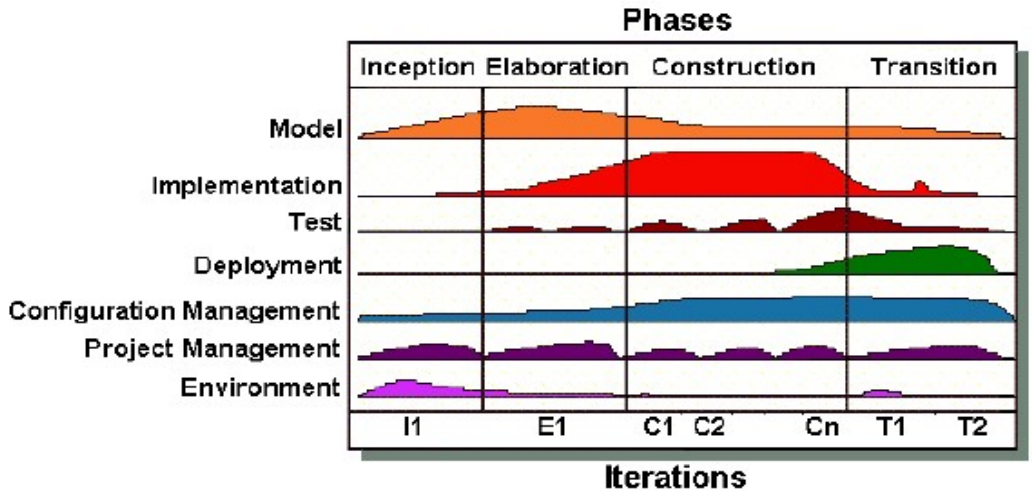


Figure 2.3 Iterations and workflow of RUP [44]

Those are used to define initially what is to be built. Based on this initial understanding the iterative process is applied to discover actual user requirements. Once requirements are defined in a better way, theycan be implemented in relatively easier steps comprising of coding and testing to assure that the system confirmedto the agreed behavior.

2.2 Agile Software Development

Agile software development methods are based on iterative and incremental style of development where value is given to short development cycles, frequent delivery of software, constant customer collaboration, face to face communication and responsiveness to change [22]. Agile development methods prove to be good

replacement for traditional, heavyweight, document driven development methods like waterfall.

In 2001, seventeen dignitaries from software development arena came together to discuss new and innovative ways of developing software and the result was the agile manifesto [48]. The Agile manifesto was designed with clear vision for producing a flexible, lightweight, and team- oriented approach for software development. The key four values of Agile Manifesto [48] are as follows:

"Individuals and interactions over process and tools,
Working software over comprehensive documentation,
Customer collaboration over contract negotiation,
Responding to change over following a plan,
That is, while there is value in the items in the right,
We value the items on the left more."

The agile software development is backed up with twelve principles primarily focused on priority to satisfy customer needs through early and continuous delivery of valuable software, welcome changing requirement even late in the development, delivering working software frequently from a couple of weeks to couple of months, building project around motivated individuals and trust them to get the job done, emphasis on effective communication and making self organizing teams.

Agile methods provides a beautiful way of accommodating changes through iterative and incremental style of development, working on small functionalities and delivering value quickly in small time boxed iteration. As agile methods emphasis on continuous customer collaboration and feedback, customers are allowed to prioritize the features according to their need.

Agile software development is an umbrella term that includes several methodologies like: Feature Driven Development (FDD), where work is divided into set of features [49]; Crystal, describes the number of methods and principles for customizing them as per the requirement of the project [50]; Dynamic Software Development Method (DSDM), an agile project delivery framework for complex problem solving tasks [51]. Adaptive Software Development (ASD), which focuses on concept and culture while ongoing development [52] and Lean framework which promotes fast software

development with less time, effort and cost. Lean works on the basic principle of eliminating waste [53]. However, the two most famous and widely used agile methodologies across the world are Scrum and XP [54]. In Scrum focus is on project management [55] while in XP focus is on team and engineering practices. A detailed description on Scrum and XP is given in the following section.

2.2.1 Scrum

The development of scrum has roots in the 1986 Harvard Business Review article, "The New New Product Development Games" by Takeuchi and Nonaka in 1986 [56]. This article provides a brief overview of the remarkable results that companies like Honda, Canon and Fuji-Xerox had achieved by using scalable team based approach [57]. This article also showcased the importance of self managing, self organizing teams and outlines the management's role in the development process [57]. Then, Jeff Sutherland and Ken Schwaber formally developed the scrum [58]. Scrum is based on sprint based development. Sprints are time boxed cycles typically between 2 to 4 weeks [59]. Scrum is a people centric framework that is based on a set of values, principles and practices that provides a solid foundation to the organizations for organizing and managing work [57]. Scrum framework is shown in Figure 2.4. A brief overview of various scrum roles, artifacts and ceremonies are provided below:

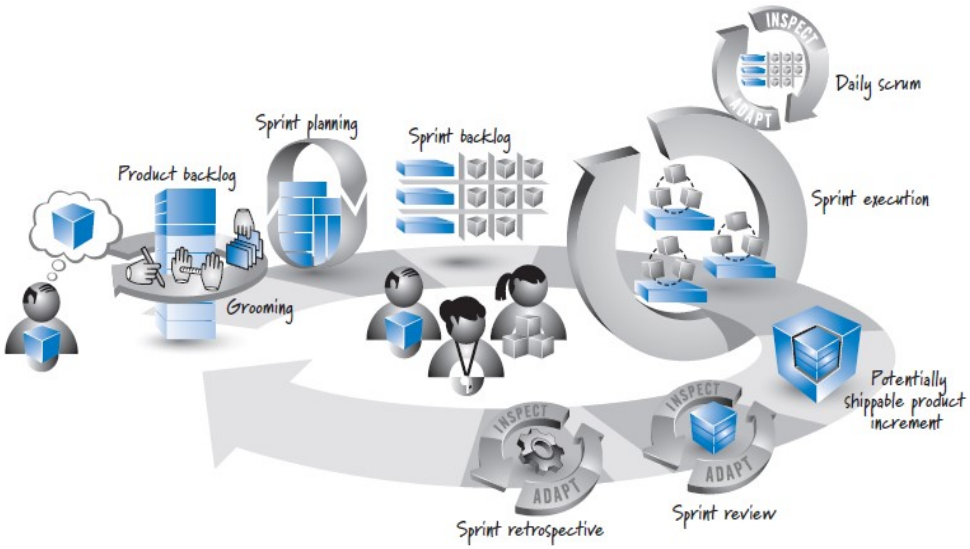


Figure 2.4 Scrum Framework [57]

- **Scrum Roles** - Scrum team comprises of three main roles: Scrum Master, Product Owner and the Development Team. Scrum framework has these three standard roles in addition to that any other role can also be possible depending on the project requirement.

- **Scrum Master** – is the key enabler who inculcates scrum values, principles and practices to the scrum team. He acts as a facilitator whose primary job is to remove any impediment which team is facing and transforming team into a high performing functional team. Scrum Master should not act as a manager rather he should be a servant leader without any authority or command and control attitude. Scrum Master shields the team from any outside interference and at the same time resolves any internal conflicts amicably within the team level. In a nutshell, Scrum Master helps the scrum team and the rest of the organization to develop their own team oriented, organization specific scrum approach.

- **Product Owner (PO)** – is the person who envisions the product. Product owner decides about the functionality and the feature that would go in the product and also the order in which they are going to be built. PO is the one who creates the sprint goals for the scrum team and gives them the clear vision regarding their future course of work. Ideally, PO is the customer representative who decides whether to accept or reject the product.

- **Development Team** – is a cross functional, diverse group of individuals who are responsible for designing, implementing and testing the product. Development team has the capability to self organize and find out the best way to achieve the goals as set by the product owner. Development team size varies from five to nine people.

Scrum Artifacts

- **Product Backlog –** is the prioritized list of features that has been finalized by the stakeholders and the scrum team. It includes all the features visible to the customer as well as technical requirements needed to build the product. Its product owner's responsibility to collaborate with the internal and external stakeholders to elicit all the requirements and define them in the product

backlog. It is a constantly evolving artifact as items can be deleted, added or changed as business conditions change or as the scrum teams understanding towards the product grow.

- **Sprint Backlog** – is a subset of the Product Backlog. Each feature that is to be developed in a particular sprint is broken down into a set of tasks. The collection of these tasks along with their associated product backlog items is known as Sprint Backlog.

- **Burndown Chart** – is the visual indicator that displays the cumulative remaining work on day by day basis. It helps the team in tracking the development schedule for a particular sprint.

Scrum Ceremonies

- **Sprint Planning Meeting –** Sprint planning is the first activity before starting the sprint. During this meeting entire scrum team along with the product owner decides and reaches an agreement on the sprint goal that defines what team is going to achieve in that sprint. Then scrum team provides estimates for the features to be developed. Normally, sprint planning meeting should take around four to eight hours for two to four weeks sprint.

- **Daily Scrum –** It is daily meeting that lasts for around 15-20 minutes preferably at the same time. It is also known as daily stand up because scrum team prefers to perform this meeting in stand up position as it promotes more brevity. Scrum master leads the team in daily scum and every member is expected to answer three questions: What I have done yesterday? What I will do today? What impediments I am facing?

- **Sprint Review Meeting –** is conducted to inspect and adapt the product that is being developed. Scrum team, customers, stakeholders and other interested members from other teams can take part in this meeting. The agenda of this meeting is to discuss the status of just completed features with respect to the overall development effort. It's the opportunity for the scrum team to get the feedbacks and work better to achieve the sprint goal.

- **Sprint Retrospective –** It is also an inspect and adapt activity performed at the end of every sprint. Ideal time for conducting the sprint retrospective is after the sprint review meeting and before the next sprint planning meeting. During this meeting, Scrum Master, Development Team and Product Owner comes together to discuss what works well and what did not work during the sprint and how they can improve in the next sprint.

2.2.2 Extreme Programming (XP)

Kent Beck was the creator of extreme programming [60]. XP can be defined as "a light weight methodology for small to medium sized teams developing software in the face of vague and rapidly changing requirements" [60]. XP aims to produce high quality products by organizing people effectively and following technical practices. Traditional software development models were prone to risk of cancellation of the project, non adherence to planned schedules and delivering features that could not please the customers. XP handled all these problems quite effectively by having short release cycles, by making customer as part of their team to get fast feedback so that business impact is minimal. In XP, only high priority features are implemented and features with no business value are placed at the low priority levels. XP is driven by some fundamental values, roles, and practices which are described below [60].

XP Values

- **Communication:** XP based projects are expected to build and deliver at a faster rate. But lack of effective communication between the team members often affects the delivery. In XP projects, communication needs to be very prompt and regular between the developers, testers, product owners and managers.

- **Simplicity:** In XP, prominence is on finding the simplest solution for the problem. It implies that team's focus should be on design and coding for the present problem instead of thinking for future problems. It follows the KISS principle ("Keep It Simple, Stupid!").

- **Feedback:** Practicing unit testing and periodic integration testing after any change, provides feedback to the developers on the state of the system.

Customer provides the feedback by means of executing acceptance or functional tests. For every new requirement, team provides the estimate on how much time they will take to develop and deliver particular user stories. Feedback is closely associated with the communication. The mechanism of feedback provides a way to periodically assess the system and improve it.

- **Courage:** Courage is needed to encourage the team to work on today's problems and not to get bogged down with tomorrow's problem that are not relevant today. Refactoring of the code is done to review the existing system and changes are made to make it suitable for use. Simultaneously, old and obsolete code is thrown away.

- **Respect:** Here, respect implies the sense of belongingness between the team members. Developers should not commit the code that could introduce compilation errors or that can fail existing unit tests. There should be respect of work for each of your peer. Team as a whole is appreciated and rewarded for the good work, nobody should feel isolated. Respect comes from effective collaboration and communication among all the team members. Trust is a crucial factor in establishing good communication and respect between the developers, senior management and the customers.

XP Practices – there are 12 practices in XP [60] as shown in Figure 2.5. These practices are further grouped into four categories:

1.) Final Scale Feedback

- Planning game - It is a type of meeting that is conducted once per iteration or once a week. This meeting is like project planning meeting where decisions are being made related to the scope, budget, schedule, features to be developed and other estimates. The purpose behind this meeting is to bring all the stakeholders on the same page and to guide the product from the concept to the delivery.

- Pair programming - It is a technique where two coworkers sit together on one workstation and program together. One of them, the driver writes the code and the other, the observer reviews each line of the

code. The roles are often switched between the driver and the observer. This technique brings the more strategic view as observer watches and suggests any improvement to the system.

- Test driven development - It is a methodology where test cases are designed first, then code is developed that would be enough to pass those tests [61].

- Whole Team -XP advocates constant customer collaboration. Customer is someone who actually uses the system, he should be available for questions. Moreover, development is seen as a collective responsibility of the whole team including customer.

2.) Shared Understanding

- Coding standards – Entire development team adheres to certain set of rules throughout the project which are termed as coding standards. The style and format of coding is governed by the coding standard. Coding standards may vary from project to project and depends on the chosen programming language [62].

- Collective code ownership – Implies that code is owned by the entire team and anyone can change any part of the code. It facilitates quick fix in case an error is encountered by any developer during pair programming. By designing and running good unit tests, a team can avoid occurrence of bugs.

- Simple design – Teams are encouraged to choose simple design to avoid complex solutions to simple problems.

- System Metaphor – It is the process of simplifying the requirements or user stories for the agile teams. It is the story that briefs about the system functionality.

3.) Continuous Process

- Continuous integration–It is the practice of integrating every code change into the central repository. These check in's then verified by

executing a set of automated tests which detect any bug or defects early in the development cycle.

- Refactoring – It is the process of modifying the existing/ legacy code without changing the actual/external behavior of the system. Refactoring helps in discovering and removing duplicate codes, improves communication and makes the system flexible.

- Small releases - help teams to develop and deliver features quickly over short iterations.

4.) Programmer Welfare

- Sustainable pace – This practice deals with the fundamental factor of programmers well-being and to shield them from the unwanted stress. According to this practice programmers should not work more than 40 hours a week. Any overtime in a week is followed by no overtime in the subsequent week.

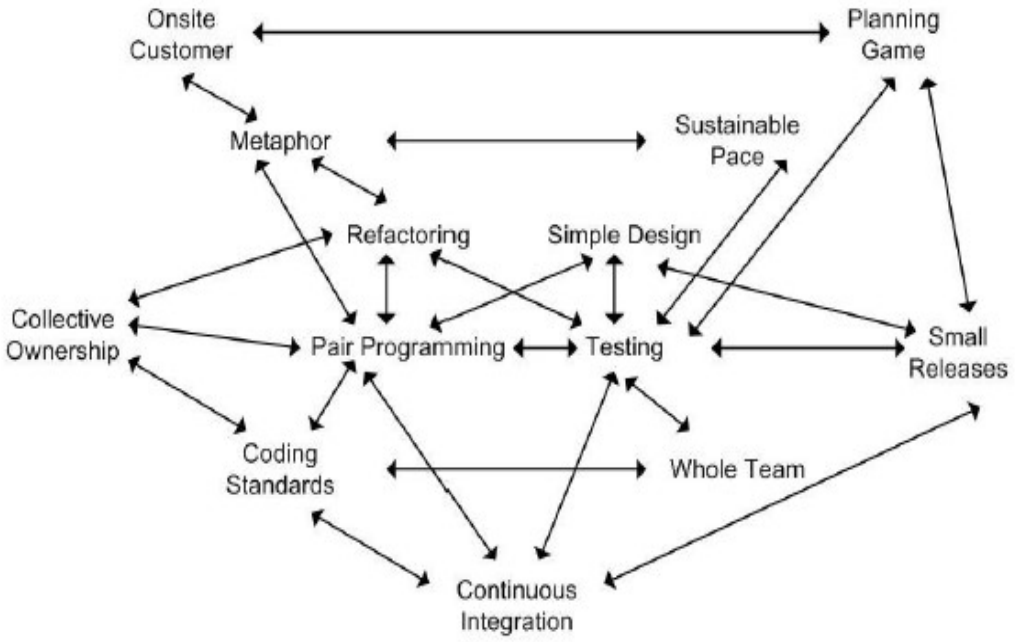


Figure 2.5 – XP practices [60]

2.3 Testing in Agile Software Development

Agile testing does not just mean testing on agile projects but testing an application with a plan to learn about it [63]. Agile testing is driven by customer information and feedback by following agile values [63]. In contrast to traditional testing, testers in agile software development have different roles where they help business by refining user stories, help developers in writing automated unit tests, automate regression tests and provide quick feedback to the team [64].

Agile has changed the way software had been developed traditionally. In agile, major focus is on testing and quality [64]. In agile projects testers are not treated as a separate entities, but are very much part of the team. As quality is the foremost concern of the agile teams, it reinforces the importance of involving the testers right from the onset of the project. In agile testing, different set of tests are performed at different levels [65]. Figure 2.6 shows the agile testing quadrants which describes the nature and reasons for performing different tests. Quadrant 1 (Q1) represents the technology facing tests which focuses testing against the technical requirements. Quadrant 2 (Q2) represents the business facing tests which verify business requirements. Quadrant 3 (Q3) represents business facing tests to criticize the product. Quadrant 4 (Q4) represents technology facing tests to criticize the product.

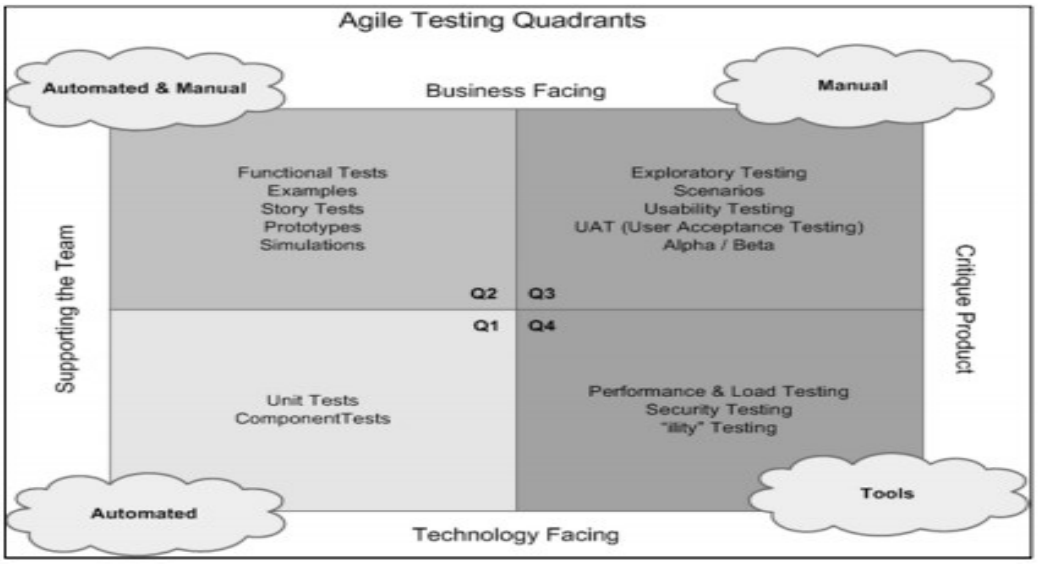


Figure 2.6 Agile Testing Quadrants [66]

In traditional waterfall model, testing focuses almost exclusively on the right side (Q3 and Q4), the functional tests are made to criticize the product but not playing a productive part in supporting the creation of the product [67]. Moreover, it is involved late in the development to detect failures but not to prevent them. Agile testing is all about the regular interaction between developers, testers and customers to prevent any failure. Agile testing is a challenge for testers that used to work in traditional waterfall projects mainly because they do not have to wait for system delivery at the end of the project to start the test activities but they need to be proactive and start automating the tests right from the beginning of the sprint [65].

It's imperative for agile development teams to practice test driven development (TDD), automated testing and continuous integration in order to avoid the technical debt [57]. According to Einsenberg [68], test automation is an integral part of the agile development projects. Software organizations that are undergoing agile transformation have faced several issues related to the maintainability and time consuming development of automated tests [69]. In agile environment, emphasis should be on making automated tests reusable so that they can be utilized in the case of repetition of tests [70]. A study by Berner et al. [71] highlighted the importance of test automation in detecting bugs at the earlier stages in the sprint development, thus saving cost, effort and improving the quality of the product. Sfetsos et al. [72] performed the systematic literature review on the quality of agile practices. In the light of reviewed literature, three main categories were identified that ensures quality on agile development projects namely: test driven development, pair programming and refactoring. Several studies [73-77] reported that use of test automation while incorporating developing test driven development on agile development projects yields multiple benefits pertaining to low defect count, significant improvement in the quality and reduction in code and design complexity. Table 2.1 presents the review of studies on functional test automation on agile development projects.

Holmes et al. [78] have used selenium IDE for functional test automation for nightly builds. Selenuim IDE was used in automation and execution of acceptance test cases. Automation testing framework by using selenium webdriver was developed for automating regression tests [79]. Data driven capability was incorporated in this framework to make it maintainable [79]. Cervantes et al. [80] developed a Software Test Automation Framework (STAF)

Table 5.3 Comparative review of studies based on functional test automation on agile development projects

Name of Study	Framework Developed	Type of automated test	Automation tool used	Test reusability	Test script maintainability
Holmes et al. [78]	No	Functional test/ Acceptance test level	Selenium IDE	No	No
Satish et al. [79]	Automation testing framework	Functional/Regression testing	Selenium WebDriver	No	Yes
Cervantes et al [80]	Software Test Automation framework(STAF)	Functional test/ System testing	STAF + STAX	Yes	No
Chaiprasert h et al [81]	Test automation Framework (TAF)	Functional test/Acceptance testing	Test Runner	No	No
Razak et al [82]	PyTestFramework	Functional/ Regression test level	Selenium Webdriver	No	No
Thopate et al [83]	Chameleon model based test automation framework	Functional/Regression	WGET	Yes	No
Liu [84]	Spreadsheet/Data driven framework	Functional/Regression	Quick Test Professional (QTP)	No	Yes
Arya et al [85]	Keyword Driven framework	Functional/Regression test level	Selenium WebDriver	Yes	No

using resuable functions written in python to automate system level test cases. The STAF framework was mainly used to automate GDS system level test cases and although initial results were satisfactors but with the limiataion of maintainability of test scripts. Another study [81] reported the use of Test Automation Framework

(TAF) using test runner tool developed for Point of Care Testing (POCT) medical diagnostic system. Acceptance level test cases were automated using TAF but no maintiablity and resuability were achived. Razak et al. [82] develped a PyTestFramework using webdrive 2.0 for agile devlopment projects. Testing efficiency have increased using PyTestFramework. Another study [83] highlights the advantages of atuomating ever changing test cases in agile environment with the help of chameleon model based framework, WGET tool is used for automating the tests. Liu et al. [84] reported the data driven test automation framework using Qucik Test Professional (QTP) tool. This framework was designed for automating regression test that were maintainable with the use of script less coding. In a study [85] authors presented a keyword driven automation testing framework for testing web applications. This framework is used for creating reusable automated regression tests.

2.4 Social Aspects of Agile Teams

Foremost objective of any agile development project is to deliver quality software at shorter intervals that can yield significant economic returns. Agile development projects are built around motivated team members who are fundamental entities of the project [66]. Agile manifesto itself emphasizes on people and their interaction during software development process [66]. The manifesto opened avenues for the researchers to study the social aspects of agile development. Agile teams are democratic in nature where team members are at the same level without any senior or junior position. Agile team members own their decisions and are expected to be self organized to perform their tasks [86]. Many studies [87-89] have depicted team member's personal attributes and social factors as the biggest cost drivers for any software development projects.

Team performance of software development teams gets impacted by the social factors such as team communication and coordination [90]. A Study by Abrahamsson et al. [91] demonstrated the improvement in team performance on embedding certain agile practices. Some of the XP practices like pair programming and planning game encouraged teams to build strong social communication that resulted in improved team performance. An ethnographic study [92] reported the role of onsite customer in XP teams and found this role to be unfeasible because of the constant demands placed to the onsite customer.

A grounded theory research study conducted by [93] highlights the importance of motivation and team cohesion during various agile practices like daily and weekly meetings. This study also discussed the role of information radiators like burn down charts that help the team in visualizing the progress of the project and act as continuous source of the motivation. However, the study also revealed that most of the activity on agile projects is centered and revolves around developers role which often resulted in the ignorance of other roles like testers, business people and technical writers [93].

2.4.1 Human Dynamics in Agile Teams

Quality in agile development methods is also related to the human factors involved, mainly because of the implicit mode of development in agile methods that depends heavily on human effort, allowing them to choose their work, deciding the priorities and strong emphasis on interpersonal communication. In the agile development methods, traits of the individuals involved in the development also affect the quality significantly [94]. Organizations that are transforming to agile from traditional waterfall approach often need to overcome challenges related to culture and mindset of the people involved in the project [95]. Other set of human related issues are unfavorable working conditions, conflicts between the team members and poor staffing on agile development projects [96]. Several studies [97, 98] mentioned that pair programming, mentoring and transferring inter personal knowledge are the key activities in XP and scrum projects that improve quality. In agile development projects emphasis is on disciplined joint development of the software with the amalgamation of communication, interaction and collaboration among the team members [94]. Sfetsos et al. [99] proposed an adaptive pair formation model to build high performance pairs of developers. Pairing is done on the basis of developers temperaments and personalities. A systematic literature review of empirical studies by Dyba et al. [100] found out thirty six empirical studies on agile software development, of which seven talked about the role of social and human factors. These studies showed the impact of team social practices on the software development performance. These studies reflected that social factors and motivated team members can contribute in improving the team performance on agile projects. Several researchers [101, 102] asserted and urged the creation and use of theories especially in agile software development.

CHAPTER 3

RESEARCH DESIGN

In software engineering research, it is not only important but essential to use proper research methods. A research design gives the roadmap of how to carry out research and collect the data. This chapter sheds light on some popular research methods. This is followed by a description of the Grounded Theory research method which has been chosen for this particular study. This chapter sheds the light on the research method used in this study along with different research perspectives.

3.1 Research Approaches

There are two basic approaches to research viz. Quantitative and Qualitative. Quantitative research is carried out by a fixed research design and qualitative research is performed through a flexible design. This section provides a description of qualitative and quantitative research approaches. Thereafter, we present our motivation behind choosing qualitative research approach.

Quantitative research approach

Quantitative research is performed to measure a natural phenomenon through a fixed design. Quantitative research involves data collection in a quantitative form which is analyzed in a formal manner [95]. Quantitative research is performed to study the cause and effect relation between variables in order to study the cause and effect relation between variables in order to study an activity or a phenomenon [103].

Qualitative research approach

Qualitative research is carried out to study phenomenon and objects in their natural settings. The research design in qualitative research is kept flexible to adapt to changes in the observed phenomenon as and when they arise and come to the notice of the researcher. Qualitative research studies are informed by visual and textual data that is gathered from in depth interviews, observations or both. Qualitative research approach is adept at research related to human and social behavior. The objective behind qualitative research studies is to explore the social and human dynamics of the research participants [104]. The nature of behavioral sciences and software

engineering has much in common with respect to human involvement. Qualitative research approach has been widely used and is a time tested approach for studying the phenomenon and objects in behavioral and social sciences. Recently, there has been gradual increase in the number of research studies investigating different social and human aspects of software engineering by using qualitative research methods [105-107]. Taking cue from behavioral sciences, a qualitative research approach has been adopted for the purpose of this study primarily because software development is inherently a social activity involving people developing software. Our motive is to understand the behavior and actions of software teams. Performance of software teams is also seen to be influenced by the social processes [90].

3.2 Overview of Research Methods

This section briefly provides a description of some popular research methods and also presents our motivation for choosing the grounded theory as our research method.

3.2.1 Case Study

A case study is described as "an empirical inquiry that investigates a contemporary phenomenon within its real-life context, when the boundaries between phenomenon and context are not clearly evident, and in which multiple sources of evidence are used" [108]. Case study based research identifies the cases according to the pre-formulated research questions [109] and then thorough investigation is required to understand the phenomenon within the given context and time [108].Case studies may be based on a single case or multiple cases.

3.2.2 Survey

A Survey is a system for collecting information from or about people to describe, compare or explain their knowledge, attitudes and behavior [110]. Questionnaires and interviews are used to collect quantitative or qualitative data in a survey. A sample which is representative of the total population is used to collect data. The results derived from the analysis of the sample are generated for the entire population. Researchers develop questionnaires in order to efficiently use the survey method [111]. This research also incorporated a questionnaire development for assessment of factors that contributes in improving trust levels within agile teams. To ensure the reliability and validity of the questionnaire numbers of statistical methods are

employed. In order to validate the construct validity, exploratory factor analysis [112] is performed on the items. Cronbach's alpha was used to validate internal consistency [113].

3.2.3 Ethnography

Ethnography gets its roots from Anthropology. Ethnography is the study of people in their own environment through the use of methods such as participant observation and face-to-face interviewing [114]. According to [115] "ethnography documents cultural similarities and differences through empirical fieldwork, also it can help with scientific generalizations about human behavior and the operation of social and cultural systems". Many studies [116, 117] have used ethnography research method to study social aspects of agile teams.

3.2.4 Grounded Theory

Grounded Theory (GT) method is a systematic research method where prominence is on the generation of theory derived from systematic and rigorous analysis of data [118]. GT is one of the widely used research methods across multiple disciplines like social sciences, psychology and healthcare [119,120]. The emphasis in GT is on new theory generation which means rather than beginning with a pre-conceived theory in mind; the theory evolves during the research process itself and thus the product of continuous interplay between data collection and analysis of that data [121]. In GT, main focus is on generating a substantive theory about the phenomenon being investigated rather than finding answers to specific research problem or questions in particular [122, 123].

3.3 Selecting our research method

Researchers using qualitative research approach may opt case study, ethnography or grounded theory as their research method. However, for the purpose of this study ethnography and case study research methods were not adopted because of following reasons:

- To pursue research using case study method, research question must be formulated beforehand after the extensive literature review on the topic of interest. Our objective was to understand the real concerns of the agile

practitioners that should be coming from the real project settings and not from the existing literature.

- Ethnography research expects the researcher to spend a considerable amount of time in the real surrounding of the community under study. Researchers are also expected to observe the field settings, to interact with the participants. It requires lot of time and fund to conduct an ethnography study.

Grounded Theory has been chosen as our research method for the following reasons:

1. Agile manifesto gives strong emphasis on people and interactions and GT has also been used extensively to study social interactions and behaviors of people [124].
2. GT is most suited to areas of research that have not been explored in detail, and according to our knowledge, there are relatively less number of empirical studies that speak about quality in the agile software development context.
3. GT focuses on theory generation, rather than extending or verifying existing theories [125].
4. GT is being liberally used to study the agile teams [93, 107, 126-128].

3.4 Research Perspectives

Researchers in computer engineering disciplines often use *Positivist*, *Interpretive* and *Critical Philosophical* research perspectives [129, 130]. In this section, brief overview on these research perspectives is provided.

Positivist: Positivist view asserts that real world events can be observed empirically and their explanation should be based on logical analysis. Positivist view is concerned with testing so that predictive understanding can be made about the phenomena" [104, 131]. Survey Research and Case Studies based research prefers to use positivist perspectives [104].

Interpretivism: In interpretivism perspective, reality is based on the subjective experience of the people involved in the research which tends to follow that reality is constructed socially [132]. Interpretive perspective is more about people concern and

their perceptions about the phenomenon. Taking an objective view of the world is discouraged in this perspective. Dependent and independent variables are not predefined in interpretative research studies rather it concentrates on the complexity of human sense making as the situation emerges [132]. Interpretive perspective aims to explain the subjective reasons and meanings that lie behind social action. We have used grounded theory with an interpretive perspective in this research mainly in two ways: firstly focus of this study would be on new theory generation rather than verifying old one. Secondly all the findings from this study are grounded in the data that is collected from the real surroundings of the context studied [104].

Critical: This perspective assumes that 'reality' is apprehendable. Advocates of critical theory acclaimed that reality is created and shaped by social, political, cultural, economic, ethnic and gender-based forces that have been crystallized over time into social structures that are taken to be natural or real [133]. Action research is the most commonly used research method associated with critical perspective which mainly used for studying the conflicts in society [133].

3.5 Researcher Role

In interpretive GT research studies, researcher plays a very vital and central role in conducting the overall study. As the researcher is interested in exploring the research phenomenon through the perspectives of participants, it is imperative for the researcher to have the clear vision to understand the complex human process [134]. The entire data collection and its analysis heavily rely on the competency and skills of the researcher [135].

Grounded Theory studies requires a researcher who has a good reasoning ability, good in questioning and prompt in understanding the connections between the concepts and categories. Combinations of all the above qualities are important for conceptualizing and formulating an emerging theory from the data [136]. It is also important for a researcher to address the researcher bias through self-reflection in the early phases of the research; this can be done by self-disclosing any assumption and belief that is used to shape the research study.

After completing my Masters in Technology (M.Tech.) with honors distinction in Information Technology from University School of Information and Technology,

Guru Gobind Singh Indraprastha University, Delhi. I got campus placement in Tech Mahindra, one of the leading IT solution provider and outsourcing multinational company in India. Initially I worked as a developer, then moved to QA and worked for different projects in different roles. There was a brief exposure of agile development towards the end of my job days at Tech Mahindra and Indian software industry was also witnessing the marked move towards agile development methods at the same time. Although, I have a good experience in teaching different software engineering courses, I ensured that my industry as well as teaching experience would not cloud my research with any preconceived ideas. Being nascent to agile development field, I approached the interviews as well as observations with fairly open mind. I have gained knowledge on Grounded Theory by extensively reading literature from the well acclaimed grounded theory sources [137-139]. Also I took help from my supervisors and distinguished faculty members from management and social sciences discipline from reputed universities like Jawaharlal Lal Nehru University (JNU), and Guru Gobind Singh Indraprastha University, Delhi, India on the grounded theory research method.

3.6 Grounded Theory

This section provides a brief overview of grounded theory and also details about our application of grounded theory in this research.

3.6.1 Brief Overview of Grounded Theory

Grounded Theory (GT) is defined as "General method of research that uses systematic methods to generate an inductive theory about a substantive area which is based on the data and its rigorous analysis" [125]. Barney G. Glaser and Anslem L. Strauss jointly developed grounded theory and laid its foundation by publishing a book on grounded theory "Discovery of Grounded Theory" in 1967 [136]. GT research focuses mainly on exploring and generating the theory rather than verifying it [140]. Theoretical sampling and constant comparative method are two fundamental characteristics of GT which drives the overall data collection, its analysis that includes constant comparison on the abstract concepts and categories. This process ensures the development of integrative theory that is grounded in the real data.

GT research method offers the set of guidelines to identify concepts, categories and how to establish relationships between them [141]. While generating a theory using GT, researchers aims to uncover the key concern of the research participants and also attempts to find out the ways of resolving that concern. Since the inception of grounded theory, the Glaser and Strauss alliance gradually separated until each was developing a different version of GT. While Glaser maintains that GT is independent from any philosophical stance, Strauss along with Corbin gradually shifted toward interpretivism [109]. Meanwhile, Charmaz's (student of Glaser), developed 'constructivist' grounded theory by reinterpreting GT from a constructivist's stance [142] that closely resembles interpretivism. Glaser GT states that researchers should start with the general 'Area of Interest' rather than with specific research questions, as formulating specific questions and focusing on them may lead to preconceived ideas or hypothesis of the research phenomenon [136]. Strauss and Corbin argue that research question should be pre-set as it defines the boundaries around the research area. In Glaserian GT, an extensive literature review is avoided until the theory starts to emerge to prevent any influence from existing concept on emerging theory. Whereas in Straussin GT, the literature may be consulted throughout the research and concepts from the literature may be used if needed. Table 3.1 provides brief overview on the three version of GT. Because of these different versions of GT, it is necessary for every researcher to explicitly indicate the version of GT they are using. This research study employs the Glaserian/Classic GT as large number of software engineering research studies effectively used this version of GT and also due to plenty of resources available [143].

In the following section, the fundamental components of GT are described followed by the application of GT in this research. A working example from this research study are also presented for better understanding the application of GT. Text in italics represents the concepts and categories that emerged from the analysis of the data. The process of GT is explained with the help of Figure 3.1 by providing overview of different procedures of GT.

Table 3.1 Brief comparison among three GT versions

Element	Classic/ Glaserian GT [132]	Straussian GT [109]	Charmaz's GT [142]
Research Question	Should not be defined a priori. The researcher should start with an 'area of interest'	Research question may be defined upfront, can be derived from the literature	Research begins with the initial research question which evolve throughout the study
Role of literature review	An extensive literature review should be delayed until after the theory is emerging to prevent the influence of existing concepts on the emerging theory.	The literature may be consulted throughout the process, as concepts from the literature may be used if applicable	Acknowledges not only Glaser's reasons for delaying the literature review but also the impracticality of this strategy. Need to tailor a literature review to fit the purpose of the GT study
Coding procedures	Three types of coding : Open coding , Selective coding and Theoretical coding	Three types of coding : Open coding, Axial coding and Selective coding	Three types of coding : Initial coding , Focused coding and Theoretical coding
Philosophical influences	Objectivism: There exists a single, correct description of reality; the researcher therefore discovers grounded theory from data.	Pragmatism and symbolic interactionism: actors engage in a world that requires reflexive interaction.	Social constructiveness: social reality is constructed by our individual and collective action.

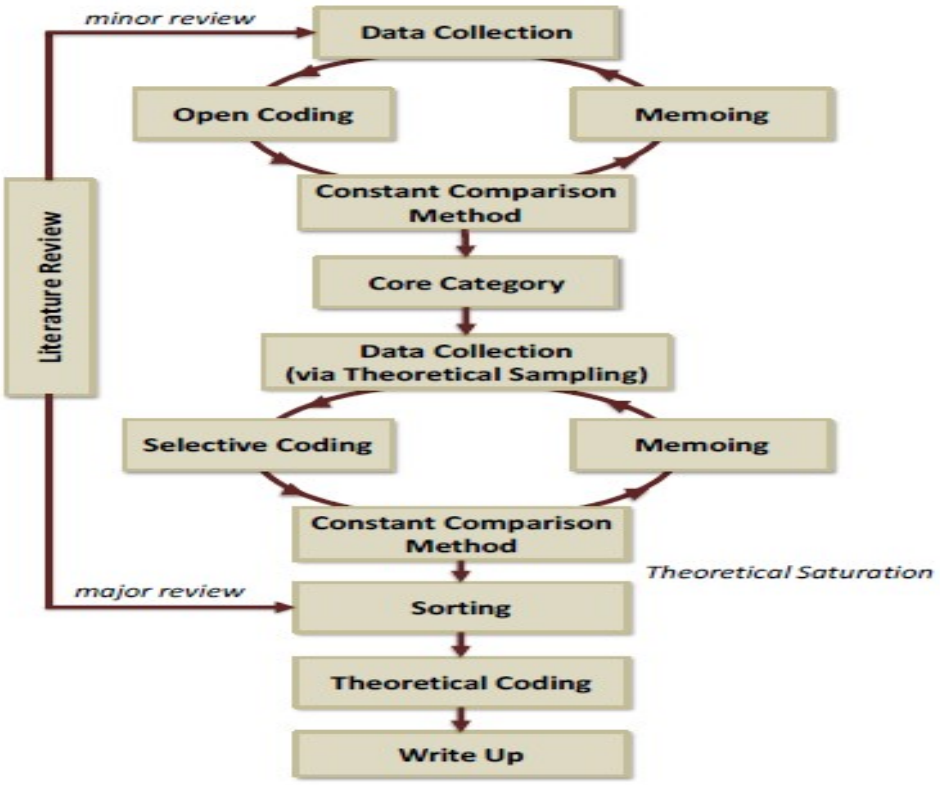


Figure 3.1 Grounded Theory Process [144]

3.6.2 Selecting the Research Area

Glaserian GT prohibits researchers from formulating a research question up front. The reasons behind this recommendation are:

- The GT method is all about generating a new theory that should be grounded in the data, and if the researcher will commence the research with preconceived research problem or research questions then it may limit the researcher in the exact exploration of the phenomenon being investigated [145].
- Ideally in GT, the main idea is to uncover the problem of the participants hence research problem should come from the participants of the study and should not be predetermined or forced by the researcher [145].
- GT is based on the fact that research problem along with its key concerns should emerge in due course of data collection and its analysis [125, 146]

Although, it is advisable to not to frame a research questions up front but researchers are required to choose a general area of interest. This research has been initiated by selecting 'Agile Project Management' as our general area of interest owing to its huge popularity in software engineering research across the world [86, 92, 147].

3.6.3 Minor Literature Review

According to the guidelines provided in the Glaserian or Classic GT, doing extensive literature review in the same area in the early stages of the research should be avoided [145]. As per Glaserian GT "Attempting an extensive literature review before the emergence of the core category violates the basic premise of GT" [148]. But still minimal literature review can be performed to get acquainted with area of research.

Following Glaserian GT guidelines, only minimum literature review was conducted at the beginning of our research. Mostly literature was used to gain the preliminary understanding on the agile methodology. As the interview process begun, our understanding on agile software development and its different paradigms gets deeper due to information provided by the participants.

3.6.4 Data Collection

This section sheds the light on demographic details of our participants, how they were recruited along with the interviews and observation details.

In GT, data collection is guided by a process called Theoretical Sampling, which is an ongoing process which guides the researcher on what data to collect next based on the emerging theory.

"Theoretical sampling is the process of data collection for generating theory whereby the analyst jointly collects, codes, and analyzes his data and decides what data to collect next and where to find them, in order to develop his theory as it emerges." [149].

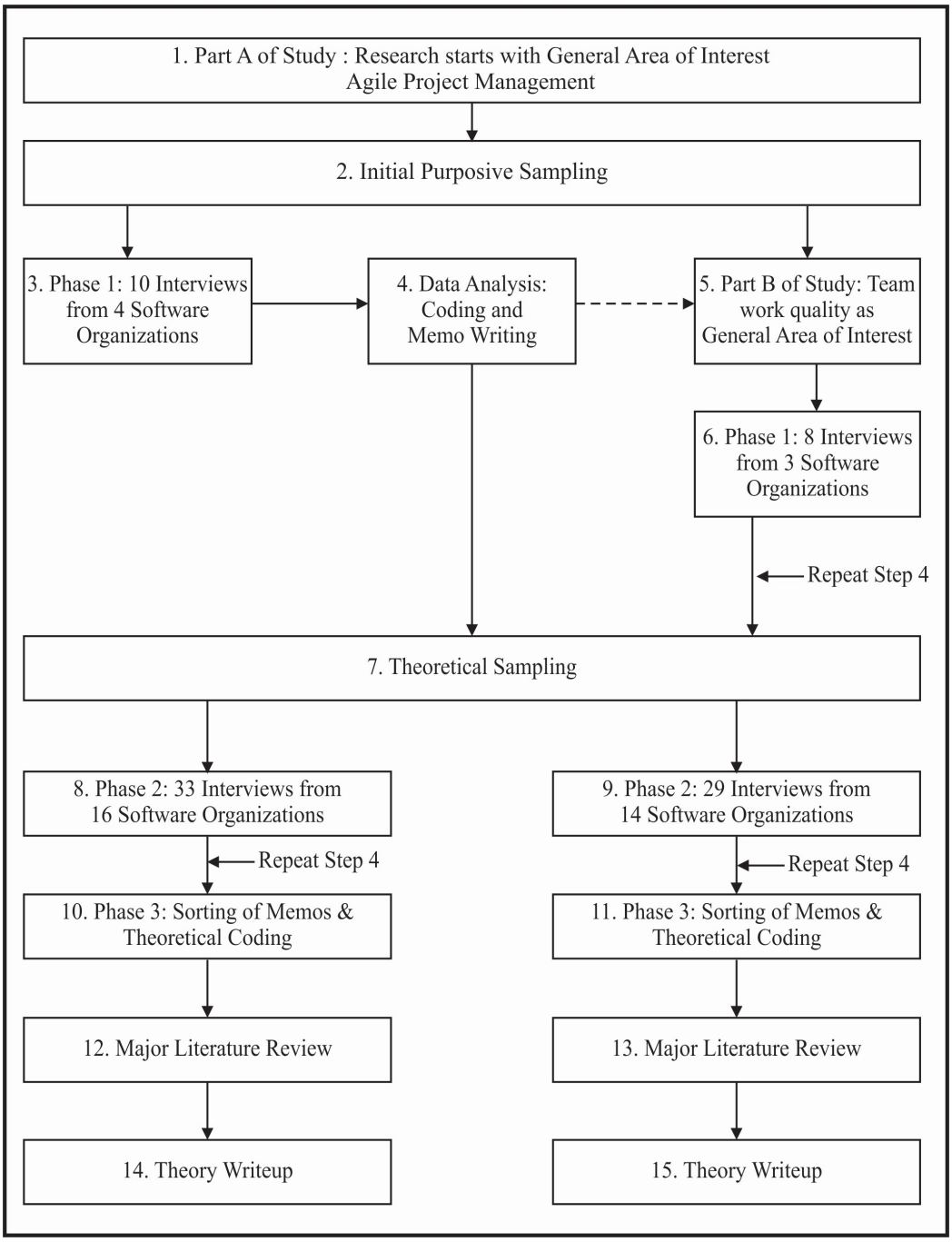


Figure 3.2 Summary of Our Research Design

Recruiting Participants

Finding participants in a GT study can be extremely challenging sometimes but fortunately being in a country which is a home to well establish IT organizations

proved beneficial for us. During our attempt to search for some group or community of agile practitioners in India lead us to the Agile Software Community of India (ASCI) which is the community of agile practitioners from different parts of India [150]. A correspondence has been made to the agile practitioners of this group. We also took part in Agile India 2016, International Conference [151] on Agile (Scrum, XP and Lean) held in Bengaluru and XP 2017, International Conference on Agile Software Development held in Cologne, Germany [152] that provided us the platform to collaborate with many agile practitioners across India and abroad. In addition to the above two source, LinkedIn was also used to contact agile practitioners. Many practitioners were agreed to be a part of our research and participated in this research study.

Our initial participants comprised of developers, testers and scrum masters but as our study progressed, need to include participants from other roles like product owners, agile coaches and business analysts was also felt. Moreover, our initial participants belonged to relatively newly formed agile teams that were undergoing agile transformation. This study started with the general area of interest "Agile Project Management" which was the focus area of our initial interviews. However, as we proceeded with the process of transcribing the interviews and their analysis, majority of participants expressed their concerns related to *Team Cohesion, Managing continuous changing requirement and teamwork quality.* This gave us direction to position our research to explore both the areas which are contributing to the 'product quality' as well as the 'teamwork quality'. Therefore, unanimously it was decided to split and pursue this research in two parts in order to cover both the aspects of quality i.e. 'product quality' and 'teamwork quality'. In the first part of study, we focused on the product quality aspect and continued our data collection using theoretical sampling process of GT. Theoretical sampling process helps us to evolve and modify our interview guide and also guides us in selecting more participants that were in good position to provide all the necessary information. Due to this, participants with different roles like agile coaches, DevOps engineers and people from senior management were also involved in this research. Figure 3.2 shows summary of our research design which describes our data collection phases and steps. First part of this study involved 43 agile practitioners from 20 different software organizations India, located in Bengaluru, Mumbai, Pune, Noida and Gurgaon. The project duration varied

from 6 to 36 months and team size varied from 7 to 20 people on different projects. The participants were chosen from the software organizations that were into wide range of domains like e-commerce platforms, product based companies, insurance, finance and telecom solution providers. The software organizations size varied from 50 to 200,000 employees. All the 43 agile practitioners were part of the geographically distributed teams. Projects were distributed across India, USA, Europe, South East Asia, Australia, New Zealand and UAE.

