

**DESIGN OF SOFTWARE TESTING TECHNIQUES  
IN AGILE DEVELOPMENT**

**THESIS**

*submitted in fulfillment of the requirement of the degree of*

**DOCTOR OF PHILOSOPHY**

*to*

**YMCA UNIVERSITY OF SCIENCE & TECHNOLOGY**

*by*

**ANITA**

**Registration No. YMCAUST/PH11/2010**

*Under the Supervision of*

**DR. NARESH CHAUHAN**

**PROFESSOR**



**Department of Computer Engineering**

**Faculty of Engineering & Technology**

**YMCA University of Science & Technology**

**Sector-6, Mathura Road, Faridabad, Haryana, India**

**AUGUST, 2016**

## **CANDIDATE'S DECLARATION**

I hereby declare that this thesis entitled **DESIGN OF SOFTWARE TESTING TECHNIQUES IN AGILE DEVELOPMENT**, being submitted in fulfillment of the requirements for the Degree of Doctor of Philosophy in **DEPARTMENT OF COMPUTER ENGINEERING** under Faculty of **ENGINEERING & TECHNOLOGY** of YMCA University of Science & Technology Faridabad, during the academic year 2011-2016, is a bona fide record of my original work carried out under guidance and supervision of **DR. NARESH CHAUHAN, PROFESSOR, COMPUTER ENGINEERING** and has not been presented elsewhere.

I further declare that the thesis does not contain any part of any work which has been submitted for the award of any degree either in this university or in any other university.

(ANITA)

Registration No. YMCAUST/PH11/2010

## **CERTIFICATE OF SUPERVISOR**

This is to certify that the thesis entitled “**DESIGN OF SOFTWARE TESTING TECHNIQUES IN AGILE DEVELOPMENT**” by ANITA submitted in fulfillment of the requirement for the award of Degree of Doctor of Philosophy in **DEPARTMENT OF COMPUTER ENGINEERING** under Faculty of **ENGINEERING & TECHNOLOGY** of YMCA University of Science & Technology Faridabad, during the academic year May 2011-Aug 2016, is a bonafide record of work carried out under my guidance and supervision.

I further declare that to the best of my knowledge, the thesis does not contain any part of any work which has been submitted for the award of any degree either in this university or in any other university.

**(Signature of Supervisor)**  
**DR. NARESH CHAUHAN**  
**PROFESSOR**

Department of Computer Engineering  
Faculty of Engineering & Technology  
YMCA University of Science & Technology Faridabad,

Dated: August , 2016

## ACKNOWLEDGEMENT

*I am thankful to God for all of His Blessings.*

I would like to express my sincere gratitude to my supervisor, **DR. NARESH CHAUHAN**, for giving me the opportunity to work in this area. It would not have been possible for me to take this thesis to this level without his innovative ideas, support and encouragement. He provided me continuous guidance, valuable advice, constructive criticism throughout the span of this course. He gave me the opportunity to learn various new things, and taught me a lot about research, teaching, and life.

I would like to thank my batchmates cum Faculty member, Ms. Rashmi Popli and Mr. Harish Kumar, and all other faculty members at YMCA University of Science & Technology for their valuable technical discussions and cooperation. I am also thankful to *family members of Dr. Naresh Chauhan* who helped me indirectly in completing my research work.

I am thankful to my mother and grandparents for their love, encouragement, and support throughout my education.

I am also thankful to my family and friends for their continuous and valuable support for motivating me time to time.

A word of special thanks to my father **Sh. Nand Lal** who made many compromises to let me start and finish my Ph.D. I am thankful to him for his never-ending patience, guidance and support. Without his blessings this thesis would not have been completed.

Thanks to all of you!

(ANITA)

## ABSTRACT

The long term goal of customer satisfaction is the main concern for any service provider in the software industry. More specifically, the team has to put on more efforts for reaching to that particular level. In an Agile software development, small team which is committed to small duration sprint handles several challenges while delivering deliverable to the client. On the other side, frequent changes are added by the client so as to have product which is as per the market standard or competitor product's feature. These changes may bring along additional burden for the team. This burden may be in terms of test suite management, working in distributed management. For handling, test suite management, an Agile testing life cycle has been proposed which is focused on regression testing.

In Agile, changes are framed into user story for the sprint and then there is a need to analyze these framed user stories so as to have effective test suite management. This test suite management is based on the proposed testing scenarios used in the Agile development environment. The proposed scenario is scattered among various testing activities/types which are performed by all team members collectively as compared with traditional software development testing scenario.

In Agile, team works in pair programming style of culture under the supervision of Scrum master in SCRUM methodology. When XP methodology is used then self organizing principle is followed. Using pair programming and refactoring practices in distributed environment, simplified processes are adopted by distributed stakeholders. Further, for pair programming practice, a pair may be formed using proposed technique which is based on self centric approach. Furthermore, usage of refactoring has been extended using object oriented principles. This proposed extension comprises simplification of code for future scalability which is based on changes proposed by client. The post simplification step includes test suite/case management. The test suite/case management has been implemented using three test case prioritization techniques.

The first proposed technique is based on pre-identification of risk for user stories and which further laid to prioritization of these user stories and further, prioritization of their test cases. The second proposed technique for test suite management is based on proposed test pattern for common problem of test suite management. This approach is based on the object

linkages among primary and secondary classes. Further, dependence diagram has been used for identifying dependencies among dependent set of classes.

The last approach for test case prioritization is based on linguistic parameters such as noun, verb and punctuations. Using punctuations or pauses in the user story, user stories can be prioritized. Generally, more pauses in a user story means more linkages and more effort required for handling more issues for these more linkages. After prioritizing user stories, next step is to manage test cases of these user stories using nouns and verbs identification in the respective user stories. Further, after identifying nouns and verbs in the user stories, sentence priority score may be calculated which is based on story point of the user story, customer priority, number of nouns and number of verbs in the test cases of the user stories.

Furthermore, the effectiveness of proposed test case prioritization has been proved using average percentage of fault detection.

# TABLE OF CONTENTS

Candidate's Declaration	i
Certificate of Supervisor	ii
Acknowledgement	iii
Abstract	iv
Table of Contents	v
List of Tables	vi
List of Figures	vii
List of Abbreviations	viii
<b>CHAPTER 1: INTRODUCTION</b>	<b>1</b>
1.1 Agile Software Development	1
1.2 Motivation And Goals	2
1.3 Challenges Related To Testing In Agile Distributed Environment	3
1.4 Organization of Thesis	4
<b>CHAPTER 2: LITERATURE SURVEY</b>	<b>7</b>
2.1 Introduction	7
2.2 Software Development Life Cycle (SDLC)	7
2.2.1 Linear Sequential Model	8
2.2.2 Prototyping Model	8
2.2.3 Iterative Incremental Model	9
2.2.4 Spiral Model	10
2.3 Agile Development Approach	11
2.3.1 History and Basic Principles of Agile	12
2.3.2 Guidelines For Creating Agile Environment	14
2.4 Agile Software Development Life Cycle	18
2.5 Agile Practices	21
2.5.1 Pair Programming	22
2.5.2 Test Driven Development (TDD)	23

2.5.3	Collective Ownership	23
2.5.4	Continuous Integration	23
2.5.5	Planning Game	23
2.5.6	Refactoring	23
2.6	Scrum	24
2.7	Color Coding Scheme For Bug Tracking	24
2.8	Testing Quadrant Matrix	25
2.9	Traditional Risk Register	26
2.10	Regression Testing	27
2.11	Communication in Distributed Environment	30
2.11.1	Communication	32
2.11.2	User Story	37
2.11.3	Modified User Story	38
2.12	Linguistic Parameters	43
<b>CHAPTER 3: AGILE TESTING LIFE CYCLE</b>		<b>45</b>
3.1	Introduction	45
3.2	Agile Testing Life Cycle	46
3.2.1	Agile Inner Testing	50
3.2.2	Online Bug Tracker	51
3.2.3	Sprint Flow Diagram	52
3.2.4	Benefits of Agile Testing Life Cycle	55
3.3	Conclusion	56
<b>CHAPTER 4: AGILE TESTING IN DISTRIBUTED ENVIRONMENT</b>		<b>57</b>
4.1	Introduction	57
4.2	Agile Testing in Distributed Environment	58
4.3	Proposed Framework for Distributed Environment	59
4.4	Pair Programming In Distributed Environment	62
4.4.1	Proposed Buddy Identifier	62
4.5	Refactoring in Distributed Environment	67