In second part of the study, "Teamwork Quality" was the general area of interest around which our initial interviews were based. 20 participants who were part of the distributed agile teams were reused from the first part of the study and 17 new participants were recruited from 8 different organizations. New participants were also a part of the distributed agile teams located in India, Germany, USA, Australia, and New Zealand. A total of fifteen interviews were conducted over Skype that includes five in first part of the study and ten in second part of the study. The software organizations were mainly from the healthcare, digital marketing, e-commerce and stock market domains. Participants were having different levels of experience in agile software development; some were very new to agile while some were having experience of over three years in agile software development. Demographics details of the participants are shown in Table 3.2. Agile teams were distributed in a sense that for some teams, development team was in Germany and QA team was in India and for some teams, development and QA teams were located in India and Australia and Product owner was in USA.

Interviews and Observations

Several semi-structured interview were conducted with the participants using open-ended questions over a period of thirty six months. Duration of each interview was approximately one hour. Although in grounded theory studies piloting interview protocols are not necessary as data collected from each interview guides the further interviews [109] but it is still advisable to have an initial interview protocol that helps in guiding future course of interviews [153]. We designed the interview protocol and sent it to three senior professors in the same area to check it for the correctness and relevance of our questionnaire to extract valid information. Their views were received on our questionnaire and their suggestions were incorporated in our interview

protocol. Our improved interview protocol (Annexure A) was used for the pilot study phase and then subsequently, we modified it for conducting further interviews as per theoretical sampling process of GT. All the interviews were audio recorded with the consent from our participants. Every interview was transcribed for further analysis post completion. Every interview was transcribed for further analysis post completion [127].

Data collection and its analysis were carried out in tandem. Print outs of interview transcripts were taken out and coding was performed along the margins by hand using pen on paper. Emerging concerns derived from the constant comparison of interview transcripts helped us in guiding our future interviews. Using this mechanism of theoretical sampling to identify our next interview samples, we continued data collection until we reached theoretical saturation i.e. when no new concepts were emerged. Our initial interview questions were revolved around participants real project experiences while working on agile development projects, different agile project management activities and challenges they faced while performing those activities. As our interviews were open ended and mostly conversation driven, some of the initial questions were as follows:

- How is your experience so far while working on agile development project? What is the difference in working on agile development project and on traditional waterfall model based projects?
- What is your role and responsibilities on the project?
- What are the different challenges that you have faced while working on this project and how you overcome those challenges?
- How you ensure quality product delivery at the end of every time boxed sprint?

Table 3.2 Demographic details of the participants

Organization (Code)	Interviewee(s) Profile	Project Location	Agile Methods	TeamSize	Sprint Duration (Weeks)	Study part
O1	Tester, Scrum Master	India-UK	Scrum	10-12, 16-18	2	A+B
O2	Senior Tester, Product Owner	India-USA	Scrum & XP	10	2- 3	A+B
O3	Scrum Master, Senior Tester	India-South East Asia	Scrum & XP	12-14, 12	3- 4	A
O4	Tester, Senior Quality Analyst	India-Europe	Scrum & XP	18	3-4	A
O5	Tester, Senior Developer	India-USA	Scrum & XP	14	1-2	A+B
O6	Senior Developer, Dev Ops	India-Australia	Scrum & XP	20	3-4	A
O7	Agile Coach, Chief Technical Officer	India-USA-Australia	Scrum	14-15	2-4	A+B
O8	Agile Coach x 2	India-New Zealand	Scrum & XP	7-8	2-3	A
O9	Developer, Agile Consultant	India-UK	Scrum & XP	12-13	3-4	A
O10	Test Analyst, Vice President	India-UK	Scrum & XP	9-10	2-3	A
O11	Scrum Master, Agile Coach	India-Western Europe	Scrum	18-19	3-4	A
O12	Agile Coach, Developer	India-USA	Scrum & XP	10-12	3-4	A+B
O13	Tester, Developer	India-Canada	Scrum & XP	9-10, 12	3-4	A
O14	Senior Tester, Test Analyst	India-Australia	Scrum & XP	10, 9-10	1-2, 2-3	A+B
O15	Senior Quality Analyst, Scrum Master	India-South East Asia-Australia	Scrum	8-9	3-4	A
O16	Tester, Vice President, Developer	India-Western Europe	Scrum	12-13	1-2	A
O17	Tester, Scrum Master, Product Owner	India-Western Europe	Scrum & XP	10-12	2-3	A
O18	Scrum Master, Vice President	Germany-USA-Australia	Scrum & XP	12-14	2-3	A+B
O19	Senior Agile Coach, Scrum Master, Product Owner	India-Europe	Scrum & XP	8-9	3	A+B
O20	Scrum Master, Test Manager	India-Europe	Scrum & XP	10-11	3-4	A
O21	Senior Tester, Developer	Germany- India	Scrum	15-18	2-3	B
O22	Tester, Agile Coach	India-USA	Scrum	10-11	3-4	B
O23	Scrum Master, Developer	Germany-India	Scrum & XP	12-14, 18	2-3 1-2	B
O24	Product Owner, Agile Consultant	Germany-India	Scrum	20	2-3	B
O25	Test Analyst, Scrum Master X2	Germany-Neteherlands	Scrum	8-10, 12-15	2-3	B
O26	Developer, Quality Analyst	Germany-Netherlands	Scrum	10	2	B
O27	Agile Coach x 2	Germany- Australia	Scrum & XP	12-14, 12	3- 4	B
O28	Tester, Senior Quality Analyst	India-Europe	Scrum & XP	18	3-4	B

- Any other important information related to agile development approach that your project follows.

Non Participant observations were also carried out in three different agile development projects from three different software organizations in addition to the interviews in order to validate the findings of the interview data. Primary author passively observed different project activities and gathered the data in the form of field notes. Important agile practices like sprint planning meetings, sprint review meetings, daily smoke and regression tests were observed carefully and field notes were taken along the way. Our observation protocol (Annexure B) was based on the guidelines provided by Creswell [154]. Data collection from two different sources i.e. (interviews + observations) sources helped us in achieving triangulation.

Figure 3.2 shows the large scrum board centrally mounted in open workspace of organization X. Organization X uses the large scrum board to monitor the task status of the sprints. Figure 3.3 shows the use of agile project management tool by organization Z in distributed environment to assign the task and track task status.

Figure 3.3 Scrum Board in organization X showing task status

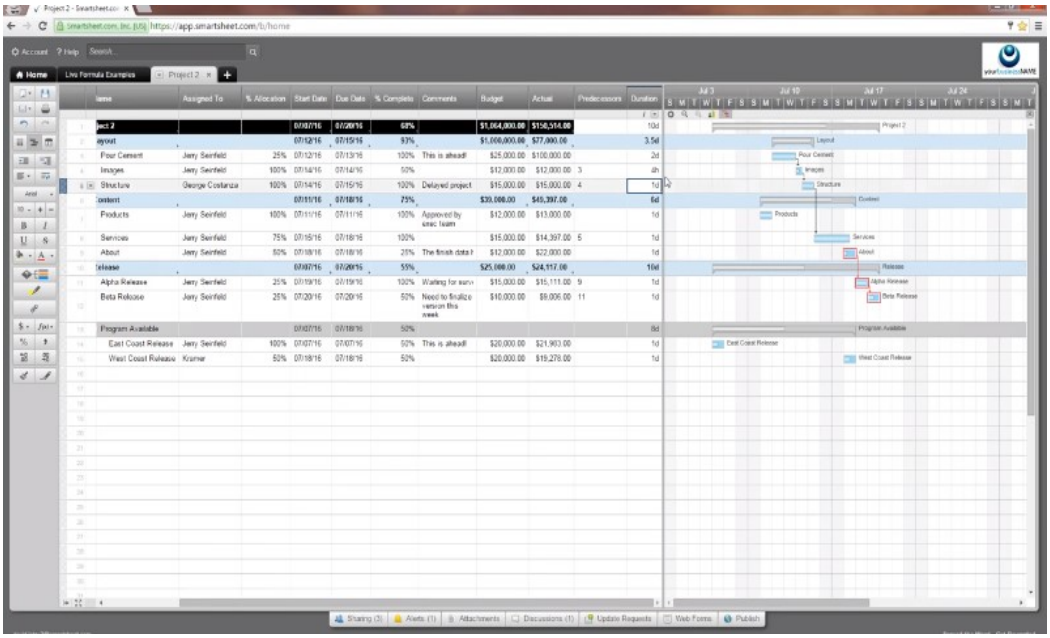


Figure 3.4 Use of agile project management tool by distributed team members of organization Z

Field Note: Observation openly collaborated workspace of organization X

"It was a cubicle less, openly collaborative workspace, entire agile team sits together with centrally places scum based featuring task status in 'To Do', 'In Progress' , 'To verify' and 'Done' format. Centrally located scrum board acts as a information radiator for the entire team. Agile teams have the better visibility of their tasks and they can manage the work quite easily using the scrum board. Scrum boards consist of user stories in their current status column. Also, the user stories that are dependent on each other are joined together with the help of lines."

Field Note: Observation on use of virtual project management tool in distributed set up by organization Z

"The agile team is distributed across Germany and India. Project used number of collaboration tools to perform the different agile activities like sprint meetings, daily stand up meetings and sprint retrospectives. Agile project management tools are used to assign and keep track of the status of tasks. Dashboard of the project activities and task status updates dynamically whenever it is updated by any of the team member at any of the location. Project management tool helps the team in sharing and communicating the information in a timely and effective manner. This not only brings

transparency but also adds accountability of the team members towards the assigned tasks for the particular sprint."

3.6.5 Data Analysis

In Grounded Theory, analysis of data involves conceptualization and categorization of data and generating a theory from the emerged concepts and categories [136]. Grounded Theory is based on the concept-indicator model [145]. An indicator symbolizes a word, phrase, sentence or group of words, phrases or sentences in the data that represent some concern. Concept-indicator model in GT is driven by the coding process applied on the data. Coding process gives rise to two types of codes: Substantive codes and theoretical codes. The substantive codes represent the conceptually developed categories that are emerged from in depth analysis of the data. Emerged categories conceptually describe the real life concern or phenomenon being investigated [145]. Theoretical codes conceptualize implicitly on how the substantive codes are related with each other as an interrelated, multivariate set of hypothesis [155]. In the following section, two coding mechanisms - open coding and selective coding that gives rise to substantive codes and theoretical coding that gives rise to the theoretical codes are described.

Open Coding

Data Analysis in GT begins with open coding, where interview transcripts are being analyzed in detail and key points are identified from each interview transcript [144]. In the next step, the key points are collated and particular code is assigned to each key point [144]. Code is a phrase used to summarize the key point in 2 to 3 words. Open coding process is explained with the help of example from working interview transcript.

Interview quotation:"*Single tool can't support all different interfaces. It's absolutely based on what you are testing, if it's a normal GUI application you may not need more than [one tool]. But things change when you automate a buy flow for an end to end test on multiple devices like tablets and mobile phones on different platforms [like iOS or android]. Here you need a different set of tools altogether and it's a big deal to choose the right one." – O5, Tester*

Key Points: *"Single tool can't support different interfaces", "Automating a buy flow on different platforms and devices require different set of tools"*

Code: Tool to support different interfaces, Tool to support different platforms, Tool to support different devices

The codes from each interview were compared constantly with the codes from the same as well from other interviews. This is known as GT's constant comparison method [144]. The constant comparison and grouping of similar codes lead to the second level of abstraction, called *concepts* in grounded theory. Different levels of data abstraction used in Grounded Theory are shown in Figure 3.4.

Concept: Choosing a tool to support different interfaces, devices and platforms

Similarly, other concepts that emerged from other interview transcripts are: Choosing a tool to automate different test types, Choosing a tool to automate continuous integration and deployment, Choosing a tool to automate regression testing, Choosing a tool to support scripting as shown in Figure 3.5.

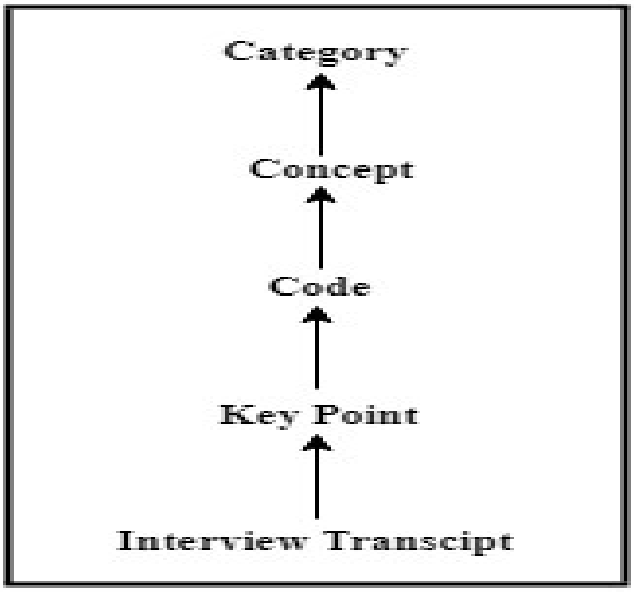


Figure 3.5 Different levels of data abstraction in GT

This constant comparison method is repeated on concepts to produce the third level of abstraction called categories. On comparing concepts emerged from the one interview to the concepts emerged from the other interviews, respective categories have been reached out.

Category: Choosing the right tool

Choosing the right tool is one of the *technical challenges* that agile practitioners faced while adopting test automation on agile development projects. Chapter 4 presents the detailed description on the other associated challenges under *technical challenges* category. Other categories emerged in the similar fashion and presented in the form of diagrams throughout this thesis.

Core Category

Open coding ends on identifying the core category Glaser and Strauss [136]. Core category is the one that best explains the central concern of the study participants [156]. Without a core category, there is no grounded theory. Criteria's for choosing the core category are: it must be central; it must be related to several other categories and their properties; it must re-occur frequently in the data; it must relate meaningfully and easily with other categories; and it must account for most variations in data [145]. It is not necessary for a core category to account for all of the behavior under study, but it accounts for the specific behavior that is most relevant to the participants involved in the study [156].

Selective Coding

When core category has been established, then researcher proceeds with the selective coding phase. In selective coding phase, researcher stops doing open coding and selectively codes for the core category by delimiting the data collection, analysis and theoretical sampling to only those categories that are related to the core [144]. Selective coding phase continues until the core category is sufficiently elaborated and integrated along with its theoretical connections to other relevant categories [144]. Moreover, when further data collection and its analysis yielding the same results and no more new concepts emerge then this stage is known as theoretical saturation stage [144]. When researcher start finding repetitive results from the data, they should stop collecting data and coding. In this research, when further data collection and its analysis were leading us to the same categories with no new data, we found out that our categories have reached saturation.

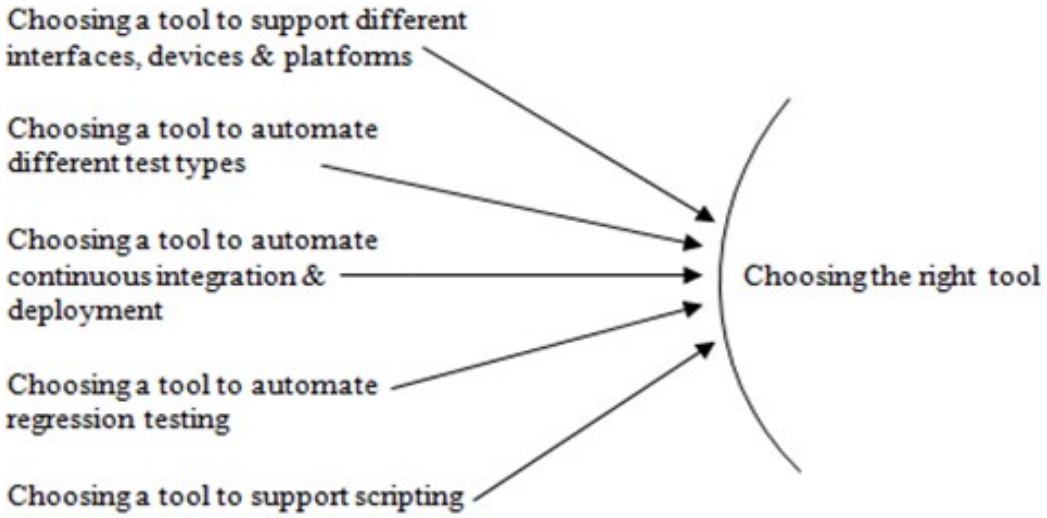


Figure 3.6 Emergence of category *choosing the right tool* from underlying concepts

3.6.6 Memoing

Memos are "theoretical notes about the data and the conceptual connections between categories written down as they strike the researcher" [144]. Memo writing guides researchers to put their ideas together about certain codes, concepts and categories which enable them to capture conceptual connections between the categories [143].

Memos were modified to cater to the new ideas emerged during further data collection and analysis phases. By following Glaser's guidelines, coding process was halted and memos were written as and when new ideas emerged. Figure 3.6 shows the memo on "Test Environment Management".

Throughout this research study, we wrote memos as and when they strike us during the data collection and analysis process. Memos can be seen as an audit trial of the researcher's conceptual and analytical approach towards the phenomenon under study. While writing memos, we took utmost care to put its focus on enlightening the conceptual connections between the emerged categories.

When you are deploying the software, you are not only deploying the software but you are deploying the environment too. The physical environment might be already provisioned or the virtual environment or environment at the cloud and you just need to make some configuration changes but if it's not provisioned you need to provision it first for the effective and valid testing. Quality depends on the sanity of the environment

Figure 3.7 Memo on "Test Environment Management"

3.6.7 Sorting

Sorting is a process to put the fractured data back together for formulating a theory [157]. Sorting of theoretical memos can be started as soon as theoretical saturation stage has been reached. According to the Glaser [157], researchers aim should be on conceptually sorting the ideas and not the data. Basically, sorting has to done on a conceptual level rather than attempting chronological ordering. Sorting performed at the conceptual level results in the outline of the theory that depicts the relationships between different categories and the core category.

To perform sorting, print out of the memos were taken out, then they were sorted by their topics and related topics were grouped together in an orderly manner. Sorting of the memos helped us in generating outline of the emergent theory which later formed the outline of this thesis.

3.6.8 Major Literature Review

Once our findings and substantive theory seemed sufficiently grounded and developed, the literature review was performed on the findings of our study. The objective behind delaying major literature review until analysis part is twofold: a) to protect the findings from any preconceived notion or ideas b) to map and relate our research findings to the literature by integration of ideas [145].

The benefit of performing literature review in later stages of GT is that it allows concepts and categories to emerge naturally from the data. Practically, doing an extensive literature review upfront in the grounded theory research can prevent

original concern to come out and eventually could affect the originality and quality of the research.

As per Glaserian GT guidelines minor literature review was performed and is included in chapter 2 for the better readability and ease for the reader. However, major literature review was performed at the end of the research, connecting the results into the existing literature. In result chapters 4 and 6 of this thesis, first our research findings are presented in detail followed by the discussion on our findings in the light of the existing literature.

3.6.9 Theoretical Coding

Theoretical coding is used to explore the relationships among different emerging categories and how they fit for integrating into a theory [145]. Theoretical coding is considered as the completion stage of theory development because emerging theory start to condense and become parsimonious [145].

Glaser has recommended around 50 different theoretical coding families to help the researchers in conceptualizing the relationships among different categories. Some of the coding families are: Degree Family (describes range, limit, etc.); Six C's family (causes, contexts, contingencies, consequences, covariance's and conditions); Type Family (form, styles, classes, kids); Strategy (strategies, tactics, mechanisms, managed, etc); Dimension family (dimensions, elements, divisions, aspect, etc.). However, assigning the theoretical code to the findings is not the mandatory requirement of the GT, but theoretical codes helps in explicating the emerging theory [155]. Theoretical coding is applied in the later stages of this research as per the Glaserian GT guidelines [125].

3.6.10 Theory Write Up

Final result of the Grounded Theory research study is a substantive theory that is grounded on the data. Writing up the theory is the final step in the GT studies, which constitutes a theoretical outline of the concepts emerged during sorting and theoretical coding. Our theory write up describes how product quality is improved by adopting test automation on agile projects and how teamwork quality is improved by building trust in agile development teams. Our write up is presented in Chapters 4 and 6.

3.6.11 Evaluating a Grounded Theory

According to Glaser, grounded theory must be evaluated on the basis of four criteria namely: fit, work, relevance, and modifiability [148].

Fit refers to "the ability of the categories and their properties to fit the realities under study in the eyes of the subjects, practitioners, and researchers in the area" [146]. It means that the concepts, categories and their properties, which define the theory, have emerged naturally from the data, and were not forced by preconceived ideas or hypothesis.

Work refers to "the ability of the theory to explain the major variations in behavior in the area with respect to the processing of the main concerns of the subjects" [146]. It means a grounded theory should be able to explain what happens in the data, predict what will happen, and interpret what is happening in the area studied.

Relevance is achieved when a grounded theory "allows core problems and processes to emerge" [145]. That is, participants and practitioners must consider that the theory is useful, meaningful and applicable to the people working in the substantive area.

Modifiability refers to "the quality of the theory to be ready for changes to include variations in emergent properties and categories caused by new data" [146]. That is, a theory can continue to be extended by other researchers when more data and new ideas are compared to the existing theory.

We evaluated our Grounded Theory against these four basic criteria in section 8.3.

CHAPTER 4

A TRANSITION JOURNEY OF ADOPTING TEST AUTOMATION ON AGILE DEVELOPMENT PROJECTS

This chapter presents our Grounded Theory of adopting test automation on agile development projects. This theory explains the challenges, strategies and benefits during transition journey of adopting test automation on agile development projects.

4.1 Introduction

Software testing is considered as the key quality assurance activity for ensuring reliable software product [158]. According to [159] software testing can incur fifty percent cost of the software development project. The role of testing in any form of software development is very important and particularly in agile it is even more decisive as development is iterative with two to four week sprints. Software testing runs in parallel with the development phases on agile projects [160]. When a computer program is written for executing the set of tests that would otherwise need to be executed manually, this process is called test automation. In test automation, we automate tests and use the tools to control the execution of tests [161]. Software development projects with stringent deadlines depends heavily on test automation as it helps in faster test execution and getting wider test coverage [162]. Test automation is of prime importance on agile development projects because testing and development goes hand in hand in agile [163]. It is apparent from above studies that test automation is one of the key components of agile software development projects.

This chapter presents our substantive theory of adopting test automation on agile development projects.

4.2 Research Method

Our research motivation was to increase the understanding of different contributing factors that facilitate agile project management practices in delivering quality product iteratively. Grounded Theory has been used in this study as our research method. In Section 3.6 we have already discussed about the grounded theory as our research

method and our data collection process in detail. Figure 4.1 shows the application of different grounded theory components in our research.

4.2.1 Data Analysis

4.2.1.1 Phase 1: Identifying the Core Category

In Grounded theory studies, data collection is guided by theoretical sampling process which recommends the researchers to "jointly collect, code, and analyze data and decide what data to collect next and where to find it in order to develop the emerging theory" [145].Emerging concepts and categories drive the theoretical sampling process in GT and provide the direction to find the future avenues for the data collection.

However, there has to be a starting point for the data collection process in GT. Researchers using GT uses purposive sampling in order to find the initial participants based on certain criteria's [149]. Afterwards, theoretical sampling process guides the further sampling sites and participants [149].

This research also started with finding the initial participants using purposive sampling based on the basic criteria: 1) Working experience of one or more years on agile development projects, mainly scrum and XP or both. 2) Total IT industry experience should be three or more years. 3) Working experience on at least one agile transformation project. Our data collection process embarked upon with ten interviews from four different software organizations on our general area of interest "Agile Project Management" [146].

Initial interview questions revolved around participant's experience after working on agile development project, their roles and responsibilities, different agile practices they follow in order to deliver quality software within time and different challenges associated with the agile development approach.

All the interviews were audio recorded and then transcribed for analysis. Two levels of substantive coding: Open coding and selective coding were performed on the data to generate codes, concepts and categories as per the Glaserian GT [164].

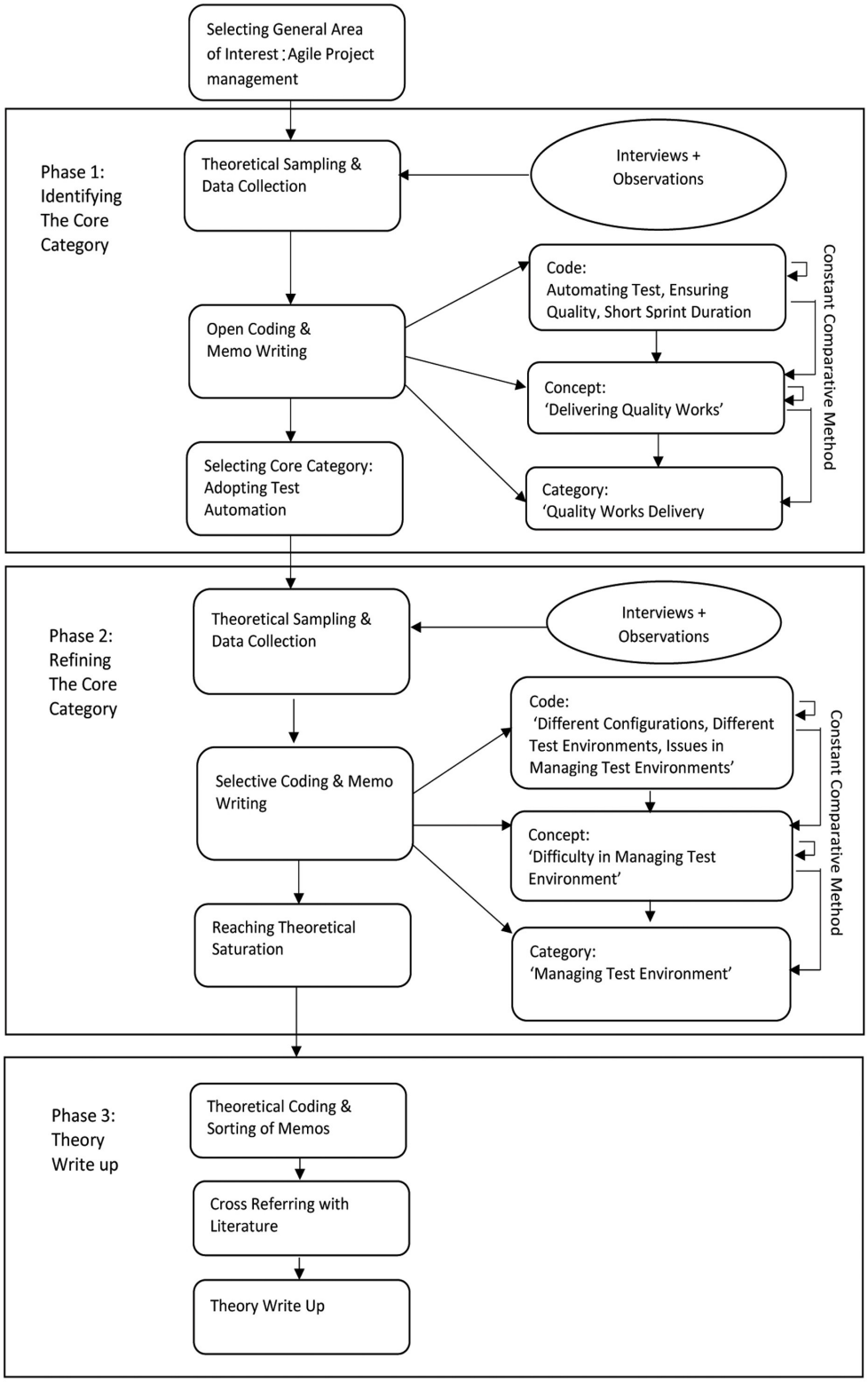


Figure 4.1Application of grounded theory in our research

4.2.1.2 Open Coding and Memo Writing

In open coding interview transcripts are broken down into discrete parts and then analyzed in detail for incidents that explain some phenomenon. Those incidents that represent the key points from each segment of the interview transcript are put together and particular code is assigned to them [145] [121]. Code is a "phrase used to summarize the key point in 2 to 3 words" [144]. The constant comparison of codes from one interview with the codes generated from other interviews and then grouping of similar codes lead to the second level of abstraction, called concepts. Furthermore, this method is repeated on concepts to produce the third level of abstraction, called categories. Table 4.1 shows the example of open coding process with the help of working interview transcripts and field notes.

Memos are written as and when they emerge during the coding process. Our coding process fetched multiple codes and memos which acted as inputs for our subsequent interviews, due to which our interview guide was updated constantly. The process of comparing incident to incident and code to code is repeated on the same as well as with the other interviews and also with the codes from the data gathered from the field notes of observations.

4.2.1.3 Selecting the Core Category

Open coding ceases once core category has been identified. Main attribute of the core category is that "it accounts for the large portion of variation in a pattern of behavior". It also signifies the main concern of the participants. Main criteria for choosing the core category includes: core category must be central; it must relate with other categories and their properties; it must re-occur frequently in the data.

Our Open coding process ended on identifying our core category *"Adopting Test Automation"*. Two potential near core categories also emerged during open coding: *"Quality Work Delivery"* and *"Manage Changing Requirements"*, but we selected *"Adopting test automation"* as our category as it is related to most other categories in a meaningful way and represented the most common concerns of our participants. Furthermore, our selection of core category tends to be fit as we were able to integrate the theory around it.

4.2.1.4 Phase 2: Refining the Core Category

We further followed the theoretical sampling process of GT and progressed into phase 2 of our study [148]. In this stage, our sample size expanded and we also interviewed senior agile coaches, people from senior management people like Chief Technical Officer (CTO) and Director from organizations with diverse domains like telecom, insurance, healthcare to gain well rounded perspective from participants working on different agile positions. Theoretical sampling guided us on expanding our sample participants and software organizations that gave us the confidence in the emerging theory. Phase 2 of our study involved 38 more interviews from 20 different software organizations. Through theoretical sampling and constant comparative method we ensured that agile practitioners with different roles and wide range of domains could be covered.

4.2.1.5 Selective Coding and Memo Writing

Once core category has been identified, we stopped open coding. Coding process is now only focused on those interviews that were related to the core category. Core category is one that relates to the other categories and their properties as much as possible. Purpose of selective coding is to integrate and refine the emerging categories into a theory [165]. Figure 4.2 explains the selective coding process with the help of some interview and observation data.

4.2.1.6 Reaching Theoretical Saturation

One of the important aspects in GT is to reach closure on theoretical sampling process. Ideally, it should be stopped on reaching theoretical saturation that is when further data collection yields no new data or concepts [121]. In our case as well, when we were unable to find any new data or concepts and all our categories became saturated, it was decided to stop further sampling and data collection.

Table 4.1 Example of open coding process

Open Coding	Statement / Field Note	Key Point	Code	Concept	Category
Interview Quotation – P5, Scrum Master	"Most important question...wheth er or not your project is truly time driven, whether or not you are delivering high quality product, time is speed for us and we can achievethat [speedand quality] by embracing automation."	Need for timely delivery of quality products, Achieving speed, Quality through automation	Timely delivery, Quality products, Embracing automation	Achieving quality and speed by embracing automation	Adopting Test Automation
Observation (Org.: Y, Project: Delta)	Acceptance testing was practiced manually till sprint 3, consuming lot of time and effort. UI changes were frequent due to constant new product launches, decision to automate acceptance tests, acceptance tests automation started.	Manual acceptance consumes time and effort, Frequent UI changes, Automating acceptance tests	Time and effort loss, Constant UI changes, Acceptance tests automation	Achieving speed by embracing automation	

4.2.1.7 Phase 3: Substantive Theory Write Up

Once the data collection is finished and all the categories have reached saturation then researcher conceptually sorts the theoretical memos. Sorting of the memos constitutes the theoretical outline around the emerging theory [145]. Ideas should be sorted not the data while sorting memos. We sorted theoretical memos conceptually based on the properties of the underlying categories, similarities and connections between the ideas. Glaser [145] suggests that theoretical coding should be performed at this stage.

Theoretical coding involves conceptualizing how the categories and their properties relate to each other as a hypothesis to be integrated into a theory. We followed

Glaser's guidelines and performed theoretical coding at the later stages of analysis. Glaser proposed several common structures of theories know as theoretical coding families [155]. On comparing our data with theoretical coding families, "Strategy" family was best suited for our data than rest of the coding families [155]. The Strategy family comprises of strategies, tactics, handlings, mechanisms and techniques. In our research study 'strategy' family fits perfectly to describe our theory of overcoming transition challenges faced by agile teams while practicing test automation on agile projects.

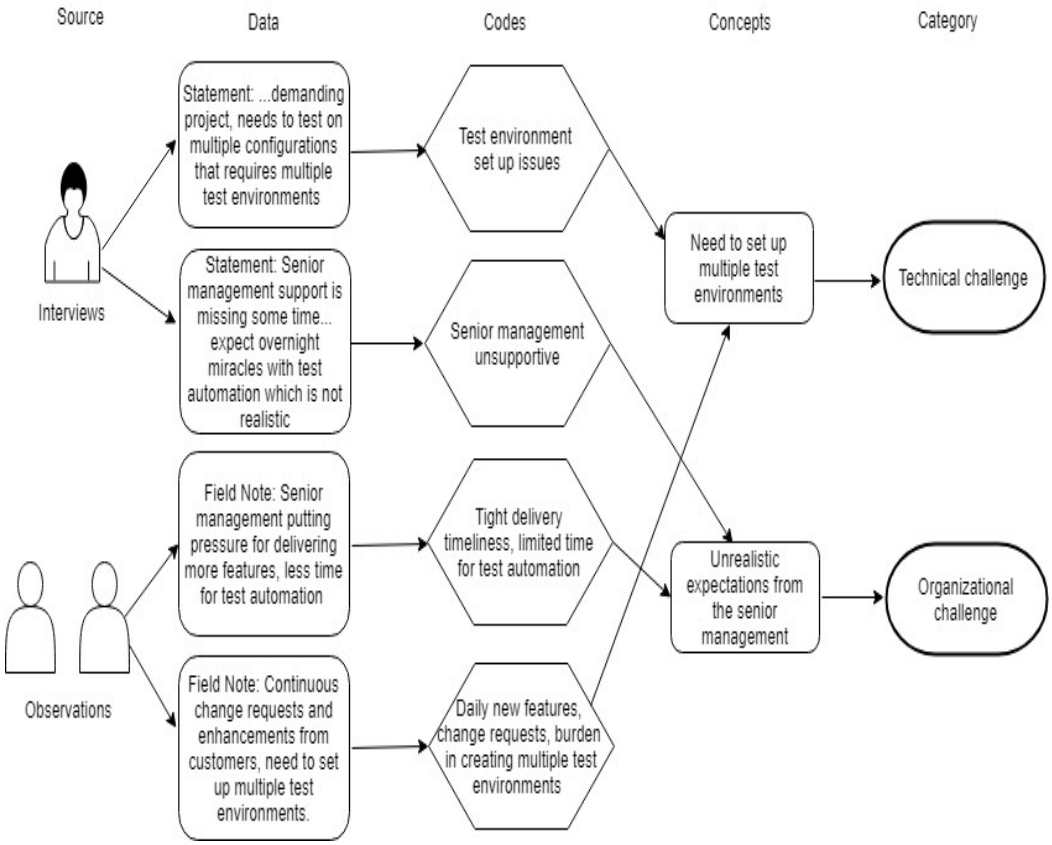


Figure 4.2 Example of selective coding process

Theory write up is the concluding stage in GT, it follows the theoretical outline produced from the theoretical coding and sorting of memos. Our theory presents the transition journey of adopting test automation on agile projects that explains various transition challenges faced by agile teams, overcoming strategies and finally the benefits of adopting test automation. In the following section, we present the research findings from our study.

4.3 Results

In this section, we present our grounded theory: "A Transition Journey of Adopting Test Automation" on Agile Projects. Selected interview quotations are provided under each category to better explain it in the present context. We discovered that test automation is a core contributor to the success of any agile project. However, *Adopting Test Automation* is a transition journey on agile development projects comprising of three stages: *Pre-Transition*, *In-Transition* and *Post-Transition* stage. *Pre-Transition* stage reflects the different types of impediments faced by the participants while practicing test automation. The strategies adopted to overcome these challenges are the focus of attention in the *In-Transition* stage. Finally, in the *Post- Transition* stage agile teams reap the benefits of test automation on successfully practicing the test automation in their projects.

4.3.1 Pre-Transition and In-Transition Stage

We now discuss the challenges and overcoming strategies during *Pre-Transition* and *In-Transition* stages while practicing test automation on agile projects. Test automation is core practice in agile software development projects. When agile teams try to inculcate test automation in their projects they face certain challenges. Based on the interviewee's responses and coding of the data, three broad categories of challenges emerged - Technical, Organizational, and Human. The key concepts from each respective category of challenge are displayed with the help of bar graphs in the following section. Only those concepts are included that were mentioned by at least ten or more interviewees.

4.3.1.1 Pre-Transition Stage: Technical Challenges

Figure 4.3 presents technical impediments pertaining to infrastructure, tools, test environment management and script maintenance. Selecting the right test automation tool is one of the key concerns for most of the agile practitioners. Whether it's about choosing the tool for automating acceptance tests or choosing a tool for automating continuous integration & continuous deployment process it was one of the impediments for agile teams.

"Now here comes the real problem, we are attaching sprint 1 with sprint 2, many things need to be taken care of like code coverage should be of this percentage, code

versioning should be done, matching of variables from sprint 1 and sprint 2. ...need tools to automate our continuous integration & deployment process." – O9, Developer

15 out of 43 agile practitioners have found that record and playback tools do not fit for practicing test automation in agile environment because in agile projects requirements changes are frequent and test script needs to be updated regularly hence only record and playback tools are not a good choices for agile development projects.

"It is difficult to decide which tool should be used to automate; record-playback tools simply don't work in an agile environment as we have to wait until the whole GUI is complete." – O2, Senior Tester

It is essential to know the project requirements, which tests needs to be automated and what tools are needed. Agile practitioners admitted that while transitioning to scrum and XP, they were still using traditional record and playback tool which were fetching unsatisfactory results.

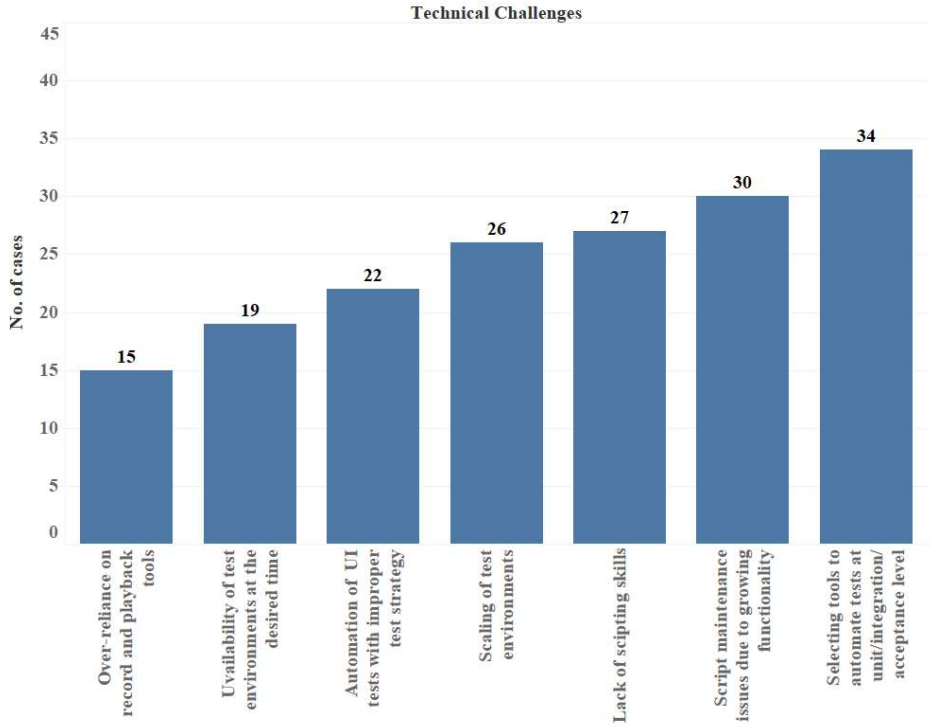


Figure 4.3 Technical challenges faced by agile teams during pre-transition stage

Figure 4.3 shows that agile practitioners consider test script maintenance as the other critical impediment while practicing test automation. In E-Commerce based projects test script is based on the page elements of the application under test (AUT). However, page elements undergo changes multiple times in one sprint and script also needs to be modified at all the locations where that page element is being used and this whole exercise leads to script maintenance issues.

"We have 200 odd test cases that interact with a submit button on user details registrations page and the development changed the element locator for this button , then either we would be maintaining 100 test cases or 100 lines of automation scripts, maintainability of [scripts] is to deal with proper care otherwise test automation will not give you that ROI." – O13, Tester

For every new addition or modification in feature, test script needs to be modified and maintained for the entire duration of the projects with multiple sprints and this was a challenge for them.

"As functionality grows with every sprint, automated regression testing must be performed to ensure that existing functionality has not been impacted by the introduction of new functionality in every subsequent sprint. The scale of regression testing grows with each sprint and so does the test scripts, so how you would add more test cases to the existing regression test suite? How would you solve test script maintenance issue?" – O20, Test Manager

It is also evident from the Figure 4.3 that scaling of test environments and managing them is also the crucial technical impediment faced by many agile practitioners. More than half of the participants expressed their concern related to the unavailability of the right test environment at the desired time, whenever they wanted to run automated test in pre-production or production environment they need to struggle with the availability issues of the test environment with the right configuration.

"I am working on agile project, which is too demanding, our sprint is 3-4 day long, teams working by 24x7 rotational shifts, they [business team] decide on tomorrows offer so that we have better customer inflow than our competitors. Automation can rescue in this situation, but … struggle with the availability of right test environment

for particular workflow and configuration. Sometimes we need to set up more than 100 different test environments for a single application." – O14, Test Analyst

"Keeping test environment as close as possible to production environment ensures the quality of the test automation and if an agile team does that [Keep test environment close to production environment] then they don't have to prove their tests valid."– O15, Senior Quality Analyst

But this was a tough task while adopting test automation in their projects for most of the participants.

"Why it is worth to have Test Automation in agile projects because it helps you in achieving your quality objectives, test environment should be a replica of your live [production] environment, you should have live data loaded into your test environment to ensure effective testing and if you practice this then the code that go into upper [production] environment would meet quality criteria." – O7, Agile Coach

Before performing smoke and regression testing in any sprint, it is a prerequisite to clear the test environment, reinstalling OS, refreshing databases to ensure reliable results. It is a crucial part of test automation and cannot be ignored. Managing the test environment to ensure effective automation testing is a tedious task.

4.3.1.2 Pre-Transition Stage: Organizational Challenges

Figure 4.4 shows different organizational impediments that agile teams suffer during their transition journey towards adopting test automation. Majority of the participants identified the traditional mindset of the senior management toward test automation as the main bottleneck while practicing test automation. Senior management support is required to create the required infrastructure for the test automation. But setting unrealistic expectations and expecting immediate results from the team shows their unwelcoming attitude towards the test automation.

"Test automation is not a magic stick; you cannot expect over nightly results. It takes time in streamlining the test automation process on agile projects. Senior management must not set unrealistic expectation right from the onset of the project." – O17, Tester

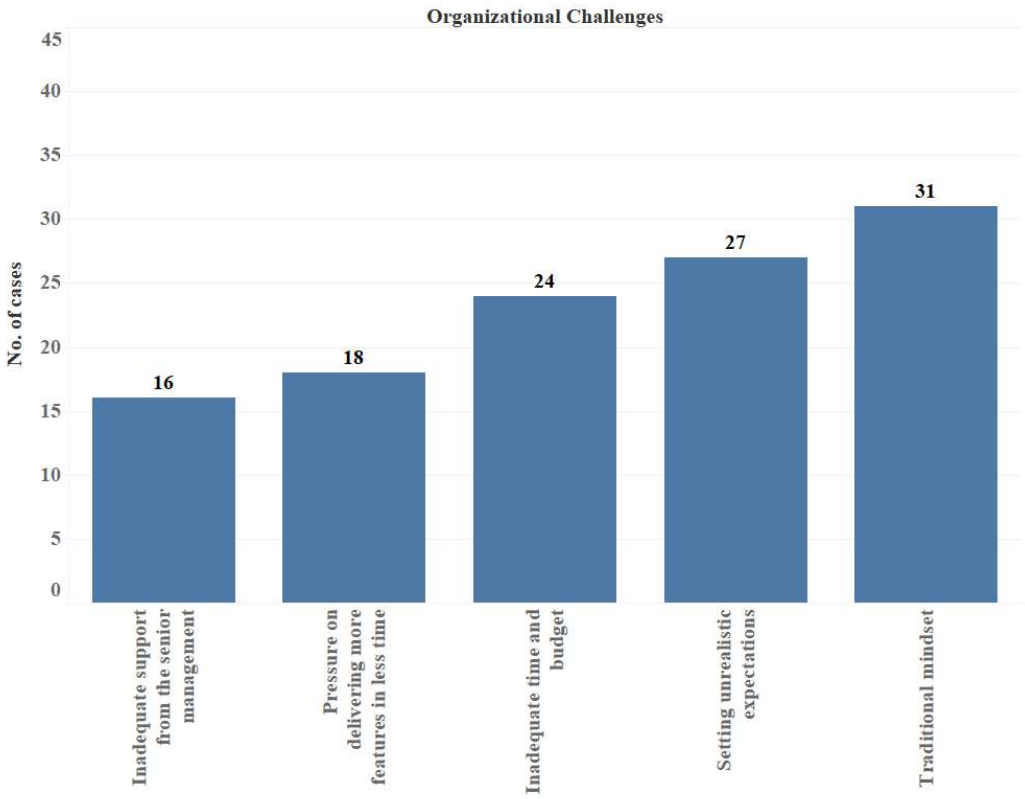


Figure 4.4 Organizational challenges faced by agile teams during pre-transition stage

Senior management was not providing enough budget and time that is required to build and sustain the test automation infrastructure on agile projects. Moreover, 18 participants acknowledged that management pressure to deliver more features in less time forced them to manually execute the test cases.

"... Have pressure to deliver more and more set of features in each sprint that hardly left us with any time that we can devote to automate our tests." – O21, Senior Tester

Another significant organization bottleneck was their weak process and support for implementing test automation on agile projects right from the early stages. Many agile practitioners mentioned that weak organizational support and lesser initiatives to promote test automation on projects were giving the agile teams the false impression to consider the test automation as a choice on agile projects.

"Test automation on agile is not an optional thing, it is by default the key ingredient of successful agile projects which aims to deliver quality software." – O18, Scrum Master

The case of test automation in agile projects is different from traditional SDLC phases as test automation have to be introduced right from the beginning whether its TDD (Test Driven Development) or ATDD (Acceptance Test Driven Development) to identify any problem right at the onset of the project. But most of the participants experienced this problem of overcoming traditional mindset.

"When we started doing agile in our project, most of team members were preoccupied by the traditional SDLC mindset, it took a while to move on from that test last mindset to test first and bringing the test automation to sprints as early as possible to accomplish TDD goals." – O20, Scrum Master

4.3.1.3 Pre-Transition Stage: Human Challenges

Figure 4.5 shows that majority of agile practitioners identified traditional mindset and delaying testing till the last phases of the development as the main human challenge while practicing test automation. As most of the projects are undergoing transition phase from the traditional waterfall model to agile, they really find it difficult to make it a habit to perform testing in parallel with the development.

"... shift left model is the only solution to the traditional code first and test later models. To adopt test automation you need to follow test first and code later methodology, this can only be possible if you get rid of that traditional mindset." – O23, Scrum Master.

Furthermore, 23 out of 43 participants have mentioned lack of communication between testers and developers as one of the impediment while practicing test automation on agile projects as it often leads to inaccurate automation planning and effort estimates. Developers and testers should engage in a dialog without waiting for any formal communication meetings and communicate freely otherwise whole project would suffer.