4.6 Refactoring Example	68
4.7 Benefits	71
4.8 Conclusion	72
<b>CHAPTER 5: AGILE REGRESSION TEST SELECTION TECHNIQUE</b>	<b>73</b>
5.1 Introduction	73
5.2 Proposed Approach for Regression Test Selection	74
5.3 Case Study	75
5.4 RTS Tool	83
5.5 Conclusion	83
<b>CHAPTER 6: AGILE REGRESSION TEST CASE PRIORITIZATION</b>	<b>87</b>
6.1 Introduction	87
6.2 Proposed Risk Based TCP Technique	88
6.2.1 User Story Graph	89
6.2.2 Complete User Story Matrix	91
6.3 Outcomes	96
6.4 Proposed Pattern based TCP Technique	98
6.4.1 Proposed “Test” Pattern	99
6.4.2 Case Study	101
6.4.3 Results & Analysis	104
6.4.4 Phases in Test Pattern	106
6.5 Linguistic Based Object Oriented TCP Technique	107
6.5.1 Story Prioritization	109
6.5.2 Case Study	111
6.5.3 Implementation	122
6.6 Conclusion	127
<b>CHAPTER 7: CONCLUSIONS AND FUTURE WORK</b>	<b>129</b>
<b>REFERENCES</b>	<b>131</b>

## LIST OF TABLES

<b>Table</b>	<b>Page No.</b>
Table 2.1 Comparison Chart	11
Table 2.2 Agile Terminology	16
Table 2.3 Actor Activity Chart	21
Table 2.4 Task Chart	34
Table 2.5 User Stories	42
Table 3.1 Agile Testing Abbreviations	48
Table 3.2 Online Bug Tracker	51
Table 4.1 Testing Abbreviations	69
Table 5.1 Requirements	76
Table 5.2 Non-Existing Edges	79
Table 5.3 Existing Edges	80
Table 5.4 Optimal Path	81
Table 5.5 Second Level Optimization	82
Table 6.1 Story Effort	90
Table 6.2 First Level Dependence Matrix	91
Table 6.3 Questions-Volatility Rate	92
Table 6.4 Questions-Implementation Dependency	93
Table 6.5 Effort Data	94
Table 6.6 Complete User Story Matrix	96
Table 6.7 Risk Measure Matrix	97
Table 6.8 Test Pattern	100
Table 6.9 Dependence Diagram	104
Table 6.10 Story Board	105
Table 6.11 Generated Test Suite	106
Table 6.12 New Test Suite	106
Table 6.13 User Requirements	112
Table 6.14 Noun table	112
Table 6.15 Verb Table	112

Table 6.16 Ready Story 1- Acceptance Criteria	113
Table 6.17 Ready Story 2- Acceptance Criteria	113
Table 6.18 Ready Story 3- Acceptance Criteria	113
Table 6.19 Ready Story 4- Acceptance Criteria	113
Table 6.20 Ready Story 5- Acceptance Criteria	113
Table 6.21 Ready Story 6- Acceptance Criteria	113
Table 6.22 Ready Story 7- Acceptance Criteria	114
Table 6.23 Ready Story 8- Acceptance Criteria	114
Table 6.24 Ready Story-Noun Table	114
Table 6.25 Ready Story-Verb Table	114
Table 6.26 New Requirement	115
Table 6.27 New Requirement- Noun Table	115
Table 6.28 New Requirement- Verb Table	115
Table 6.29 Ready Story 9- Acceptance Criteria	115
Table 6.30 New Ready Story-Noun Table	115
Table 6.31 New Ready Story-Verb Table	115
Table 6.32 Noun Dependent Story Table_a	115
Table 6.33 Noun Dependent Story Table_b	116
Table 6.34 Verb Dependent Story Table_a	116
Table 6.35 Verb Dependent Story Table_b	116
Table 6.36 Test Cases Story 1	117
Table 6.37 Test Cases Story 2	117
Table 6.38 Test Cases Story 3	118
Table 6.39 Test Cases Story 4	118
Table 6.40 Test Cases Story 9	119
Table 6.41 Sentence Priority Score Story 9	120
Table 6.42 Sentence Priority Score Sorting	121
Table 6.43 TCP Ordering	126
Table 6.44 No Ordering	127

## LIST OF FIGURES

<b>Figure</b>	<b>Page No.</b>
Fig. 2.1 The Classic Waterfall Model	8
Fig. 2.2 Spiral Model	10
Fig. 2.3 Agile Methodologies	14
Fig. 2.4 Agile Manifesto	19
Fig. 2.5 Agile Life Cycle	20
Fig. 2.6 Pair Programming	22
Fig. 2.7 Scrum Life Cycle	25
Fig. 2.8 Color Coding Tracking Scheme	26
Fig. 2.9 Testing Matrix	27
Fig. 2.10 Normal Agile Testing	29
Fig. 2.11 Week Plus Testing	29
Fig. 2.12 Agile Scenario	33
Fig. 2.13 Team Communication	35
Fig. 2.14 Verbal Communication	36
Fig. 2.15 Non-verbal Communication	37
Fig. 2.16 Story Format	37
Fig. 2.17 Confirming Points	38
Fig. 2.18 Stakeholders Understanding	39
Fig. 2.19 Researcher Database	40
Fig. 2.20 Project Requirement	41
Fig. 2.21 Story Script	43
Fig. 2.22 Confirming points_10	44
Fig. 3.1 More is less	46
Fig. 3.2 Agile Testing Life cycle	47
Fig. 3.3 Regression Testing Matrix	49
Fig. 3.4 Testing Scenario in Pre-execution Phase	53
Fig. 3.5 Testing Scenario in Execution Phase	54
Fig. 3.6 Testing Scenario in Post-execution Phase	55

Fig. 4.1 Framework for Distributed Environment	61
Fig. 4.2 Ego centric graph	62
Fig. 4.3 Common Habit Representation	63
Fig. 4.4 Common Cluster	64
Fig. 4.5 South Cluster	65
Fig. 4.6 Overlapping Cluster	65
Fig. 4.7 A pair	66
Fig. 4.8 Buddy Rotation	66
Fig. 4.9 Refactoring using Object Oriented Principles in Distributed Environment	69
Fig. 5.1 A Path Strength based RTS Technique	75
Fig. 5.2 Weighted Story Graph	77
Fig. 5.3 Optimized Weighted Story Graph	82
Fig. 5.4 Snapshot 1	84
Fig. 5.5 Snapshot 2	84
Fig. 5.6 Snapshot 3	85
Fig. 5.7 Snapshot 4	85
Fig. 5.8 Snapshot 5	86
Fig. 6.1 Agile Environment	89
Fig. 6.2 Risk Based Model	90
Fig. 6.3 User Story Graph	91
Fig. 6.4 BFS Based Indirect Path Implementation	95
Fig. 6.5 Working Model for pattern based TCP	101
Fig. 6.6 Patent Grant Procedure	102
Fig. 6.7 Relationship Diagram	103
Fig. 6.8 Test Pattern	107
Fig. 6.9 Linguistic TCP Technique	109
Fig. 6.10 Total Punctuation	110
Fig. 6.11 Story Prioritization	111
Fig. 6.12 Sequential Approach	122
Fig. 6.13 Noun Count	124