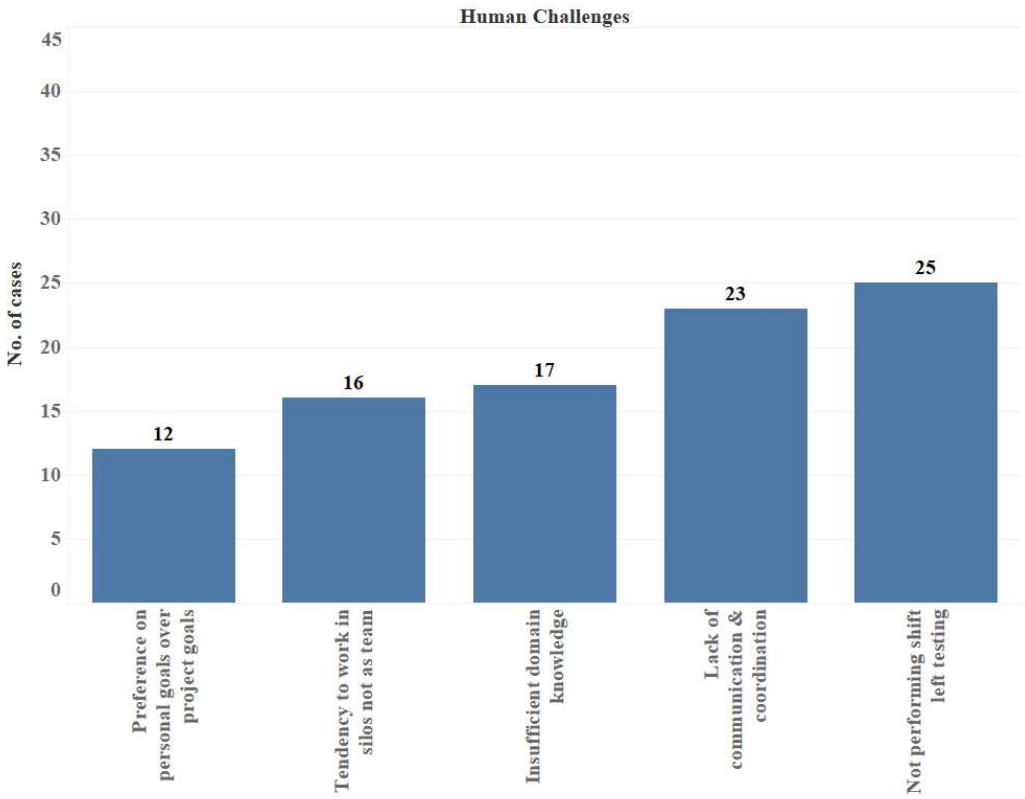


Figure 4.5 Human challenges faced by agile teams during pre-transition stage

Test automation also requires reasonable programming knowledge in order to automate tests and perform scripting. Lack of programming skills among agile team members was also one of the impediments in adopting test automation. In addition to that, insufficient domain knowledge about the application under test was also seen as a key challenge when agile teams try to automate tests either at unit level or acceptance level.

"You must understand the domain of the application which you want to automate, if its telecom or ecommerce or healthcare application, their automation flow would differ from each other." – O11, Agile Coach

Other noticeable challenges were related to the team member's personality traits that include habit of working in silos not as part of one single team and working for personal goals rather than project goals which eventually affects the implementation of test automation on agile projects.

4.3.1.4 In-Transition Stage: Overcoming Strategies

During In-Transition stage, agile teams try to overcome challenges they encountered in Pre-Transition stage by adopting some strategies. Figure 4.6 shows that cost is the biggest driver while implementing test automation on agile projects, more than half of the agile practitioners admitted that by doing proper Return on Investment (ROI) evaluation and performing Cost to Benefit Analysis (CBA) before procuring any licensed tool or any other infrastructure related issue, helped them in gaining the confidence of the senior management towards test automation.

"It's important to prove the worth of time , money and effort spent on building the automation infrastructure on project, performing ROI evaluation and showing it to the senior management helped us in gaining their confidence." – O9, Agile Consultant

Creating test automation awareness in agile team members is a important aspect that organizations should consider while getting on board with the agile development project. This can be achieved by educating team members on importance of effective test automation practices among agile team members by arranging test automation awareness boot camps and workshops help agile teams in knowing the actual use and utility of test automation in projects as mentioned by 17 participants. To overcome the communication related challenges, agile teams were encouraged to practice proactive communication with each other with all forms of possible communication media available to them.

"Proactive communication is the key to success, it bridges the [communication gap] between the testers and developers and PO's." – O8, Agile Coach

Involving testers in daily stand up meetings and developers in automating acceptance test inject the one team approach in them which further helps them to do work and think as one single team.

18 out of 43 agile practitioners reported that use of automation testing framework proved to be very useful in solving test script maintenance related issues. Data driven framework, keyword driven framework, hybrid framework and behavioral driven framework were the most commonly used frameworks used by the agile teams.

"Data driven and hybrid frameworks are the two automation frameworks that we follow in our project to solve test script maintenance related problems." – O10, Test Analyst

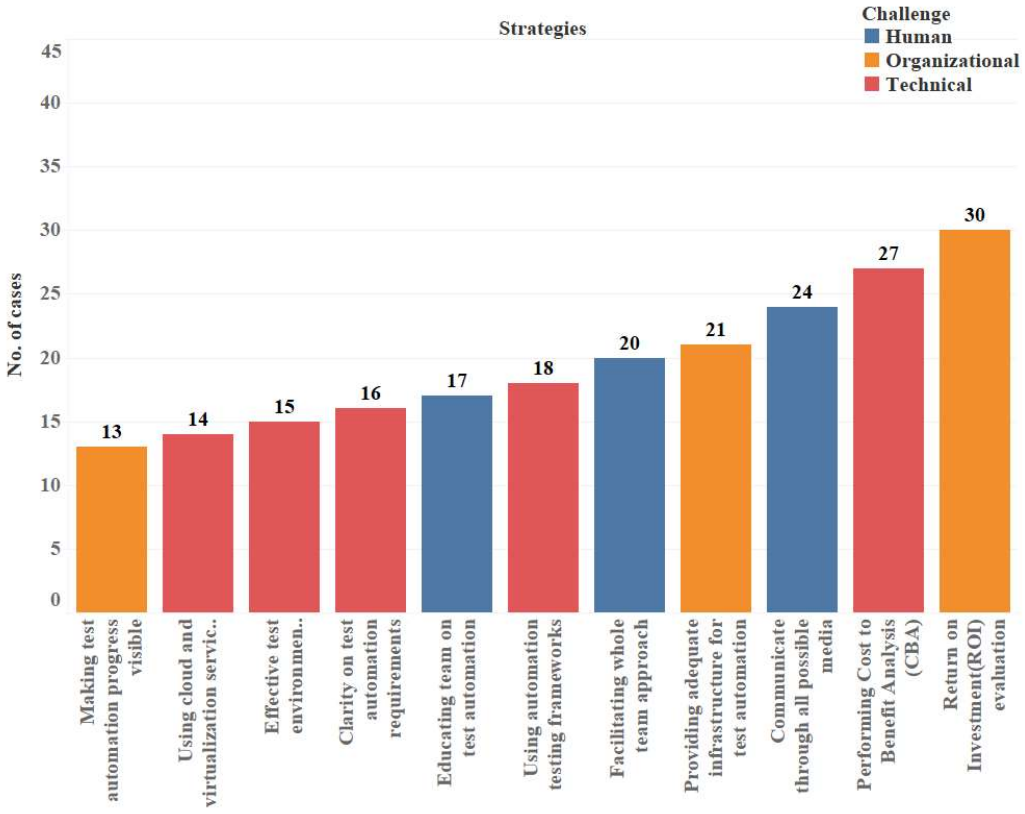


Figure 4.6 Overcoming strategies used by agile teams during in-transition stage

14 out of 43 participants mentioned that by using cloud and virtualization services they were able to solve test environment management related problems.

"Using cloud and bring out the virtualization aspect of it , helped us in setting up multiple test environments according to our need andcan be scaled up/ down …" – O6, Devops

Other important strategy used by agile practitioners was to create an upfront plan for managing test environment. Participants used spreadsheets to store all the information related to all the configurations, profiles, and test data required for any particular test environment setting.

4.3.2 Post-Transition Stage: Benefits

Post-Transition is the stage where agile teams have addressed most of the challenges of *Pre-Transition* stage by opting overcoming strategies in the *In-Transition stage*. Once they reach *Post-Transition stage* they have learned from their experiences and opts some strategies to overcome test automation adoption challenges in their projects. *Post-Transition* stage yields several *Benefits* which strengthen test automation practices of agile teams. *Benefits* achieved during the *Post-Transition* stage provided the confidence and momentum that turned out to be a driving force in pursuing test automation further for many interviewed agile practitioners.

Practicing test automation on agile projects yields number of benefits to the agile teams as shown in Figure 4.7. As there are multiple change requests, multiple bug fixes and constant addition of new features on sprint by sprint basis, verifying them with automated smoke and regression tests before moving to the next sprint becomes the quality gateway for most of the agile projects.

We don't have to worry about the last bug fix or feature change as we check the existing functionality at the onset of next sprint. Automating our smoke and regression tests ensures the quality for us."– O15, Senior Quality Analyst

Agile software development teams don't have to shrink the scope of the testing if proper test automation infrastructure is in place. Automated regression tests, automated acceptance tests prove to be strong determinant of the quality on agile projects.Agile teams often encounter inadequate test coverage as code changes are frequent and could not be anticipated beforehand so they must use test automation to improve the test coverage.

"You should be doing, automated regression tests for every change to verify that nothing is broken in it [Build]. This is the only way to proceed and improve your test coverage." –O10, Vice-President

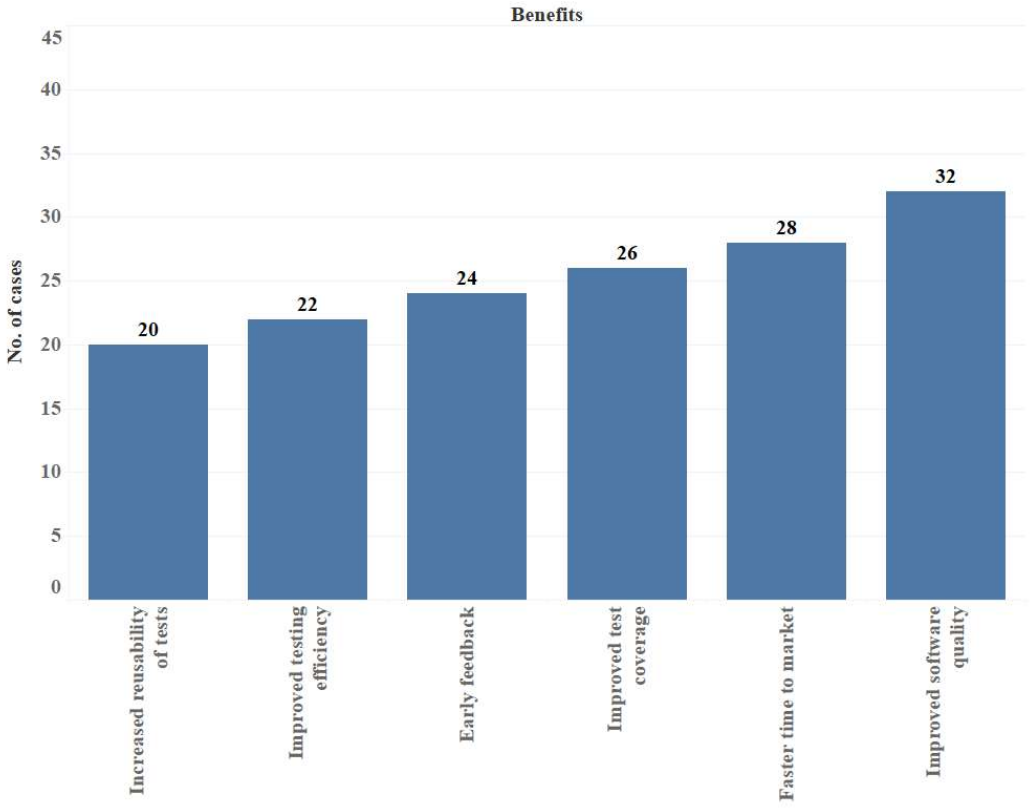


Figure 4.7 Benefits experienced by agile teams during post-transition stage

Repetition of same tests manually is not only laborious but also not recommended in agile set up. Automating regression tests and keep on updating the regression suite is the wise way for increasing test coverage. Moreover, the tests which require multiple sets of data or inputs also need to be automated.

One of the primary goals of any agile development project is timely release of quality software. Test automation makes important contribution in shortening the overall development cycle by performing different types of automated tests in every sprint, thus helps in developing quality software in less time. 28 out of 43 participants found faster time to market as one of the key benefit of adopting test automation as shown in Figure 4.7.

"Umm, Test automation not only reduces the test cycle time but it also reduces the overall development cycle time too by automated execution of different types of tests and providing us [Developers] the fast feedbacks." – O12, Developer

Test automation is a key tenant of agile projects. It is inevitably important to have automated tests to verify every code change or a new feature addition so that immediate feedback can be provided to the developed regarding the testing results.

"We just cannot miss this [Test automation] in our project. Automated tests [smoke, regression] provide us the continuous and early feedbacks that help in reducing risk and repeating manual effort." – O21, Senior Tester

Improving testing efficiency is another important aspect of test automation; many agile practitioners admitted that they were able to perform automated testing and scale up their regression test cycles on sprint by sprint, which helped them in improving their overall testing efficiency.

"*Testing efficiency is an important measure to ascertain our quality goals for the release and after practicing test automation...improved our testing efficiency considerably.*" – O19, Product Owner

Test automation yield another benefit that is related to reusability of tests, by using good automation testing frameworks, reusable automated test scripts can be created.

"By using Automation frameworks, we have made our tests usable to the greater extent, now we don't need to automate the scripts from the scratch. Only correct framework has to be used to have the optimum utilization of automation." – O16, Developer

By making reusable test suites, different applications from different projects could be tested which saved time as agile teams used the same sets of tests over multiple sprints.

4.4 Discussion

Following the grounded theory research procedures, first data collection and its analysis were performed. Then, once our findings were developed and sufficiently grounded on data, literature on test automaton in agile context was reviewed. This section discusses our results in the light of related literature.

Agile software development professes practices unforeseen in traditional software development. The adoption of agile practices like daily stand up meetings, continuous

integration and greater customer and stakeholder interaction puts forth challenges that need to be addressed. Adopting test automation is a transition journey for agile development organizations where agile teams experience some challenges while practicing test automation, this transition process is very closely mapped to the Bridges transition model [166] as shown in Figure 4.8.

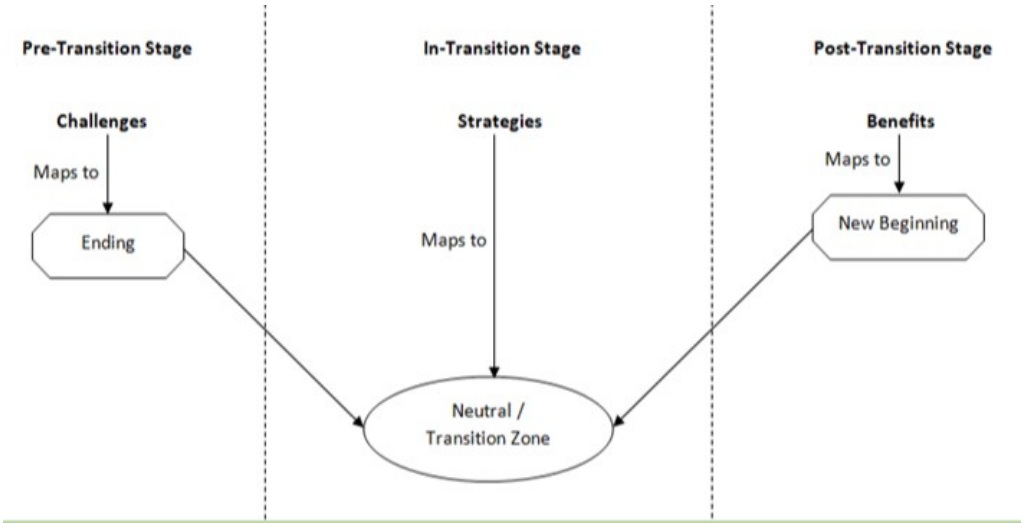


Figure 4.8 Mapping of transition stages with bridges transition model

Bridges transition model depicts that change is a transition process that includes three phases: 1) Ending phase – this phase corresponds to that state where people need to let go the past practices before they embrace new things. 2) Neutral zone or Transition phase – where people experience challenges and come out of their comfort zone for embracing a new change. 3) The Beginning phase – where people start to embrace the new change and develop the understanding regarding the new change. The problems associated with change are mostly not because of the change itself but due to the transition involved with the change, and when agile teams undergo this transition of letting go their traditional test practices and mindset, they face some problems. In this study, *Adopting Test Automation* on agile projects is also a change that different software organizations look upon for producing good quality software in less time. In *Pre-Transition* stage, agile teams face certain *Challenges* related to mindset towards test automation, selecting the tools, managing the test environments and communication related issues which needs to be addressed and this stage is aligned with the Ending stage of the Bridges transition model.

In *In-Transition stage*, agile teams adopt some overcoming *Strategies* to address the challenges of the *Pre-Transition stage*, this stage is mapped to the Neutral or Transition phase of the Bridges model. When agile teams streamline their test automation practices by using different *Strategies* in the *Post-Transition* stage they start getting the *Benefits*, this stage is mapped to the New Beginning phase of the Bridges model where people embrace new change. Effectiveness of *Adopting Test Automation* on agile development projects mainly depends on the fact that how well agile teams understand the challenges while practicing test automation and how they overcome those challenges. One of the most cited technical challenge was related to selecting right tool for the test automation. Agile teams need to overcome challenges related to test automation tools, they should inspect and adapt according to their project requirements and needs [167]. Yoder et al. [168] discussed the importance of deciding the right time to automate tests and selecting right automation tools.

The other set of technical challenges were related to test environment management and test script maintenance. In one of the study by Karhu et al. [169], managing test environment, maintaining test scripts and implementation time were the main contributing factors in affecting test automation infrastructure. Test automation could provide huge benefits if test script maintenance cost can be controlled [170].

For implementing successful test automation infrastructure on agile development projects organizational challenges must also be addressed and taken into account, management support plays a very crucial role in effective implementation and adoption of test automation practices in agile development projects. Diirr and Santos [171] identified management support as one of the critical success factors in improving overall IT service processes including test automation. Management should provide enough time and facilities to build up the test automation infrastructure [172]. Overcoming human challenges is also very important in order to adopt test automation on agile projects. Many people with traditional mindset prefer to perform testing in the later stages of development. This mindset is the biggest bottleneck in adopting test automation, shift left testing approach should be adopted on agile projects [173]. Other important human aspect is effective communication between the developers and the testers. Smooth and free flow of information through efficient communication helps in making both development and testing better [173]. In a study

[168], whole team approach was suggested to improve the mindset of the team while practicing test automation.

Test Driven Development (TDD) facilitates automated unit testing that helps in improving code quality and overall quality of the software [174]. According to Rafi et al. [175], test automation yields various benefits including better test coverage and improves product quality. Due to repeatability nature of the automated tests, cost and time can be saved that result in faster time to market of the product. Aniche and Gerosa [174] reported the importance of TDD in terms of continuous execution of automated unit tests that provide rapid feedbacks to the developers on their code, thus acts as a safety net for them.

CHAPTER 5

DEVELOPMENT OF REUSABLE HYBRID TEST AUTOMATION FRAMEWORK FOR WEB BASED SCRUM PROJECTS

5.1 Introduction

Web engineering is one of the primary areas of computing that have embraced agile software development. Most of the web based applications are developed using scrum framework of agile development method [176]. Wide spread usage of web applications invariably makes testing of any web application necessary before it goes online. Agile methods advocate automated testing, which assures faster and thorough testing of any web application [177].

In scrum, project progresses in a series of time boxed units known as sprints. Ideal duration of sprint is between two to four weeks [178]. To deliver a quality product at the end of every sprint it's important to overcome certain challenges like short development cycles, automating daily activities like smoke testing, regression testing, continuous integration and deployment process. Cost of fixing a defect can be too high if issues of one sprint carry on undetected to the next sprint. With manual testing there is a strong likelihood to miss out important tests. On the other hand, automated testing assures thorough testing in spite of tight deliverable schedules. According to [179] code constantly changes in an agile project which causes automated test cases to become obsolete. Most of the agile projects overrun on the test maintenance cost as compared to the possible Return on Investment (ROI) achieved due to test automation efforts [179]. Tyagi et al. [180] reported test script maintenance as one of the key challenges faced by agile practitioners while practicing test automation on their projects. In this chapter, we take up following research objectives with respect to test automation:

Objective 1: How to develop a test automation framework that can provide high reusability and maintainability of test scripts?

Objective 2: How to use this framework in actual agile software development methods like scrum?

In order to investigate above research objectives, it is important to establish the understanding of different test automation frameworks and automation tools that are prevalent in software industry.

Test Automation Frameworks

An automation framework is defined as an overall architecture that contains test data, test cases, test scripts, test automation tool, test reporting and proper strategy to execute a testing problem. Automation frameworks are broadly of three types, namely: 1. Data Driven framework- It is primarily used for executing set of test steps with multiple sets of data. In this framework sets of test input are read from data files like CSV files, Excel files, ADO objects etc. and are loaded into variables in captured or manually coded script. 2. Keyword Driven framework -In this framework each test step of every test case is associated with a keyword. Keywords are stored in excel file and driver script reads keywords from excel file to perform user actions. 3. Hybrid Framework- It is a combination of both data driven and keyword automation framework. This framework stores keyword for each test case step and at the same time test data can also be read from excel files to run multiple sets of data [181].

Many test automation tools are available in the market today like QTP (Quick test functional) also known as Unified Functional Tester (UFT), Test complete and Selenium. We have used selenium tool to implement Reusable Hybrid Test Automation Framework (RHTAF) mainly due to following reasons:

1) Selenium is a defacto standard in software industry for automating web based applications [182].
2) Selenium is an open source tool and does not levy any license fee.
3) Selenium gives leverage to automate web applications with multiple programming languages support like Java, C# and Python.

Selenium comprises of four components - Selenium IDE, Selenium RC, Selenium Grid and Selenium WebDriver. Recently, selenium RC and WebDriver have been merged to form Selenium 2.0 which offers many advantages like direct communication with the browser, Simple API's without any redundancy and easy

architecture [183]. Selenium 2.0 supports multiple web browsers like chrome, Firefox, Internet explorer and Safari. It also supports AJAX applications. The main goal of selenium 2.0 is to improve support for modern web application testing problems [183].

Following section explains our proposed framework and its three layered architecture. Thereafter, Implementation, experimental analysis and insprint automation strategy of RHTAF is presented. Section 5.4 presents evaluation and experimental validation of our RHTAF. Section 5.5 presents our results followed by discussion.

5.2 Proposed framework

In this chapter, we propose a Reusable Hybrid Test Automation Framework (RHTAF). Our proposed framework used Page Object Model (POM). Use of page object model in designing this hybrid test automation framework makes it unique. POM is a design pattern which models the web pages as objects [182]. The need of continuous delivery of features and change requests with every subsequent sprint in an agile development project makes leads to higher maintenance costs and development time. Without a proper test automation framework it is very difficult to cope up with such challenges. Our test automaton framework which is based on POM inherently makes script maintenance easier and also makes test execution simpler and faster.

Eclipse Integrated Development Environment (IDE) is used to develop RHTAF using Selenium 2.0. Eclipse is preferred over visual studio due to many reasons like eclipse is open source IDE, test script creation and test execution is faster and easier in eclipse IDE.

5.2.1 Three Layered Architecture of the proposed Reusable Hybrid Automation Framework

We have implemented the reusable hybrid test automation framework using three layered architecture as shown in Figure 5.1.

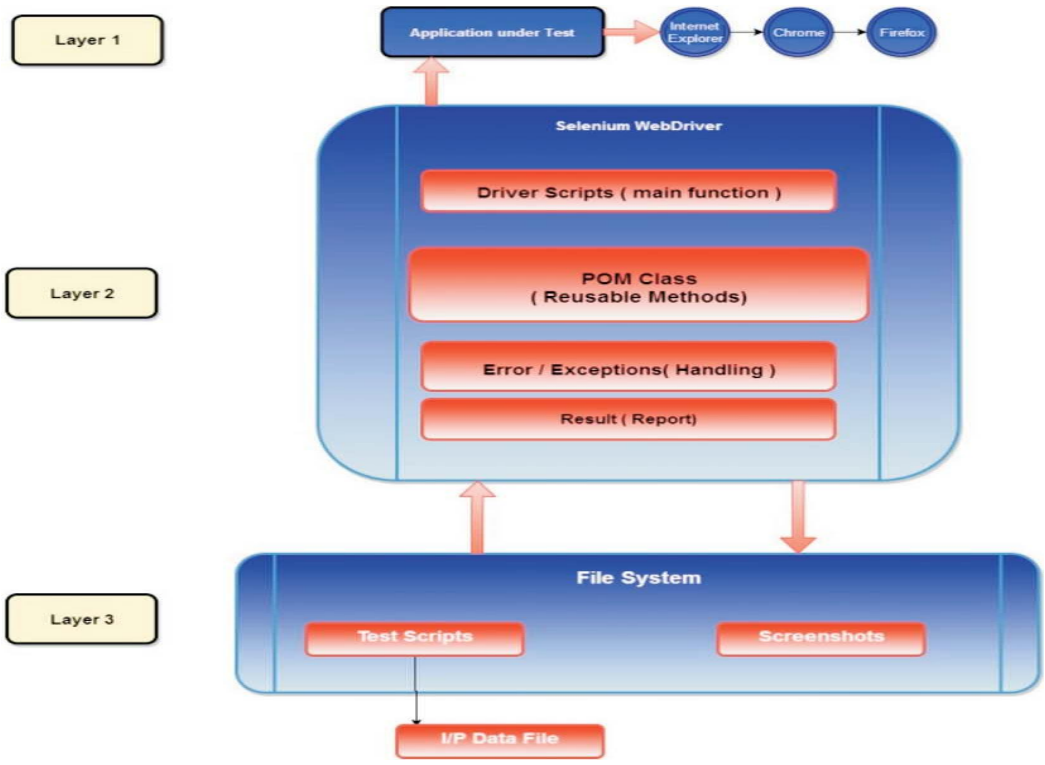


Figure 5.1 Three layered architecture of the RHTAF

Layer 1 -It comprises of the Application under Test (AUT) that could be run on any browser such as Internet explorer, Google chrome or Firefox. This study used two famous Indian e-commerce portals - www.amazon.in and www.flipkart.com as its web applications under test. Chrome is used as the web browser to test our AUT.

Layer 2 - Actual hybrid test automation framework that is built using selenium webdriver is implemented at this layer. Major components of layer 2 are as follows:

Driver Script - Driver script consist of main() function that consist of set of tests that call the reusable methods under the respective POM Class along with the ReadExcelFile() function. This function reads the excel file containing sheets row by row which shows the hybrid nature of framework (Combination of Keyword and Data Driven Framework) as shown in Figure. 5.2.

POM Class - It contains all the webpage objects and actions to be performed on those webpage objects within a particular webpage. Example-POM class LoginPage.java is created for automating the login functionality. Reusable methods were written for automating different user actions like enter username, enter password etc. on that

page. Similarly for each webpage, a separate POM class has been created which contains its own member functions.

Error/Exception handling - These functions take care of unexpected errors such as error or warning messages during execution of the test case.

Test Case	Keywords (Objects) Keyword Driven	Locator Type	Locator Value	Data(Data Driven)	Additonal Details
T1	Base_URL			https://www.amazon.in	
T2	Signin				
T3	Signin				
T4	Add Address				
T5	Delete Address				
T6	Edit Address			Full Name	Mr. Michael Scofield
T7	Edit Address			Pincode	null
T8	Edit Address			Mobile Number	9821321301
T9	See Cart				
T10	Search Product			Asus Zenfone Max	Black, 16
T11	Add to Cart				
T12	Delete from Cart			$1	
T13	Search Product			Apple Iphone 7	Black, 32
T14	Add to Cart				
T15	Proceed to Checkout				
T16	Login				
T17	Change Address				
T18	Add Address				
T19	Choose delivery speed			Two Day Delivery	
T20	Choose payment method				
T21	Place your order				

Figure 5.2 Input excel file containing keywords and data for amazon.in

Result (Report) - This refers to the reporting of the test cases execution report in the prescribed format for the analysis. We used extent report plug-in to generate the reports for every executed test case.

Layer 3 - It comprises of input data files in .xls format containing test data for each test scenario. First column contains all the reusable keywords for each web page of our application under test and other column is for the data driven framework mainly in the form of excel file as shown in Figure 5.2. Our framework also has a screenshot functionality which captures the screenshot in case of failed test case to record any error or warning messages during execution of the test script.

5.2.2 Implementation and Experimental set up for Analysis of RHTAF in Scrum Environment

The implementation of our Reusable Hybrid Test Automation Framework (RHTAF) was performed on www.amazon.in and www.flipkart.com, two widely used e-commerce websites in India. RHTAF is implemented and experimentally analyzed by simulating scrum environment. Initially manual test cases were written for the business critical functionality for the two websites. Thereafter, we split the total number of test cases in three successive sprints. Our sprint duration was two weeks per sprint, RHTAF was applied to automate the test cases on sprint by sprint basis.

5.2.2.1 InSprint Test Automation Test Strategy

We propose Insprint automation test strategy to use our RHTAF on web based projects developed in scrum environment. Insprint automation test strategy was developed to effectively apply test automation on scrum projects. It involves identifying important test scenarios with respect to business critical functionality of system under test. Thereafter, regression test cases are designed based on the test scenarios. In order to apply InSprint test automation strategy, we initially wrote manual test cases for business critical functionality for the two ecommerce websites. Thereafter, we split the total number of test cases in three successive sprints. Our sprint duration was two weeks per sprint and RHTAF was applied to automate the regression test cases on sprint by sprint basis as shown in Table 5.1. In Sprint 1, regression test cases that cover important business critical test cases were automated that includes Login, Add Address, Delete Address and Edit Address functionalities for both the websites. These test cases from T1-T8 became an initial part of our regression test suite denoted by R1.1 which means regression test suite for release 1 and sprint1. Similarly, in Sprint 2, test cases from T9-T14 were automated and added to regression test suite R1.1 that became our regression test suite R1.2. In the same way, we automated test cases from T15 to T20 that were added to regression test suite R1.2 and we got our completed regression suite ready for release 1 that is R1.3 as shown in Figure 5.3 and our complete regression test suite can be calculated as: Complete regression suite (R1) = (T1-T20) = R1.3

Automating our regression tests using RHTAF on sprint by sprint basis gave us leverage to have one subset of regression suite ready by the end of sprint 1 which we ran at the start of sprint 2. Similarly, we had one subset of regression suite ready by the end of sprint 2 which we ran at the beginning of sprint 3 and finally at the end of

sprint 3 our whole regression test suite was ready which we ran at the hardening sprint (Last sprint used to execute whole regression test suite) to check overall functionally of our application under test.

Table 5.1 Different regression test cases automated over 3 sprints

Sprint Number	Test case number	Test Case Name	Test case scenario (positive or negative)
1.	T1	Open browser	Positive
1	T2	Login	Negative
1.	T3	Login	Positive
1.	T4	Add address	Positive
1.	T5	Delete address	Positive
1.	T6	Edit address	Negative
1.	T7	Edit address	Positive
1.	T8	Cart lookup	Positive
2.	T9	Search product	Positive
2.	T10	Add to cart	Positive
2.	T11	Delete from cart	Positive
2.	T12	Search product	Positive
2.	T13	Add to cart	Positive
3.	T14	Proceed to checkout	Positive
3.	T15	Login	Positive
3.	T16	Change address	Positive
3.	T17	Add address	Positive
3.	T18	Choose delivery speed	Positive
3.	T19	Choose payment method	Positive
3.	T20	Place your order	Positive

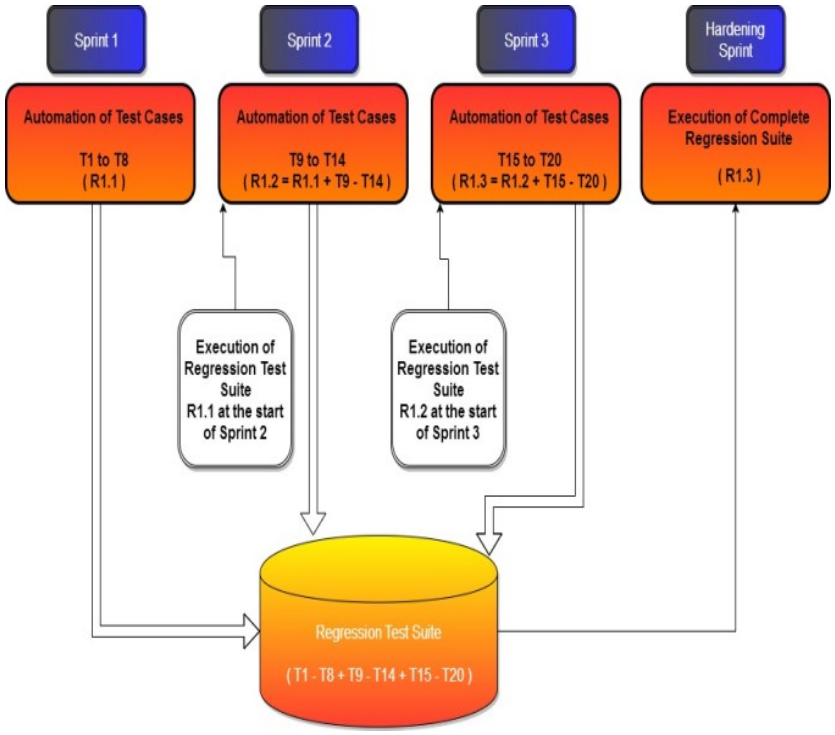


Figure 5.3 Implementation of RHTAF over three sprints in scrum environment

5.3 Implementation details of RHTAF using Page Object Model

In this section we explain the implementation of RHTAF with the help of selected screen shots from every sprint

- **Sprint 1:** Firstly, Driver script that is main function() of amazon.in calls the ReadExcelFile() function as shown in Figure 5.4. ReadExcelFile() function reads the input excel file row by row. On reading the second element (keyword) of a row as shown in Figure 5.2, the corresponding page object class is instantiated and corresponding member function reusable method is invoked.

```
}

public static void main(String[] args) throws InterruptedException
{
  System.setProperty("webdriver.chrome.driver","C:\\Softwares\\selenium-java-3.4.0\\chromedriver.exe");     //Chrome driver location
  driver = new ChromeDriver();
  driver.manage().window().maximize();    //Maximize chrome browser

  report = new ExtentReports("F:\\Studies\\Project\\Activation Framework Project\\Amazon.html"); /*A report of HTML type. PAth
                                                    where report is going to be stored*/
  for(int i=1;i<=20;i++)
  {
    String testcase = String.format("VerifyTestCase %d", i);     //Name of the test
    logger = report.startTest(testcase);                  //Starts the test and keeps log of the test and saves the name of the test
    ReadExcelFile(1,i);
  }
  report.endTest(logger);                          //Ends the Test Cases
  report.flush();                                 //Writes all the logs to the HTML report

  driver.get("F:\\Studies\\Project\\Activation Framework Project\\Amazon.html");//To get the report after executing all tests
}
```

Figure 5.4 Driver (Main) function of amazon.in

```
}

public static void main(String[] args) throws InterruptedException
{
  System.setProperty("webdriver.chrome.driver","C:\\Softwares\\selenium-java-3.4.0\\chromedriver.exe");     //Chrome driver location
  driver = new ChromeDriver();
  driver.manage().window().maximize();    //Maximize chrome browser

  report = new ExtentReports("F:\\Studies\\Project\\Activation Framework Project\\Flipkart.html"); /*A report of HTML type. PAth
                                                                                                  where report is going to be stored*/
  for(int i=1;i<=20;i++)
  {
    String testcase = String.format("VerifyTestCase %d", i);     //Name of the test
    logger = report.startTest(testcase);                    //Starts the test and keeps log of the test and saves the name of the test
    ReadExcelFile(1,i);
  }
  report.endTest(logger);                                   //Ends the Test Cases
  report.flush();                                          //Writes all the logs to the HTML report

  driver.get("F:\\Studies\\Project\\Activation Framework Project\\Flipkart.html"); //To get the report after executing all tests
}
```

Figure 5.5 Driver (Main) function flipkart.com

Automating Signin test case (T2), main function() calls the ReadExcelFile() function which reads the input excel file. On finding "Signin" keyword, it calls the LoginPage.java POM class which in turn invokes the corresponding member function for executing login test case for amazon.in as shown in Figure 5.6 and Figure 5.7.

```java
package pageObjects;

import org.openqa.selenium.By;

public class LoginPage
{
    public static void Credentials(WebDriver driver,String username, String password,String site) throws InterruptedException
    {
        if(site.equals("amazon"))
        {
            try
            {
                driver.findElement(By.partialLinkText("Hello. Sign in")).click();
                Thread.sleep(2000);
                driver.findElement(By.name("email")).sendKeys(username);
                driver.findElement(By.name("password")).sendKeys(password);
                driver.findElement(By.id("signInSubmit")).click();
                Thread.sleep(2000);
            }
            catch (Exception e)
            {
                Thread.sleep(2000);
                driver.findElement(By.name("email")).clear();
                driver.findElement(By.name("email")).sendKeys(username);
                driver.findElement(By.name("password")).sendKeys(password);
                driver.findElement(By.id("signInSubmit")).click();
                Thread.sleep(2000);
            }
        }
        else
        {
            try
            {
                driver.findElement(By.linkText("Log In")).click();
                Thread.sleep(2000);
                driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[1]/input")).clear();          //Clears the email text box
                driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[1]/input")).sendKeys(username); //Email
                Thread.sleep(2000);
                driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[2]/input")).sendKeys(password); //Password
                Thread.sleep(2000);
                driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[3]/button")).click();//Login Button
                Thread.sleep(2000);
```

Figure 5.6 Sprint 1 - Automation of login functionality using page object model for amazon.in

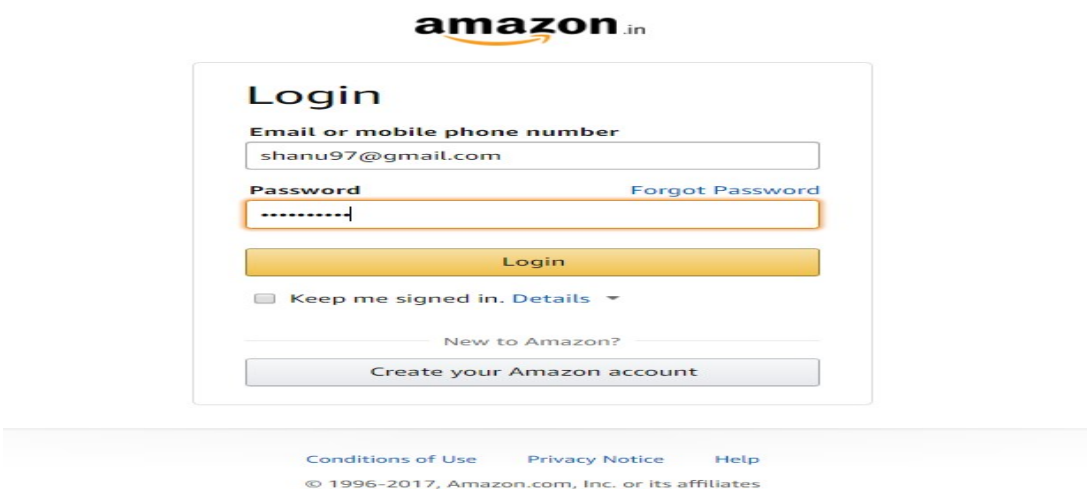


Figure 5.7 Sprint 1 - Execution of login test case of amazon.in

For flipkart.com, operation is exactly similar as with amazon.in. Driver script or Main function() of flipkart.com calls the ReadExcelFile() function as shown in Figure 5.5

which reads the input excel file row by row as shown in Figure 5.8 and when it reads the second element (keyword) of the row then corresponding page object class is called and corresponding member function (reusable method) is being executed.

Test Case	Keywords (Objects) Keyword Driven	Locator Type	Locator Value	Data (Data Driven)	Additonal Details
T1	Base_URL			https://www.flipkart.com	
T2	Signin				
T3	Signin				
T4	Add Address				
T5	Delete Address				
T6	Edit Address			Full Name	Mr. Michael Scofield
T7	Edit Address			Pincode	null
T8	Edit Address			Mobile Number	9821321301
T9	See Cart				
T10	Search Product			Asus Zenfone Max	Black,16
T11	Add to Cart				
T12	Delete from Cart			$1	
T13	Search Product			Apple Iphone 7	Black,32
T14	Add to Cart				
T15	Proceed to Checkout				
T16	Login				
T17	Change Address				
T18	Add Address				
T19	Choose delivery Speed			Two Day Delivery	
T20	Choose payment method				
T21	Place your order				

Figure 5.8 Input excel file containing keywords and data for flipkart.com

The inherent advantage of this framework is the use of Page Object Model which makes it reusable. Reusable methods implemented in this framework are used to automate both amazon.in and flipkart.com.

Hence, for second test case i.e. Signin for flipkart.com, main function() calls the ReadExcelFile() function which reads the input excel file as shown in Figure 5.8 and on finding “Signin” keyword, it calls the LoginPage.java POM class which in turn executes the corresponding member function for executing login test case for flipkart.com as shown in Figure 5.9 and Figure 5.10.

```
            driver.findElement(By.name("email")).sendKeys(username);
            driver.findElement(By.name("password")).sendKeys(password);
            driver.findElement(By.id("signInSubmit")).click();
            Thread.sleep(2000);
        }
    }
    else
    {
        try
        {
            driver.findElement(By.linkText("Log In")).click();
            Thread.sleep(2000);
            driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[1]/input")).clear();          //Clears the email text box
            driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[1]/input")).sendKeys(username); //Email
            Thread.sleep(2000);
            driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[2]/input")).sendKeys(password); //Password
            Thread.sleep(2000);
            driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[3]/button")).click();//Login Button
            Thread.sleep(2000);
        }
        catch(Exception e)
        {
            driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[1]/input")).clear();          //Clears the email text box
            driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[1]/input")).sendKeys(username); //Email
            Thread.sleep(2000);
            driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[2]/input")).clear();          //Clears the password text box
            driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[2]/input")).sendKeys(password); //Password
            Thread.sleep(2000);
            driver.findElement(By.xpath("/html/body/div[3]/div/div/div/div/div[2]/div/form/div[3]/button")).click();//Login Button
            Thread.sleep(2000);


        }
    }
}
public static String getLog()
{
    return "Signed In Successfully";
}

}
```

Figure 5.9 Sprint 1 - Automation of login functionality using page object model for flipkart.com

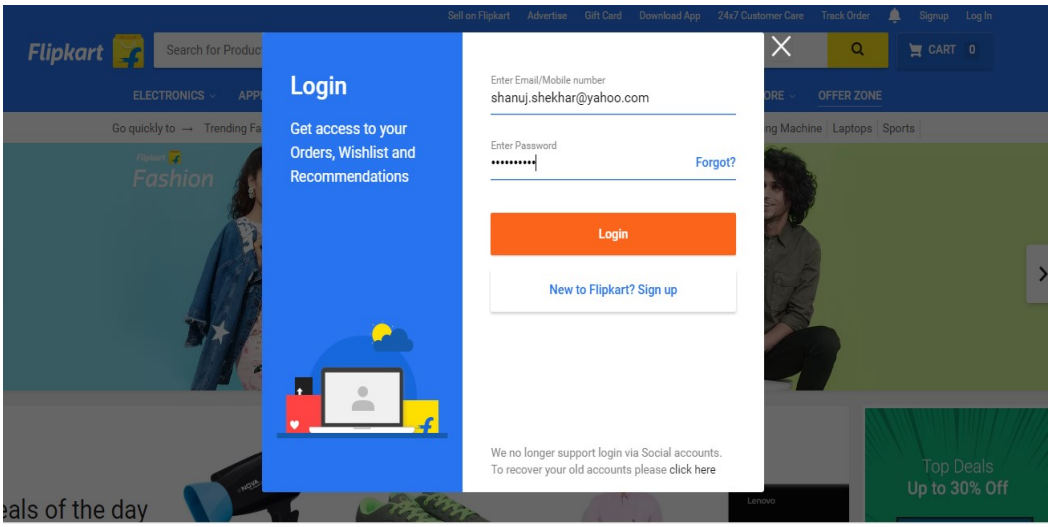


Figure 5.10 Sprint 1 - Execution of login test case of flipkart.com

Sprint 2: In Sprint 2, for Search Product test case; main function calls the ReadExcelFile() function which reads the excel file row by row and on finding "Search Product" keyword, it calls the SearchProduct.java POM class which in turn executes the corresponding member function for executing Search Product test case for amazon.in as shown in Figure 5.11 and Figure 5.12.

```
package pageObjects;

import java.util.ArrayList;

public class SearchProduct
{

  public static void search(WebDriver driver,String text,String Add_details,String site) throws InterruptedException
  {
    if(site.equals("amazon"))
    {
      driver.findElement(By.id("twotabsearchtextbox")).sendKeys(text);
      driver.findElement(By.className("nav-input")).click();                    //Click on search button
      Thread.sleep(2000);
      driver.findElement(By.partialLinkText(Add_details)).click();
      Thread.sleep(2000);
      ArrayList<String> tabs = new ArrayList<String> (driver.getWindowHandles());
      //Close the current tab
      driver.close();
      //Switch to New Tab
      driver.switchTo().window(tabs.get(1));
      }
```

Figure 5.11 Sprint 2 - Automation of search product functionality using page object model for amazon.in

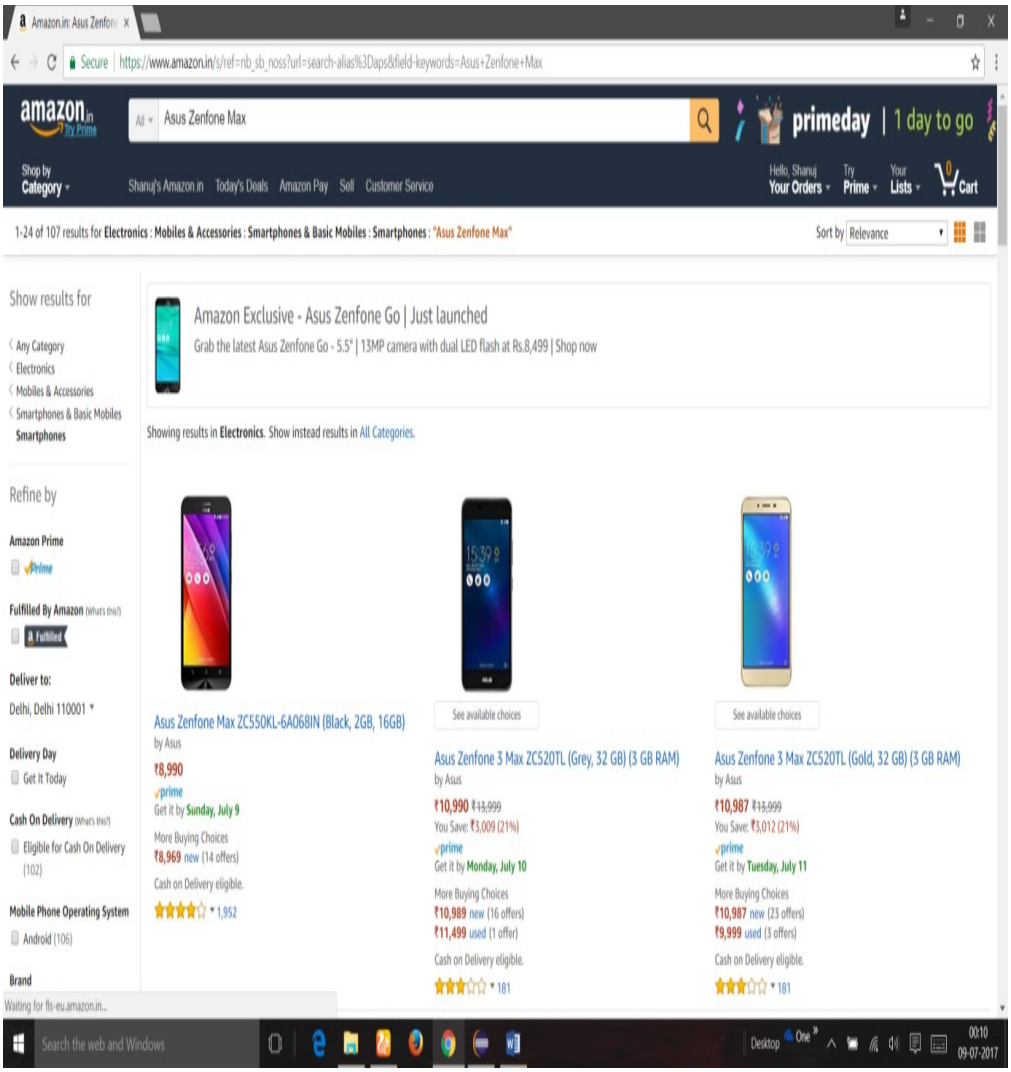


Figure 5.12 Sprint 2 - Execution of search product test case of amazon.in

Similarly, in Sprint 2, for Search Product test case; main function calls the ReadExcelFile() function which reads the excel file row by row as shown in Figure 5.8and on finding "Search Product" keyword, it calls the SearchProduct.java POM class which in turn executes the corresponding member function for executing Search Product test case for flipkart.com as shown in Figure 5.13 and Figure 5.14.

```
else if(site.equals("flipkart"))
{
    driver.findElement(By.name("q")).sendKeys(text);
    Thread.sleep(2000);
    try
    {
        driver.findElement(By.className("vh79eN")).click();                    //Click on search button on home page
    }
    catch(Exception e)
    {
        driver.findElement(By.xpath("//*[@id='fk-header-search-form']/div/div/div[2]/input[1]")).click();     //Click on the search button on add to cart page
    }
    Thread.sleep(2000);
    driver.findElement(By.partialLinkText(Add_details)).click();
    Thread.sleep(2000);
    ArrayList<String> tabs = new ArrayList<String> (driver.getWindowHandles());
    //Close the current tab
    driver.close();
    //Switch to New Tab
    driver.switchTo().window(tabs.get(1));
    Thread.sleep(10000);
    try
    {
      try
      {
        System.out.println("In try");
        driver.findElement(By.id("pincodeInputId")).clear();
        driver.findElement(By.xpath("//*[@id='pincodeInputId']")).sendKeys("110002");  //Fill it with 110002 pincode
        System.out.println("After input");
        driver.findElement(By.xpath("//*[@id='container']/div/div[1]/div/div/div[1]/div/div[2]/div[6]/div[1]/div[2]/div/div[2]/div[1]/span")).click();   //click on Check Link
        Thread.sleep(2000);
        System.out.println("in try 1");
      }
      catch(NoSuchElementException e1)
      {
        System.out.println("after try");
```

Figure 5.13 Sprint 2 - Automation of search product functionality using page object model for flipkart.com

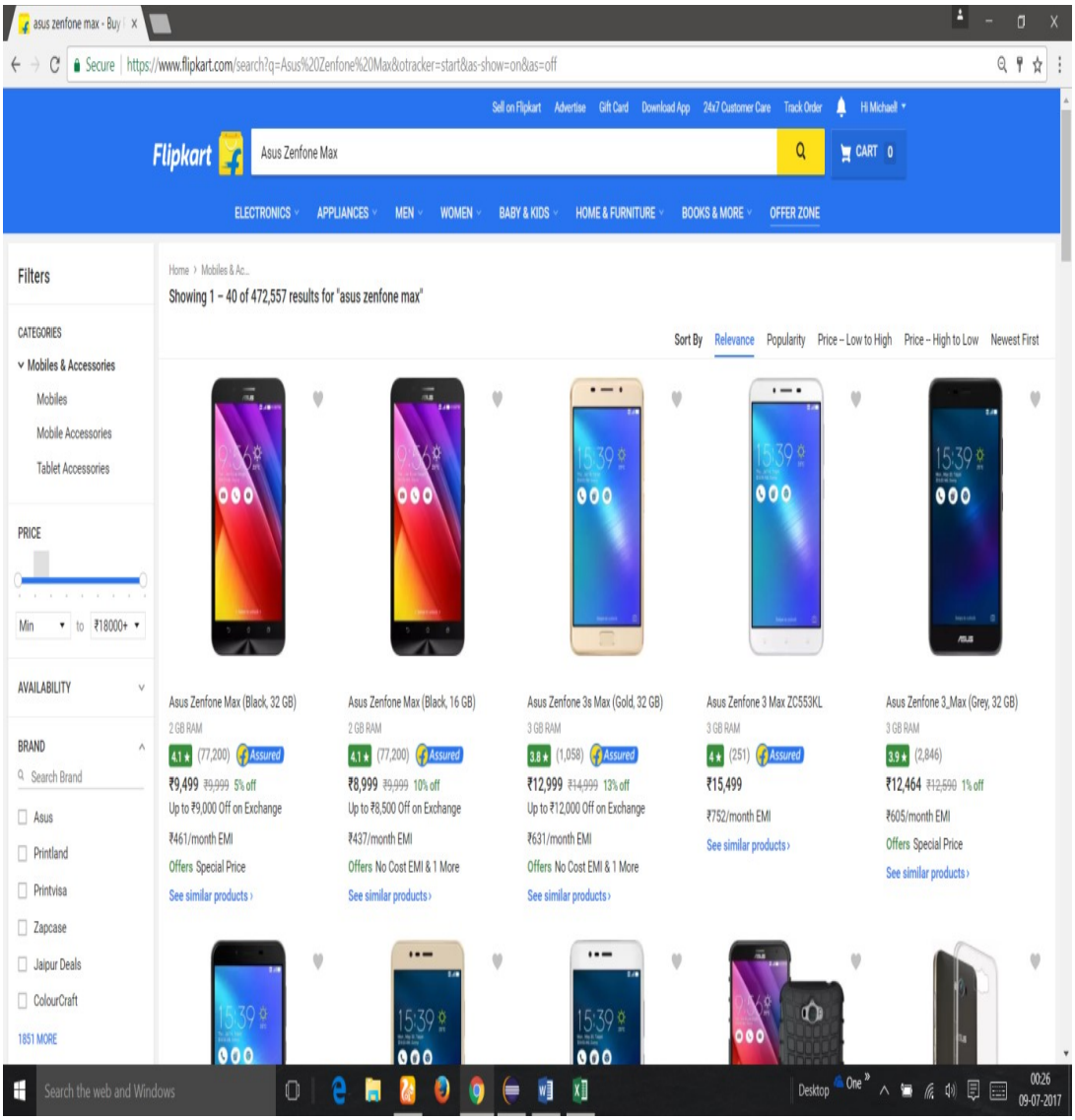


Figure 5.14 Sprint 2 - Execution of search product test case for flipkart.com

Sprint 3: In Sprint 3, for Choosing a Payment Method test case; main function calls the ReadExcelFile() function which reads the excel file row by row and on finding "Payment By" keyword, it calls the PaymentBy.java POM class which in turn executes the corresponding member function for executing Search Product test case for amazon.in as shown in Figure 5.15 and Figure 5.16.

workspace - Java - SummerProjectAutomationFramework/src/pageObjects/PaymentBy.java - Eclipse

```
{
  if(site.equals("amazon"))
  {
    flag = 0;
    try
    {
      Thread.sleep(2000);
      driver.findElement(By.id("pm_0")).click();  //Already Existing Credit/Debit Card
      Thread.sleep(2000);
    }
    catch(Exception e1)
    {
      flag = 1;
      switch(Type)
      {
        case "Credit Card" :  System.out.println("In Credit Card");
                Thread.sleep(2000);
                driver.findElement(By.id("pm_new_verified_credit_card")).click();          //To select Credit Card
                Thread.sleep(500);
                driver.findElement(By.id("vccName")).sendKeys(Name);                       //To write Name
                Thread.sleep(500);
                driver.findElement(By.id("newVerifiedCreditCardNumber")).sendKeys(card_number);        //To write Credit Card Number
                Thread.sleep(500);
                List<WebElement> dropdowns = driver.findElements(By.className("a-dropdown-prompt"));   //To have elements having same classname
                String dt;

                if(Exp_date.length()==7)
                  dt = Exp_date.substring(0, 2);
                else
                  dt = Exp_date.substring(0, 1);

                dropdowns.get(0).click();                                  //To open the drop down menu of date
                Thread.sleep(2000);
                dropdowns.get(0).click();
                String date = String.format("//*[@id='2_dropdown_combobox']/li[%s]/a", dt);
                Thread.sleep(500);
```

Figure 5.15 Sprint 3 - Automation of choosing payment method functionality using page object model for amazon.in

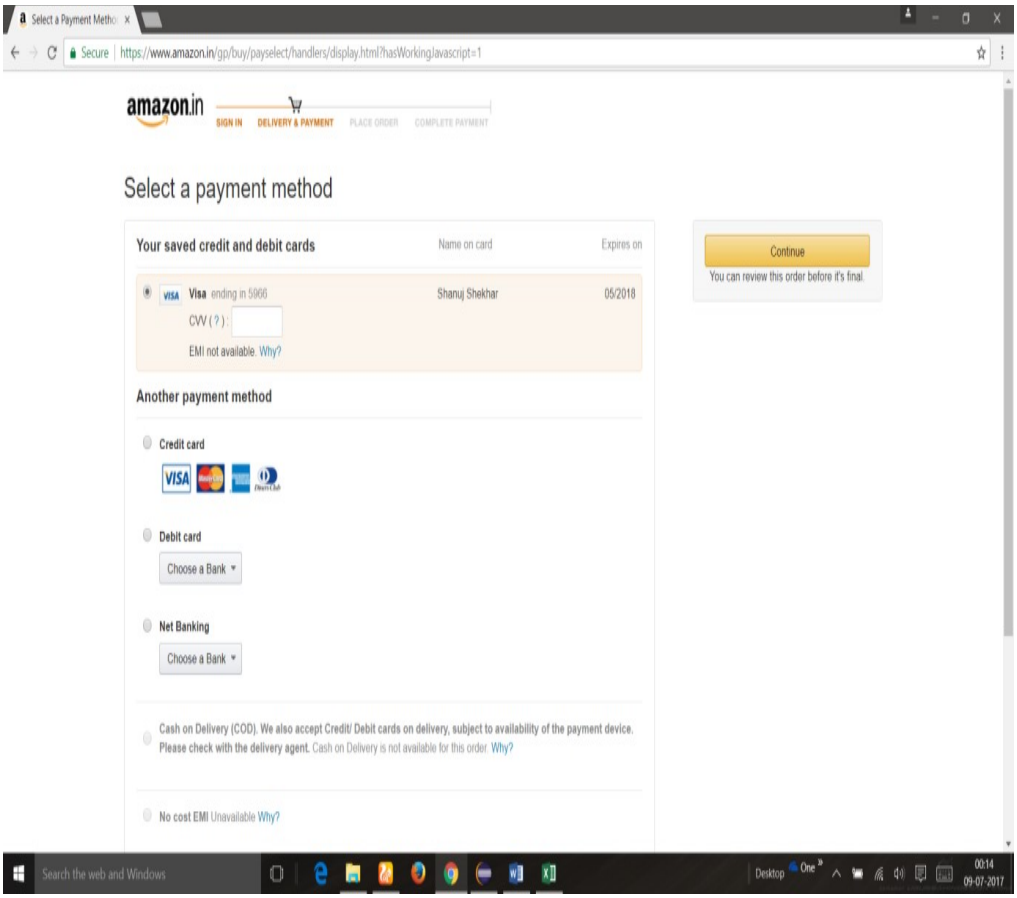


Figure 5.16 Sprint 3 - Execution of choosing payment method test case for amazon.in

Similarly, in Sprint 3, for automating Choosing a Payment Method test case for flipkart.com; main function calls the ReadExcelFile() function which reads the excel file row by row as shown in Figure 5.8 and on finding “Payment By” keyword, it calls the PaymentBy.java POM class which in turn executes the corresponding member function for executing Search Product test case for flipkart.com as shown in Figure 5.17 and Figure 5.18.

```
else if(site.equals("flipkart"))
{
    try
    {

        System.out.println("In payment");
        JavascriptExecutor jse = (JavascriptExecutor)driver;
        jse.executeScript("window.scrollBy(0,280)", "");//*[@id="container"]/div/div[1]/div/div/div/div[2]/div/label[2]/div[2]/div/div/form/div[2]/div/div/input
        Thread.sleep(2000);
        try
        {
            driver.findElement(By.xpath("//*[@id='container']/div/div[1]/div/div/div/div[2]/div/label[2]/div[1]")).click();
        }
        catch(Exception e)
        {
            System.out.println("No click");
        }
        Thread.sleep(2000);
        driver.findElement(By.name("cardNumber")).sendKeys(card_number);
        //driver.findElement(By.class.Name("_208vszPyFYht2kUNkYpLO")).sendKeys(card_number);
        //driver.findElement(By.xpath("//*[@id='container']/div/div[1]/div/div/div/div[2]/div/label[2]/div[2]/div/div/form/div[1]/div/div/input")).sendKeys(card_number);
        Thread.sleep(2000);
        String month;

        if(Exp_date.length()==7)
            month = Exp_date.substring(0, 2);
        else
            month = Exp_date.substring(0, 1);

        int index = Integer.parseInt(month);
        Select month_select = new Select(driver.findElement(By.name("month")));
        month_select.selectByIndex(index+1);

        Thread.sleep(2000);
        String year;
        year = Exp_date.substring(Exp_date.length()-2, Exp_date.length());
```

Figure 5.17 Sprint 3- Automation of choosing payment method functionality using page object model for flipkart.com

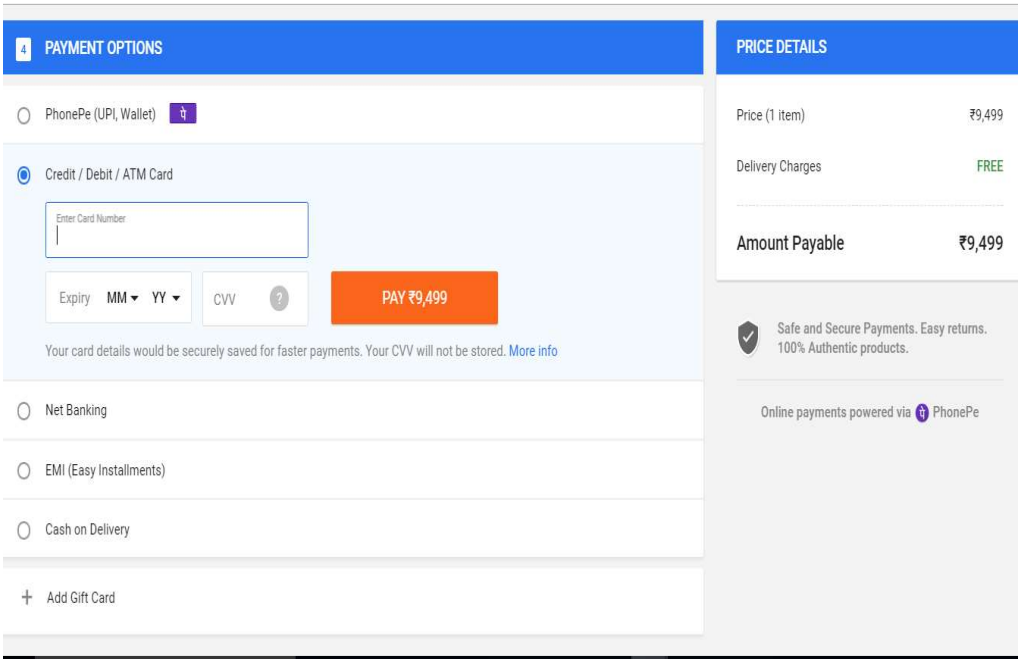


Figure 5.18 Sprint 3 - Execution of choosing payment method test case for flipkart.com

5.4 Evaluation and Validation of RHTAF

In this section, we describe the evaluation and validation of RHTAF:

5.4.1 Evaluating RHTAF

Three popular web applications namely: snapdeal.com, myntra.com and shopclues.com were used for assessing the code coverage and code duplication using RHTAF. Automating test cases using selenium webdriver consists of two functions: first to identify the web elements of the page object that needs to be automated in web application under test and second to perform the required operation on that page object.

Driver.findElement(By.name("Identify_WebElement").sendkeys("Perform_Operation"). ---------- Statement 1

Statement 1 denotes the method to automate some functionality of any web application. Suppose if this functionality is used in 100 test cases and web element in statement 1 changes then all corresponding 100 test cases needs to be updated in order

to change the web element. This is because of code duplication which causes poor maintainability of test cases. In order to validate reusable and maintainability aspect of our RHTAF, we used RHTAF on three popular e-commerce web applications viz. snapdeal, shopclues and myntra. Code duplication and code coverage metrics are used to validate our RHTAF on these three web applications. We automated 20 test cases without using RHTAF and then automated these 20 tests cases by using RHTAF then comparison on both these before and after using our RHTAF is presented using sonar cube reports. Sonar cube code analyzer is integrated with eclipse and reports have been generated to show the code duplication (Less code duplication leads to more maintainable tests) and code coverage metrics. Figure 5.19 shows the automation of web application snapdeal using RHTAF. Furthermore, page object class contains reusable methods. These reusable methods were used across all the three web applications which show the reusability of our RHTAF.

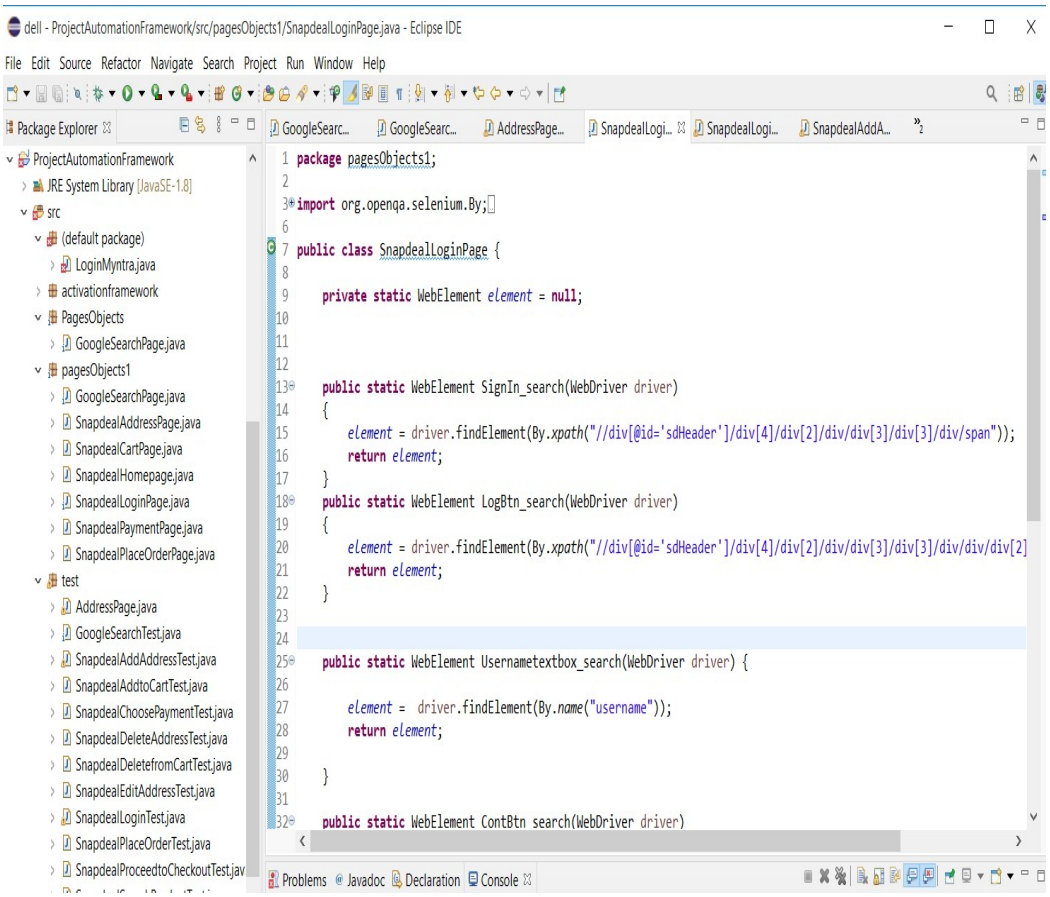


Figure 5.19 SnapdealLoginPage class

```
import pagesObjects1.SnapdealLoginPage;


public class SnapdealLoginTest {

    private static WebDriver driver = null;


    public static void main(String[] args) {


        googleSearch();
    }

    public static void googleSearch( ) {

        System.setProperty("webdriver.chrome.driver", "C:\\Users\\Dell\\Desktop\\NewChrome\\chromedriver.exe");
        driver=new ChromeDriver();

        driver.get("https://www.snapdeal.com/");

        SnapdealLoginPage.SignIn_search(driver).click();
        SnapdealLoginPage.LogBtn_search(driver).click();
        SnapdealLoginPage.Usernametextbox_search(driver).sendKeys("tyagisulabh2k@gmail.com");
        SnapdealLoginPage.ContBtn_search(driver).click();
        SnapdealLoginPage.Pwdtextbox_search(driver).sendKeys("Mohit@1234");
        SnapdealLoginPage.Login_button_search(driver).click();
```

Figure 5.20 SnapdealLoginTest class

Two separate classes were created under two different packages: pagesObjects1 package consist of all the page classes of the web application under test and test package consist of all test case classes to automate the required functionality. SnapdealLoginPage class was created to capture all the web elements of login page of snapdeal.

```java
public class SnapdealAddAddress1Test {

    private static WebDriver driver = null;


    public static void main(String[] args) {


        googleSearch();
    }

    public static void googleSearch( ) {

        System.setProperty("webdriver.chrome.driver", "C:\\Users\\Dell\\Desktop\\NewChrome\\chromedriver.exe");
        driver=new ChromeDriver();

        driver.get("https://www.snapdeal.com/");

        SnapdealAddAddress1Page.Profile_search(driver).click();
        SnapdealAddAddress1Page.Link_order(driver).click();
        SnapdealAddAddress1Page.SavedAddress_search(driver).click();
        SnapdealAddAddress1Page.AddNewAddress_search(driver).click();
        SnapdealAddAddress1Page.AddPincode_search(driver).sendKeys("201011");
        SnapdealAddAddress1Page.AddName_search(driver).sendKeys("Sulabh");
        SnapdealAddAddress1Page.AddAddress_search(driver).sendKeys("B31/F3, Surya Nagar");
        SnapdealAddAddress1Page.AddLocality_search(driver).sendKeys("Near Ram Mandir");
        SnapdealAddAddress1Page.AddCity_search(driver).sendKeys("Ghaziabad");
        SnapdealAddAddress1Page.AddMobile_search(driver).sendKeys("9871697952");
        SnapdealAddAddress1Page.AddLandline_search(driver).sendKeys("9999999999");
```

Figure 5.21 SnapdealAddAddress1Page class

SnapdealAddAddress1Page class was created to capture all the web elements of add address page of snapdeal as shown in Figure 5.21. SnapdealAddAddress1Test class was created to write the test methods to perform operations like click on profile link, click on order link, click on add address link, adding complete address details of the user and finally click on save address button as shown in Figure 5.22.

```
import org.openqa.selenium.By;

public class SnapdealAddAddress1Page {

    private static WebElement element = null;



    public static WebElement Profile_search(WebDriver driver)
    {
        element = driver.findElement(By.xpath("//div[@id='sdHeader']/div[4]/div[2]/div/div[3]/div[3]/div/span[2]/i
        return element;
    }
    public static WebElement Link_order(WebDriver driver)
    {
        element = driver.findElement(By.xpath("//div[@id='sdHeader']/div[4]/div[2]/div/div[3]/div[3]/div/div/div/c
        return element;
    }


    public static WebElement SavedAddress_search(WebDriver driver) {

        element =  driver.findElement(By.id("dotAccountSavedAddressesLi"));
        return element;

    }

    public static WebElement AddNewAddress_search(WebDriver driver)

    {
```

Figure 5.22 SnapdealAddAddress1Test class

Sonar cube code analyzer was used to generate report. Figure 5.23 shows the shows sonar cube report of snapdeal.com for automation of 20 test cases without using RHTAF. Code coverage is 55% and code duplication is 67% without using RHTAF.

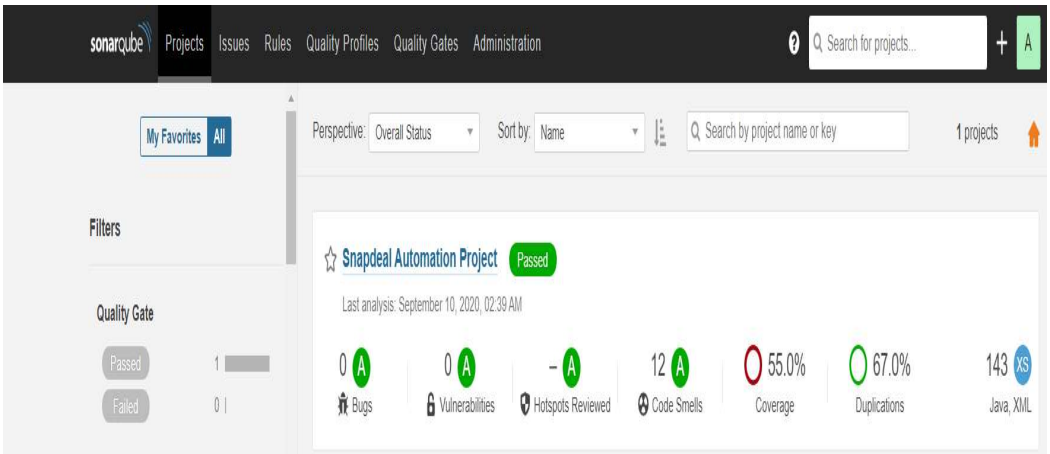


Figure 5.23 Sonar cube report of snapdeal.com automation without using RHTAF

Figure 5.24 shows sonar cube report of snapdeal.com for automation of 20 test cases using RHTAF. Test coverage has increased to 90% and code duplication has reduced to 12% by using RHTAF.

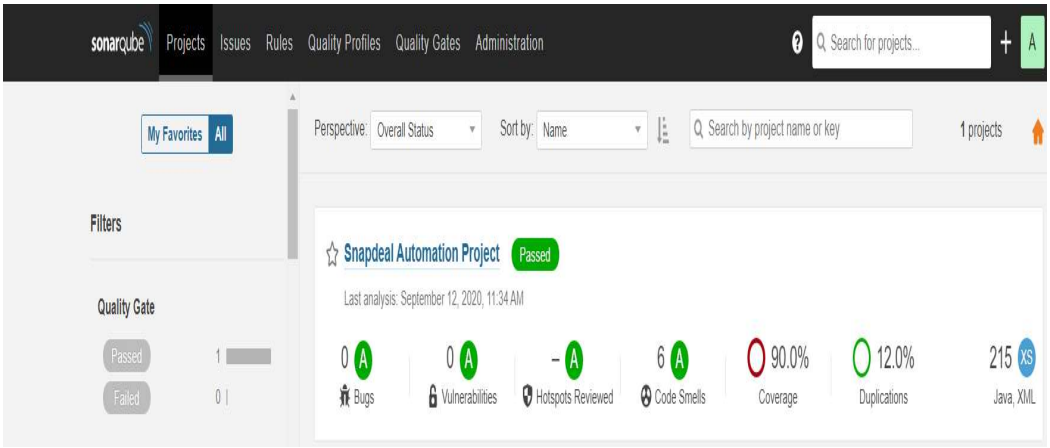


Figure 5.24 Sonar cube report of snapdeal.com automation using RHTAF

We also automated myntra.com using RHTAF. Firtsly, MyntraLoginPage class was created to capture all the web elements of login page of myntra as shown in Figure 5.25. MyntraLoginTest class was created to write the test methods to perform operations like click on profile link, click on login link, entering username, entering password and click on login button as shown in Figure 5.26.

```java
package pagesObjects1;

import org.openqa.selenium.By;

public class MyntraLoginPage {

    private static WebElement element = null;

    public static WebElement textbox_search(WebDriver driver) {

        element =  driver.findElement(By.name("q"));
        return element;

    }


    public static WebElement button_search(WebDriver driver) {

        element = driver.findElement(By.name("btnK"));
        return element;

    }


}
```

Figure 5.25 MyntraLoginPage class

```java
package test;

//import org.openqa.selenium.By;
import org.openqa.selenium.Keys;


public class MyntraLoginTest {

    private static WebDriver driver = null;


    public static void main(String[] args) {
        // TODO Auto-generated method stub

        googleSearch();
    }

    public static void googleSearch( ) {

        System.setProperty("webdriver.chrome.driver", "C:\\Users\\Dell\\Desktop\\NewChrome\\chromedriver.exe");
        driver=new ChromeDriver();

        driver.get("https://google.com");

        //driver.findElement(By.name("q")).sendkeys("https://www.myntra.com/");

        MyntraLoginPage.textbox_search(driver).sendKeys("Myntra");

        MyntraLoginPage.button_search(driver).sendKeys(Keys.RETURN);
```

Figure 5.26 MyntraLoginTest class

MyntraAddAddressPage class was created to capture all the web elements of add address page of myntra.com as shown in Figure 5.27.

```
import org.openqa.selenium.By;

public class MyntraAddAddressPage {

    private static WebElement element = null;



    public static WebElement Profile_search(WebDriver driver)
    {
        element = driver.findElement(By.xpath("(//header[@id='desktop-header-cnt']/div[2]/div[2]/div/div/s
        return element;
    }

    public static WebElement Order_search(WebDriver driver)
    {
        element = driver.findElement(By.xpath("(//header[@id='desktop-header-cnt']/div[2]/div[2]/div/div[2
        return element;
    }


    public static WebElement LinkAddress_search(WebDriver driver)
    {
        element = driver.findElement(By.xpath("(//a[contains(@href, '/my/address')])[2]"));
        return element;
    }


    public static WebElement AddAddress_search(WebDriver driver)
    {
        element = driver.findElement(By.cssSelector("div.addAddressBar-addText"));
        return element;
    }
```

Figure 5.27 MyntraAddAddressPage class

MyntraAddAddressTest class was created to write the test methods to perform operations like click on profile link, click on order link, click on link address, click on add address, adding complete address details of the user and finally click on save address button as shown in Figure 5.28.

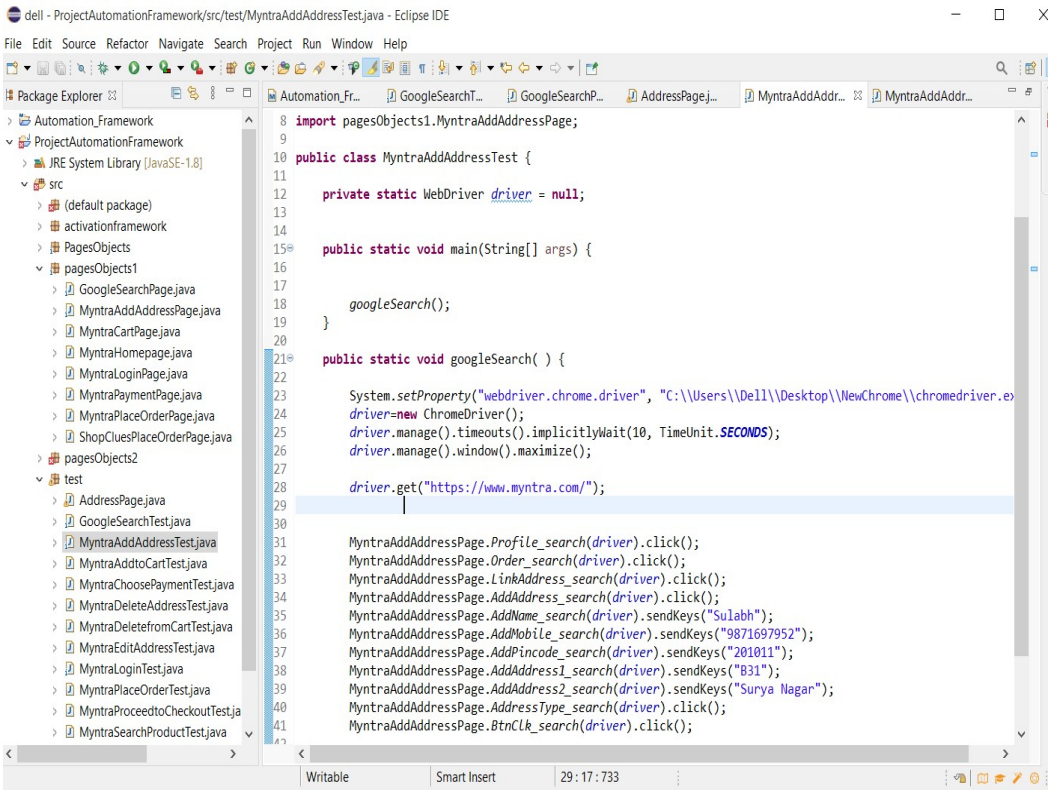


Figure 5.28 MyntraAddAddressTest class

Sonar cube code analyzer was used to generate report. Figure 5.28 shows sonar cube report of myntra.com for automation of 20 test cases without using RHTAF. Code coverage is 55% and code duplication is 68% without using RHTAF.

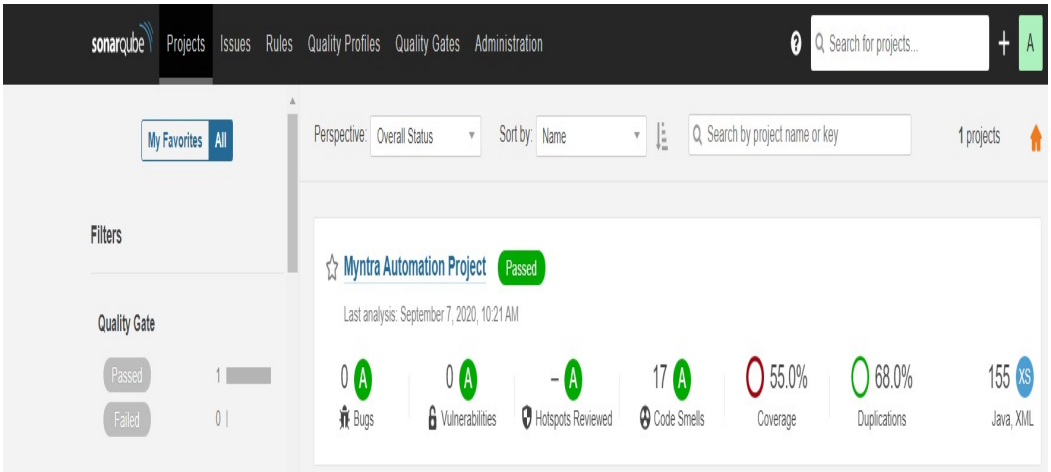


Figure 5.29 Sonar cube report of myntra.com without using RHTAF

Figure 5.29 shows the sonar cube report of automation test execution of 20 test cases using RHTAF. Test coverage has increased to 88% and code duplication has reduced to 10% by using RHTAF.

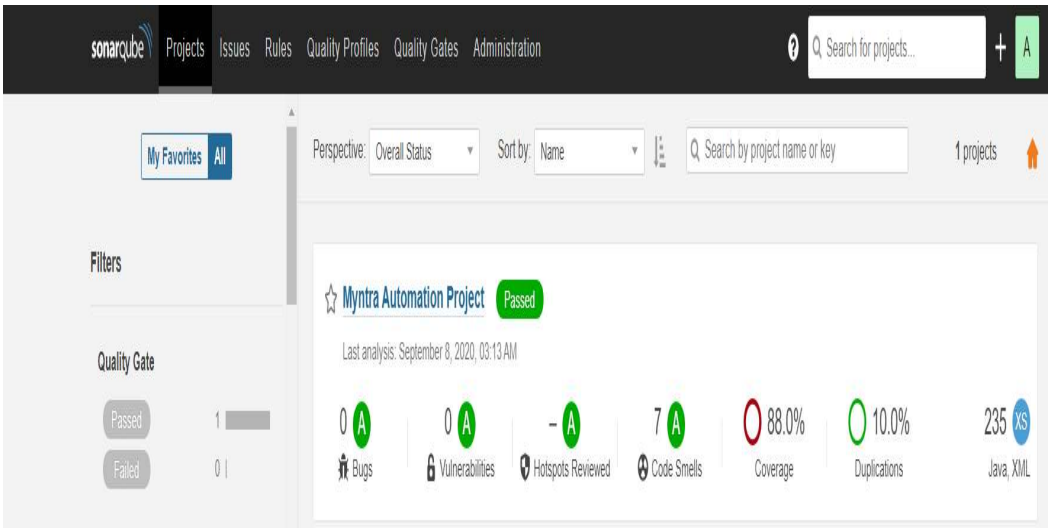


Figure 5.30 Sonar cube report of myntra.com automation using RHTAF

ShopCluesILoginPage class was created to capture all the web elements of login page of shopclues.com as shown in Figure 5.30. ShopCluesLoginTest class was created to write the test methods to perform operations like click on sign in link, entering username, entering password and click on login button as shown in Figure 5.31. Add address functionality was also automated using our RHTAF.

```java
package pagesObjects1;

import org.openqa.selenium.By;

public class ShopClueslLoginPage {

    private static WebElement element = null;



    public static WebElement SignIn_search(WebDriver driver)
    {
        element = driver.findElement(By.xpath("(//a[contains(@href, 'javascript:void(0);')])[10]"));
        return element;
    }

    public static WebElement Usernametextbox_search(WebDriver driver) {

        element =  driver.findElement(By.name("user_login"));
        return element;

    }

    public static WebElement Pwdtextbox_search(WebDriver driver) {

        element = driver.findElement(By.name("password"));
        return element;

    }
```

Figure 5.31 ShopCluesLoginPage class

```java
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;


import pagesObjects1.ShopClueslLoginPage;


public class ShopClueslLoginTest {

    private static WebDriver driver = null;


    public static void main(String[] args) {


        googleSearch();
    }

    public static void googleSearch( ) {

        System.setProperty("webdriver.chrome.driver", "C:\\Users\\Dell\\Desktop\\NewChrome\\chromedriver.exe");
        driver=new ChromeDriver();

        driver.get("https://www.shopclues.com/");

        ShopClueslLoginPage.SignIn_search(driver).click();
        ShopClueslLoginPage.Usernametextbox_search(driver).sendKeys("tyagisulabh2k@gmail.com");
        ShopClueslLoginPage.Pwdtextbox_search(driver).sendKeys("Mohit@1234");
        ShopClueslLoginPage.Login_button_search(driver).click();
```

Figure 5.32 ShopCluesLoginTest class

ShopCluesAddAddressPage class was created to capture all the web elements of add address page of shopclues.com as shown in Figure 5.32.

```
package pagesObjects1;

import org.openqa.selenium.By;

public class ShopCluesAddAddressPage {

    private static WebElement element = null;



    public static WebElement Link_search(WebDriver driver)
    {
        element = driver.findElement(By.xpath("(//a[contains(@href, 'javascript:void(0);')])[7]"));
        return element;
    }
    public static WebElement LinkProfile_search(WebDriver driver)
    {
        element = driver.findElement(By.xpath("//li[@id='sc_uname']/div/ul/li[4]/a"));
        return element;
    }


    public static WebElement MyAddressBook_search(WebDriver driver)
    {

        element =  driver.findElement(By.xpath("//div[@id='content']/div/div[5]/div/div/div[2]/div/div/div/div/ul/
        return element;

    }
```

Figure 5.33 ShopCluesAddAddressPage class

ShopCluesAddAddressTest class was created to write the test methods to perform operations like click on search link, click on profile link, click on my address book link, click on add address, adding complete address details of the user and finally click on save address button as shown in Figure 5.33

```
public class ShopCluesAddAddressTest {

    private static WebDriver driver = null;

    public static void main(String[] args) {

        googleSearch();
    }

    public static void googleSearch( ) {

        System.setProperty("webdriver.chrome.driver", "C:\\Users\\Dell\\Desktop\\NewChrome\\chromedriver.exe");
        driver=new ChromeDriver();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        driver.manage().window().maximize();

        driver.get("https://www.shopclues.com/");

        ShopCluesAddAddressPage.Link_search(driver).click();
        ShopCluesAddAddressPage.LinkProfile_search(driver).click();
        ShopCluesAddAddressPage.MyAddressBook_search(driver).click();
        ShopCluesAddAddressPage.AddNewAddress_search(driver).click();
        ShopCluesAddAddressPage.AddFirstName_search(driver).sendKeys("Sulabh");
        ShopCluesAddAddressPage.AddLastName_search(driver).sendKeys("Tyagi");
        ShopCluesAddAddressPage.AddMobile_search(driver).sendKeys("9871697952");
```

Figure 5.34 ShopCluesAddAddressTest class

Figure 5.34 shows the sonar cube report of shopclues.com for automation of 20 test cases without using RHTAF. Code coverage is 64% and code duplication is 72% without using RHTAF.

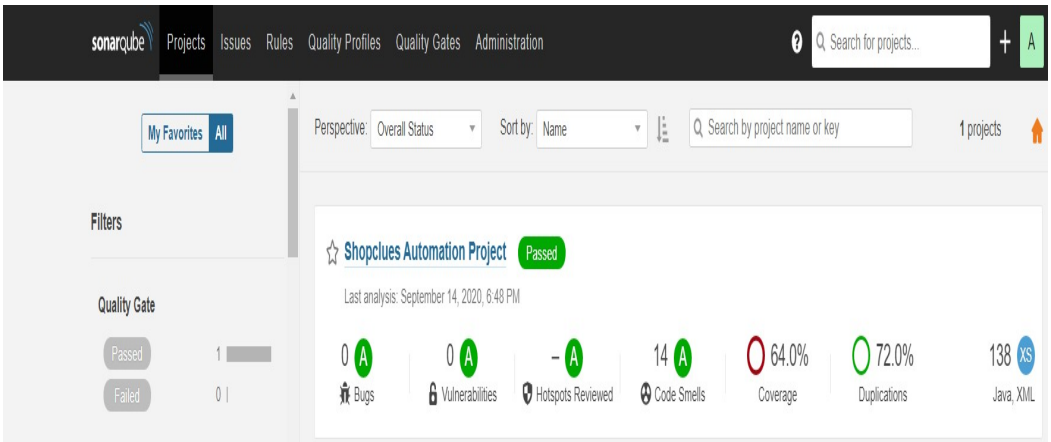


Figure 5.35 Sonar cube report of shopclues.com without using RHTAF

Figure 5.35 shows the sonar cube report of shopclues.com for automation of 20 test cases using RHTAF. Code coverage is increased to 86% and code duplication is reduced to 10% using RHTAF.

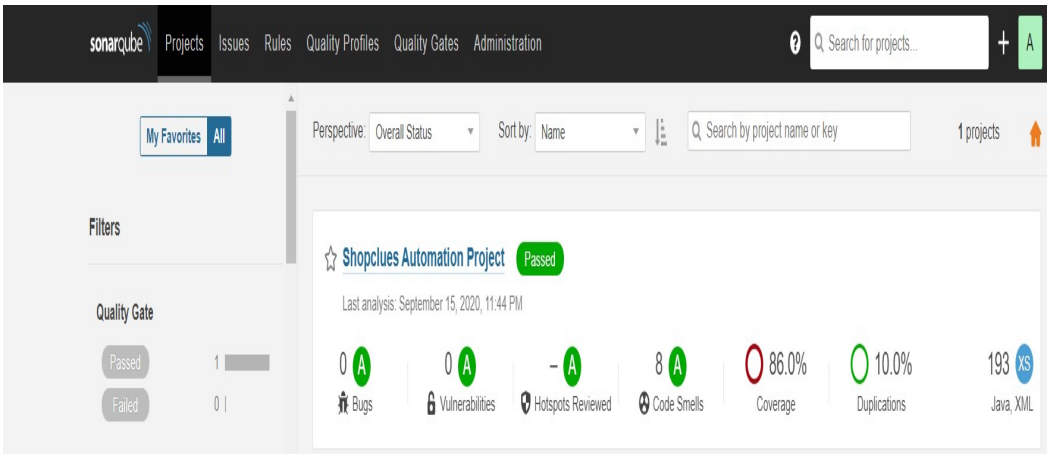


Figure 5.36 Sonar cube report of shopclues.com using RHTAF

In section 5.2, we have already explained the automation of amazon.in and flipkart.com using RHTAF. Figure 5.36 shows the sonar cube report of amazon.in for automation of 20 test cases without using RHTAF. Code coverage is 60% and code duplication is 70% without using RHTAF.

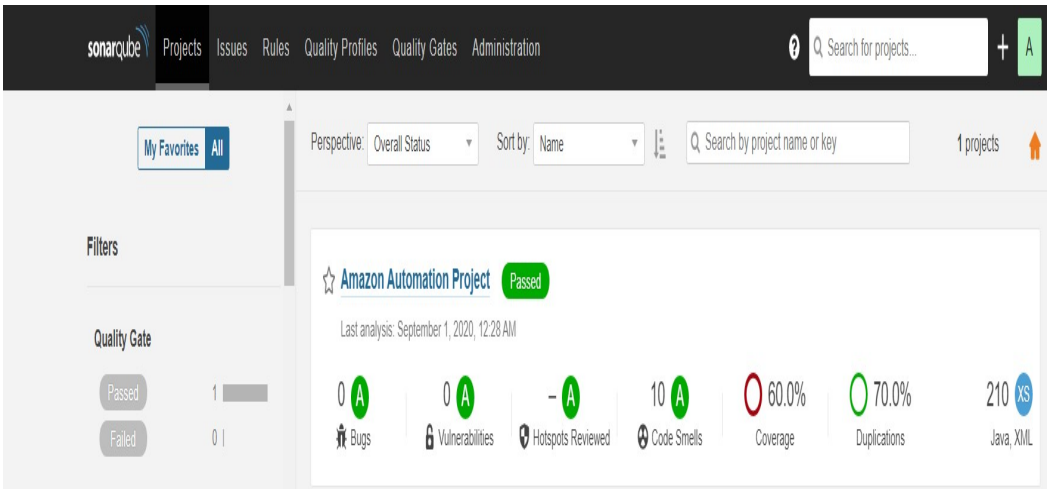


Figure 5.37 Sonar cube report of amazon.in without using RHTAF

Figure 5.37 shows the sonar cube report of amazon.in for automation of 20 test cases using RHTAF. Code coverage is increased to 90% and code duplication is reduced to 10% using RHTAF.

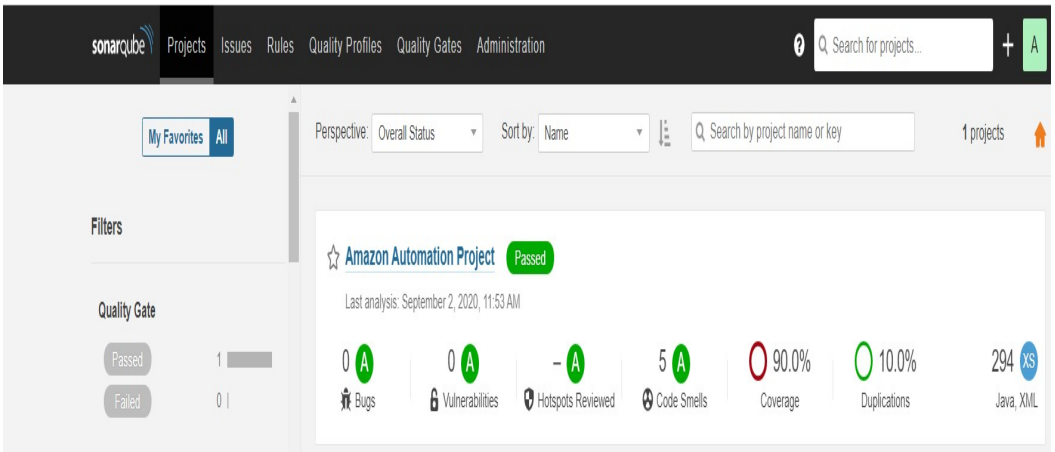


Figure 5.38 Sonar cube report of amazon.in using RHTAF

Figure 5.38 shows the sonar cube report of flipkart.com for automation of 20 test cases without using RHTAF. Code coverage is 55% and code duplication is 68% without using RHTAF.