Fig. 6.14 Verb Count	125
Fig. 6.15 Effect	125
Fig. 6.16 Comparison	128

## LIST OF ABBREVIATIONS

Software Development Life Cycle	SDLC
Software Requirements Specification	SRS
Multi National Company	MNC
Feature Driven Development	FDD
Rational Unified Process	RUP
Dynamic Software Development Method	DSDM
Object Oriented Design	OOD
Graphical User Interface	GUI
Test Driven Development	TDD
Scrum Master	SM
Product Owner	PO
Product Backlog List	PBL
Sprint Backlog List	SBL
Extreme Programming	XP
Test Driven Development	TDD
Information Technology	IT
Sprint level regression testing	SLRT
End to end regression testing	EERT
Agile Software Development	ASD
Client Representative	CR
Voice Text Markup Language	VTML
Definition Of Done	DOD
Video Conference	VC
Face To Face	FTF
Daily Scrum Meetings	DSM
Knowledge Process Outsourcing	KPO
Test Case Prioritization	TCP
Regression Test Selection	RTS
Average Percentage Of Fault Detection	APFD

# CHAPTER I

## INTRODUCTION

### 1.1 AGILE SOFTWARE DEVELOPMENT

Various SDLCs are available such as linear sequential model, prototyping model, iterative and incremental model, spiral model and many more. These traditional models follow the static working strategy in which everything is fixed for example cost of the project, requirements or scope of the project and schedule which is to be followed during development of the project. Moreover, duration of the project is in years. Every one of them suffers from one or more issues.

Agile software development (ASD) [1, 25, 27, 37, 66] is amalgamation of best practices which may be implemented by team members using Agile core values such as commitment, communication, coordination, continuous learning and continuous improvement. It may introduce a unique evolution of responding to change rather than sticking to a fixed plan as it inherits the features of iterative and incremental model [59] along with adaptability [41] and predictive feature. ASD is based on light weight methodologies [43] having dynamic nature. Not only methodologies are dynamic rather Agile itself is dynamic. This dynamic nature of Agile is its major strength and its nimbleness is reflected in each and every activity which is performed before the iteration, during the iteration and after the iteration. In short, Agile follows ‘less is more’ approach having less number of requirements, less team members in a team, small iterations/sprints, small duration, less documentation, small meetings and many more. This leads to improve quality for the software products delivered to the customer. Many software organizations have started to use Agile principles and practices for the purpose of attaining quality in their deliverables. Largely, organizations are accepting this model as it provides an environment to accommodate frequent changes which are as per the market standard and as per the customer need.

Some of the popular methodologies of Agile are extreme programming (XP) [21, 34] and scrum [49]. Mostly, these methodologies are supported by pair programming, refactoring, simple design and test driven development (TDD) [18, 20, 82]. Also, on-site



customer presence is one of the evolutionary steps in ASD. Frequent feedback and frequent delivery helps in promoting long term goal of customer satisfaction.

## **1.2 MOTIVATION AND GOALS**

Quality is valuable to the customer as he is the major stakeholder in any development process. Here, onus lies on development team to satisfy the customer. Also, quality in the product is reflection of an organization. Software testing is one of the phases of software development life cycle (SDLC) which is used by many of the software organizations to achieve quality. In ASD, testing is not performed in single phase like traditional models rather it is an ongoing activity, so, management of test cases is a challenging task. Agile promotes that it is based on responding to change over following a strict plan. The occurrence of frequent changes in Agile may have crucial aftereffects if necessary steps are not taken at right time. Thus, cumulated test suite may become a hurdle after number of sprints in a large scale project. So, there is a need to perform regression testing by effective techniques so as to reduce the size of pending backlogs of user stories and test suite respectively. Further, in distributed [39, 62] Agile, testing using pair programming practice may be cumbersome as first team members of the pair may be at one location and second member of the pair may be at different location. In that scenario, other issues also arise such as language barrier, cultural barrier, time zone barrier for doing effective communication among team members of the sprint. Therefore, quality may lag in software products.