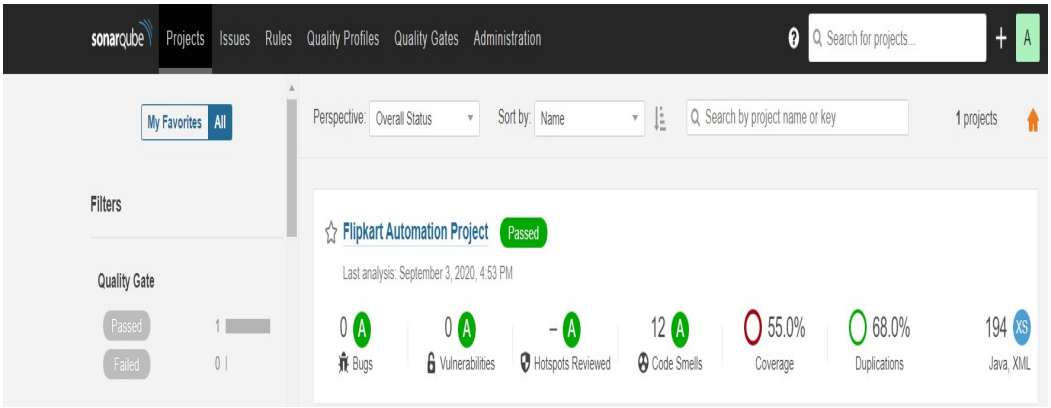


Figure 5.39 Sonar cube report of flipkart.com without using RHTAF

Figure 5.39 shows the sonar cube report of flipkart.com for automation of 20 test cases using RHTAF. Code coverage is increased to 85% and code duplication is reduced to 13% using RHTAF.

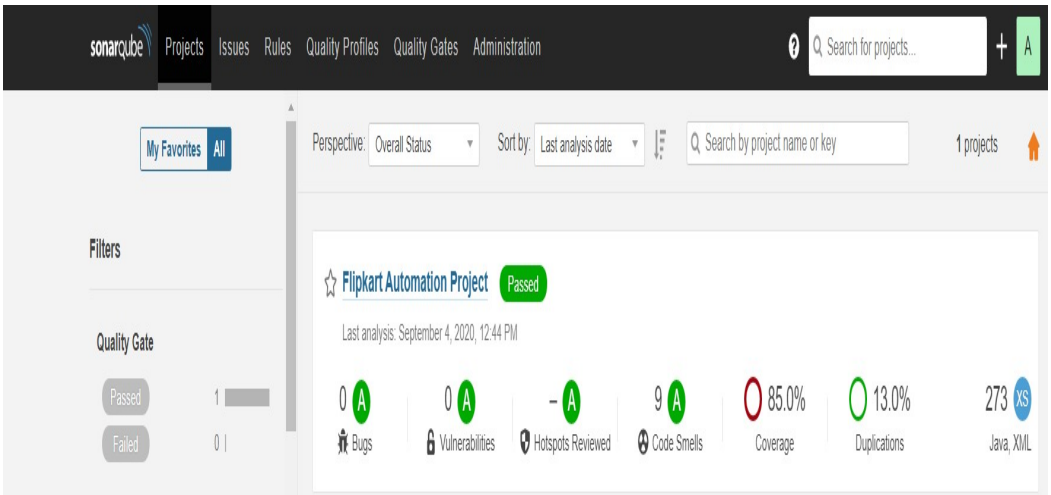


Figure 5.40 Sonar cube report of flipkart.com using RHTAF

5.4.2 Validation with respect to automated test case execution time

In order to compare the performance of automated test execution of test scripts w.r.t test execution time developed using RHTAF, we also developed automated test scripts using Katalon tool using keyword driven framework. Thereafter, test execution time of our RHTAF using selenium 2.0 and keyword driven framework using katalon are compared as shown in Table 5.3. Test suite of 20 test cases as shown in table 5.1 has been created for business critical functionality for amazon.in and flipkart.com. Figure 5.40 and 5.41 shows the automated script creation for amazon.in using Katalon tool. Katalon tool is used to capture web elements as keywords, thereafter methods have been written for performing actions during test case execution.

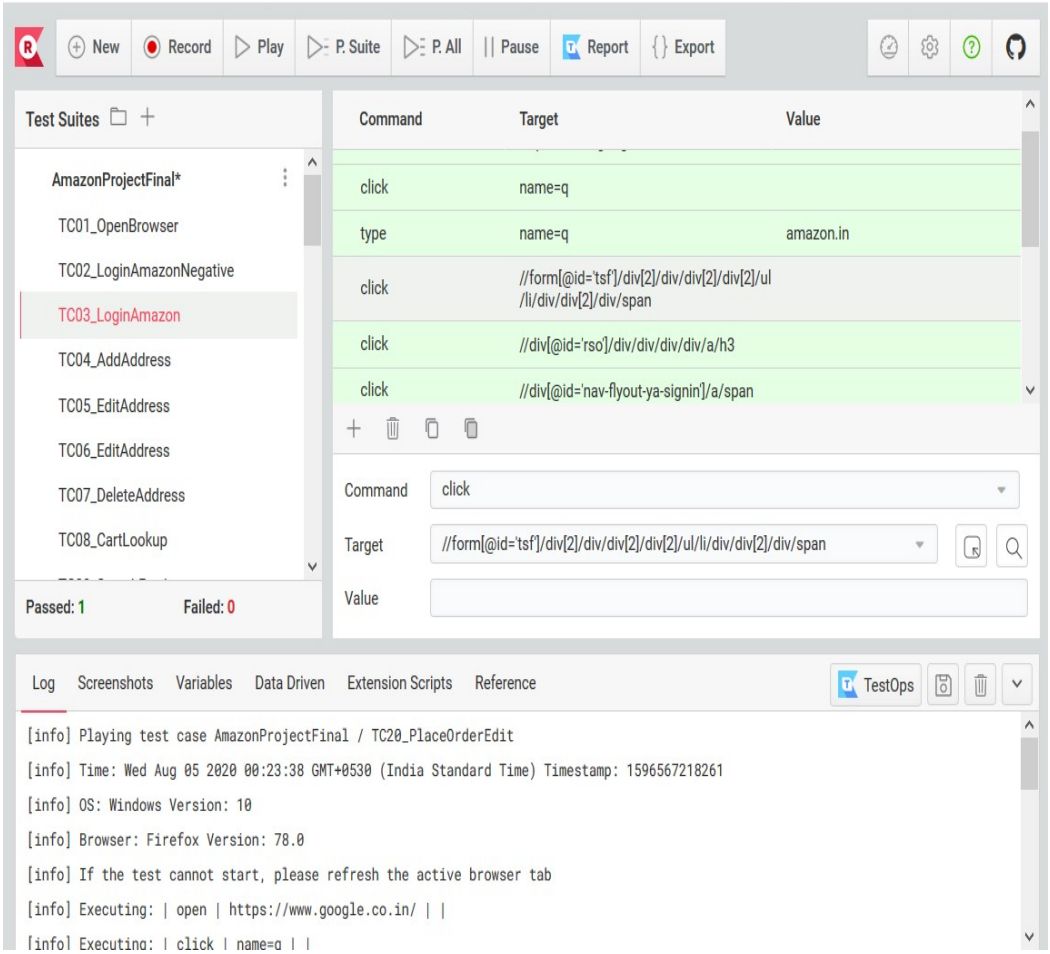


Figure 5.41 Automated script development for amazon.in using Katalon tool

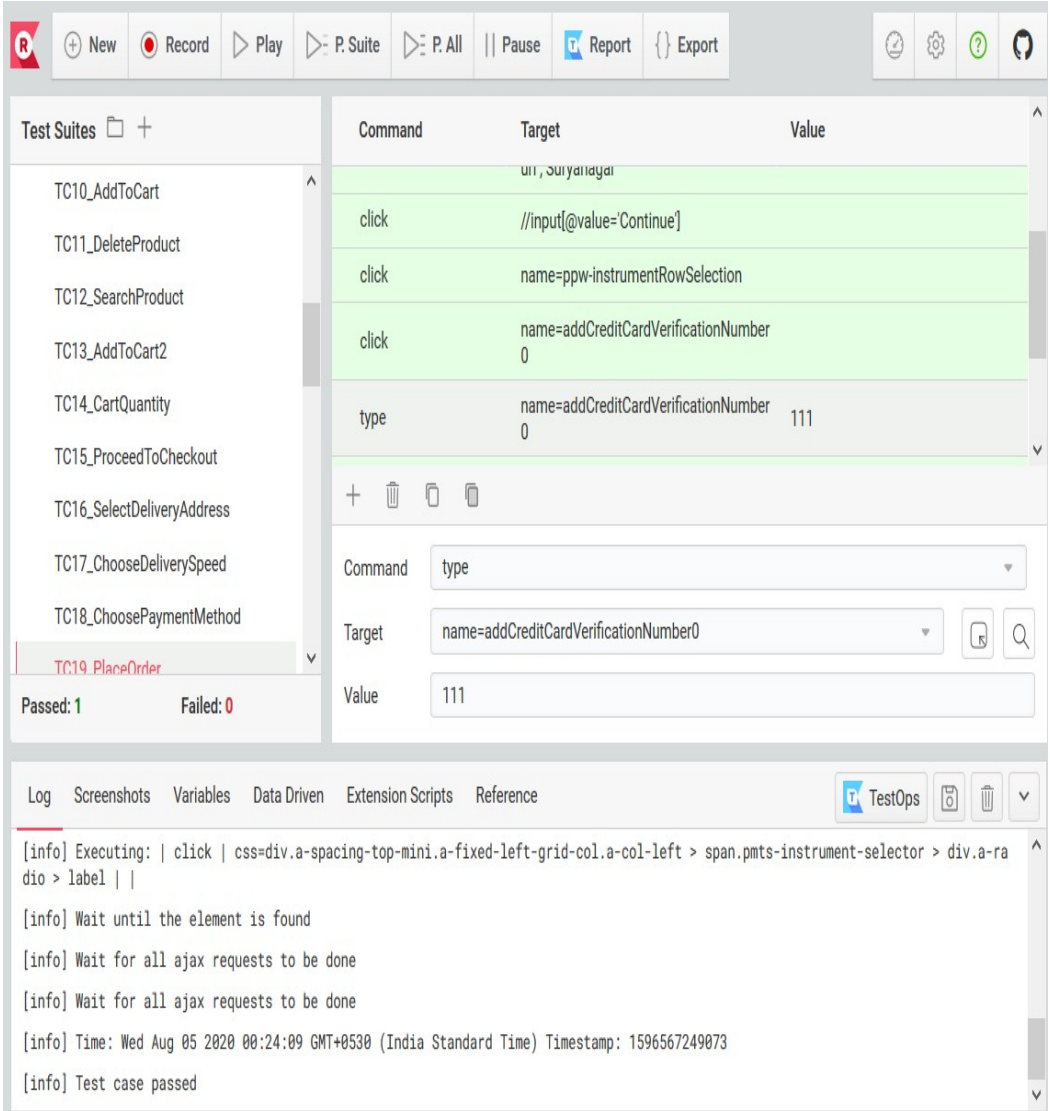


Figure 5.42 Automated script development for amazon.in using Katalon tool

We automated the complete buy flow consisting of 20 test cases for amazon.in and flipkart.com. Similarly, test suite of 20 test cases has been automated for flipkart.com as shown in Figure 5.42 and 5.43.

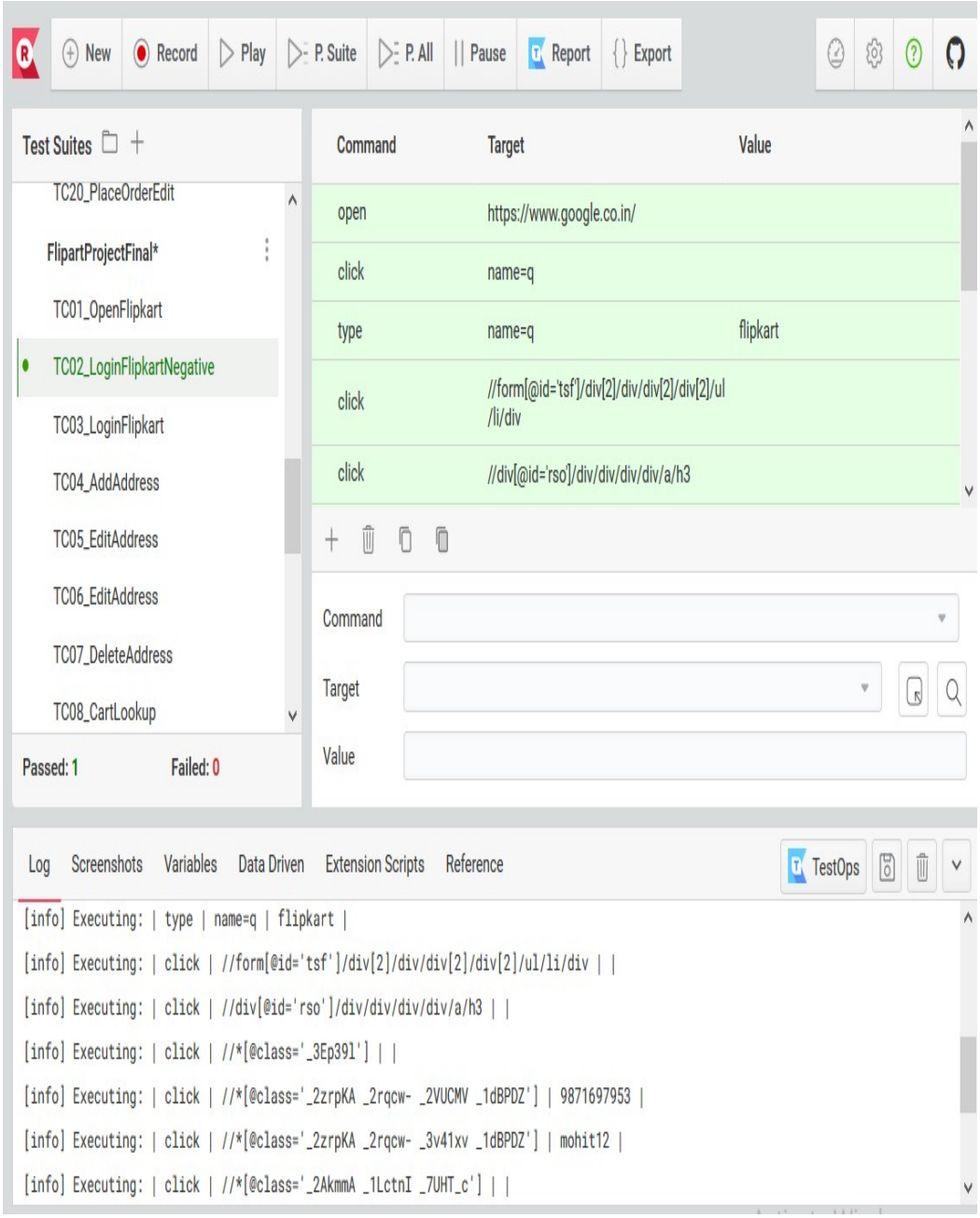


Figure 5.43 Automated script development for flipkart.com using Katalon tool

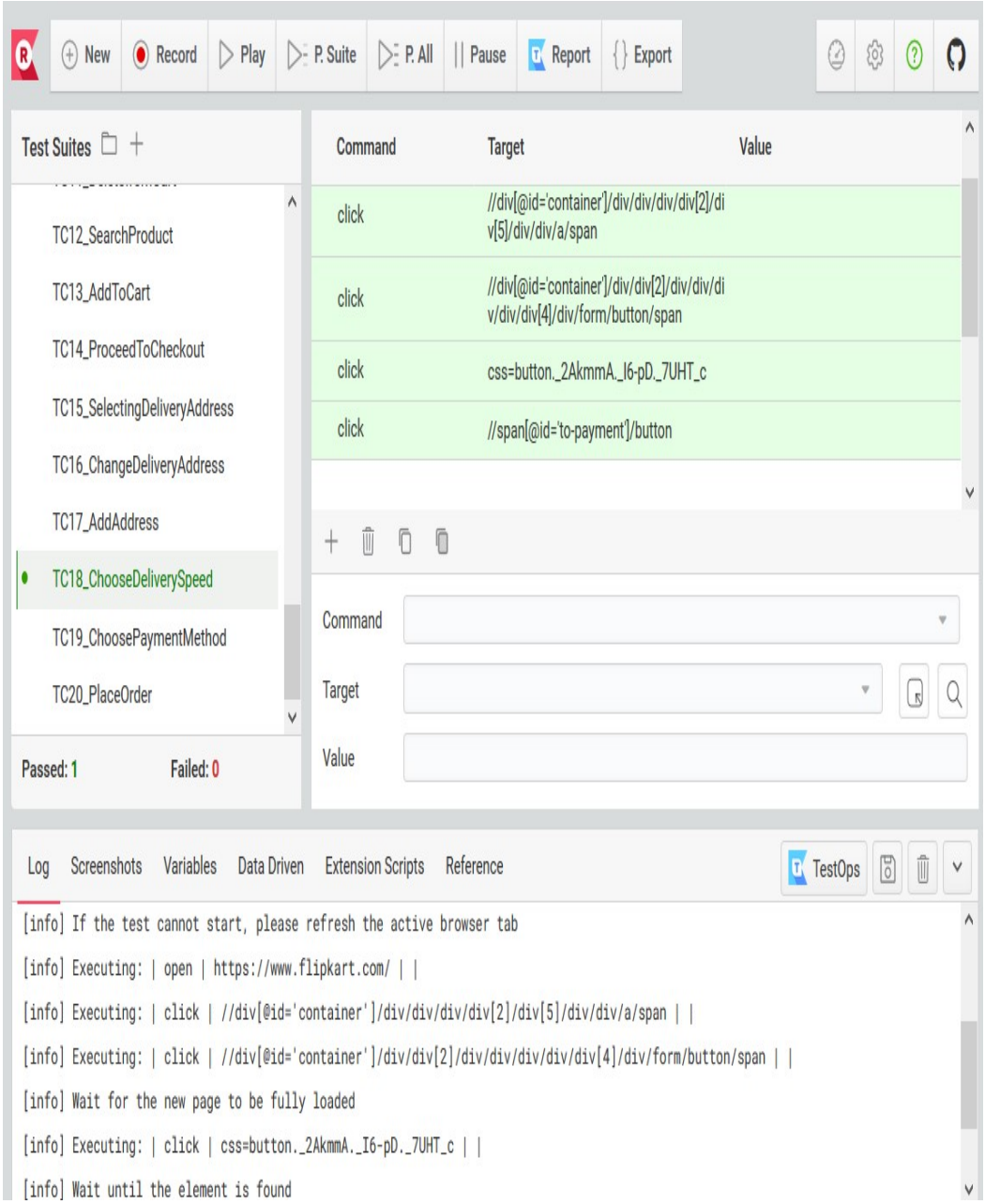


Figure 5.44 Automated script development for flipkart.com using Katalon tool

Test execution summary of katalon tool for both amazon.in and flipkart is shown in Figure 5.44 and Figure 5.45.

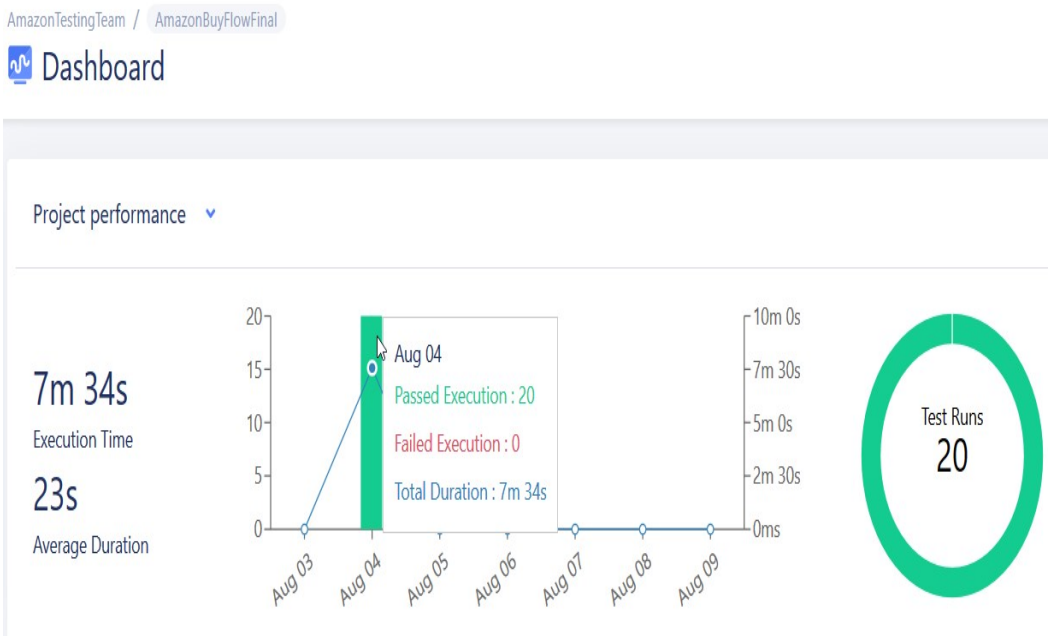


Figure 5.45 Test execution summary along with total test execution time for amazon.in

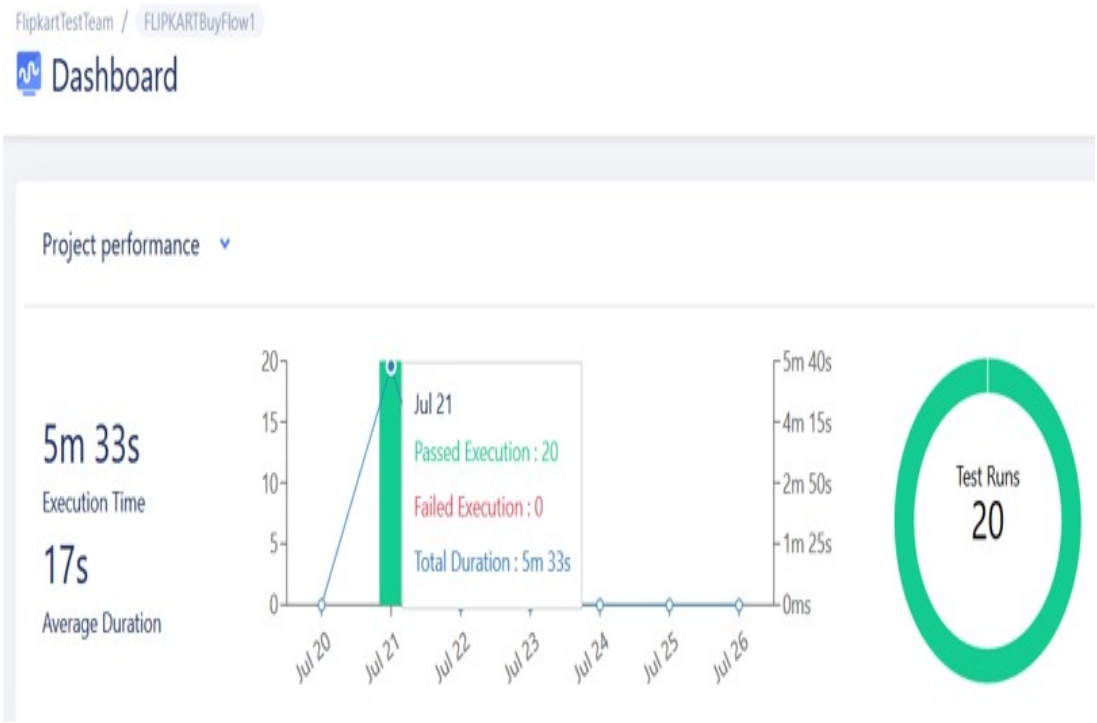


Figure 5.46 Test execution summary along with total test execution time for flipkart.in

5.5 Results

We have successfully implemented reusable hybrid test automation framework using page object model for web based applications in simulated scrum environment. Our experimental analysis of our RHTAF on amazon.in and flipkart.com showed that our regression testing efficiency is improved remarkably as we were able to run our regression test suite right from the beginning of Sprint 2 which is very much useful in the agile software development projects. Also by using page object model we were able to make this framework highly reusable and maintainable as we have used the same framework for two different e-commerce websites and also in section 5.3.1 we have applied our RHTAF on three other e-commerce web applications viz. snapdeal, myntra and shopclues to validate the reusability and maintainability of our framework. Tables 5.2 summarize the different metrics before and after applying our RHTAF on these three web applications. Moreover, Test execution time has also been reduced considerably using our RHTAF as compared to the automated test execution using Katalon recorder tool as shown in Table 5.3 which addressed our research objective 1.

Table 5.2 summary of different metrics values before and after applying our RHTAF on three web applications

E-commerce Website	Code coverage before using RHTAF	Code coverage after using RHTAF	Code duplication before using RHTAF	Code duplication after using RHTAF
Amazon	60%	90%	70%	10%
Flipkart	55%	85%	68%	13%
Snapdeal	55%	90%	67%	12%
Myntra	55%	88%	68%	10%
Shopclues	64%	86%	72%	10%

We saved around thrice the test execution time by automating our regression tests using RHTAF as compared to the manual test case execution and around twenty percent reduction has been achieved as compared to the automated test case execution using Katalon recorder.

Our Insprint test automation strategy proved to be very useful in applying RHTAF right from the Sprint 1 and we had the subset of regression suite ready that was executed before the beginning of the next sprint, this addressed our research objective 2. Moreover, we also integrated the result reporting facility in our framework with the help of which entire test summary is readily available after test case execution and also graphical representation of test result summary is also displayed to have a quick view on the pass/fail status of the test cases as shown in Figure 5.46 for amazon.in and flipkart.com.

Table 5.3 Performance comparison between automated test case execution using RHTAF on selenium and keyword driven framework on katalon

E-commerce Website	Total Time Taken (Manual test execution)	Total Time Taken (Automated test execution using Katalon)	Total Time Taken (Automated test execution using RHTAF)
Amazon	15m 36 sec	7m 34 sec	6m 8 sec
Flipkart	10m 10 sec	5m 33 sec	3m 42 sec

5.6 Discussion

Today most of the software industries, especially e-commerce industries and software companies with maintenance projects are migrating from bulkier development methods to lightweight agile methods. In agile methods test automation strategies are followed in order to improve quality of the product, reduce cycle time and for optimal utilization of manpower [184]. According to [167], technical debt accumulates if development team opts shortcut approach, hacks in quick fixes, skips writing

automating test and eventually code base gets harder and harder to maintain and this leads to the higher maintenance cost and lower team velocity.

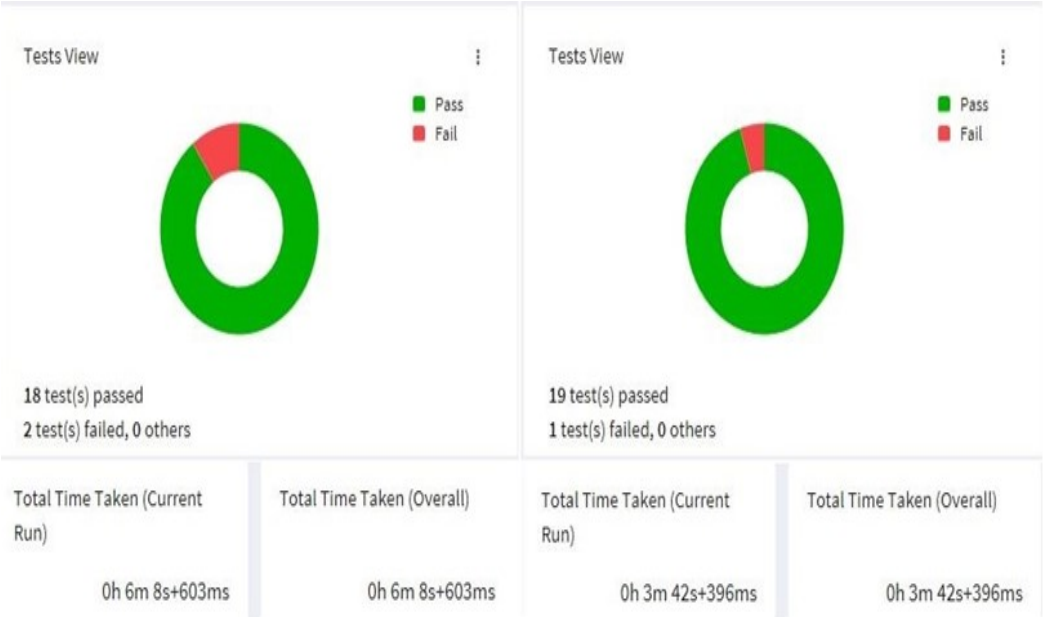


Figure 5.47 Graph showing test result summary along with total test execution time for amazon.in and flipkart.com

Tyagi et al. [180] reported the use of test automation frameworks in solving the test script maintenance problem for agile software development projects. Moreover, other studies [158,169] also depicted the importance of handling the test script maintenance and effective test automation.

Rafi et al. [175] discussed the superiority and benefits of test automation when several regression rounds are needed. A study [185] reported that creating automation testing framework using selenum webdriver adds value in the qulaity and reliability of the web applications. AutoJet tool is developed as an extension to the selnium webdriver for automating regression tests for web applications [186]. This tool is used in integration with TestNG and cucumber for writing tests cases in Behavioural Driven Development (BDD) format [186] . However, in contrast to this framework out RHTAF offers maintainability and resuability featurs which makes it an effictent test automation framewrok for web based scrum projects.

CHAPTER 6

EMPIRICALLY DEVELOPED FRAMEWORK FOR BUILDING TRUST IN DISTRIBUTED AGILE TEAMS

6.1 Introduction

Teamwork quality is one of the crucial ingredients for successful software projects [187]. Hoegl and Germuenden [188] studied the impact of team work quality on the software quality and identified six teamwork quality (TWQ) factors: coordination, communication, mutual support, member contribution, effort and cohesion that influence the software quality. Weimar et al. [189] further extended the teamwork quality model and proposed three additional factors viz. coordination of expertise, value sharing and trust that contribute to high quality team performance. Trust is one of the key factors that influences team performance and acts as a foundation for effective teamwork [190].

The primary driving force behind the huge popularity and sustainability of agile software development methodology is the preference given to human and social factors like interpersonal relationships among the team members during project development as traditional software development methods [191]. According to VersionOne survey [192], many organizations are scaling agile beyond one project, one location and one country. Organizations are embracing agile in distributed environment due to various factors like reduced costs, faster time to market and managing late requirement changes [193]. Executing agile development project in distributed set up has its own set of challenges pertaining to cultural differences, spatial and temporal distances, coordination and communication related issues among team members [194, 195]. In agile, teamwork, collaboration and self organization are the key attributes of the agile teams and trust acts as the foundation for building all these attributes [196]. Presence of trust is vital between the leader and the team, and also among the team members for effective team performance [196].

In distributed environment, agile team members are distributed across multiple cities within the same or different countries. Different cultural backgrounds, ethnicities and

time differences add on to the geographical distances for weakening the cohesiveness of agile teams.

The importance of trust in team effectiveness and project success has been highlighted by numerous studies [197, 198]. Trust is one of the key factors that influences team performance and acts as a foundation for effective teamwork [189, 190]. Even though, it is well known that trust plays a critical role particularly in distributed agile teams, little research has been done to develop a comprehensive model or theory on how to build trust in agile teams in a distributed environment. Badreddin [199] reported that role of trust in distributed agile teams has not been formalized in theory and highlighted the opportunity in taking up empirical study to investigate the role of trust in distributed agile development projects. Early trust research measured a single trust characteristic [200, 201] while more recent research argues for a more complex, multidimensional structure [202-204]. A popular conceptualization proposed by Mayer et al. [205] explores trust consisting of three dimensions: ability, benevolence, and integrity. Majority of the research that investigated the role of trust in distributed teams reported it from the socio cultural perspective [86] [206]. But we understand that trust is a multidimensional concept and should be investigated from a multidimensional perspective. To address this gap, we undertook this study to increase the understanding of different factors that help to build trust in distributed agile teams from a multidimensional perspective. To provide an empirical insight in this area, a Grounded Theory (GT) study has been conducted that involved in depth interviews of 40 agile practitioner's from 19 different software organizations distributed across India, UK, Netherlands, Germany, USA, Australia, and New Zealand. The interview data is further strengthened by the observation of two different agile development projects distributed across India and Germany.

The contributions of this chapter are as follows:

1. This chapter provides a comprehensive framework to build trust in distributed agile teams. This framework has been empirically developed using Grounded Theory (GT) research method. This framework consists of factors divided across five categories which denote the multidimensional perspective of trust.

2. Further, the trust factors across five different categories were mapped to the dimensions of the ability, benevolence and integrity of Mayer's model of

organizational trust. This mapping explained the impact on these dimensions of trust by our trust factors.

The rest of the chapter is structured as follows: next section presents a background on trust and distributed agile software development. The third section provides overview of our research methodology. The fourth section describes the findings of this study; the fifth section discusses these findings, sixth section evaluates our GT findings.

6.2 Background and related work

6.2.1 Trust

Trust is both an emotional and logical act. Emotionally, it is where you expose your vulnerabilities to people, believing they will not take advantage of your openness. Logically, it is where you have assessed the probabilities of gain and loss, calculating expected utility based on hard performance data, and concluded that the person in question will behave in a predictable manner. According to [207] trust can be seen from interpersonal and collective aspects and can be expressed at three levels within an organization: individual, teams and organizational. According to [205] trust is defined as the "willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to trustor, irrespective of the ability to monitor or control that other party". Here, definition of trust implies both an individual and relational component of trust.

Team interaction histories provide rich insights on intentions and motives of others that attributed towards studying trust as a collective phenomenon [208]. Costa and Anderson [208] defined trust between team members as a latent construct based on the individual's own propensity to trust others and on the perceived trustworthiness of the other team members. Oza et al. [209] and Babar et al. [210] have investigated factors necessary to build and maintain trust in software outsourcing companies. It is apparent from above research studies that trust plays a very crucial role in team based organizations.

6.2.2 Significance of trust in Distributed Agile Teams

When software development teams are distributed across different cities, different countries or different continents then they are termed as Distributed Software Development (DSD) teams. Global Software Development (GSD) is often used interchangeably for DSD. Today, many software organizations are using agile methods in DSD set up to take the benefits of both of these approaches [211]. Agile teams face numerous challenges pertaining to collaboration, face to face communication, and team bonding during daily stand up, sprint retrospective, and sprint planning meetings in distributed environment [212].

Collaboration and face to face communication are two biggest problems when agile methods are adopted in GSD set up [213]. Vax et al. [214] reported challenges of DSD that are related to face to face communication, language barrier, cultural difference and lack of trust.

According to [215] distributed agile teams face many challenges that slow down software development processes. According to [216] major difficulties when agile practices are used in global software engineering projects are related to communication, trust, culture and different time zones. A study [217] identifies trust as a key factor in developing cohesiveness and collaboration in agile teams in a distributed set up. Korkala et. al. [218] conducted a single case study in a large global distributed software company using agile development methods for two years. This study revealed that since customer was not involved in implementation of the project due to lack of trust so this impacted information sharing amongst distributed team members. Another significant work has been done in a case study conducted in large multinational financial services organization [219]. They reported the role of agile practices like daily stand up, sprint planning meeting and sprint retrospective meeting in building trust among agile team members in distributed set up [219].

Hole and Moe [220] reported that trust is important in distributed agile software development projects to reduce the need for standardization and direct supervision. Ramesh et al. [221] described the need of trust in forming high performing and cohesive teams in distributed agile environments. Trust is seen as a key enabler in building cohesive agile teams in globally distributed environment. Passivaara and Lassenius [222] performed multiple case study on four distributed scrum projects.

They also reported different benefits of frequent visits and multiples communication modes in developing trust amongst distributed team members.

Lee et al. [223] described difficulty in fostering team bonding and collaboration in distributed agile development environment. In distributed agile set up, teams are distributed at geographically distant location with little or no face to face interaction between the team members. Wildt and Prikladnik [224] presented their experience of a global multinational company while transitioning from distributed and traditional to distributed and agile environment. Authors have recommended that in order to have a smooth transition to distributed agile environment, scrum master must know how to deal with the distance, trust and cultural diversity issues. Moe et al. [225] reported challenges of virtual teams related to lack of communication, time zone difference and difficulty in building and maintaining trust. Distributed team members also experienced problems related to knowledge sharing.

A study by Dorairaj et al. [226] investigated the role of trust in developing effective communication among team members in distributed agile environment. Another study by Dorairaj et al. [126] reported a role of trust in promoting team interaction between distributed agile teams. Hoda et al. [227] discussed the role of organizational culture in building the environment of trust between agile team members and senior management in order to facilitate them to become self organizing. Trust is key factor in distributed agile teams, trust is crucial in determining the success or failure of distributed agile projects [206]. Dorairaj et al. [228] argue that trust is crucial for blending agility with distributed software development. Trust amongst distributed team members is important to bridge spatial, temporal and socio cultural distances. Trust fuels team performance and contribute in building an effective and cohesive team.

The motivation behind this research study is to develop a comprehensive framework to build trust in distributed agile teams.

6.3 Research Method

The proposed framework has been developed empirically using Grounded Theory (GT) research method.

6.3.1 Data Analysis

6.3.1.1 Phase 1: Identifying the Core Category

Theoretical sampling approach is used by researchers to find the data sites and samples in grounded theory studies [145]. However, there has to be a starting point for the data collection process in GT. Purposive sampling is used in order to find the initial participants based on certain criteria [149]. Thereafter, theoretical sampling process guides us in selecting further interview sites and samples [149].

The initial selection of participants using purposive sampling was based on the following criteria: 1) Participant should be working in distributed agile team. 2) Participant working experience on agile development projects should be more than one year. 3) Total IT industry experience of participant should be more than three years. Our data collection process embarked with eight interviews from three different software organizations on our general area of interest "Teamwork Quality" in distributed agile teams [146]. Initial interview questions revolved around participant's experience of working on agile development project, their roles and responsibilities, different agile practices they follow in order to deliver quality software within time and different challenges associated with the agile development approach.

All the interviews were audio recorded and then transcribed for the analysis. Two levels of substantive coding: Open coding and selective coding was performed on the data to generate codes, concepts and categories as per the Glaserian GT [164].

6.3.1.2 Open Coding and Memo Writing

In open coding interview transcripts are being broken down into discrete parts and then analyzed in detail for any incidents that explains some phenomenon. Several incidents were generated from the coding of the interviews, incidents that occurred more than thrice in an interview have been represented with the help of word cloud in Figure 6.1, trust emerged as the main concern of the participants working in the distributed agile set up as shown in the Figure 6.1. Those incidents that represent the key points from each segment of the interview transcript are put together and particular code is assigned to them [145] [125]. Code is a "phrase used to summarize the key point in 2 to 3 words" [144]. The constant comparison of codes from one interview with the codes generated from other interviews and then grouping of similar codes leads to the second level of abstraction, called concepts. Further, this method is

repeated on concepts to produce the third level of abstraction, called categories. Table 6.1 shows an example of open coding process with the help of working interview transcripts and field notes.

Figure 6.1 Incidents emerged from coding of interview transcripts

Memos were written as and when they emerged during the coding process. Our coding process fetched multiple codes and memos which acted as inputs for our subsequent interviews, due to which our interview guide was updated constantly. The process of comparing incident to incident and code to code is repeated on the same as well as with other interviews and also with the codes generated from the field notes of the observations.

6.3.1.3 Selecting the Core Category

Open coding process ends when core category has been identified. Main attribute of the core category is that "it accounts for the large portion of variation in a pattern of behavior".

Without a core category, there is no grounded theory [136]. Main criteria for choosing the core category are: core category must be central; it must relate with other categories and their properties; it must occur frequently in data.

Table 6.1 Example of open coding process

Open Coding	Interview Quotation – P10, Scrum Master	Observation (Org.: Y, Project: Delta)
Statement / Field Note	"Let me tell you my experience with my new team. It's like newly formed divorce team...Has team members from different continents. Team lacks on trust obviously as it's a new team .Lack of commitment, avoidance of accountability were evident that eventually hampering the performance"	Team members distributed across three locations, two in India, one in US, Distributed team members joins daily stand up via phone call sometimes they miss daily stand ups. There was a disconnect between Indian team members and US team members, Information sharing was not good and smooth. Trust related issues were observed ,
Key Point	Disconnect between team members of newly formed team across multiple locations, Lack of trust between team member lead to accountability and performance related issues.	Distributed team members have disconnect among team members that leads to poor information sharing and lack of trust.
Code	Lack of trust, accountability issues, performance issues	Lack of trust, disconnected team members, poor information sharing
Concept	Need to build trust among team members	Need to build trust among team members
Category	Building trust	

We ended open coding on identifying our core category "Building trust" in distributed agile teams. "Effective communication" also emerged as one of the potential near core category, but we selected "Building trust" in distributed agile teams as our category as it was related to the other category in a meaningful way. Furthermore, our selection of core category tends to be fit as we were able to integrate the theory around it. Figure 6.2 shows the emergence of our core category "Building Trust" in distributed agile teams from the underlying concepts.

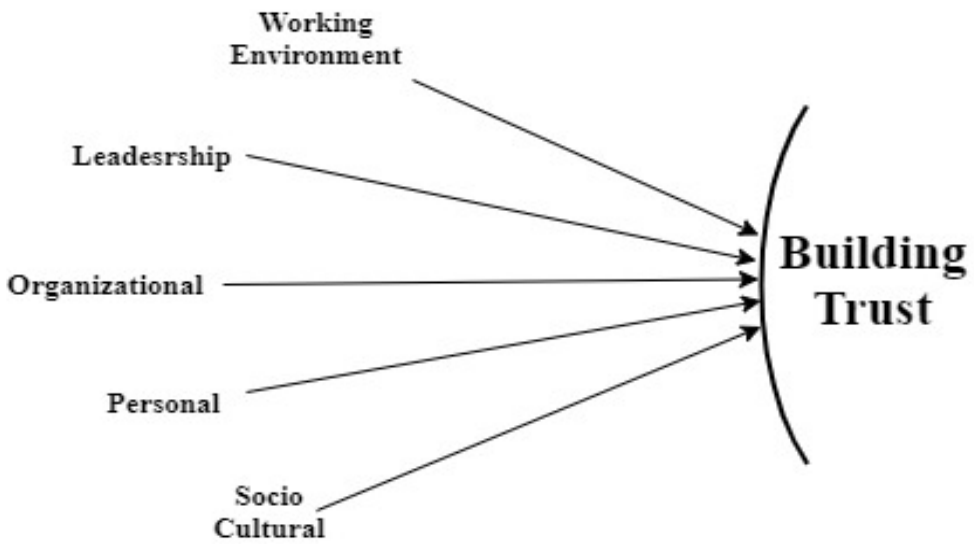


Figure 6.2 Emergence of core category building trust from underlying concepts

6.3.1.4 Phase 2: Refining the Core Category

Theoretical sampling process of GT continued in phase 2 of our study and guided us in selecting our new sites and interviewees. Based on the analysis of data from the phase 1 of our study, we decided to bring diversity in our data in terms of project roles, project domains and project distribution [148]. Domain of our sample was expanded to include other job roles like senior agile coaches, people from senior management people like chief technical officer, vice president and directors from organizations with diverse domains like telecom, insurance, and healthcare. Phase 2 of our study involved 29 further interviews from 14 different software organizations. Through theoretical sampling and constant comparative method we ensured that agile practitioners with different roles and wide range of domains were covered.

6.3.1.5 Selective Coding and Memo Writing

The objective of selective coding is to identify a key category that can be used as a fulcrum of the study results [164]. Therefore, only those interview transcripts were coded that were related to our core category i.e. “Building Trust” in distributed agile teams. Constant comparative method was used on interview transcripts and observations to find out codes, concepts and finally the categories related to our core. Table 6.2 shows an example of selective coding process.

Table 6.2 Example of selective coding process

Selective Coding	Interview Quotation – P6, Tester	Observation (Org.: X, Project: Sigma)
Statement / Field Note	"We make an effort to have one face to face interaction with our geographically distributed members at least in 3 months so that we get to know each other better by working and socializing with each other, this is the best way to build trust between team members"	Three QA were at client side in US. They travel to offshore in India every six months for the kick off planning of next release. This allows the offshore team to develop that bond, cultural understanding and trust with the onsite team members when they physically work with them.
Key Point	Socializing with distributed team members builds trust among team members.	Socialization of onsite and offshore team members enhance cultural understanding and trust with each other.
Code	Importance of socialization in building trust	Importance of socialization in building trust
Concept	Trust building through socialization	Trust building through socialization
Category	Socio Cultural	

6.3.1.6 Reaching Theoretical Saturation

One of the important aspects in GT is to reach closure on theoretical sampling process. Ideally, it should be stopped on reaching theoretical saturation that is when further data collection yields no new data or concepts [125].

On reaching a stage where further data collection and its analysis were leading us to the same categories with no new data, we found out that our categories have reached saturation.

6.3.1.7 Phase 3: Theory Write Up

Once the data collection is finished and all the categories have reached saturation then researcher can conceptually sort the theoretical memos. Sorting of the memos constitutes the theoretical outline around the emerging theory [145]. Ideas should be sorted not the data while sorting memos. We sorted theoretical memos conceptually based on the properties of the underlying categories, similarities and connections between the ideas. Theoretical coding was performed at this stage as suggested by Glaser [145]. Theoretical coding involves conceptualizing how the categories and their properties relate to each other as a hypothesis to be integrated into a theory. Glaser proposed several common structures of theories know as theoretical coding families [155].

On comparing our data with theoretical coding families, “Model” family was best suited for our data than rest of the coding families [155]. Our research study can be used as a "Framework" similar to a "Model" for building trust in distributed agile teams. This framework can be utilized by the organizations and senior management to build and foster the environment of trust in teams working in a distributed set up.

Our theory represents the key contributing factors for building trust in distributed agile teams. Figure 6.3 presents our empirically developed framework for "Building Trust" in distributed agile teams. In the following section, we present the research findings from our study. Selected interview quotations are provided under each category to better explain it in the present context.

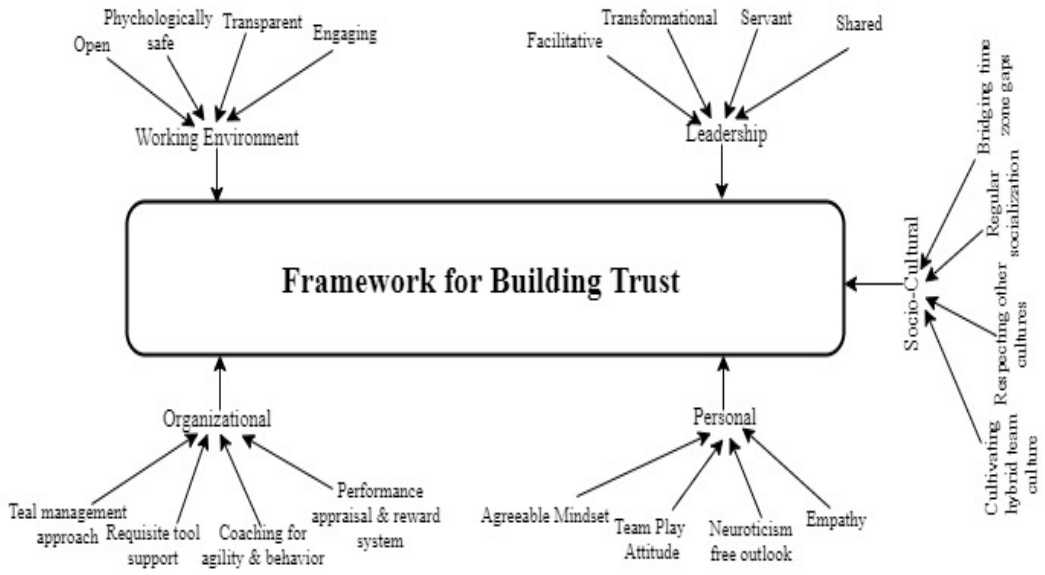


Figure 6.3 Our grounded theory model for building trust in distributed agile teams

6.4 Results

In this section, we present the key results of our study. The following sections describe trust building factors in detail. Selected interview quotations are included in description so that better understanding can be developed for that category.