The objective of this research is to improve the quality in Agile projects and satisfy the customer by performing regression testing in distributed environment in an efficient manner. To achieve this objective, the work on following goals has been performed in this thesis:

- To design a transitional testing model by mentioning tester specific activities during Agile life cycle.
- To design a framework for Agile testing in distributed environment.
- To design regression test selection (RTS) technique to manage test cases in an Agile environment.

- To design test case prioritization (TCP) techniques to manage test suite of user stories (requirements defined by customer) in distributed Agile environment.

### **1.3 CHALLENGES RELATED TO TESTING IN AGILE DISTRIBUTED ENVIRONMENT**

In Agile, testers work in different roles unlike traditional models. Here, tester is involved with every stakeholder. As the number of sprints increases, test suite size also grows, so management of test cases becomes a problem in a distributed environment. A critical look at the available literature [40, 46, 57, 67] indicates that the following issues need to be addressed towards testing activities in an Agile environment.

**Agile testing life cycle with testing activities:** The issue is how an Agile tester can perform other activities along with testing activities. Also, sequence of various activities along with regression testing in Agile testing life cycle is missing in the existing literature [6, 24].

*Solution: An Agile testing life cycle has been proposed wherein a tester is interacting with all stakeholders. Specifically, tester role has been defined by mentioning testing activities for delivering quality product to the customer. Further, the effective role of regression testing has been identified.*

**Agile testing in Distributed environment:** Agile testing is implemented using pair programming practice which is followed to get the quality product. The issue is how to perform testing when team members are not collocated or customer is not collocated. Another issue is how to make pairs for pair programming or testing [55].

*Solution: A framework for distributed environment [7] has been proposed which involves solutions for issues which are faced by distributed team members and customer. More specifically, refactoring [14] has been recommended for handling distributed barriers by following simple design practice. Further, a pattern, having an evolved solution from many best existing solutions, may be used to resolve the issue of similar problems in distributed environment. Furthermore, for Pair testing/Pair programming, a self centric approach (common attributes of team member) has been proposed to form a pair among team members.*

**Regression testing in Agile environment:** Since responding to change is frequent in Agile, this may have substantial effect on the dependent modules [4, 40, 44, 57] and accordingly number of test cases increases. Thus, need of regression testing increases. Therefore, a regression testing technique is required for managing the increased set of test cases.

*Solution: A regression test selection technique [10] has been proposed which is based on the user story graph, having connections among user stories. Further, two parameters namely average path value and average path length have been used to find an optimal path out of all the paths including non existing path and existing path in the user story graph. Further, a tool has been designed for the proposed regression test selection technique in Microsoft excel.*

**Test case prioritization in distributed environment:** The issue is to manage test cases of a sprint in distributed environment when responding to change is frequent and existing user stories are affected because of that change [71].

*Solution: A linguistic approach [8] has been proposed for test case prioritization which is based on number of pauses identified in user stories for story prioritization and number of noun and verbs in user stories for test cases prioritization. Further, a risk based technique [11] has been proposed which is based on identifying high risky story for doing prioritization of test cases. Also, a tool has been designed for the proposed test case prioritization technique in Microsoft excel.*

## **1.4 ORGANIZATION OF THESIS**

The thesis has been organized in the following chapters:

**Chapter 2:** The basic concepts of traditional software development life cycles have been briefly discussed in this chapter. Further, a detailed review of Agile software development life cycle along with its practices has been provided. This chapter also discloses in brief about regression testing. Moreover Communication has been discussed from Agile point of view using Adobe Captivate Tool.

**Chapter 3:** An Agile testing life cycle from a tester perspective has been proposed. Further, Sprint flow diagram in line with the Agile testing has been proposed by considering its three phases.

**Chapter 4:** The fourth chapter is related to proposed framework for Agile distributed environment. In this chapter, pair programming and refactoring has been discussed in detail for managing issues of distributed environment.

**Chapter 5:** In fifth chapter, a Regression test selection technique has been proposed. Further, the proposed technique has been implemented, by considering case study, using Microsoft Excel.

**Chapter 6:** This chapter is related with Agile TCP techniques. Three techniques have been proposed to manage test cases. The first technique is based on risk measure matrix. Second technique is based on object oriented relationship and the last technique is based on linguistic parameters such as noun and verbs. Further, effectiveness of TCP has been proved using APFD Metric.

**Chapter 7:** It concludes the outcome of the work proposed in this thesis. It also endeavors to explore the possibilities of future research work based on the proposed design.



# **CHAPTER II**

## **LITERATURE SURVEY**

### **2.1 INTRODUCTION**

The software to be released by the development team passes through several phases and several team members of the development team work as per the assigned tasks. These phases are milestones for looking at the intermediate progress of the software. The collection of these intermediate stages is software development life cycle (SDLC) for any software project. There are numerous SDLCs that are used by software professionals since decades. Specifically, an organization may follow traditional SDLCs like Linear Sequential Model, Prototyping Model [70], Iterative-Incremental model, Spiral Model etc. Generally in traditional models, life cycle is properly defined and also phases are executed by specifying needed input and output parameters from software requirements specification (SRS) document. The description regarding all these models is given in the subsequent sections of this chapter.

Companies have started to drift from traditional SDLCs models to Agile environment for the purpose of attaining quality and for the sake of saving cost and time. Agility is bringing in responsibility and ownership in individuals, which will eventually bring out effectiveness and efficiency in software deliverables. Agile model [6] is growing in the market at very good pace. Nimbleness nature of Agile is helpful in frequent releases so as to satisfy the customer by providing frequent dual feedback. This chapter is in context of traditional SDLCs, Agile, Agile terminology, Agile methodologies, Agile life cycle [12], Agile quadrant matrix and related text.

### **2.2 SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)**

Traditional methodologies create a schedule at the beginning of a project and to stick to this schedule for the life of the software project. Complex software systems can be built in a sequential, phase-wise manner where all of the requirements are gathered at the beginning, all of the design is completed next, and finally the master design is implemented into quality software. This approach conveys that complex systems can be built in a single pass, without going back and revisiting requirements or design ideas in

light of changing business or technology conditions. Yet a common complaint is that the users keep on changing their requirements. A mapping [23, 72, 73] from traditional to Agile models is also possible.

### 2.2.1 Linear Sequential Model

It was first proposed in 1970 by W.W. Royce. Royce advocated iterations of waterfalls adapting the results of the precedent waterfall. In it, development flows steadily through requirements analysis, design implementation, coding, verification & validation testing, and maintenance as shown in Figure 2.1. This model formed the basis for the many software frameworks. In it, with every phase one deliverable is compulsory. This is basically document driven model in which proper sequence is maintained. Problem of this classic approach is mainly inflexibility which is related with change in customer mind set by seeing changing needs of time. Also, bugs keeps on propagating from one phase to another. On the other side, in Agile environment [52, 74, 77] requirements keep on changing as per the market need and business value.

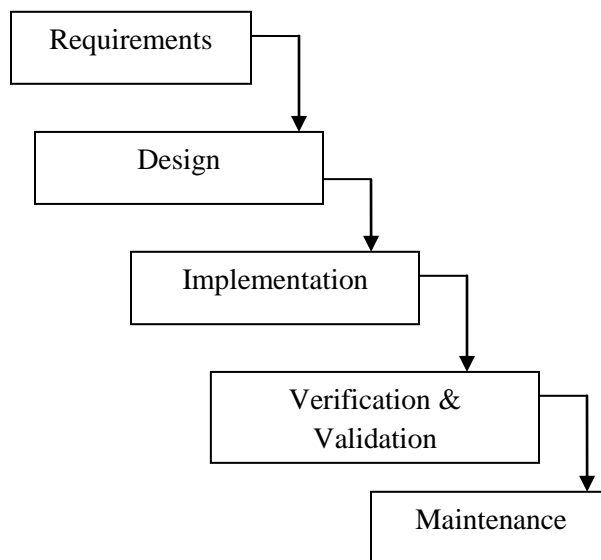


Figure 2.1 The Classic Waterfall Model

### 2.2.2 Prototyping Model

The process of this model involves many small activities viz identification of basic requirements, developing of initial prototype, review and enhancing of prototype. There are two types of prototyping including close-ended or throwaway prototyping and