6.4.1 Working Environment

Working environment factors correspond to the working conditions and general operating mechanisms that include communication, coordination and collaboration amongst distributed agile team members. Table 6.3 shows the emergence of concepts from the underlying codes that give rise to the working environment category.

6.4.1.1 Concept - Open working environment

Majority of agile practitioner's have mentioned that changing their workspace from cubicle types to the open work area proved to be very advantageous for them. In large open workspace, agile team members work collaboratively and feel connected with each other. They use smart agile boards, wikis and virtual team walls to communicate,

share and visualize the tasks with their distributed counterparts. This promotes a feeling of one single team and builds the foundation of trust between them.

"We have wide open room without any cubicles, entire team sits together in one room, we did not build open collaborative workspace, what we build is open collaborative culture and we fit our workspace to a culture so what you seen that a workplace we been, is a physical space that is a reflection of openness, transparency and the relationship we want to build between people in our team and eventually becomes the foundation for trust between our team members".– O26, Quality Analyst.

6.4.1.2 Concept - Psychologically safe working environment

Agile software development teams are autonomous and self managing. Autonomy gives freedom to members to independently raise concerns and ask doubts from the product owner and senior management. Team members should not deter from taking decisions and innovating in fear of doing mistakes. They should feel safe in approaching their fellow team members for discussions at any location without fear of negative social evaluation.

"Building psychologically safe agile workspace is very important, your team members should feel that they will not be criticized or questioned for sharing their ideas or concerns with each other or with senior management. It can improve the trust within agile teams tremendously". – O27, Agile Coach

6.4.1.3 Concept - Transparent working environment

Majority of interviewed participants mentioned transparent working environment as one of the key enablers for building trust among agile team members. Sharing information amongst distributed team members through mediums like online scrum and discussion boards brings in transparency about the progress of the project.

"Visibility is the most powerful feedback loop, people need recognition not judgment, Its always better to have a big scrum board in the center of your work area so that entire team can track on the task status, this also contributes in invoking that culture of collaboration and trust within your team".– O20, Scrum Master

6.4.1.4 Concept - Engaging working environment

Another concept mentioned by many agile practitioners is to develop engaging work environment for the distributed agile teams. In computer-mediated teams, members have a feeling of less connectedness because of lack of face to face interactions. An engaging working environment can be created through frequent interactions. Engaging working environment helps team members to collectively work towards shared goals.

"*Team member's feel disconnected when they don't get clarity on product vision, they cannot strategize the product development road map unless they have the continuous discussion with PO on the product features. Engaging work environment brings the sense of enjoyment of work amongst distributed team members*" O15 – Senior Quality Analyst

When every member of a distributed agile team is given clear authority and responsibility, they feel more engaged. Engaging working environment helps team members collectively to coordinate their efforts and builds trust amongst them.

Table 6.3 Emergence of concepts from the codes for working environment category

Codes	Concepts
Building open collaborative culture, helps in bringing openness, know your team, building rapport, open work climate	Open working environment
No fear for asking questions, no fear for raising concerns, creating safe environment, approachability of senior management	Psychologically safe working environment
Make work visible to team members, visibility of work status, information radiators, on-line task sharing.	Transparent working environment
Performance cannot be managed, manage work environment, help people grow better, engage people	Engaging working environment

6.4.2 Leadership

Leadership factors imply the factors related to leadership that help in building trust amongst team members working in distributed agile software development.

Leadership role is very crucial in agile software development as it is the leader who empowers the team, who facilitates the team during any impediment, helps team as a servant leader in solving problems and in decision making. Table 6.4 shows the emergence of concepts from the codes for leadership category.

6.4.2.1 Concept - Facilitative leadership

Many agile practitioners mentioned that the commonality of purpose amongst distributed team members enhance the trust levels. Facilitative leader creates shared vision amongst distributed team members where they work towards common strategic goals. Facilitative leadership is a predictor of team reflexivity which improves trust amongst distributed team members which in turn attributes towards high team performance.

"*Leadership plays a very significant role in aligning the distributed agile teams in the same direction. Leader must facilitate in getting each team member know that they should work together for the common goals. He is the one who takes initiative in orienting distributed team members towards the shared vision.*" O16 – Vice President

6.4.2.2 Concept - Transformational leadership

Another important aspect of leadership is to make distributed teams self managing. Transformational leader empowers distributed agile team members to be self motivated and self driven. Distributed team members should be encouraged to self allocate the tasks during sprint planning meetings. They should be allowed to make their decisions and take full ownership of the work they are doing.

"*When you have team working from multiple locations, giving them the autonomy for work is a very crucial thing. This form of leadership [transformational] empowers the team to work together in a united manner.*" O17 – Scrum Master

6.4.2.2 Concept - Servant leadership

In agile development teams, leadership has a very pivotal role, there is no scope of bossism and micro management. All team members should be treated as equal regardless of their positional authority. Leadership should be more of a servant leadership type that reflects the attitude of self less service for the team. Servant

leadership builds trust among the distributed team members and they work without any fear or restriction.

"There was a member of the executive team [one of our clients] that if something went wrong he want to find somebody to blame. Sometimes my team made mistakes. one of my teams made a really critical mistake. So this person wanted to find who he could blame. so what I did to to be honest, it was an interesting technology decision but it turned out to be bad technology decision that really hurt the project, now I am not that smart, I had no idea inwhat they were experimenting with in terms of technology but when he tend to blame somebody i told this guy hey listen get off, it was my decision, I am the one who decided to do this because I wanted to shield them from blame because if there is a blame there is no trust". – O7, CTO

Table 6.4 Emergence of concepts from the codes for leadership category

Codes	Concepts
Boss less teams; facilitating the team; involving team in decision making.	Facilitative leadership
Leader fostering trust in team; allowing self-allocation of tasks; team taking ownership of tasks; empowering the team	Transformational leadership
Becoming servant leader; care for your team; allow team to experiment and make mistakes; command and control less leadership	Servant leadership
Team members assist each other, team members lead each other towards the common goal, more proactiveness from the team, shared responsibility	Shared leadership

6.4.2.4 Concept - Shared leadership

To create high performing distributed agile team that sail high on trust, sharing the leadership amongst team members is very crucial. Particularly if team members are located across different geographical locations, then they must trust each other's knowledge and competencies. Shared leadership enables sharing of decision making authority amongst distributed agile teams.

"As an agile leader, your job is to help people grow at the personal level and professional level. To build trust, leader needs to develop that sense of leadership among team members which encourages them to share the decisions amongst them on the basis of their skills and knowledge."– O3, Scrum Master

6.4.3 Organizational

Organizational factors correspond to facilities, infrastructure and agile friendly culture of an organization. Organizational support is essential in order to create an environment of trust amongst distributed agile team members.

Table 6.5 Emergence of concepts from the codes for organizational category

Codes	Concepts
Flat organizational structure, support for inspect and adapt principle, encouraging self managing teams, encouraging self organizing teams, deliver value quickly	Teal management approach
Virtual team collaboration tool support, agile project management tool support, video conferencing and high speed internet support	Requisite tool support
Conducting online agile trainings, coach teams on agile principles and values, develop agile culture, providing budget to attend agile conferences	Coaching for agility and behavior
Team based appraisal system, reward the team, using short assessment cycles, using 360 degree feedback	Performance appraisal and reward system

Table 6.5 shows the emergence of concepts from the codes for organizational category.

6.4.3.1 Concept – Teal management approach

Majority of participants consider organizational structure and management style as one of the key enablers in building trust in agile teams. Organization should follow flat structure rather than hierarchical one that supports the distributed agile teams by

allowing them to try out new things, provide them sufficient time to adapt to agile and to learn from their mistakes.

"Organizational support in terms of letting the team experiment and try new things while doing agile, this kind of support really helps in improving trust amongst agile teams and gives them the confidence to do something out of the box. Moreover practicing teal organizational structure is one of the key enabler for trust in distributed agile teams."– O17, Scrum Master

Teal based management approach is highly recommended by agile practitioners in distributed set up. In Teal organizations, hierarchical organizational structure is replaced by flat decentralized structure which is powered by a group of self managing and self organizing teams. This enhances trust in distributed agile team members significantly as there is no status difference among the team members. They can participate in decision making freely and feel more empowered.

6.4.3.2 Concept – Requisite tool support

As distributed agile teams are dispersed across multiple locations, it is very important to have a communication facility through which team members can connect with each other. Also, proper Application Lifecycle Management (ALM) tools are required for effective tracking and management of projects.

"Organizational support is really crucial because if your team would lack on the basic infrastructure facilities like video conferencing facility or good internet access or good project management tools, then it would be very challenging to build the trust especially when you have your team members located at different parts of the world". – O24, Agile Consultant

6.4.3.3 Concept – Coaching for agility and behavior

To make people agile aware is of fundamental importance in distributed agile teams. As team members work across different location their knowledge on fundamental agile concepts, values and principles should be very well articulated and aligned. Agile teams should know what real agile is and how to practice it.

"*Agility comes with right coaching and training, when team members work across different locations they should understand the importance of all the agile events, and*

values. They should know the difference between doing agile and being agile. Organizations must organize online agile training's, webinars and encourage team members to attend meet ups and conferences on agile." – O14, Senior Tester

6.4.3.3 Concept – Performance appraisal and reward system

Many participants also mentioned that organization perspective on performance appraisal and their zeal towards employee satisfaction plays a major role in building trust in distributed teams. Rewarding a team oriented behavior leads to collective interests towards accomplishing shared goals.

"One of the organization goals is to take employees satisfaction on highest priority through early and continuous feedback on their performance which is aligned to agile values (Agile Manifesto and Principles). Early and continuous performance assessment cannot be achieved without adhering to simplicity principle. Keep the assessment/feedback mechanism light weight. Using simple star rating or voting techniques for feedback (wherever possible). Doing face to face review discussions over skype for setting objectives rather than doing electronically."– O12, Agile Coach

6.4.4 Personal Factors

Personal factors correspond to self motivated attitude, task ownership, work ethics and sense of belongingness of each member of the agile team. Table 6.6 shows the emergence of concepts from the codes for personal category.

Table 6.6 Emergence of concepts from the codes for personal category

Codes	Concepts
Global mindset, growth mindset, openness, working for common goals, ability to handle ambiguity	Agreeable mindset
Team first attitude, ignore personal interests, display team play, share information within the team, share knowledge, help other team members	Team play attitude
Emotional stability, positive outlook, calm and compose attitude, importance of team achievement, building friendly rapport	Neuroticism free outlook
Sense of belonging, sense of ownership, understanding others, able to initiate communication, providing support and care to others, feeling pain of others	Empathy

6.4.4.1 Concept – Agreeable mindset

Team members in distributed agile teams need to have an agreeable mindset. Agreeable individuals are those who are considerate, trusting, and friendly. In a distributed set up every team member's decision and opinion holds same value irrespective of the location. Agreeable mindset is essential for a global mindset where they need to think from a wider perspective.

"People need alignment, some team members do not agree on some issues, there are disagreements. You need clarity of work and understand the importance of working towards common goal. We practice it by setting achievable sprint goals and then get all the team members together over zoom or Skype at least once daily. Team members must develop growth mindset and should be open for any kind of change." – O13, Tester

6.4.4.2 Concept – Team play attitude

Many participants find team play as one of the major boosters for motivating distributed agile teams and building trust amongst them. Team play is above

teamwork, it's about going up the level to help other team members, and it's about showing compassion and sense of belonging to your team.

"*I look to push myself before pushing anybody else in my team; I need to do that first for [team] to show that energy, especially when you are working far from your team mates, you need to create that feeling that you are there for any kind of problem or issue. It's about display your team play, putting your team before your personal interests.*" – O15, Scrum Master

6.4.4.3 Concept – Neuroticism free outlook

Agile development projects mainly have short release cycles, frequent changes, and continuous testing. Agile team members need to have an emotionally stable mindset in order to avoid being anxious, impatient or being upset if projects do not meet deadlines.

"*At times delivered functionality does not turn out as expected, sometimes customers feedback is not positive but Product Owner and team members should stay calm and should not get anxious*." – O19, Senior Agile Coach

Neuroticism free outlook gives positivity and enhances the emotional quotient of distributed agile team members. Many agile practitioners mentioned that it's very common to lose your cool and going too hard on yourself when things are not getting the way they should. In distributed settings where most of the team members are separated geographically it's important to develop positive outlook so that positive energy and vibes flow through the team.

6.4.4.4 Concept – Empathy

Empathy is one of the most important traits of agile teams in a distributed environment as team members do not meet each other frequently. Team members develop bonding over a period of time. They need to be empathetic towards each other and be able to understand the concerns of each other.

"*It is not necessary that every time a fellow team member is doing the task with the same speed as you do. He may be undergoing some problem. Being empathetic is one of the most important traits when you work in a distributed environment where team*

members are at different locations and moreover this also builds trust." – O22, Agile Coach

Table 6.7 Emergence of concepts from the codes for socio-cultural category

Codes	Concepts
Team travels to offshore, in person meeting once per release, personal connect amongst team members, whole team feeling	Cultivating hybrid team culture
Different team members belong to different cultures; respect for all cultures, respect for all religions, equality amongst team members	Respecting other cultures
Arranging virtual coffee on calls, playing on-line team games, introductory sessions on web, encourage team bonding	Regular socialization
Different time zones, clashes of time zones, conduct meetings at suitable time, share the time zones, share the pain	Bridging time zone gaps

6.4.5 Socio-Cultural

Socio cultural factors correspond to challenges due to regional, social and cultural differences among agile team members located across different locations. Table 6.7 shows the emergence of concepts from the codes for socio cultural category.

6.3.5.1 Concept – Cultivating hybrid team culture

Feeling of working in a one single team despite being located at geographically distant location helps agile teams in building trust as mentioned by many agile practitioners. People develop a personal touch and exhibit familiarity when they meet with their distant team members onsite or offshore.

"We make an effort to have one face to face interaction with our geographically distant located members at least once in 3-4 months so that we can socialize with

each other, this gives the team members the feeling of working in one single team. This is the best way to improve trust between our teams." – O19, Senior Agile Coach

A whole team approach melds cultural differences to build a hybrid team culture.

6.4.5.2 Concept – Respecting other cultures

To develop trust with distributed team members needs patience and time. Every team member should respect the culture of fellow team members. This develops when team members knows the background, personality and cultures of other distributed team members.

"*To tell you one of my experience with my Indian team members, here in Germany we have different accent and they have different, sometimes repeatedly asking my Indian team member what he said make him embarrass to the extent that he starts replying in yes or no, At times language barrier comes in between whether its verbal or written communication*" – O15, Scrum Master

To build trust in a distributed set up where team are working across different cities and countries, it is important to respect each other's religion, culture and national holidays.

6.4.5.3 Concept – Regular Socialization

It was also evident from multiple agile projects that frequent socialization bring the distributed team members closer, they can share their concerns and ideas freely with each other. Above all, holding socializing events like virtual pizza Friday's, remote coffee on call, planned project outings once per release have huge impacts on agile teams in terms of building and improving trust with each other.

"Consciously choosing to take out time to get to know each other as people not as just machines, sharing food, eating out together, have virtual pizza Fridays with members at onsite, start with small introductory videos of all team members both onsite and offshore every release, this way you can gel the team together with great levels of trust." – O17, Scrum Master

6.4.5.4 Concept – Bridging time zone gaps

Another key concern of agile teams working in distributed set up is time zone difference. Agile practitioners explained that better utilization of overlapping hours proved very useful in solving time zone gaps.

"*Doing relevant discussion and setting up priorities may result proper utilization of overlapping hours with our distributed team mates. We are having some disagreements say on an API design, we try to solve it in sprint planning itself without dragging it too long*" – O13, Developer

Trust between team members gets compromised due to cross team dependencies and coordination issues in distributed set up if temporal distance is not being handled properly.

6.5 Discussion

We now present the mapping of our trust factors with multidimensional constructs of perceived trustworthiness of the Mayer's [20] model of organizational trust. In the following section we give a brief overview of Mayer's model followed by the relevance of our findings with respect to it.

6.5.1 Mapping with Mayer's Model of Trust

Mayer et al. [205] proposed a model of organizational trust that involves two specific parties: trustor (trusting party) and a trustee (party to be trusted). The model focuses on the specific relationship between a trustor and a trustee along with the reasons why a trustor would trust a trustee. In the proposed model, on the side of trustor, propensity to trust is the general willingness to trust others [229]. Different people have different propensity to trust depending on their developmental experiences, personality types and cultural backgrounds. On the side of trustee, perceived trustworthiness refers to the attributes and actions of the person to be trusted which lead that person to be more or less trusted. Perceived trustworthiness has been treated as a multidimensional construct that has three contributing factors namely ability, benevolence, and integrity. All the three are separate factors that vary independently of one another. A continuum of the three types of trust is what determines overall trustworthiness of the trustee. Ability of a trustee entrusts that person with the tasks in the technical area he/she is adept at. Benevolence is the extent to which a trustee

would want to do good to the trustor apart from egocentric profits. Integrity relates to a set of principles that the trustor finds acceptable in the trustee for continuing the relationship [204]. Our factors of trust contribute towards these three dimensions of perceived trustworthiness of a trustee. We map our factors to the three dimensions of the trust i.e. ability, benevolence and integrity. The mapping is shown in Table 6.8. An entry in the table is indicative of our factors mapping to one or more dimensions of perceived trustworthiness of the trustee.

6.5.1.1 Working Environment

The *Working Environment* category corresponds to the working conditions and operating atmosphere of distributed agile teams. An *open working environment* encourages frequent interactions and interpersonal communication amongst team members. This helps in enhancing their knowledge and skills. This also brings about ease in personal relationships and openness amongst team members which is conducive for developing trust. *Open working environment* maps to the ability and benevolence dimension of trust. Distributed team members feels united when they work in open collaborative workspace, this not only encourages communication but also builds trust among them [230]. *Psychologically safe working environment* promotes innovation and team learning in distributed teams. *Psychologically safe working environment* makes people liberated, they start to innovate and perform to best of their abilities. *Psychologically safe working environment* maps to the ability dimension of trust as it gives them the freedom to do mistakes and learn new things. A study [231] reported that psychological safe work environment is positively related to the team performance and job satisfaction of software development teams. *Transparent working environment* facilitates information sharing. This improves coordination and communication and reduces conflict amongst distributed team members. *Transparent working environment* maps to both ability and benevolence dimension as it tends to grow both skills and reliability amongst distributed team members. In an *engaging working environment* distributed team members have shared vision and clear understanding on product development roadmap. Distributed agile teams feel engaged when their goals are aligned with the business goals and they value the relationship with their coworkers. *Engaging working environment* maps to the benevolence and integrity dimensions of trust.

6.5.1.2 Leadership

Leaders play a primary role in establishing and developing trust in teams [232, 233]. *Facilitative leadership* promotes team reflexivity which in turn encourages team innovation. *Facilitative leadership* maps to the ability dimension of trust. *Transformational leadership* is about motivating team members at different levels. *Transformational leadership* creates that connection between the team leader and team members and empowers them to perform with best of their abilities even in a distributed environment. *Transformational leadership* maps to the ability dimension of trust. Mayner [234] have reported the strong positive relationship between the transformational leadership and employee engagement while adopting agile methodology. Transformational leadership also positively influences employees' readiness for change which further improves trust amongst team members and the leader [235,236]. *Servant leadership* is concerned with both professional as well as personal well being of the team members. *Servant leadership* maps to the ability and benevolence dimension of trust. According to [237] servant leadership is a predictor of affective trust which eventually predicts a team performance in distributed teams. Servant leadership plays an important role in contributing towards social aspects of agile teams like trust and empowerment [238]. A study [239] identifies servant leadership as a predictor of scrum team's effectiveness. Leadership should be delegated to the lowest possible level among distributed teams. Leadership is rotated among team members in a shared leadership structure on the basis of domain knowledge, skills and abilities. *Shared leadership* maps to the ability dimension of trust. Moe et al. [240] argue that shared leadership is essential in distributed agile teams but proper mechanisms should be in place for learning and division of external and internal autonomy amongst team members.

6.5.1.3 Organizational

Organizational factors are the steps that an organization takes to build trust amongst distributed agile teams. Lack of organizational support causes poor planning and ineffective team communication that further leads to the team failure. *Teal management approach* requires no hierarchy and consensus, peer to peer relationship with shared goals are the core values of teal organizations. This makes distributed agile teams to be self managed, high on competence and trust.

Table 6.8 Mapping of trust factors to the trust dimensions of Mayer's model

Trust Factors	Trust Dimensions		
	Ability	**Benevolence**	**Integrity**
Working Environment Factors			
Open working environment	X	X	
Psychologically safe working environment	X		
Transparent working environment	X	X	
Engaging working environment		X	X
Leadership Factors			
Facilitative leadership	X		
Transformational leadership	X		
Servant leadership	X	X	
Shared leadership	X		
Organizational Factors			
Teal management approach	X		X
Requisite tool support	X	X	X
Coaching for agility and behavior	X	X	
Performance appraisal and reward system	X		
Personal Factors			
Agreeable mindset		X	X
Team play attitude		X	X

Neuroticism free outlook		X	
Empathy		X	X
Socio Cultural Factors			
Hybrid team culture		X	X
Respecting other cultures		X	X
Regular socialization		X	X
Bridging time zone gaps	X	X	

According to Kramer [72] distributed team members are more willing to trust one another in a teal based software organizations as they expect the same from other distributed team members.

When distributed team members are aligned towards common goals and attributed towards finding similarities then they tend to have the integrity dimension of trust. *Teal management approach* maps to the ability and integrity dimensions of trust. Teal based organization structure along with senior management support is crucial in order to sustain the difficulties of working in dispersed environment [241]. Some provisions like flexible work schedules, fast feedbacks help team members to gain trust. *Requisite tool support* is another important factor for building trust amongst distributed team members. Right from the daily stand up meetings to the sprint planning and sprint retrospective meetings across different locations need adequate communication mechanisms (like video conferencing, slack boat, IM's etc.). When team members have access to all necessary communication and Application Lifecycle Management (ALM) tools they build stronger trust with each other across all three dimensions of the trust. Organizations also encourage agile learning by sending their employees to attend agile conference and workshops. This would enable them to eliminate silos and foster climate of trust. In a distributed set up, members may have different working styles and beliefs that may cause conflicts in the absence of right knowledge of agile value and principles. Hence, *coaching for agility and behavior* maps to the ability and benevolence dimensions of trust. *Performance appraisal and reward system* of an organization also plays a important role in building trust across

distributed team members. Performance appraisal should be team centric and should be based on collective efforts of the team. As whole team works for the organizational goals from the different locations, they should be evaluated and rewarded in a team not as an individual. *Performance appraisal and reward system* maps to the ability dimension of trust. Teams with low trust have members focused on the equity of performance assessment and rewards while teams with high trust were more accepting of rewards based on collective team performance [242].

6.5.1.4 Personal

Team trust is dynamic in nature, it can increase or decrease based on the interactions between the individuals. Personal traits of the distributed team members also influence the level of trust amongst them. Agile team members irrespective of their positional roles need to have an aligned vision and agreeableness in order to build trust amongst them. *Agreeable Mindset* is essential in order to develop growth mindset in a distributed environment. According to [243] team members having personal traits that exhibit agreeableness are more trustworthy. *Agreeable Mindset* maps to the benevolence and integrity dimensions of trust. *Team play attitude* shows the helpful and humble personality trait of the team member. Team dynamics amongst the distributed team members improves as members are more concerned with the welfare of the others. Agile team members resolve conflicts, disagreements and find similarities when they develop *team play attitude*. *Team play attitude* maps to the benevolence and integrity dimensions of trust. Moreover, lack of trust not only hampers collaboration but also the effective decision making between the members of the distributed agile team [117]. Interviewed participants acknowledged the fact that as agile development projects have rapid decision, development, testing and delivery cycles they encounter unexpected delays, negative feedbacks, and surprising errors but they need to keep their calm and should remain emotionally stable. *Neuroticism free outlook* brings positivity and make people to get along with fellow team members nicely that results in increased level of trust. *Neuroticism free outlook* maps to the benevolence dimension of trust. A study [244] reported the relationship between the team personality composition and team climate. This study confirms findings of our study as it shows the positive effect of agreeableness on agile team climate where as neuroticism have negative effect on it. *Empathy* develops feeling of care and

compassion amongst distributed team members which further builds trust between them. *Empathy* maps to the benevolence and integrity dimensions of trust.

6.5.1.5 Socio Cultural

Distributed agile teams are transnational with members from two or more nations. A difference in cultural backgrounds, norms and beliefs is a source of communication barrier. A *hybrid team culture* helps in reducing the communication barrier. *hybrid team culture* is the set of work based norms and behaviours that heterogeneous team members create amongst themselves based on their similarities after some interactions. Cultivating a *hybrid team culture* assists in building trust in cross cultural agile teams. *Hybrid team culture* help team members to feel connected and cooperate with each other. *Hybrid team culture* improves the benevolence and integrity dimensions of trust. As different people belong to different cultures in a distributed set up, one should have a respect for fellow team members culture, religion and festivals. This is important as it creates a bond of respect and integrity. *Respecting other cultures* maps to the dimensions of benevolence and integrity of trust. *Regular socialization* has an impromptu effect on trust formation in distributed agile teams. Distributed team members form social bonds through informal virtual meetings, regular site visits, informal chats and video conferencing. Poor communication skills owing to cultural differences, causes lack of trust between the team members and product owner distributed across different parts of the world [245]. Ramesh et al. [211] argued that regular visits by distributed team members and building a collaborative culture contributes in building trust amongst geographically distant located teams. Our study also confirms that by conducting face to face meetings at the starting of the iteration or once per release increases team bonding and cohesion between the distributed agile team members. *Regular socialization* maps to the benevolence and integrity dimension of trust. Another important socio cultural factor is *bridging time zone gaps* which address concerns related to task dependency and coordination amongst distributed team members. When distributed team members utilize their overlapping time zones and share the pain of each other they build foundation of trust amongst them. *Bridging time zone gaps* maps to the ability and benevolence dimension of trust. In order to have effective team meetings with onsite members and clients, it is important to conduct the meetings at a suitable time so that

every team member can attend [86] [211]. Team members distributed across different locations should rotate the timings of meetings either weekly or biweekly to share the pain of time zone difference.

CHAPTER 7

A TRUST ASSESSMENT TOOL FOR MEASURING AND IMPROVING TRUST IN AGILE TEAMS

7.1 Introduction

Software process and software engineering practices followed in project affect software quality [246]. However, software quality also depends on teamwork, specifically with respect to the interaction and trust within the teams [247]. Teamwork quality is one of the primary factors for success of a software project [248]. Hoegl and Germuenden [188] studied the impact of team work quality on software quality and identified six teamwork quality factors (TWQ) factors: coordination, communication, mutual support, member contribution, effort and cohesion that influence software quality. Coordination of expertise, value sharing and trust are amongst the main components of teamwork quality model for improving team performance [189]. Trust acts as glue when team members work together and influences team performance [190].

Trust is subjective in nature, one cannot measure trust directly although there are certain factors that affect trust in teams and can measure trust indirectly by measuring those factors. The indirect method relies on measuring and assessing the factors that are likely to affect trust levels within the team. The direct method focuses on observing the behaviors of team members and asking for their opinions through interviews or questionnaires [249]. To talk more about trust in general, trust is both an emotional and logical act. As suggested by Straker [250] emotionally, it is where you expose your vulnerabilities to people, but believing they will not take advantage of your openness. Logically, it is where you have assessed the probabilities of gain and loss, calculating expected utility based on hard performance data, and concluded that the person in question will behave in a predictable manner. Actually, trust is a blend of both “I trust you because I have experienced your trustworthiness and because I have faith in human nature”. In particular “Trust means being able to predict what other people will do and what situations will occur, if we can surround ourselves with people we trust then we can create a safe present and an even better future” [251].

Agile software development falls under iterative and incremental style of development. There are many flavors of agile development methods like scrum, feature driven development, eXtreme programming (XP), adaptive software development and dynamic systems development. But scrum methodology and XP are most widely used in software industry today [252]. A study [253] reported that use of agile practices reinforce trust in three agile teams, especially from the testers and product owners perspective. When we say trust measurement in agile software development teams, we should have some comprehensive factors or mechanisms to measure trust directly or indirectly within agile software development teams. Therefore, the requirement to set a benchmark for trust measurement in agile teams becomes inevitable, but amount of good empirical studies pertaining to trust measurement in agile team are very scant [217] [254].

In the current research work, we propose a trust assessment instrument based on our empirically developed framework for building trust. We name this instrument as **A**ssessment of **T**rust in **A**gile Teams (**ATA**). It takes into consideration trust measuring factors across five dimensions namely: Working Environment, Leadership, Organizational, Personal and Socio Cultural. ATA is used for calculating composite trust scores within any agile team and then provide feedback on the basis of these scores to improve trust. Relation between trust levels and performance of distributed agile teams is established using ATA. ATA comprises a questionnaire consisting of 34 items across five constructs (dimensions) of trust. Initial inspiration for item generation was from the "Teamwork Quality (TWQ)" model (Hoegl and Germuenden [188] and "The Big Five of Teamwork" model [190].

7.2 BACKGROUND AND RELATED WORK

Research work on trust measurement within teams has been done across many disciplines like psychology [208], computer science [255], social science [256], management [257] and medical [258]. Salas et.al [190] has proposed a model for team effectiveness. This model defines three coordinating mechanisms that are said to exist in all effective team work: shared mental model, mutual trust and closed loop communications.

To achieve timely goals and success, agile teams must be self organizing in nature [144]. Working in collaborated and self organizing manner necessitates the presence of trust within agile teams [127]. Manager needs to trust his team to get the job done and every individual needs to trust his team members based on technical abilities and knowledge in order to make the project a success [224]. There are certain consequences for lack of trust in distributed agile teams like lack of commitment, ineffective communication, team conflict that can develop potential risk for the project like poor team performance, lot of rework, increased cost and poor product quality [195].

Dorairaj et al. [128] used grounded theory method to identify seven techniques for building trust in distributed teams. Some of noted techniques are engender cultural identity, keep regular interaction, encourage knowledge sharing, build team relationship, and promote mutual visiting. It is beneficial for the team members to have cultural understanding of other distributed team members for a good work relationship. Hasnain et al. [259] suggested role of communication and trust in agile software teams using concept of game theory. Zolin et al [260] proposed the model of trust development in cross functional global teams based on online survey and interviews. Trust acts as a vital factor to resolve complexity in complex software development project. Social complexity approach using Cynefin framework is used to identify trust as a potential factor in agile development projects [261].

Agile teams are cross functional comprising of developers, testers, architects and product owners. Software industry needs professionals who understand the cross functional nature and ideology of agile teams which is based on team work, collaboration, trust and coordination. This tradeoff is mainly due to the lack of attention given on agile development related courses at undergraduate and graduate levels [262]. Kropp et al. [263] reported that undergraduate students have too little knowledge of agile. Agile education is important at undergraduate level in order to bridge the knowledge gap of what students are learning and what software industry is expecting [3]. Masood et al. [264] reported the challenges related to communication and coordination amongst students teams while teaching agile course in a university.

Hence, above discussion motivated us to teach industry oriented agile software development course to undergraduate students and also to teach them importance of trust in agile development methodology using Agile Trust Assessment (ATA) tool

7.3 Research Method

7.3.1 Data Collection

We carried out our research activity amongst final year undergraduate students pursuing B.Tech (CSE). Total 14 agile teams were formed with 7-10 members in each team. All of these students were part of a FastTrack program of final year students (involving students from both sector 62 and sector 128 campuses) at Jaypee Institute of Information Technology (campuses), Noida, India. A course on agile software development with DevOps was taught under this fastrack program. The course was specially designed for those final year students who got placed in campus recruitments in software organizations. All of these students had studied agile software development foundation course in fifth semester and had basic understanding of agile methodology. The duration of this course was one month that included daily 1 hr lecture and 1 hr lab. Real agile software development events and practices were used along with multiple agile tools to keep team members in synchronization with each other. Trello was used for the product backlog creation and prioritization, planning poker was used to estimate user stories. Jira tool was used for defining the workflows and managing product and sprint backlogs, slack was used for team communication. This course was designed to be executed in a simulated real world project development environment which included sprint simulations and mini projects as deliverables. Classes were held daily five times in a week from Monday to Friday just like the working timing of any software organization set up. This course was based on a hybrid of scrum and XP frameworks along with DevOps practices. This course was exclusively designed to teach as well as to give hand-on experience to undergraduate students on the best practices of scrum, XP and DevOps in a real world project development environment. Entire course was divided into 3 sprints of one week duration. Sprint artifacts such as product backlog, sprint backlog, user story estimation and workflow creation is shown in Appendix C with the help of selected screenshots.

7.3.2 Questionnaire construction and item generation

Initial inspiration for item generation was from the "Teamwork Quality (TWQ)" model [188] and "the big five of teamwork" model [190].

Table 7.1 Teamwork Quality (TWQ) Model Construct

Teamwork Quality Construct [188]	Explanation
Communication	Is there frequent, open, direct and informal communication?
Coordination	Are individual efforts are synchronized and structured within the team
Member contribution	Are team members able to contribute to best of their potential
Mutual Support	Do team members help and support each other during tasks?
Effort	Do team members work for the team tasks?
Cohesion	Do team members works with a team spirit

Table 7.1 shows the teamwork quality construct of TWQ model [188] and Table 7.2 shows the Teamwork construct of teamwork model [190].

We developed a questionnaire consisting of total 34 items. These 34 items were generated using both TWQ model and big five teamwork model as well as our trust framework developed using grounded theory in chapter 6. This trust framework comprises of five factors: working environment, leadership, organizational, personal and socio cultural factors. We developed the trust assessment instrument across five dimensions of trust as shown in table 7.3.

7.3.3 Questionnaire procedure and design

We used Likert rating scale, which meant that for each item respondents were asked to rate and indicate the degree to which they felt the statement was true with respect to their agile team experience. Respondents rated on a five point Likert scale from 1 to 5 (where 1 = low agreement, and 5 = high agreement to the statement).

Table 7.2 A big five teamwork model construct

A big five teamwork model [190]	Explanation
Adaptability	Ability to adjust strategies based on information gathered from the environment through the use of backup behavior and reallocation of intra-team resources
Backup behavior	Ability to anticipate other team members' needs through accurate knowledge about their responsibilities.
Closed-loop communication	The exchange of information between a sender and a receiver irrespective of the medium
Mutual performance monitoring	The ability to develop common understandings of the team environment and apply appropriate task strategies to accurately monitor teammate performance.
Mutual trust	The shared belief that team members will perform their roles and protect the interests of their teammate
Shared mental models	An organizing knowledge structure of the relationships among the task the team is engaged in and how the team members will interact.
Team leadership	Ability to direct and coordinate the activities of other team members, assess team performance, assign tasks, develop team knowledge, skills, and abilities, motivate team members, plan and establish a positive work atmosphere
Team orientation	Propensity to take other's behavior into account during group interaction and the belief in the importance of team goal's over individual members' goals.

7.3.4 Controlled experiment using ATA

The sample consisted of 68 male students and 52 female students (N= 120). Age ranged from 21-23. Different roles were assigned within the teams like product owner, scrum master, developers, testers and business analyst to create a real agile development environment. Scrum master acted as a team leader for their respective teams. We provided basic technical training on scrum and XP methodologies to every team before starting this research. Faculty acted as an agile coach to mentor students on agile software development.

Table 7.3 Items in Questionnaire

Construct (no. of items)	Item no., Statement
Working Environment	1. Open and collaborative working environment is provided for teams 2. Team members can share information openly and freely within the team. 3. Team members can approach and communicate freely with the agile coach and senior management 4. Team members are getting timely information on task and sprint goals 5. Important project information is shared with team members in a timely manner 6. Work in progress is visible to every team member through proper information radiators
Leadership	1. Team leader prevents conflicts amongst team members. 2. Team leader allow team members to self allocate the tasks. 3. Team leader involves team members in decision making. 4. Team leader allow team members to experiment and make mistakes. 5. Team leader helps in removing impediments and facilitate the team 6. Team leader distributes responsibility equally amongst team members 7. Team leader motivates the team members and empower them
Organizational	1. Team members have been provided adequate infrastructure to carry our project. 2. Team members have access to adequate communication audio/video tools 3. Team members job roles and responsibilities are properly defined. 4. Flat team structure is followed in the organization 5. Clear goals are defined at task and subtask level for the team 6. Proper agile trainings are conducted to enhance agile knowledge of team members. 7. Team members are encouraged to attend agile meet ups and conferences. 8. Performance appraisal and reward system is team centric not individual centric.
Personal	1. Team members are able to work our any performance differences. 2. Team members respect decisions of other teammates. 3. Team members work towards common team goals 4. Team members are ready to help fellow team members 5. Team members have friendly rapport with fellow team members 6. Team members maintains positive mindset and stay task focused 7. Team members have empathy for the fellow team members
Socio Cultural	1. There is equality amongst team members irrespective of race, religion or ethnicity 2. There is no communication related issue in team due to language difference 3. Team members have frequent social communication with each other 4. Team has collective culture orientation towards team goals 5. Team members openly discuss and share ideas irrespective of different cultural backgrounds 6. Team meetings are arranged at common time zones

Thereafter, every team developed a software project using scrum or XP methodologies, each project had 3 sprints with sprint duration of one week. Trust measurement exercise was conducted across all the teams at the end of each sprint retrospective meeting. We used trust scoring sheets to collect scores from all agile teams on all the trust factors, and then scores from every agile team member were averaged out to get a score for that particular team for each factor.

The scores were analyzed and the trust factors with low scores were identified and appropriate improvement measures were taken to improve the trust scores. Then same exercise was repeated for rest of the sprints and finally we compared the results from all the sprints for all the factors based on their respective trust scores.

Each agile development team developed software projects by following agile development life cycle activities with our trust assessment framework in the sequence as shown in Figure 7.1.

7.4 Statistical Data Analysis and Validation

We performed analysis of the data in three phases. In the first phase, we analyzed mean trust score of all the agile software development teams for each trust factor for all the three sprints according to their respective category. This provides us the assessment of trust score improvements or falls based on the trust factors.

To calculate the mean trust score for every factor in its category for every sprint we used the following formula:

$$Mean\ Trust\ Score = \frac{\Sigma_{for\ all\ teams}\left(\left(\Sigma_{for\ all\ members\ per\ team}\ Score\right) \Big/ No.\ of\ members\ per\ team\right)}{No.\ of\ teams} \qquad \text{Equation-1}$$

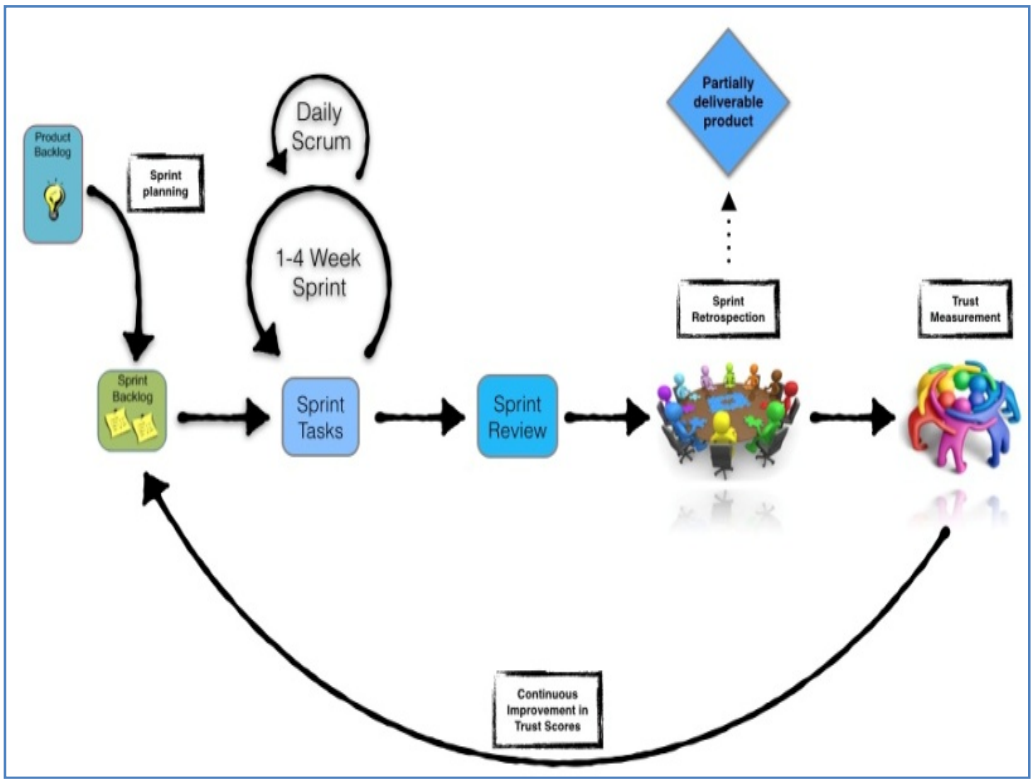


Figure 7.1 Scrum Framework with Trust Assessment

The above double summation to calculate the Mean Trust Score can be interpreted as follows:

First, the average of each trust factor for every team is calculated by summing the scores given by all team members and taking the quotient on division with the number of team members as the average. Then, these values were summed up and divided by the number of teams to get the mean trust score value for each trust factor in its category for every sprint.

In the second phase, we analyzed correlation among the trust factors, we used Principal Component Analysis (PCA) extraction method and perform factor analysis. Factor analysis helps us in integrating the trust factors that are strongly correlated with each other and have the same source of attributes. Factor analysis helps us to capture largest possible variance from the original data contained in respective trust factors. PCA is a well-known mathematical procedure used to reduce correlated variable into a smaller number of new synthetic variables (Principal Components) through linear combinations of the original variables with minimum loss of information [28]. Then, we used Pearson correlation coefficient to get the correlation

values of trust factors between Sprint 1, Sprint 2 and Sprint 3. Pearson correlation coefficient is used to measure the strength of a linear association between two variables. When the value r = +1, it means a perfect positive correlation and value of r = -1 means a perfect negative correlation.

Finally in the third phase, we used Cronbach's Alpha test to measure internal consistency. Cronbach's alpha test helps us to check how closely related the trust factors were as a group. Cronbach's alpha is an index of reliability associated with the variation accounted for by the true score of the "underlying construct". A Cronbach alpha indicates the level of reliability with its value ranging from 0 to 1. A value close to 1 indicates high level of reliability. We used IBM SPSS Statistics 27 [265] software for data analysis.

7.4.1 Factor Analysis using PCA

We performed pre-analysis test for validating suitability of this data set for factor analysis. Exploratory Principal Factor Analysis (PFA) is used to test the construct validity of the questionnaire [266]. Reason behind using PFA is that it allow us to explore the underlying constructs, which cannot be measured directly [267]. PFA assumes that individual item scores are result of an underlying factor or construct [268]. PFA mainly group the variables if they correlate and explain much of the same variance [268].

The Kaiser-Mayer-Olkin (KMO) was 0.655 and the Barlett test for sphericity was significant at $p<0.001$, indicating the suitability of this data for factor analytic procedures. In addition to that communalities were found well above 0.3 (see Table 7.4), indicating that every item has some common variance with other item. Then we carried out factor analysis with all 34 items. We performed varimax rotated factor analysis to identify the main factors that influence trust in agile teams. In varimax rotated factor analysis method, set of multiple inter-correlated variables are replaced by small number of independent variables (factors) by orthogonal transformations (rotations). This is done by computing the correlation matrix of the variable, i.e., by computing their Eigen values and Eigen vectors. Factor loadings after the varimax rotation give the correlation between the variables and the factors. We used highest loadings to determine to which factor an item belonged. A total of 25 items load at 0.50 or above into the ten–factor structure. The first three factors of principal

component analysis covers 41% of variance out of which first factor explained 26% of the variance and termed as trust factor category I and, the second factor 8% of the variance and termed as trust factor category II, the third factor 7% of the variance and termed as trust factor category III, the rest of the factors had eigen values of just over one, with each factor explaining 3-5 % of the total variance. We preferred three factor solution, which explained 41 % of the variance. Results obtained by varimax rotated factor analysis are given in table 3. Factor I has total five items of which three items were showing leadership as a dominant factor on agile development teams. Factor loading suggest that rest of the two factors which were from personal category may also be influenced by leadership attributes. Factor II has total four items of which three were strongly related with working environment of the agile development teams. Factor III has total of three items of which two are strongly related with organizational facilities of agile development teams. As a result of PFA analysis, it is observed that factors related to leadership, working environment, organizational, and personal categories have a profound impact on trust amongst the team members. However, factors concerning cultural and personal categories have a lesser impact on trust within agile development in our scenario. This was possibly due to the fact that team members belong to same demography and ethnicity.

7.4.2 Correlation

Pearson correlation was used to find out correlation between the sprints which is depicted in table 7.5, strong correlation was observed between sprints 1 and sprint 2 equals to 0.851 , again a strong correlation was observed between sprint 2 and sprint 3 equals to 0.864 which indicates correlation of scoring patterns over three sprints on the trust factors and indicates that all factors are strongly correlated over all the 3 sprints on which trust scores were taken and affecting trust within agile software development teams.

7.4.3 Reliability

The reliability of a questionnaire can be inferred as the consistency of the survey results. Internal consistency is used to evaluate the consistency of a questionnaire.

Internal Consistency: Internal consistency shows the extent to which the questionnaire items are inter-correlated. Internal consistency also proves if questionnaires are consistent in measurement of the same construct [113]. A detailed

analysis of the selected trust factors was performed using Cronbach's α to check internal consistency. If Cronbach's α value is 0, it indicates no internal consistency (none of the items are correlated with each other) and if Cronbach's α value is 1, then it indicates perfect internal consistency (all the items are perfectly correlated with each other).

Table 7. 4 Factor analysis with varimax rotation using PCA

	Trust Factor Category I	**Trust Factor Category II**	**Trust Factor Category III**	**Communalities**
Team leader involve team members in decision making	**0.50**	0.34	0.11	**0.73**
Team leader distributes responsibility equally amongst team members	**0.54**	-0.11	0.32	**0.72**
Team leader helps in removing impediments and facilitate the team	**0.68**	0.11	0.25	**0.66**
Team members work towards common team goals	**0.70**	0.16	0.11	**0.66**
Team members maintains positive mindset and stay task focused	**0.70**	0.23	0.03	**0.66**
Team members can approach and communicate freely with the agile coach and senior management	0.33	**0.71**	-0.01	**0.75**
Open and collaborative working environment is provided for teams	0.29	**0.65**	0.17	**0.66**
Important project information is shared with team members in a timely manner	0.08	**0.62**	0.42	**0.73**
Performance appraisal and reward system is team centric not individual centric	0.21	**0.67**	0.20	**0.66**
Team members job role and responsibilities are clearly defined	0.24	0.20	**0.66**	**0.67**
Proper agile trainings are conducted to enhance agile knowledge of team members	0.08	0.25	**0.55**	**0.72**
Team members are ready to help fellow team members	0.17	0.14	**0.81**	**0.77**
Percent of variance %	26.35	8.00	7.54	
Cumulative percentage %	26.35	34.35	41.89	

Loading >0.5 are considered to be statistically significant
Extraction method: Principal Component Analysis

However, in order to have adequate internal consistency Cronbach's α value should be above 0.70 [113]. As indicated in table 7.6 the standardized alpha for all the five categories is well above .70 indicating acceptable level of internal consistency and reliability for all the five categories of our trust framework

Table 7.5 Pearson correlation coefficient for the three sprints.

Correlations			
	Sprint 1	Sprint 2	Sprint 3
Sprint 1	1	0.851**	0.572**
Sprint 2	0.851**	1	0.864**
Sprint 3	0.572**	0.864**	1
**. Correlation is significant at the 0.01 level (2-tailed).			

Table 7.6 Trust Measurement Framework-Reliability

Factor (Category)	No of Items	Alpha (Standardized)
Working Environment	6	0.997
Leadership	7	0.999
Organizational	8	0.999
Personal	7	0.997
Socio Cultural	6	0.780

7.4.4 Content Validity

Content validity is used to determine the extent to which items in a questionnaire are representative of the theoretical construct which questionnaire is intended to measure [269]. Items in questionnaire are generated using TWQ model and big five teamwork model and also from our empirically developed trust framework. Initial draft of questionnaire was sent to two agile coaches working in reputed software organizations and to one professor in JNU in order to evaluate the content validity of our questionnaire. We received the feedback from all of the three experts and incorporated their feedbacks in our questionnaire.

7.5 RESULTS

Using the mean trust scores as represented in Equation-1 above, bar plots were prepared which were used to compare the trust scores for the corresponding factor throughout the three sprints.

7.5.1 Improvement in trust factors under working environment category

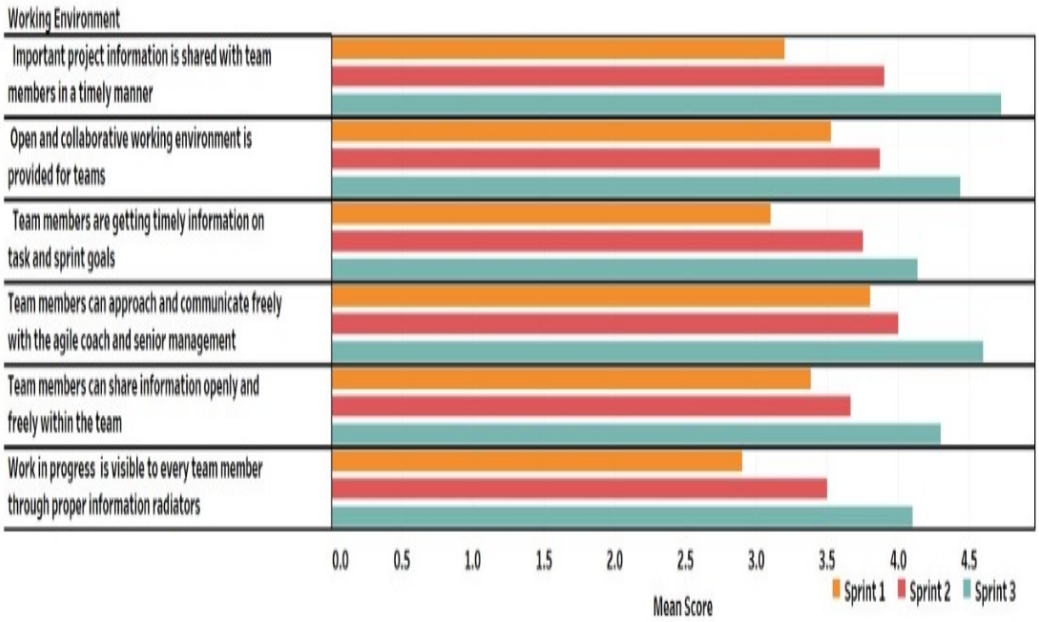


Figure 7.2 Mean trust score of working environment factors over three sprints

Factors under working environment category showed significant increase with each subsequent sprint as shown in Figure 7.2. Trust assessment score of sprint 1 showed poor information flow amongst agile team members. Proper strategies were adopted in order to increase the information flow amongst the team members like trello and jira tools used to create a scrum board and workflows. Workflow of each user story has been created along with notification in jira so that every team member receives update on all the changes made on the project. Scrum board created in trello helped team members to visualize the work in progress. Mean trust score were increased considerably using those strategies. Second most significant increase was observed in factor related to team members approachability and ease of communication with the agile coach and senior management. In our case, agile coach role was played by the course faculty. Open communication environment has been created so that team

member of any team can approach agile coach. Moreover, agile coach was also added across slack channels of all the teams in order to address any type of concern. Trust assessment score of other factors related to open and collaborative working environment and visibility on work in progress also improved as open collaborative working culture was adopted. Every team member was involved on the project since the beginning. Agile project activities were planned and executed with the help of tools like planning poker was used for user story estimation. Agile team games were also played in order to create an open working environment. We brought out open communication culture in which each team member can communicate with the respective team leader and agile coach without any hesitation.

7.5.2 Improvement in trust factors under leadership category

As inferred from the Figure 7.3. for the leadership trust factors, trust assessment score showed significant increase across all the factors. Maximum increase was reported in factor concerned with the team leader removing impediments and facilitating the team. Facilitative leadership was practiced on all the projects right from the onset. Agile coach emphasized on facilitative leadership style where team leader align every team member with respect to the team goals and also facilitate the team in achieving those goals. Achievable sprint goals were assigned to team members and team leader along with an agile coach ensured that team should work on sprint goals and if any problem was encountered then proper resolution was made without blaming any team member. Improvement was also observed in factor related to team leader preventing conflict amongst team members. Clear division of work and duties right from the project beginning ensured that there were fewer conflicts amongst team members. Agile coach also participated in sprint review and retrospective meeting in order to prevent any conflict amongst team members. Team leader also acts as a motivator for the team. Agile coach help team members on every aspect of agile right from the execution of any agile events or interpersonal issues related to teamwork. Every team members work has been valued and properly recognized. This motivated and empowered agile teams. Also, team members were allowed to experiment and make mistakes. Blame culture was discouraged and team members responsibilities were divided equally without any biasing.

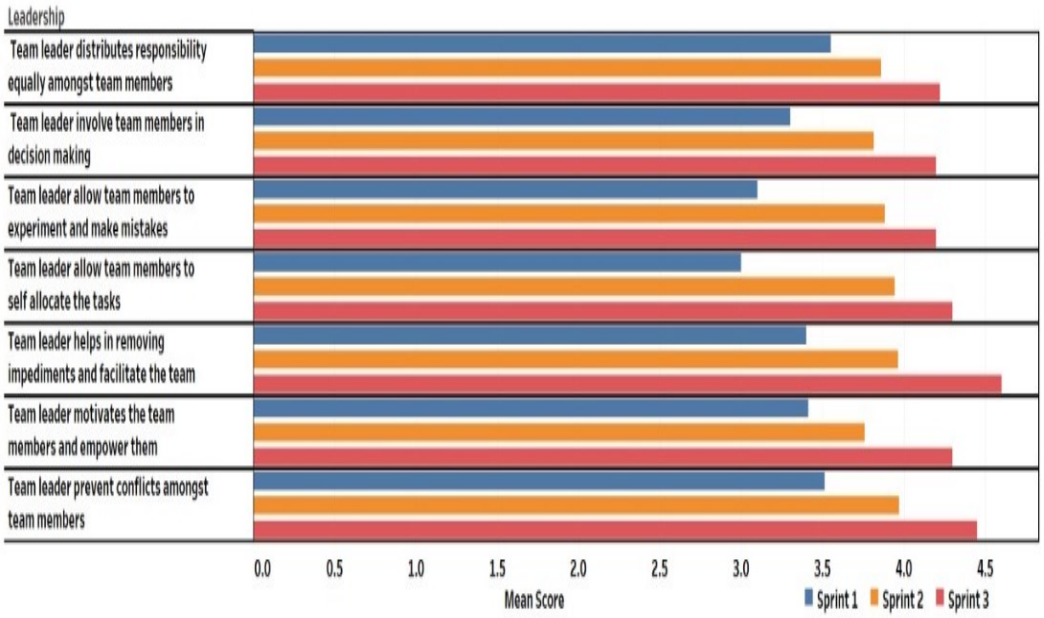


Figure 7.3 Mean trust score of leadership factors over three sprints

7.5.3 Improvement in trust factors under organizational category

Trust assessment scores of all the factors under organizational category showed significant increase over three sprints as shown in Figure 7.4. The maximum increase was achieved in the factor: performance appraisal and reward system should be team centric. All the projects were directly associated with the curriculum of the students. The teacher assessment marks of this course on agile software development were directly mapped to the performance of students in these projects. Project evaluation was team based not individual based. Moreover, we also announced appreciation letters for top 2 teams at the onset of the project. This led to a very healthy competitive environment. Furthermore, proper agile trainings were also conducted in order to increase the agile knowledge of the teams. Two expert lectures cum hands on sessions were arranged for the students by a software industry agile coach. These session helped team members to work in a real life agile project environment. Also, all the teams have been provided with adequate infrastructure to carry out their projects in an agile environment. Agile Lifecycle Management (ALM) tools like jira, asana, trello were provided to all the teams. Devops tools like git, Jenkins and chef were also provided to all the teams. Team communication tools like slack, google meet were used in order to have effective communication with all the team members.

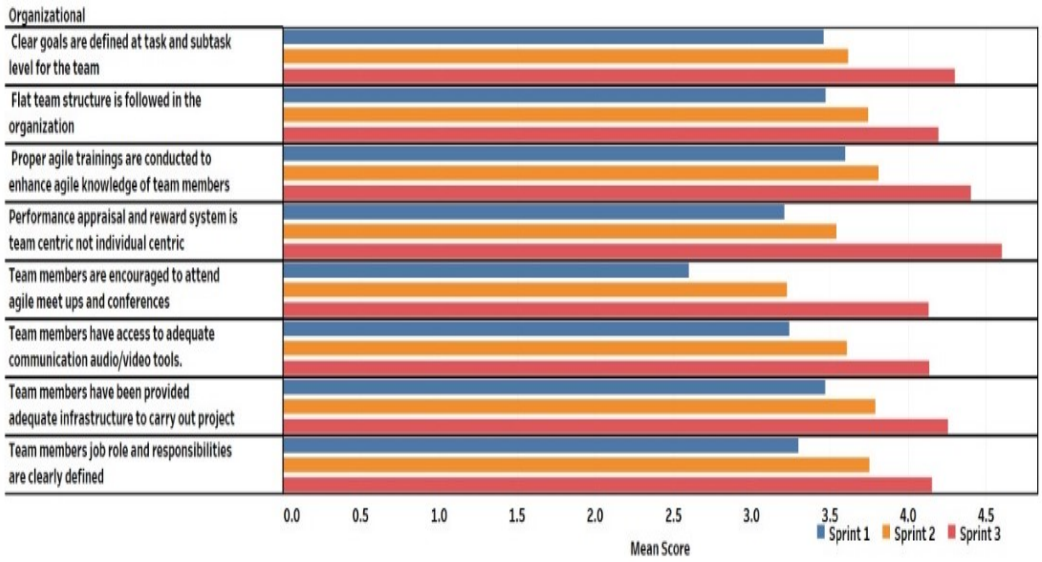


Figure 7.4 Mean trust score of organizational factors over three sprints

7.5.4 Improvement in trust factors under Personal Category

As inferred from the Figure 7.6, trust assessment score of factors under personal category showed significant increase over three sprints:

Team members maintain positive mindset and stay task focused showed maximum increase. Agile coach played a very important role in building positive mindset of team members. Agile coach always encouraged team members to conduct all the agile events and helped team members to overcome any bottleneck. As team members were working in close cooperation with each other under the guidance of agile coach, there team play attitude improved significantly and team members were supportive to each other. Sprint retrospective meetings were conducted at the end of every sprint, every team member shared the challenges and problems they encountered during the sprint. Every team member's opinion and decision was respected in sprint retrospective meeting. Team members were encouraged to work out any performance differences during sprint retrospective meetings. As whole project environment was team centric and team members were working for common goals, they were in constant touch with each other, which improved empathy between them.

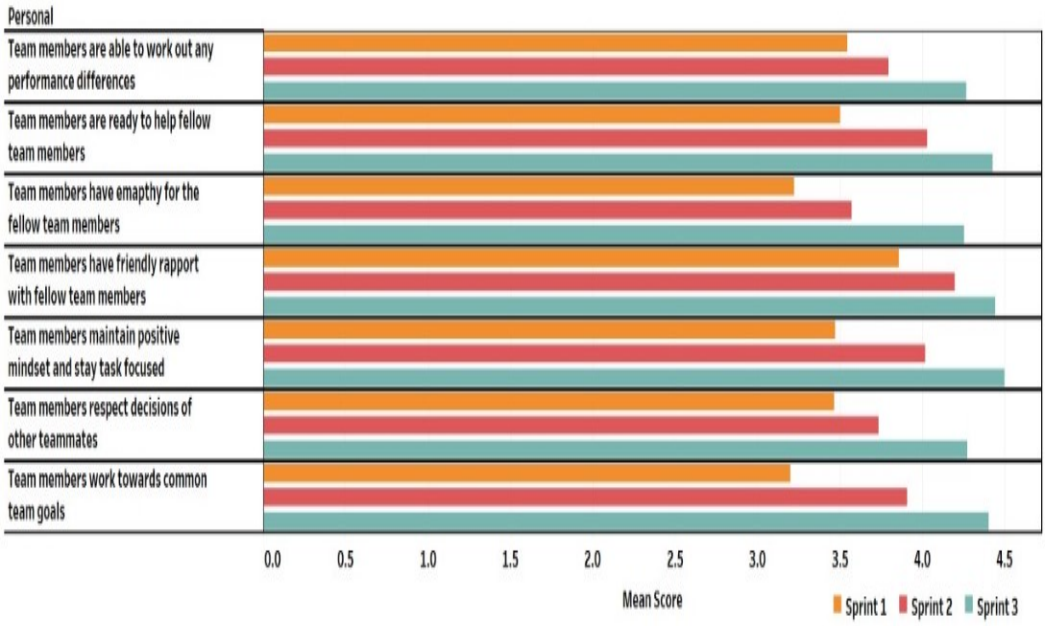


Figure 7.5 Mean trust score of personal factors over three sprints

7.5.4 Improvement in trust factors under Socio Cultural Category

Trust assessment of socio cultural trust factors has also shown improvement over three sprints as shown in Figure 7.7. Maximum increase was achieved in the following two factors: team members openly discuss share ideas irrespective of different cultural backgrounds. All the team members were from India but belong to different regions. One team environment was created where all team members were very much part of one single team and there was no discrimination based on the cultural background. Agile teams showed collective culture orientation as they were proactively involved in performing toward collective team goals.

Teams socialize during or after sprint release: Social outings like team lunch meetings were organized for the students after every sprint so that the team could spend some time bonding with each other outside of working hours. Many times project meetings were conducted outside working hours (after college timings). In that case, it was advised that meetings were conducted at time which was convenient to all the team members.

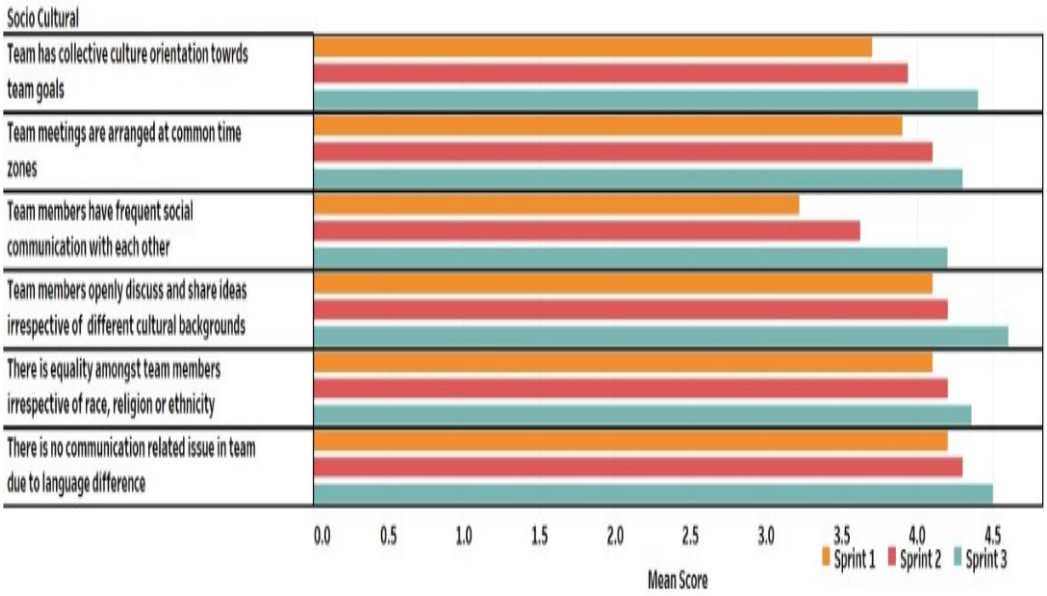


Figure 7.6 Mean trust score of socio cultural factors over three sprints

7.5.5 Relation between trust and performance

We observed an increase in performance across most of the agile software development teams as the trust scores among them increased with each sprint. Figure 7.8 showed the burn down charts of one of the agile teams, this burn down chart has been normalized with the number of total hours required to complete a particular sprint at the start of the sprint to get a uniform scale of 0-1 rather than different hours required for different sprints. The normalized value is calculated by dividing each value in the burn down by the total time required for the entire sprint.

As evident from the normalized burn-down chart in Figure 7.8, the slope is maximum for the third sprint depicting an increase in the performance and a decrease in the development cycle time. This increase can be attributed to improvement in trust levels over three sprints.

7.6 Discussion

Results from a quantitative analysis show a positive relation between trust and team performance of agile teams. Our ATA tool used for calculating mean trust score of agile teams across five different categories: working environment, leadership, organizational, personal and socio cultural shows the effectiveness of these factors in improving trust levels amongst agile team members. Wheelan [270] developed a

Group Development Questionnaire (GDQ) which comprises of four parts. The third part of (GDQ3) is about "Trust and Structure". GDQ3 is about creating an environment of trust and structure group in such a way that clear job roles are defined. Open communication should be emphasized that should focus on tasks. GDQ 3 also suggests that role of a leader should be consultative rather than authoritative. Sidky [271] proposed an agile adoption framework which is based on agile principles. This framework uses the Sidky's Agile Measurement Index (SAMI) to determine the agility level of projects and organizations. SAMI has components namely: "agile levels", "agile principles", "agile practices" and "items or indicators". This framework has two levels of assessment: first at the project level and second at the organizational level. A study [272] partly validated the Agile Adoption Framework and suggested few more categories viz. open communication, dedication to team work, leadership style, honest feedback and agile planning. Sample size in this study consisted of 45 participants from two software organizations and questionnaire comprised of 29 items. Exploratory Principal Factor Analysis is used for the statistical analysis and to group the variables which correlate with other. Another study [273] asserted to focus on group development aspect while building an agile team. A survey comprises of 31 items developed using Sidky's agile measurement tool [271] and GDQ4 questionnaire [272] and used on a sample size of 66 participants. Gren et al. [272] reported that group development is strongly related to dedication to teamwork, open communication and agile planning.

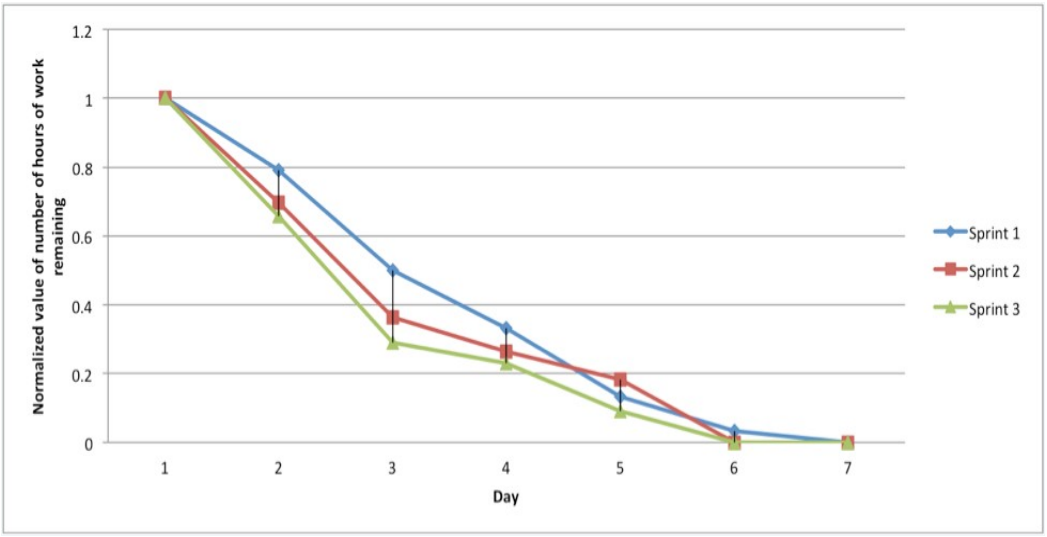


Figure 7.7 Normalized Sprint burn down chart

Lindsjorn [273] identified positive effect of teamwork quality on the team performance in agile teams. Shared decision making, supportive leadership, frequent interaction and contribution from all team members are some of the critical components of building high performance teams. A study by Fairbairn [274] argued that trust acts as an enabler for building agile muscle memory. Building community surrounding people, allowing people to experiment, sharing knowledge, and open communication are four key components for building trust competencies. Moe et al [275] highlights the importance of trust if agile team members need to work on collective product goals. Effective tool support both for communicating with team members and for managing project activities are important for building trust in agile development teams [275]. A study [276] highlights the positive affect of affective trust between team members and customer during agile requirement engineering practices such as requirement elicitation and management. Barbosa et al. [277] reported trust as an important contributor in perceptions of interdependence when people are working in a team. Furthermore, Trust is identified as important psychological factor that can influence team work and degree of interdependence.

CHAPTER 8

CONCLUSION AND FUTURE WORK

This chapter summarizes the results of our study. This is followed by discussion of limitations of this study. Finally, we present suggestions for future work.

8.1 Conclusion

This thesis presents a comprehensive study on two important aspects of quality in agile development projects: 1) test automation adoption to improve the product quality on agile projects and 2) Empirically developed framework to build trust amongst distributed agile teams to improve the teamwork quality. This study is based on Grounded Theory research method that involved 60 agile practitioners from 28 different software organizations in India, USA, Germany, Australia and New Zealand. Summary of the findings and main contribution of this study are as follows:

(i). A substantive theory that explains transition journey of adopting test automation on agile development projects is proposed. This theory explains different challenges faced by agile teams while practicing test automation and different strategies used by them in order to overcome those challenges. *Adopting Test Automation* is a transition journey on agile development projects comprising of three stages: *Pre-Transition*, *In-Transition* and *Post-Transition* stage. *Pre-Transition* stage reflects the different types of impediments faced by the participants while practicing test automation. The challenges faced by agile practitioners are further categorized into three broad categories namely: Technical, Organizational and human. Challenges were pertaining to the tools, test environment management, scrip maintenance, senior management support, mindset, automation awareness. The strategies adopted to overcome these challenges are the focus of attention in the *In-Transition* stage. Finally, in the *Post-Transition* stage agile teams reap the benefits of test automation on successfully practicing the test automation in their projects. Then, these transition stages are further mapped to the Bridges transition model [12].Bridges transition model depicts that change is a transition process that includes three phases: 1) Ending phase – this phase

correspond to that state where people need to let go the past practices before they embrace new things. 2) Neutral zone or Transition phase – where people experience challenges and come out of their comfort zone for embracing a new change. 3) The Beginning phase – where people start to embrace the new change and develop the understanding regarding the new change. The problems associated with change are mostly not because of the change itself but due to the transition involved with the change, and when agile teams undergo this transition of letting go their traditional test practices and mindset, they face some problems. In this study, *Adopting Test Automation* on agile projects is also a change that different software organizations look upon for producing good quality software in less time. This study provides the roadmap to the agile practitioners towards their test automation adoption journey by making them aware about different types of impediments and ways to overcome them which is based on experiences gained from the real life agile projects.

(ii). We developed a hybrid test automation framework with Selenium 2.0 using Page Object Model (POM). Normally, web application comprises of multiple web pages and each webpage has a different set of objects also known as UI elements to interact with. Moreover, most of the time a single UI element is being used in multiple test cases sometimes as many as twenty or thirty odd test cases so if that UI element undergoes any change than one has to traverse these twenty or thirty odd test cases to update that change. By using POM, change has to be made under that particular page object class containing the reusable method for that particular UI element and all the test cases need to call that reusable method of that particular POM class. Hence, a page object model simply models the objects within the test code. This reduces the amount of duplicated code and thus, if the UI changes frequently, the fix needs only to be applied at one place. This test automaton framework which is based on POM would inherently make script maintenance easier, increases reusability of tests and also makes test execution simpler and faster.

(iii). Software industry is witnessing a paradigm shift from traditional models to adaptive agile development methods. Agile methods ensure maximum participation of stakeholders in the entire development process. More and

more software organizations are using best practices of agile development methods in distributed software development. However, using agile methods in geographically distributed set up has its own challenges related to spatial distance, cultural differences, alignment of team members and time zone differences. Organizations are in constant search for different techniques to overcome these challenges and build an environment of trust within distributed agile teams. This study proposes a comprehensive framework for building trust in distributed agile teams. This framework is developed empirically by conducting GT study over a period of 24 months that involved 40 agile practitioners from 19 different software organizations. We investigated the multidimensional perspective of trust from agile practitioners view through their real life project experiences using GT.

Proposed framework has five components related to working environment, leadership, organizational, personal and socio cultural perspectives. The influence of these components in building trust within distributed agile teams is also the main contribution of this study. Organizations and software practitioners can utilize this study to have plan of action and devise out the strategy while practicing agile in distributed environment.

(iv). We developed a trust assessment instrument based on our empirically developed framework for building trust. We name this instrument as **A**ssessment of **T**rust in **A**gile Teams (**ATA**). It takes into consideration trust measuring factors across five dimensions namely: Working Environment, Leadership, Organizational, Personal and Socio Cultural. ATA is used for calculating composite trust scores within any agile team and then provide feedback on the basis of these scores to improve trust. Relation between trust levels and performance of distributed agile teams is established using ATA. ATA comprises a questionnaire consisting of 34 items across five constructs (dimensions) of trust. Statistical analysis and validation of ATA was also performed using IBM SPCC software.

(v). In this thesis, Grounded theory research method, its components and its application in software engineering research is also described in detail. We use all the components of the grounded theory viz. theoretical sampling, memoing,

open coding, selective coding, theoretical coding and theory generation as per GT guidelines. Our research shows that Grounded Theory research method is used successfully in the field of software engineering research.

8.2 Limitations

The inherent limitation with grounded theory research study is that the research findings are grounded in the specific contexts that are explored in the research. The context in this study was governed by our choice of research destinations and the availability, accessibility and willingness of agile practitioners to participate in this study.

Data gathered from the interviews may be prone to bias [124], Mainly, four types of data can be perceived to the researcher: (i) Baseline Data, the best description provided by the participant (ii) Interpreted, description given by the trained professional who wants others to perceive the data from his professional perspective (iii) Properline data, data provided by participant which he thinks is proper for the researcher (iv) Vaguing it out, when participant is providing the vague information to the researcher [145]. In software engineering research, above kind of biases can be encountered by the researchers. Collecting the relevant information from the participant and discern the type of data while interviewing is a learning process itself. Moreover, it need skills to ask counter questions to check the relevancy of the data provided during an interview. Using semi-structured interview with open ended questions gives the opportunity to researchers to ask detailed and specific questions.

Triangulation of the data source used in collecting the data is another way to ensure authenticity in qualitative research [124]. The best way is to supplement the interview data with field observations of the project in their natural settings [124]. In our case, observations data supported our findings from the interview data thereby strengthening it. A research study based on Grounded Theory produces a “mid-range” theory, which implies that the theory can be modified with constant comparison of new data from the same context or new context. We do not affirm that our findings are universally generalizable to different contexts. However, findings of this study accurately characterize the contexts studied. In this thesis, a detailed description of the contexts of our research is presented which can be used by other researchers. Grounded Theory research method is used on new data to determine whether a

substantive finding fits the new data, or the findings are transferable to another context [146].

8.3 Evaluating the Grounded Theory

Qualitative research studies such as GT can be evaluated using different perspectives and approaches [278, 279]. This study used two approaches to evaluate our emerging grounded theories in chapter 4 and chapter 6 which are described in the next section.

8.3.1 Glaser's approach

Glaser and Strauss believe that GT is a strong method for building theory, not verifying it [136]. Therefore, verification of emergent theories are left to other research methods that are better suited to that purpose with the help of follow up studies [125]. Glaser advocates that a theory, which is based and grounded on a real data, in some sense has already been verified: "A grounded theory gets its concepts from the data; it does not bring ideas to force the data that need to be subsequently tested" [145]. Glaser, however, proposes four criteria: fit, workability, relevance, and modifiability for evaluating credibility of a grounded theory [125] [145]. Fit criterion is about emerging codes, concepts, categories and theory from the data instead of researchers own preconceived ideas. GT should also explain experiences of other practitioners that do not participate in theory generation process. Workability is used to assess the integration of the related categories with the core category. The theory should have the predictive and explanatory capability in the area under study. Moreover, it should also explain how the major concerns of the participants are continually resolved [125]. Relevance indicates how well a GT identifies a core problem and allows it to emerge. According to Glaser [145] “GT arrives at relevance, because it allows core problems and processes to emerge”. Modifiability means that emerged theory must be changeable in such a way that it maintains its validity and fits with the data.

Since the beginning of this research, major literature review has been avoided in the initial stage. We gained the knowledge on different perspectives and concerns related to our general area of interest i.e. “Agile Project Management” and “Teamwork Quality" on agile development projects with the help of interviews during data collection process. Our data collection was based on theoretical sampling which guided us on our next data sites based on the emerging theory. In order to have well

rounded and diverse perspectives and to gain relevant information on the emerging concepts, participants form different roles, locations, and teams have been recruited for the interviews. Moreover, our interview questions kept on changing continuously to keep focus on concerns of the participants that were central to the emerging concepts, categories and theory [145]. Theoretical sampling was stopped on reaching theoretical saturation when no new concepts were found. All the concepts, categories and their properties were integrated with the central category. The resultant grounded theory emerged from the substantive data without being forced by preconceived idea or hypotheses. Therefore, we believe that our grounded theory and its related categories fit the realities under the current study.

Work and Relevance: Our codes, concepts and categories that were emerged from the data represented the concerns of the participants. Regular discussion with the research supervisors on the emerging categories and theory ensured that emerging theory works. We shared our findings and publications with the participants and took their feedbacks via e-mail or through telephonic communication. Findings were also shared with the agile practitioners through presentations at agile conferences and meetups. Agile practitioners provided their feedbacks. Excerpt of the written feedback from one of the agile practitioner is provided below:

"During our agile transformation journey, we encountered most of the issues as highlighted in your research, your findings accurately depicts the key concerns of the agile teams while surrounding their project around test automation." – O3, Scrum Master.

Agile practitioners often gave us comments like "most of the time", "I can relate", and "bang on" during our presentation of our emerging theory. This provided us the confidence on our substantive theory.

Modifiability: Theory development in GT is an evolving process; new data was compared with existing concepts and categories during data collection process. Furthermore, theory had evolved further as data was collected from relatively new to more experienced participants (senior developer, scrum master, agile coaches and senior management people) in agile software development. Hence, we ascertain that our theory is *modifiable* as new data brought variation in the emerging theory.

8.3.2 Lincoln and Guba's approach

Lincoln and Guba suggest some criteria to evaluate a qualitative research including *credibility*, *dependability/auditability*, *confirmability*, and *transferability* [280]. They have mentioned credibility as the most important one in establishing trustworthiness. *Credibility* is achieved mainly through *peer debriefing*, *triangulation of data*, and *member checking* [280]. How this study was evaluated against the credibility criterion is explained below.

Peer debriefing: *Credibility* was added to this study with the help of peer debriefers. The primary researcher used to discuss the GT process and the emerging concepts with the other authors and agile experts on a regular basis in order to ensure *validity* and *credibility*. This process started with handing over the complete interview transcript to the peer debriefers to read text line by line, to highlight the relevant concerns and concepts. *Peer debriefers* were then provided with the codes and memos developed by the primary researcher. In case of any disagreements on the coding, views of third party would be sought by the primary researcher. In this way, peer debriefers questioned the primary researcher about his findings and helped to clarify his interpretations, thus strengthening the *credibility* of the emerging theory.

Triangulation of data: Preliminary interviews were conducted to collect data from the participants. Thereafter, theoretical sampling technique of GT was employed to collect data from wide range of participants with different job roles so that a comprehensive view could be obtained on a research phenomenon. As, participants were from different organizations located across different locations in the world, it gave us the opportunity to understand the different social and cultural aspects in the context of our study. Non participant observations were also used to collect the data from three different projects, consequently, the data triangulation was automatically adopted in this study.

Member Checking: All the interviews were tape recorded and later transcribed for data analysis purpose. Different steps were taken in order to ensure the validity of the data collected from the participants. Participants were provided with initial findings and short technical reports through follow up email or call. Participants were requested to verify the data provided and suggest any modification or any other information. We tried to get maximum number of feedbacks on the data from our participants, even though few participants have not responded to our requests.

Researchers treated these feedbacks as another source of data that was integrated into the emerging theory through the data analysis process

8.4 Future Work

This thesis has contributed mainly by two ways: Firstly by providing the transition journey roadmap of adopting test automation on agile development projects. Secondly, by providing model comprises of multifaceted factors for building trust in distributed agile teams for improving teamwork quality. Both the contributions deemed to be novice as the phenomenon covered in this thesis has not been covered by previous theories. Hence, this study has opened several new avenues for future research:

(i). This research study suggests a roadmap of adopting test automation on agile development projects which involves three transition phases: Pre Transition, In Transition and Post Transition. Future research can explore the role of test automation in DevOps, as large numbers of agile projects are doing DevOps and practicing continuous integration and continuous delivery which involves extensive test automation. DevOps practices utilize the test automation in order to provide quality product with faster time to market. A study should be performed that could focus on how test automation is applied in devops based agile projects.

(ii). Now days, organizations are applying agile practices at the enterprise level which is known as scaled agile. In scaled agile approach multiple teams strives to implement agile practices and deliver quality product. But, as multiple teams are working across different geographical locations they face several obstacles and delays. Future research could explore the teamwork quality and role of trust in large scale agile teams distributed across geographically distant locations.

APPENDIX A

INITIAL INTERVIEW PROTOCOL

Interviewee personal details

Name of the Organization & Location:	Email:
Interviewee Name (Optional):	Designation:
Interview Date and time:	Team Size:
Total IT industry experience:	Project duration:
Total Experience in agile development:	

Consent form for audio recording the conversation

I would like to audio record our conversation for the data analysis purpose. I need your permission to audio record this interview.

Interviewee response: Yes/No

Case 1, Yes: Thank you for your permission to audio record this interview. Please let me know if you want me to pause the recording at any point of time.

Case 1, No: Thank you, I will only note down important point from our conversation.

Preliminary open ended questionnaire

1. What is the domain of your project?
2. What is Agile Project Management? How it is different from traditional software project management?
3. Which agile software development methodology you use? Scrum, XP, Hybrid or any other?
4. How many locations your project is distributed?

5. What is the sprint duration in your project?
6. What is your role in the project?
7. How you plan development and testing in your project?
8. How is your agile transformation journey? What were your initial challenges related to team, infrastructure, management or others?
9. How did you overcome those obstacles while transforming to agile software development?
10. Who takes decisions in your project?
11. How often requirements change in your project?
12. How you ensure quality of the product in your project?
13. How you collaborate with other team members?
14. Is there anything important you want to discuss on agile transformation?

APPENDIX B

OBSERVATION PROTOCOL

Project Name: Date of the Observation:

Project Domain: Observation Time:

Days of Week:

1. Describe the physical workspace settings and seating arrangement of the agile team.

   ……
   ……
   ……...……
   ……

2. How requirements are managed and prioritized in the project?

3. What types of tools are used in the project?

4. How agile teams collaborate with each other?

5. How defects are being managed in the project?

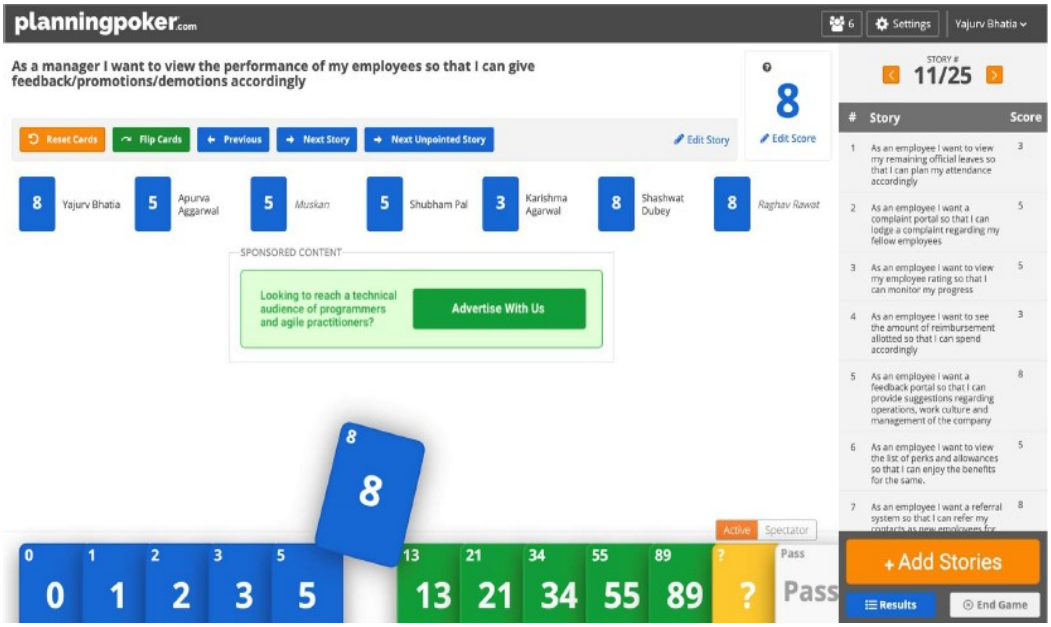
6. Any other information / comments.

7. Details of different agile project activities performed in the given project and associated observations.

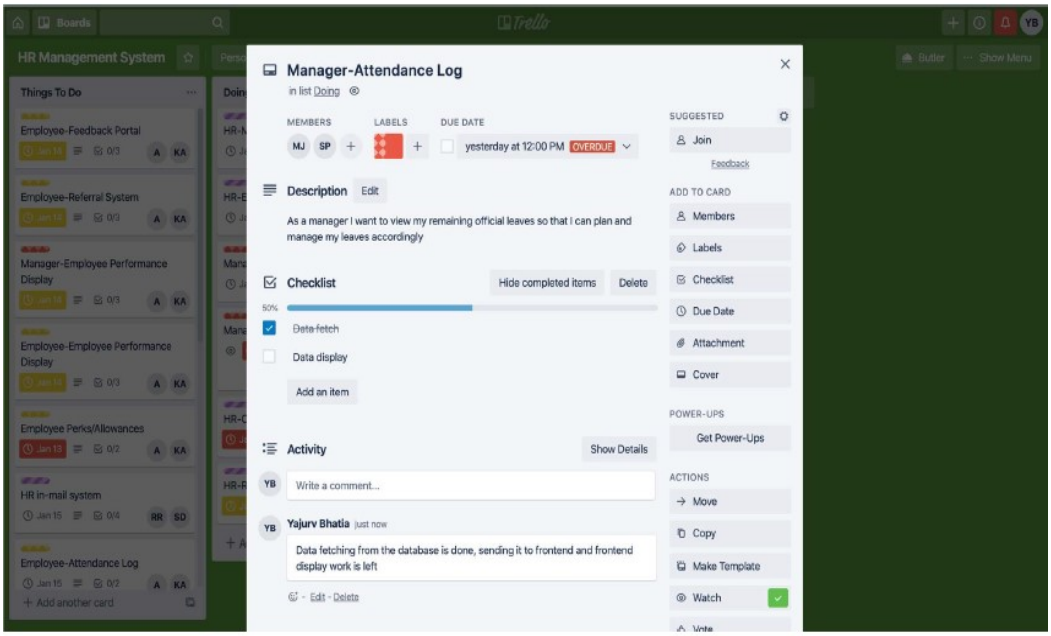
List ofactivities performed by the team	Check if Yes	Name of the participant / owner / attendee	Position and role of the participant / owner / attendee	Observations / Comments
Sprint Planning Meeting				
Sprint Review Meeting				
Daily Stand up Meeting				

Agile Project Tracking
Build Management and Version Control
End Sprint Demo
Pair Programming
Unit Testing
Integration Testing
Smoke Testing
Regression Testing
User Acceptance Testing

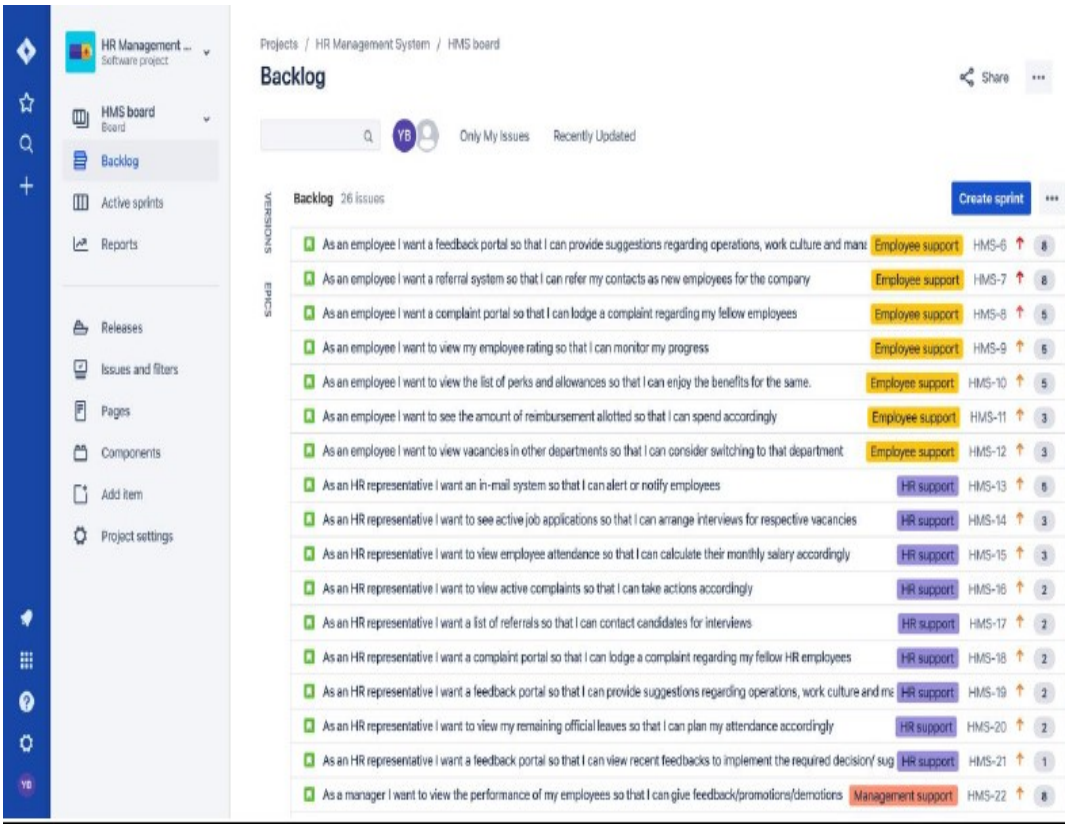
APPENDIX C



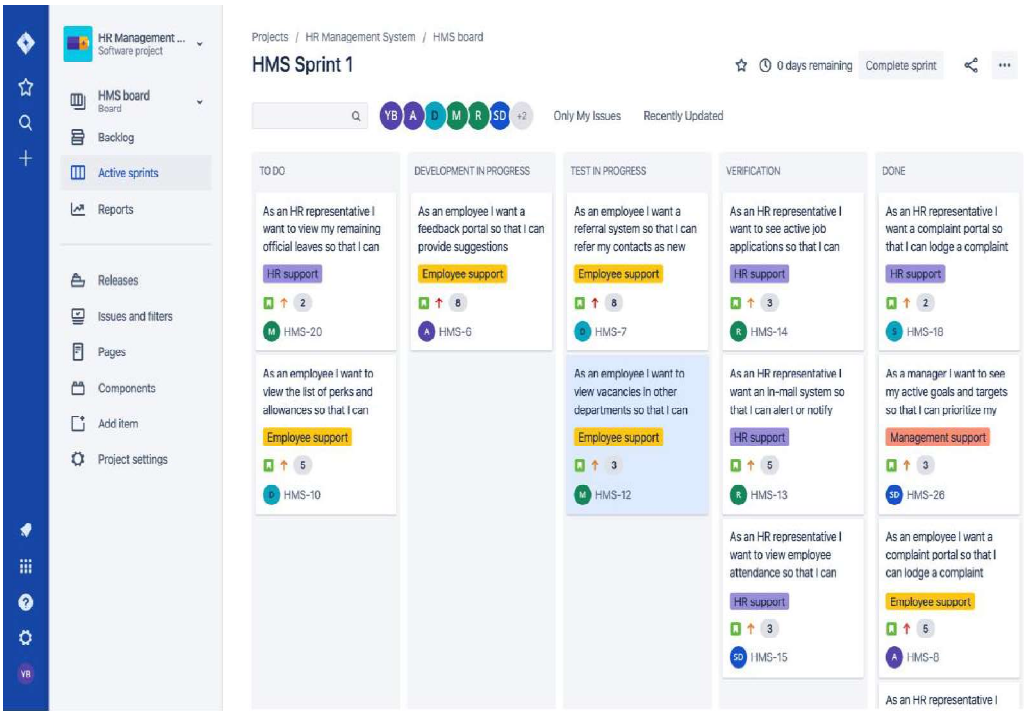
Team B: User Story Estimation using planning poker



Team B: Product backlog prioritization using trello



Team B: Creation of prioritized product backlog in Jira



Team B: Sprint 1 Scrum Board view on Jira

REFERENCES

1. VersionOne, "8th annual state of agile development survey," Available at https://www.versionone.com/pdf/8th-Annual-20State-of-Agile-Development-Survey.pdf (2018/02/08).

2. I. G. Stamelos and P. Sfetsos, *Agile software development quality assurance*. Igi Global, 2007.

3. J. M. Juran and F. M. Gryna, *Juran's Quality Control Handbook*. McGraw-Hill, 1988.

4. IEEE Std. 610.12-1990, *Glossary of Software Engineering Terminology, in Software Engineering Standard Collection*, 1990, IEEE CS Press, Los Alamitos, California.

5. E. Hendrickson, "Redefining quality" Available at http://www.stickyminds.com/ (2018/03/15).

6. E. Mnkandla and B. Dwolatzky, "Defining agile software quality assurance," in 2006 International Conference on Software Engineering Advances (ICSEA'06). IEEE, 2006, pp. 36–36.

7. B. Kitchenham, S. M.Pfleeger, *Software Quality: The Elusive Target*. IEEE Software, pp. 12-21, 1996. DOI=http://dx.doi.org/10.1109/52.476281.

8. J. P. Cavano and J. A. McCall, "A Framework for the Measurement of Software Quality," in *Proceedings of the Software Quality Assurance Workshop on Functional and Performance Issues*, New York, NY, USA, 1978, pp. 133–139.

9. ISO/IEC. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*. International Standard ISO/IEC 25010:2011, 2011.

10. J.A. McCall, M.T. Matsumoto, *Software Quality Measurement Manual,* Vol. II. Rome Air Development Center, RADC-TR-80-109-Vol-2. 1980.

11. B. W. Boehm, *Characteristics of Software Quality*. Amsterdam: North Holland, 1978.

12. N. Takeshi, S. Katsuhiko, A. Tadashi, *A Structured Approach to Software Defect Analysis*. Hewlett-Packard Journal. Palo Alto: Hewlett-Packard Co. 40 (2), pp. 50–56, 1989.

13. ISO/IEC. *Software Engineering – Product Quality – Part 1: Quality Model*. International Standard ISO/IEC 9126-1:2001, 2001.

14. ISO 9001, "Standard official page," Available at http://www.iso9001.com/ (2018/02/14).

15. J. W. Horch, *Practical Guide to Software Quality Management*. Artech House, 1996.

16. I. S. Association et al., "Iso/iec/ieee 24765: 2010 systems and software engineering-vocabulary. iso/iec/ieee 24765: 2010 25021," Institute of Electrical and Electronics Engineers, Inc, 2010.

17. P. McBreen, "Quality assurance and testing in agile projects." Available at http://www.mcbreen. ab.ca/talks/CAMUG.pdf (2017/11/21).

18. S. Ambler, "Agile documentation: strategies for agile software development, "Available at http://www.agilemodeling/com/essays/agileDocumentation.html (2018/02/01)

19. M. Huo, J. Verner, L. Zhu, and M. A. Babar, "Software quality and agile methods," in *Proceedings of the 28th Annual International Computer Software and Applications Conference*, 2004. COMPSAC 2004.IEEE, 2004, pp. 520–525.

20. M. Fowler, K. Beck, J. Brant, W. Opdyke, D. Roberts, *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.

21. R. Jeffries, "What is extreme programming?" Available at http://www.xprogramming.com/xpmag/whatisxp.html (2017/12/17).

22. M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams, and M. Zelkowitz, "Empirical findings in agile methods," in *Conference on extreme programming and agile methods*. Springer, 2002, pp. 197–207.

23. K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change (2nd ed.),* Reading, MA. Addison-Wesley Professional, 2004.

24. A. Cockburn and J. Highsmith, Agile software development: The people factor. *IEEE Computer Magazine*, 34(11), pp. 131-133, 2001.

25. K. V. Siakas, E. Georgiadou, and E. Berki, "Agile methodologies and software process improvement," in *IADIS International Virtual Multi Conference on Compute Science and Information Systems*, vol. 26, 2005.

26. P. Sfetsos, and I. Stamelos, "Empirical Studies on Quality in Agile Practices: A Systematic Literature Review." *Seventh International Conference on the Quality of Information and Communications Technology, Porto,* IEEE, 2010, pp. 44-53.

27. P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen, "New directions on Agile methods: a comparative analysis," *25th International Conference on Software Engineering, USA*, 2003, pp. 244-254.

28. A. Begel and N. Nagappan, "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study," in *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, Washington, DC, USA, 2007, pp. 255–264.

29. S. Easterbrook, J. Singer, M. A. Storey, and D. Damian, "Selecting empirical methods for software engineering research" in *Guide to Advanced Empirical Software Engineering*. Springer, 2008, pp. 285-311.

30. S. Bhasin, "Quality Assurance in Agile: A Study towards Achieving Excellence," in *2012 Agile India*, Bengaluru, IEEE, 2012, pp.64-67.

31. M. Puleio, "How Not to Do Agile Testing," *AGILE 2006 (AGILE '06)*. IEEE Computer Society, Washington, DC, USA, 2006, pp. 305-314.

32. S. Shaye, "Transitioning a Team to Agile Test Methods," *Agile 2008 (AGILE '08)*. IEEE Computer Society, Washington, DC, USA, 2008, pp. 470-477.

33. M. Cohn, *Succeeding with agile: software development using Scrum*. Pearson Education, 2010, pp. 314-316.

34. C. Sogeti, “World quality report,” Available at https://www.sogeti.com/explore/reports/world-quality-report-2014-2015/ (2017/09/17)

35. Y. Lindsjørn, G. R. Bergersen, T. Dingsøyr, and D. I. K. Sjøberg, “Teamwork Quality and Team Performance: Exploring Differences Between Small and Large Agile Projects,” in *Agile Processes in Software Engineering and Extreme Programming*, 2018, pp. 267–274.

36. T. Dingsøyr, T. E. Faegri, T. Dybå, B. Haugset and Y. Lindsjørn, "Team Performance in Software Development: Research Results versus Agile Principles," in *IEEE Software*, vol. 33, no. 4, 2016, pp. 106-110.

37. E. Salas, N. J. Cooke, and M. A. Rosen, “On Teams, Teamwork, and Team Performance: Discoveries and Developments,” *Human Factors*, vol. 50, no. 3, 2008, pp. 540–547.

38. D. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," in *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 481-494, 2003.

39. M. C. Jones and A. W. Harrison, “IS project team performance: An empirical assessment,” *Information & Management*, vol. 31, no. 2, 1996, pp. 57–65.

40. G. Asproni, “Motivation, teamwork, and agile development,” *Giovanni Asproni. Agile Times*, vol. 4, 2004.

41. T. O. A. Lehtinen, R. Virtanen, J. O. Viljanen, M. V. Mäntylä, and C. Lassenius, “A tool supporting root cause analysis for synchronous retrospectives in distributed software teams, ”*Information and Software Technology*, vol. 56, no. 4, 2014, pp. 408–437.

42. N. B. Moe and D. Šmite, “Understanding a lack of trust in Global Software Teams: a multiple-case study,” *Software Process: Improvement and Practice*, vol. 13, no. 3, 2008, pp. 217–231.

43. W. W. Royce, “Managing the Development of Large Software Systems: Concepts and Techniques,” in *Proceedings of the 9th International*

Conference on Software Engineering, Los Alamitos, CA, USA, 1987, pp. 328–338.

44. K. K. Agarwal, and Y. Singh, *Software Engineering.*(*3rd ed.*), New Delhi, New Age International Publishers, 2007.

45. D. Leffingwell, *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley Professional, 2010.

46. S. Wikipedia, "Spiral model" Available at https://en.wikipedia.org/wiki/Spiral_model (2018/03/10).

47. R. Wikipedia. "RAD model" Available at https://en.wikipedia.org/wiki/Rapid_application_development (2018/03/10).

48. J. Highsmith and M. Fowler, The Agile Manifesto. *Software Development Magazine*, 9(8), pp. 29-30, 2001.

49. S. Palmer and M. Felsing, *A Practical Guide to Feature-Driven Development*. Pearson Education, 2001.

50. A. Cockburn, *Crystal clear: a human-powered methodology for small teams*. Addison-Wesley Professional, 2004.

51. D. Wikipedia. "DSDM" Available at http://dsdmofagilemethodology.wikidot.com/ (2018/04/21).

52. J.Highsmith, *Adaptive software development: a collaborative approach to managing complex systems*. Dorset House Publishing, New York, 2000.

53. L. Framework, "Lean Framework" Available at https://www.altexsoft.com/whitepapers/agile-project-management-best-practices-and-methodologies/(2018/05/05).

54. M. Pikkarainen, J.Haikara, O.Salo, P.Abrahamsson, and J. Still, "The impact of Agile practices on communication in software development." *Empirical Software Engineering*, vol. 13, no. 3, pp. 303–337, 2008.

55. P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen, “New directions on agile methods: a comparative analysis,” in *25th International Conference on Software Engineering,* 2003, pp. 244–254.

56. H. Takeuchi, and I. Nonaka, The new new product development game. *Harvard Business Review*, vol.64, no. 1, pp. 137-146, 1986.

57. K. S. Rubin, *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison- Wesley, 2012.

58. S. Alliance, “Scrum Framework” Available at http://www.scrumalliance.org/view/scrum_framework. (2018/05/10).

59. C. Chaos, “Control Chaos” Available at http://www.controlchaos.com/old-site/rules.htm (2018/02/22).

60. K. Beck, *Extreme Programming Explained: Embrace Change (1st ed.),* Addison-Wesley Professional, 1999.

61. S. Ambler, "Agile Testing and Quality Strategies: Discipline over Rhetoric” Available at http://www.ambysoft.com/essays/agileTesting.html. (2017/12/18).

62. A. Kolawa, D. Huizinga, *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Press. p. 75, 2007.

63. L. Crispin, “Hiring an Agile Tester,” “An Agile Tool Selection Strategy for Web Testing Tools,” “Driving Software Quality: How Test-Driven Development Impacts Software Quality,” Available at http://lisa.crispin.home.att.net. (2017/11/09).

64. T. Systems “Agile Software Testing,” Available at https://www.teksystems.com/-/media/teksystems_com/Agile-Software-Testing.pdf. (2018/01/11).

65. E. Collins, A. Dias-Neto, and V. F. d L. Jr, “Strategies for Agile Software Testing Automation: An Industrial Experience,” in *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops*, 2012, pp. 440–445.

66. M. Cohn, *Succeeding with Agile: Software Development Using Scrum (1st Ed.)*, Addison-Wesley Professional. pp.314-316, 2009.

67. L. Damm, L. Lundberg, D. Olsson, "Introducing Test Automation and Test-Driven Development: An Experience Report." Electronic Notes in Theoretical Computer Science, 116 (SPEC.ISS.), 2005, pp. 3-15.

68. R. J. Eisenberg, "A Threshold Based Approach to Technical Debt," *SIGSOFT Softw. Eng. Notes*, vol. 37, no. 2, pp. 1–6, Apr. 2012.

69. A. Bertolino, "Software testing research: Achievements, challenges, dreams," in Proceedings of the Workshop on the Future of Software Engineering (FOSE 2007), pp. 85–103, 2007.

70. A. Leitner, I. Ciupa, B. Meyer, and M. Howard, "Reconciling manual and automated testing: The auto test experience," in *Proceedings of the 40th Hawaii International Conference on Systems Science (HICSS-40 2007),* 2007, p. 261.

71. S. Berner, R. Weber, and R. K. Keller, "Observations and lessons learned from automated testing," in *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE2005*, 2005, pp. 571–579.

72. P. Sfetsos and I. Stamelos, "Empirical Studies on Quality in Agile Practices: A Systematic Literature Review," *2010 Seventh International Conference on the Quality of Information and Communications Technology, Porto,* 2010, pp.44-53.

73. L. Huang and M. Holcombe. "Empirical investigation towards the effectiveness of Test First programming."*Information and Software Technology*, vol. 51, no. 1, pp. 182-194, 2012.

74. A. Gupta and P. Jalote. "An experimental evaluation of the effectiveness and efficiency ofthe test driven development." *ESEM '07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement, Washington, DC, USA,* 2007, pp. 285-294.

75. B. George and L. A. Williams. "A structured experiment of test-driven development."*Information and Software Technology, vol. 46, no. 5*, pp. 337-342, 2004.

76. N. Nagappan, E. M. Maximilien, T. Bhat and L. Williams. "Realizing quality improvement through test driven development: results and experiences of four industrial teams", *Empirical Software Engineering*, vol. 13, no. 3, pp. 289–302*,* 2008.

77. C. Sanchez, L. Williams, E.M. Maximilien. "On the Sustained Use of a Test-Driven Development Practice at IBM", *Proceedings of the AGILE 2007*, 2007, pp. 5-14.

78. Holmes and M. Kellogg, "Automating functional tests using selenium," in *AGILE 2006 (AGILE'06)*. IEEE, 2006, pp. 6–pp.

79. S. Gojare, R. Joshi, and D. Gaigaware, "Analysis and design of selenium webdriver automation testing framework," *Procedia Computer Science*, vol. 50, pp. 341–346, 2015.

80. A. Cervantes, "Exploring the use of a test automation framework," in *2009 IEEE Aerospace conference*. IEEE, 2009, pp. 1–9.

81. R. Chaipraserth, A. Leelasantitham, and S. Kiattisin, "A test automation framework in poct system using tdd techniques," in *2013 13th International Symposium on Communications and Information Technologies (ISCIT)*, 2013, pp. 600–604.

82. R. A. Razak and F. R. Fahrurazi, "Agile testing with selenium," in *2011 Malaysian Conference in Software Engineering*, 2011, pp. 217–219.

83. H. Thopate and R. R. Kachewar, "Chameleon model based automation framework design for testing in agile environments," in *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*. IEEE, 2012, pp. 1–4.

84. Liu, Q. Chen, and X. Jiang, "A maintainability spreadsheet driven regression test automation framework," in *2013 IEEE 16th International Conference on Computational Science and Engineering*. IEEE, 2013, pp. 1181–1184.

85. K. Arya and H. Verma, “Keyword driven automated testing framework for web application,” in *2014 9th International Conference on Industrial and Information Systems (ICIIS)*. IEEE, 2014, pp. 1–6.

86. S. Nerur, “Challenges of migrating to Agile methodologies.” *Commun. ACM*, vol. 48, no. 5, pp. 72-78, 2005.

87. A. Cockburn, “Agile Software Development Joins the Would-Be Crowd,” *International Journal of Information Technology and Management - IJITM*, vol. 15, 2002.

88. C. Jones, *Software Assessments, Benchmarks, and Best Practices*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.

89. S. Sawyer and P. J. Guinan, “Software development: Processes and performance,” *IBM Systems Journal*, vol. 37, no. 4, pp. 552–569, 1998.

90. S. Faraj and L. Sproull, “Coordinating Expertise in Software Development Teams,” *Management Science*, vol. 46, no. 12, pp. 1554–1568, 2000.

91. P. Abrahamsson, Extreme programming: first results from a controlled case study. IEEE, 2003, p.259.

92. A. Martin, R. Biddle, and J. Noble, “The XP customer role in practice: three studies,” in *Agile Development Conference*, pp. 42–54, 2004.

93. E. Whitworth and R. Biddle, “The Social Nature of Agile Teams,” in *Agile 2007 (AGILE 2007)*, pp. 26–36, 2007.

94. I. G. Stamelos, *Agile Software Development Quality Assurance*. Information Science Reference, 2007.

95. A. Cockburn, Selecting a project’s methodology, *IEEE Software*, pp. 64-71, 2000.

96. P. Sfetsos, I. Stamelos, L. Angelis, and I. Deligiannis, “Investigating the Impact of Personality Types on Communication and Collaboration-Viability in Pair Programming – An Empirical Study,” in *Extreme Programming and Agile Processes in Software Engineering*, pp. 43–52, 2006.

97. L. Williams and R. Kessler, *Pair Programming Illuminated*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

98. L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the Case for Pair Programming," *IEEE Software.*, vol. 17, no. 4, pp. 19–25, 2000.

99. P. Sfetsos, L. Angelis, and I. Stamelos, "Investigating the extreme programming system–an empirical study," *Empirical Software Engineering*, vol. 11, no. 2, pp. 269–301, 2006.

100. T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and Software Technology*, vol. 50, no. 9, pp. 833–859, 2008.

101. J. E. Hannay, D. I. K. Sjoberg, and T. Dyba, "A Systematic Review of Theory Use in Software Engineering Experiments," *IEEE Transactions on Software Engineering*, vol. 33, no. 2, pp. 87–107, 2007.

102. D. I. K. Sjøberg, T. Dybå, B. C. D. Anda, and J. E. Hannay, "Building Theories in Software Engineering," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D.I. K. Sjøberg, Eds. London: Springer London, pp. 312–336, 2008.

103. T. R. Black, *Doing Quantitative Research in the Social Sciences: An Integrated Approach to Research Design, Measurement and Statistics*. SAGE, 1999.

104. M. Myers, "Qualitative Research in Information Systems," *MIS Quarterly*, vol. 21, 1997.

105. J. Carver, "The Impact of Background and Experience on Software Inspections," *Empirical Software Engineering*, vol. 9, pp. 259–262, 2004.

106. C. A. Crabtree, C. B. Seaman, and A. F. Norcio, "Exploring language in software process elicitation: A grounded theory approach," in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pp. 324–335, 2009.

107. A. Martin, R. Biddle, and J. Noble, "The XP Customer Team: A Grounded Theory," in *2009Agile Conference*, pp. 57–64, 2009.

108. R. K. Yin, *Case Study Research: Design and Methods.* SAGE Publications, 2009.

109. J. Corbin and A. Strauss, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 1998.

110. A. Fink, *The Survey Handbook*. SAGE Publications, 1995.

111. S. Tsang, C. F. Royse, and A. S. Terkawi, "Guidelines for developing, translating, and validating a questionnaire in perioperative and pain medicine," Saudi journal of anaesthesia, vol. 11, no. Suppl 1, p. S80, 2017.

112. M. W. Watkins, "Exploratory factor analysis: A guide to best practice," Journal of Black Psychology, vol. 44, no. 3, pp. 219–246, 2018.

113. L. J. Cronbach, "Coefficient alpha and the internal structure of tests," psychometrika, vol. 16, no. 3, pp. 297–334, 1951.

114. N. Ethnography "NPS Ethnography: African American Heritage & Ethnography." Available at https://www.nps.gov/ethnography/aah/aaheritage/ERCb.htm. (2017/11/21).

115. H Sidky, *Perspectives on Culture: A Critical Introduction to Theory in Cultural Anthropology*, Pearson 2004.

116. H. Sharp and H. Robinson, "An Ethnographic Study of XP Practice," *Empirical Software Engineering*, vol. 9, no. 4, pp. 353–375, Dec. 2004.

117. N. B. Moe, T. Dingsøyr, and T. Dybå, "Understanding Self-Organizing Teams in Agile Software Development," in *19th Australian Conference on Software Engineering (aswec 2008)*, pp. 76–85, 2008.

118. B. Glaser, "Grounded theory institute: methodology of Barney G Glaser." Available at http://groundedtheory.org/ (Dec-2017/12/10).

119. J. Vickers, "Exploring Engagement: A Grounded Theory Study of Young People's Interactions with Healthcare Professionals," PhD, University of Salford, 2016.

120. A. Strauss and J. Corbin, “Grounded theory methodology: An overview,” in *Handbook of qualitative research*, Thousand Oaks, CA, US: Sage Publications, Inc, pp. 273–285.1994.

121. C. Goulding, *Grounded Theory: A Practical Guide for Management, Business and Market Researchers*. SAGE Publications, 2002.

122. K. Charmaz, *Constructing Grounded Theory*. SAGE, 2014.

123. A. J. PICKARD, *Research Methods in Information*. Facet Publishing, London, UK, 2007.

124. K. W. Parry, “Grounded theory and social process: A new direction for leadership research,” *The Leadership Quarterly*, vol. 9, no. 1, pp. 85–105, Jan. 1998.

125. B. G. Glaser, *Emergence Vs Forcing: Basics of Grounded Theory Analysis*. Sociology Press, 1992.

126. S. Dorairaj, J. Noble, and P. Malik, “Understanding Team Dynamics in Distributed Agile Software Development,” in *Agile Processes in Software Engineering and Extreme Programming*, pp. 47–61, 2012.

127. R. Hoda, J. Noble, and S. Marshall, “Organizing self-organizing teams,” *Proceedings -International Conference on Software Engineering*, vol. 1, 2010.

128. S. Dorairaj, J. Noble, and P. Malik, “Understanding Lack of Trust in Distributed Agile Teams: A grounded theory study,” in *16th International Conference on Evaluation Assessment in Software Engineering (EASE 2012)*, pp. 81–90, 2012.

129. G. Melnik and F. Maurer, “Introducing Agile Methods in Learning Environments: Lessons Learned,” in *Extreme Programming and Agile Methods - XP/Agile Universe 2003*, pp.172–184, 2003.

130. K. Weaver and J. K. Olson, “Understanding paradigms used for nursing research,” *J Advance Nursing*, vol. 53, no. 4, pp. 459–469, Feb. 2006.

131. W. J. Orlikowski and J. Baroudi, “Studying Information Technology in Organizations: Research Approaches and Assumptions,” *Information Systems Research*, vol. 2, pp. 1–28, 1991.

132. G. WALSHAM, “Doing interpretive research,” *European Journal of Information System*, vol. 15, no. 3, pp. 320-330, 2006.

133. RWJF, “RWJF - Qualitative Research Guidelines Project | Critical Theoretical Paradigm | Critical Theory Paradigms.” Available at http://www.qualres.org/HomeCrit-3518.html. (10-Dec-2017/12/10).

134. S. C. Brown, R. Stevens, P. F. Troiano, and M. K. Schneider, “Exploring complex phenomena: Grounded theory in student affairs research,” *Journal of College Student Development*, vol. 43, pp. 173–183, 2002.

135. R. Hoda, J. Noble, and S. Marshall, “Balancing Acts: Walking the Agile Tightrope,” in *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, New York, NY, USA, 2010, pp. 5–12.

136. B. G. Glaser and A. L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine, 1967.

137. M. Ali Babar, J. M. Verner, and P. T. Nguyen, “Establishing and maintaining trust in software outsourcing relationships: An empirical investigation,” *Journal of Systems and Software*, vol. 80, no. 9, pp. 1438–1449, Sep. 2007.

138. P. Martin, “Grounded Theory and Organizational Research,” *The Journal of Applied Behavioral Science*, vol. 22, pp. 141–157, 1986.

139. R. S. SCHREIBER, AND P. N. STERN, *Using Grounded Theory in Nursing*. Springer Publishing, Broadway, New York, 2001.

140. D. L. Rennie, J. R. Phillips, and G. Quartaro, “Grounded theory: A promising approach to conceptualization in psychology?,” *Canadian Psychology/Psychologiecanadienne*, vol. 29, pp.139–150, 1988.

141. Y. Chun Tie, M. Birks, and K. Francis, “Grounded theory research: A design framework for novice researchers,” SAGE open medicine, vol. 7, p. 2050312118822927, 2019.

142. K. Charmaz, *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*. SAGE Publications, 2006.

143. B. Glaser, Grounded Theory Institute: Methodology of Barney G. Glaser, 2010.

144. R. Hoda, J. Noble, and S. Marshall, "Developing a grounded theory to explain the practices of self-organizing agile teams". Empirical Software Engineering, 2011.

145. B. G. Glaser, *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory*. Sociology Press, Mill Valley, CA, 1978.

146. B. G. Glaser, *Doing Grounded Theory: Issues and Discussions*. Sociology Press, Mill Valley, CA, 1998.

147. A. Cockburn, People and Methodologies in Software Development. PhD thesis, University of Oslo, Norway, 2003.

148. B. G. Glaser and J. A. Holton, "Remodeling Grounded Theory" 1967.

149. J. Morse, Sampling in Grounded Theory. The SAGE Handbook of Grounded Theory. London: SAGE. pp. 229–44, 2007.

150. ASCI, "Agile Software Community of India" Available at http://www.agileindia.org/ (2017/01/21).

151. AI 2016 "Agile India 2016" Available at http://www.2016.agileindia.org/ (2016/05/28).

152. XP 2017 "18th International Conference on Agile Software Development" Available at https://www.agilealliance.org/xp2017/ (2017/07/20).

153. M. Castillo-Montoya, "Preparing for interview research: The interview protocol refinement framework," vol. 21, pp. 811–831, 2016.

154. J. W. Creswell, *Qualitative Inquiry and Research Design: Choosing Among Five Approaches*. SAGE Publications, 2012.

155. B. G. Glaser, *The Grounded Theory Perspective III: Theoretical Coding*. Sociology Press, 2005.

156. V. Stray, D. Sjøberg, and T. Dybå, "The Daily Standup Meeting: A Grounded Theory Study," *Journal of Systems and Software*, vol. 114, pp. 101–124, 2016.

157. B. G. Glaser, "Naturalist inquiry and Grounded Theory," vol. 32, pp. 114–132, 2007.

158. A. Deak, T. Stålhane, and G. Sindre, "Challenges and strategies for motivating software testing personnel," *Information and Software Technology*, vol. 73, pp. 1–15, May 2016.

159. G. Whyte and D. Lindsay Mulder, “The Impact of Software Test Constraints on Software Test Effectiveness,” *The Electronic Journal Information Systems Evaluation*, vol.14, no.2, pp.254-270, 2011.

160. N. I. Sayed, “The Case for Agile testing,” Available at https://www.cognizant.com/InsightsWhitepaper. (2017/06/18).

161. J. Z. Gao, J. Tsao, Y. Wu, and T. H.-S. Jacob, *Testing and Quality Assurance for Component-Based Software*. Norwood, MA, USA: Artech House, Inc., 2003.

162. E. Dustin, J. Rashka, and J. Paul, *Automated Software Testing: Introduction, Management, and Performance*. Pearson Education, 1999.

163. S. D. Shaye, “Transitioning a Team to Agile Test Methods,” in *Agile 2008 Conference*, 2008, pp. 470–477.

164. S. Adolph, W. Hall, and P. Kruchten, “Using grounded theory to study the experience of software development,” *Empirical Software Engineering*, vol. 16, no. 4, pp. 487–513, Aug. 2011.

165. K. Stol, P. Ralph, and B. Fitzgerald, “Grounded Theory in Software Engineering Research: A Critical Review and Guidelines,” in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pp. 120–131, 2016.

166. W. Bridges, *Managing Transitions: Making the Most of Change*. Perseus, 2003.

167. E. F. Collins and V. F. de Lucena, “Software Test Automation practices in agile development environment: An industry experience report,” in *2012 7th International Workshop on Automation of Software Test (AST)*, pp. 57–63, 2012.

168. J. W. Yoder, R. W. Brock, and H. Washizaki, “QA to AQ Part Six: Being Agile at Quality Enabling and Infusing Quality,” in *Proceedings of the 23rd Conference on Pattern Languages of Programs*, USA, pp. 23:1–23:14, 2016.

169. K. Karhu, T. Repo, O. Taipale, and K. Smolander, "Empirical Observations on Software Testing Automation," in *2009 International Conference on Software Testing Verification and Validation*,2009, pp. 201–209.

170. J. Bach, "Test Automation Snake Oil !,"*Windows Technical Journal*, pp. 40-44, 1999.

171. T. Diirr and G. Santos, "Improvement of IT service processes: a study of critical success factors," *Journal of Software Engineering Research and Development*, vol. 2, no. 1, p. 4, May 2014.

172. V. Kettunen, J. Kasurinen, O. Taipale, and K. Smolander, "A Study on Agility and Testing Processes in Software Organizations," in *Proceedings of the 19th International Symposium on Software Testing and Analysis*, New York, NY, USA, 2010, pp. 231–240.

173. O. Taipale and K. Smolander, "Improving Software Testing by Observing Practice," in *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, New York, NY, USA, 2006, pp. 262–271.

174. M. Aniche and M. A. Gerosa, "Does test-driven development improve class design? A qualitative study on developers' perceptions," *Journal of the Brazilian Computer Society*, vol.21, no. 1, p. 15, 2015.

175. D. M. Rafi, K. R. K. Moses, K. Petersen, and M. V. Mäntylä, "Benefits and limitations ofautomated software testing: Systematic literature review and practitioner survey," in *2012 7thInternational Workshop on Automation of Software Test (AST)*, pp. 36–42, 2012.

176. A. Qumer and B. Henderson-Sellers, "An evaluation of the degree of agility in six agile methods and its applicability for method engineering," *Information and Software Technology*, vol. 50, no.4, pp. 280–295, Mar. 2008.

177. Agile24 "Version One 10th Annual State of Agile Report" Available at http://www.agile247.pl/wp-content/uploads/2016/04/VersionOne-10thAnnual-State-ofAgile-report.pdf. (2017/06/10).

178. M. G. Software "Scrum Methodology and Project Management." Available at https://www.mountaingoatsoftware.com/agile/scrum. (2017/09/20).

179. P. Day, “n-tiered test automation architecture for agile software systems,” Procedia Computer Science, vol. 28, pp. 332–339, 2014.

180. S. Tyagi, R. Sibal, and B. Suri, “Adopting Test Automation on Agile Development Projects: A Grounded Theory Study of Indian Software Organizations,” in *Agile Processes in Software Engineering and Extreme Programming*, 2017, pp. 184–198.

181. Selenium “Selenium for .Net using C# language.” Available at http://seleniumdotnet.blogspot.com/ (2017/07/20).

182. M. Leotta, D. Clerissi, F. Ricca, and C. Spadaro, “Improving test suites maintainability with the page object pattern: An industrial case study,” in 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops. IEEE, 2013, pp. 108–113.

183. Selenium2.0 “Selenium Web Browser Automation.” [Online]. Available at https://www.seleniumhq.org/ (2019/01/20).

184. L. Crispin and J. Gregory, *Agile Testing: A Practical Guide for Testers and Agile Teams*. Pearson Education, 2008.

185. E. Vila, G. Novakova, and D. Todorova, “Automation testing frame-work for web applications with selenium webdriver: Opportunities and threats,” in *Proceedings of the International Conference on Advances in Image Processing*, 2017, pp. 144–150.

186. S. Kapoor and K. Sagar, “Autojet: Web application automation tool,” in *Towards Extensible and Adaptable Methods in Computing.* Springer, 2018, pp. 27–42.

187. R. G. Cooper, *Winning at New Products: Accelerating the Process from Idea to Launch*. Addison-Wesley, 1993.

188. M. Hoegl and H. G. Gemuenden, “Teamwork Quality and the Success of Innovative Projects: A Theoretical Concept and Empirical Evidence,” *Organization Science*, vol. 12, no. 4, pp. 435–449, 2001.

189. E. Weimar, A. Nugroho, J. Visser, A. Plaat, M. Goudbeek and A. P. Schouten,“ The influence of teamwork quality on software team performance,” *arXiv preprint arXiv*:1701.06146, 2017.

190. E. Salas, D. E. Sims, and C. S. Burke, “Is there a ‘Big Five’ in Teamwork?,” *Small Group Research*, vol. 36, no. 5, pp. 555–599, 2005.

191. J. Highsmith, *Agile Software Development Ecosystems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

192. C. VersionOne “7th Annual State of Agile Development Survey| CollabNet VersionOne.” Available at https://www.versionone.com/pdf/7th-Annual-%20State-%20of-Agile-Development-Survey.pdf (2018/01/22).

193. S. Jalali and C. Wohlin, “Agile Practices in Global Software Engineering - A Systematic Map,” in *2010 5th IEEE International Conference on Global Software Engineering*, 2010, pp. 45–54.

194. S. V. Shrivastava and U. Rathod, “Categorization of risk factors for distributed agile projects,” *Information and Software Technology*, vol. 58, pp. 373–387, 2015.

195. E. Hossain, M. A. Babar, H. Paik, and J. Verner, “Risk Identification and Mitigation Processes for Using Scrum in Global Software Development: A Conceptual Framework,” in *2009 16th Asia-Pacific Software Engineering Conference*, 2009, pp. 457–464.

196. A. Extension “Agile Extension | IIBA | International Institute of Business Analysis.” Available at :https://www.iiba.org/standards-and-resources/agile-extension/(2018/10/22).

197. C. S. Burke, D. E. Sims, E. H. Lazzara, and E. Salas, “Trust in leadership: A multi-level review and integration,” *The leadership quarterly*, vol. 18, no. 6, pp. 606–632, 2007.

198. M. Dayan and C. A. Di Benedetto, “The impact of structural and contextual factors on trust formation in product development teams,” *Industrial Marketing Management*, vol. 39, no. 4, pp. 691–703, 2010.

199. O. Badreddin, “Thematic review and analysis of grounded theory application in software engineering,” *Advances in Software Engineering*, vol. 2013, 2013.

200. C. Johnson-George and W. C. Swap, “Measurement of specific inter-personal trust: Construction and validation of a scale to assess trust in a specific other.” *Journal of personality and social psychology*, vol. 43,no. 6, p. 1306, 1982.

201. L. Solomon, “The influence of some types of power relationships and game strategies upon the development of interpersonal trust.” *The Journal of Abnormal and Social Psychology*, vol. 61, no. 2, p. 223, 1960.

202. T. G. Brashear, J. S. Boles, D. N. Bellenger, and C. M. Brooks, “An empirical test of trust-building processes and outcomes in sales manager-salesperson relationships,” *Journal of the Academy of Marketing Science*, vol. 31, no. 2, p. 189, 2003.

203. W. R. Clark, P. Scholder Ellen, and J. S. Boles, “An examination of trust dimensions across high and low dependence situations,” *Journal of Business-to-Business Marketing*, vol. 17, no. 3, pp. 215–248, 2010.

204. Y.-T. Hung, A. R. Dennis, and L. Robert, “Trust in virtual teams: Towards an integrative model of trust formation,” in *37th Annual Hawaii International Conference on System Sciences*, 2004. Proceedings of the. IEEE, 2004, pp. 11–pp.

205. R. C. Mayer, J. H. Davis, and F. D. Schoorman, “An integrative model of organizational trust,” *Academy of management review*, vol. 20, no. 3,pp. 709–734, 1995.

206. S. Dorairaj, J. Noble, and P. Malik, “Understanding the importance of trust in distributed agile projects: A practical perspective,” in International Conference on Agile Software Development. Springer, 2010, pp.172–177.

207. A. Puusa and U. Tolvanen, “Organizational identity and trust,” EJBO-Electronic Journal of Business Ethics and Organization Studies, 2006.

208. A. C. Costa and N. Anderson, “Measuring trust in teams: Development and validation of a multifaceted measure of formative and reflective indicators of

team trust," *European Journal of Work and Organizational Psychology*, vol. 20, no. 1, pp. 119–154, 2011.

209. N. V. Oza, T. Hall, A. Rainer, and S. Grey, "Trust in software outsourcing relationships: An empirical investigation of Indian software companies," *Information and software Technology*, vol. 48, no. 5, pp. 345–354, 2006.

210. M. A. Babar, J. M. Verner, and P. T. Nguyen, "Establishing and maintaining trust in software outsourcing relationships: An empirical investigation," *Journal of Systems and software*, vol. 80, no. 9, pp. 1438–1449, 2007.

211. B. Ramesh, L. Cao, K. Mohan, and P. Xu, "Can Distributed Software Development Be Agile?," *Communications of the ACM*, vol. 49, no. 10, pp. 41–46, Oct. 2006.

212. T. O. A. Lehtinen, R. Virtanen, J. O. Viljanen, M. V. Mäntylä, and C. Lassenius, "A tool supporting root cause analysis for synchronous retrospectives in distributed software teams," *Information and Software Technology*, vol. 56, no. 4, pp. 408–437, 2014.

213. M. Paasivaara and C. Lassenius, "Could Global Software Development Benefit from Agile Methods?," in *2006 IEEE International Conference on Global Software Engineering(ICGSE'06)*, 2006, pp. 109–113.

214. M. Vax and S. Michaud, "Distributed Agile: Growing a Practice Together," in *Agile 2008 Conference*, 2008, pp. 310–314.

215. C. Young and H. Terashima, "How Did We Adapt Agile Processes to Our Distributed Development?," in *Agile 2008 Conference*, 2008, pp. 304–309.

216. S. Jalali and C. Wohlin, "Agile Practices in Global Software Engineering - A Systematic Map," in *2010 5th IEEE International Conference on Global Software Engineering*, 2010, pp. 45–54.

217. S. Tyagi, R. Sibal, and B. Suri, "Role of trust in distributed agile software development teams-a light weight systematic literature review," *Journal on Management Studies*, no. 4, pp. 748–753, 2018.

218. M. Korkala, M. Pikkarainen, and K. Conboy, "Distributed Agile Development: A Case Study of Customer Communication Challenges," in

Agile Processes in Software Engineering and Extreme Programming, 2009, pp. 161–167.

219. O. McHugh, K. Conboy, and M. Lang, "Using Agile Practices to Build Trust in an Agile Team: A Case Study," in *Information Systems Development*, 2011, pp. 503–516.

220. S. Hole and N. B. Moe, "A Case Study of Coordination in Distributed Agile Software Development," in *Software Process Improvement*, 2008, pp. 189–200.

221. B. Ramesh, L. Cao, J. Kim, K. Mohan, and T. L. James, "Conflicts and complements between eastern cultures and agile methods: an empirical investigation," *European Journal of Information Systems*, vol. 26, no. 2, pp. 206–235, Mar. 2017.

222. M. Paasivaara and C. Lassenius, "Using Scrum Practices in GSD Projects," in *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*, D. Šmite, N. B. Moe, and P. J. Ågerfalk, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 259–278.

223. S. Lee and H.S. Yong, Distributed agile: project management in a global environment, *Empirical Software Engineering*, vol. 15, no. 2, pp.204-217, 2010.

224. D. Wildt and R. Prikladnicki, "Transitioning from Distributed and Traditional to Distributed and Agile: An Experience Report," in *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*, D. Šmite, N. B. Moe, and P. J. Ågerfalk, Eds. Berlin, Heidelberg, 2010, pp. 31–46.

225. N. B. Moe, T. E. Fægri, D. S. Cruzes, and J. E. Faugstad, "Enabling Knowledge Sharing in Agile Virtual Teams," in *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 2016, pp. 29–33.

226. S. Dorairaj, J. Noble, and P. Malik, "Effective Communication in Distributed Agile Software Development Teams," in *Agile Processes in Software Engineering and Extreme Programming*, 2011, pp. 102–116.

227. R. Hoda, J. Noble, and S. Marshall, "Supporting Self-organizing Agile Teams - What's Senior Management Got to Do with It?," in Lecture Notes in Business Information Processing, 2011, vol. 77, pp. 73–87.

228. S. Dorairaj and J. Noble, "Agile Software Development with Distributed Teams: Agility, Distribution and Trust," in *2013 Agile Conference*, 2013, pp. 1–10.

229. J. B. Rotter, "Interpersonal trust, trustworthiness, and gullibility." *American psychologist*, vol. 35, no. 1, p. 1, 1980.

230. J. Rasmussen, "Introducing xp into greenfield projects: Lessons learned," *Ieee Software*, vol. 20, no. 3, pp. 21–28, 2003.

231. P. Lenberg and R. Feldt, "Psychological safety and norm clarity in software engineering teams," in *Proceedings of the 11th international workshop on cooperative and human aspects of software engineering*, 2018, pp. 79–86.

232. K. T. Dirks and D. L. Ferrin, "Trust in leadership: Meta-analytic findings and implications for research and practice." *Journal of applied psychology*, vol. 87, no. 4, p. 611, 2002.

233. N. A. Gillespie and L. Mann, "Transformational leadership and shared values: The building blocks of trust," *Journal of managerial psychology*, 2004.

234. S. W. Mayner, "Transformational leadership and organizational change during agile and devops initiatives," Ph.D. dissertation, Capella University, 2017.

235. G. Casimir, D. A. Waldman, T. Bartram, and S. Yang, "Trust and the relationship between leadership and follower performance: Opening the black box in australia and china," *Journal of Leadership & Organizational Studies*, vol. 12, no. 3, pp. 68–84, 2006.

236. A. A. Tabassi, K. M. Roufechaei, A. H. A. Bakar, and N. Yusof, "Linking team condition and team performance: A transformational leadership approach," *Project Management Journal*, vol. 48, no. 2, pp. 22–38, 2017.

237. J. Schaubroeck, S. S. Lam, and A. C. Peng, "Cognition-based and affect-based trust as mediators of leader behavior influences on team performance." *Journal of applied psychology*, vol. 96, no. 4, p. 863, 2011.

238. R. Biddle, A. Meier, M. Kropp, and C. Anslow, “Myagile: sociological and cultural effects of agile on teams and their members,” in *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*, 2018, pp. 73–76.

239. N. Holtzhausen and J. J. de Klerk, “Servant leadership and the scrum team’s effectiveness,” *Leadership & Organization Development Journal*, 2018.

240. N. B. Moe, D. S. Cruzes, T. Dyb˚a, and E. Engebretsen, “Coaching a global agile virtual team,” in *2015 IEEE 10th International Conference on Global Software Engineering*. IEEE, 2015, pp. 33–37.

241. F. Laloux, “The future of management is teal,” *Strategy+ business*, vol. 6, 2015.

242. K. K. Merriman, “On the folly of rewarding team performance, while hoping for teamwork,” *Compensation & Benefits Review*, vol. 41, no. 1, pp. 61–66, 2009.

243. A. M. Evans and W. Revelle, “Survey and behavioral measurements of interpersonal trust,” *Journal of Research in Personality*, vol. 42, no. 6, pp. 1585–1593, 2008.

244. S. D. Vishnubhotla, E. Mendes, and L. Lundberg, “Investigating the relationship between personalities and agile team climate of software professionals in a telecom company,” *Information and Software Technology*, vol. 126, p. 106335, 2020.

245. S. Moore and L. Barnett, “Offshore outsourcing and agile development,” *Forrester Research*, Inc, 2004.

246. L. Fernández-Sanz and S. Misra, “Influence of Human Factors in Software Quality and Productivity,” in *Computational Science and Its Applications - ICCSA 2011*, 2011, pp. 257–269.

247. T.-P. Liang, J. C.-H. Wu, J. J. Jiang, and G. Klein, “The impact of value diversity on information system development projects,” *International Journal of Project Management*, vol. 30, no. 6, pp. 731–739, 2012.

248. R. G. Cooper, *Winning at New Products: Accelerating the Process from Idea to Launch*. Addison-Wesley, 1993.

249. M. McCallum, "Building trust in diverse teams: A concise facilitation handbook," Oxfam GB, 2011.

250. D. Straker, "What is Trust?" Available at http://changingminds.org/explanations/trust/what_is_trust_htm (2019/11/09).

251. B. Fitzgerald, G. Hartnett, and K. Conboy, "Customizing agile methods to software practices at Intel Shannon," *European Journal of Information Systems*, vol. 15, no. 2, pp. 200–213, Apr. 2006.

252. O. McHugh, K. Conboy, and M. Lang, "Agile practices: The impact on trust in software project teams," *Ieee Software*, vol. 29, no. 3, pp. 71–76, 2011.

253. E. Hasnain and T. Hall, "Investigating the role of trust in agile methods using a light weight systematic literature review," in *International Conference on Agile Processes and Extreme Programming in Software Engineering*. Springer, 2008, pp. 204-207.

254. S. P. Marsh, "Formalising trust as a computational concept," 1994.

255. D. Gambetta, "Trust: Making and breaking cooperative relations," 1988.

256. K. J. Blois, "Trust in business to business relationships: An evaluation of its status," *Journal of Management Studies*, vol. 36, no. 2, pp. 197–215, 1999.

257. M. A. Hall, F. Camacho, E. Dugan, and R. Balkrishnan, "Trust in the medical profession: conceptual and measurement issues," *Health services research*, vol. 37, no. 5, pp. 1419–1439, 2002.

258. E. Hasnain, T. Hall, and M. Shepperd, "Using experimental games to understand communication and trust in agile software teams," in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering* (CHASE). IEEE, 2013, pp. 117–120.

259. R. Zolin, P. J. Hinds, R. Fruchter, and R. E. Levitt, "Trust in cross-functional, global teams," *Center for Integrated Facility Engineering, Stanford University*, *Working Paper*, vol. 67, 2002.

260. W. T. Chan and F. Ying, “A social complexity approach to investigate trust in agile methodology,” in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 172–177.

261. V. Devedzic et al., “Teaching agile software development: A case study,” *IEEE Transactions on Education*, vol. 54, no. 2, pp. 273–278, 2010.

262. M. Kropp and A. Meier, “Teaching agile software development at university level: Values, management, and craftsmanship,” in *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 2013, pp. 179–188.

263. Z. Masood, R. Hoda, and K. Blincoe, “Adapting agile practices in university contexts,” *Journal of Systems and Software*, vol. 144, pp. 501–510, 2018.

264. I. SPSS, “IBM SPSS Statistics,” Available at https://www.ibm.com/support/pages/downloading-ibm-spss-statistics-27 (2020/09/05).

265. L. Gren, R. Torkar, and R. Feldt, “The prospects of a quantitative measurement of agility: A validation study on an agile maturity model,” *Journal of Systems and Software*, vol. 107, pp. 38–49, 2015.

266. B. M. Byrne, “Structural equation modeling with amos: basic concepts, applications, and programming (multivariate applications series),” *New York: Taylor & Francis Group*, vol. 396, p. 7384, 2010.

267. L. R. Fabrigar and D. T. Wegener, *Exploratory factor analysis*. Oxford University Press, 2011.

268. K. S. Shultz, D. J. Whitney, and M. J. Zickar, *Measurement theory in action: Case studies and exercises*. Routledge, 2013.

269. S. A. Wheelan, M. Akerlund, and C. Jacobsson, *Creating effective teams: A guide for members and leaders*. SAGE Publications, Incorporated, 2020.

270. A. Sidky, J. Arthur, and S. Bohner, “A disciplined approach to adopting agile practices: the agile adoption framework,” *Innovations in systems and software engineering*, vol. 3, no. 3, pp. 203–216, 2007.

271. L. Gren, R. Torkar, and R. Feldt, "Group development and group maturity when building agile teams: A qualitative and quantitative investigation at eight large companies," *Journal of Systems and Software*, vol. 124, pp. 104–119, 2017.

272. Y. Lindsjørn, D. I. Sjøberg, T. Dingsøyr, G. R. Bergersen, and T. Dyba, "Teamwork quality and project success in software development: A survey of agile development teams," *Journal of Systems and Software*, vol. 122, pp. 274–286, 2016.

273. S. T. Fairbairn, "A new muscle memory: Training systems engineers in the agile culture of trust," *INSIGHT*, vol. 21, no. 2, pp. 17–20, 2018.

274. N. B. Moe, T. E. Fægri, D. S. Cruzes, and J. E. Faugstad, "Enabling knowledge sharing in agile virtual teams," in *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*. IEEE, 2016, pp. 29–33.

275. A. Iftikhar, M. Alam, S. Musa, and M. M. Su'ud, "Trust development in virtual teams to implement global software development (gsd): A structured approach to overcome communication barriers," in *2017 IEEE 3rd International Conference on Engineering Technologies and Social Sciences (ICETSS)*. IEEE, 2017, pp. 1–5.

276. A. Alhubaishy and L. Benedicenti, "Affect and affective trust in agile requirements engineering," in *2018 1st International Workshop on Affective Computing for Requirements Engineering (AffectRE)*. IEEE, 2018, pp. 9–12.

277. I. F. Barbosa, M. P. Oliveira, P. B. Reis, T. C. Gomes, and F. Q. da Silva, "Towards understanding the relationships between interdependence and trust in software development: a qualitative research," in *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 2017, pp. 66–69.

278. M. Birks and J. Mills, *Grounded theory: A practical guide*. Sage, 2015.

279. J. W. Creswell and C. N. Poth, *Qualitative inquiry and research design: Choosing among five approaches*. Sage publications, 2016.

280. Y. S. Lincoln and E. G. Guba, "Naturalistic inquiry (vol. 75)," 1985.

LIST OF PUBLICATIONS

Journals

1. S. Tyagi, R. Sibal, B. Suri, B. Wadhwa, and S. Shekhar, "Development of reusable hybrid test automation framework for web based scrum projects," *Journal of Applied Science and Engineering*, vol. 21, pp. 455–462, 2018.
2. S. Tyagi, R. Sibal, B. Suri, "Role of Trust in Distributed Agile Development Teams: A Light Weight Systematic Literature Review," *Journal on Management Studies*, ICT, pp. 748-754, 2018.
3. S. Tyagi, R. Sibal, B. Suri, "A Transition Journey of Adopting Test Automation on Agile Projects: An Empirical Investigation," *Journal of King Saud University-Computer and Information Sciences*, Elsevier, 2020, Under Review.
4. S. Tyagi, R. Sibal, B. Suri, "Empirically Developed Framework for Building Trust in Distributed Agile Teams," *Journal of Information and Software Technology*, Elsevier, 2020, Under Review.
5. S. Tyagi, R. Sibal, B. Suri, "Assessing the Impact of Agile Software Development Education on Students Team Performance," *Computer Applications in Engineering Education*, Wiley, 2020, Under Review.

Conference Proceedings

1. S. Tyagi, R. Sibal, and B. Suri, "Adopting Test Automation on Agile Development Projects: A Grounded Theory Study of Indian Software Organizations," in *International Conference on Agile Software Development*, *Cologne, Germany*, Springer, cham, pp. 184–198, 2017

2. S. Tyagi, R. Sibal, B. Suri."Role of Trust in Distributed Agile Development Teams: A Light Weight Systematic Literature Review" *TEAMC*, *International conference on extensible computing*, *New Delhi, India,* 2018.