breadboard or evolutionary prototyping. In the former case, prototype of the requirement is created that will eventually be discarded rather than becoming part of the final delivered software. The main goal of latter is to build a robust prototype in structured manner and constantly refine and rebuilt it. This acknowledges that work on those requirements that are well understood. The problem with this prototyping approach of software development is the cost and time as in the former case prototype is thrown away after reviewed by the customer. Also, presence of customer may be an issue.

In Agile way of software development [71], customer interactions are more important and either customer or one of the representative of customer is always present with the team members so that feedback can be received for the improvement at any time and requirements can be modified by changing trends of market. Communication is the basis for the good quality of the software. Also, prototype here is not created rather user stories are developed and demo is shown to the customers.

### **2.2.3 Iterative Incremental Model**

Incremental model is an evolution of waterfall model. The product is designed, implemented, integrated and tested as a series of incremental builds. The Incremental software development model may be applicable to projects where:

- Software Requirements are well defined, but realization may be delayed.
- The basic software functionality are required early.

It generates working software quickly and early during the software life cycle. It is more flexible less costly to change scope and requirements. It is easier to test and debug during a smaller iteration. Also, it is easier to manage risk because risky pieces are identified and handled during its iteration. In this case, problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

In Agile context, builds are gradually created. Then review is done with the help of demonstrations to the customer. Also, review is possible within the existing team members or product owners.



## 2.2.4 Spiral Model

The Spiral model was first defined by Barry Boehm. It combines elements of evolutionary, incremental, and prototyping models. The term spiral refers to successive iterations outward from a central starting point. The goal of it is to identify risk and focus on it early. In theory, risk is reduced in outer spirals as the product becomes more refined. Each spiral

- starts with design goals
- ends with the client reviewing the progress thus far and future direction
- was originally prescribed to last up to 2 years

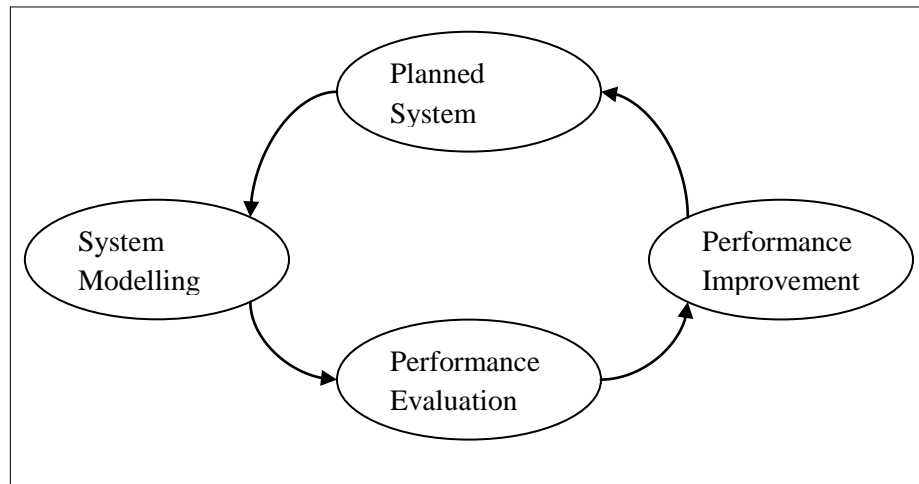


Figure 2.2 Spiral Model

The basic concepts of spiral model are planned system, system modeling, performance evaluation, performance evaluation as shown in Figure 2.2. Major applications of spiral model are mainly high risk projects in which requirements are not clearly understood, architecture is not clear, quality issues, and problem in the underlying technology. Problem with this model is its management cost as it is complex way of software development. Also, amount of documentation required in intermediate stages is a tough affair.

In Agile, manifesto says that working software is more desirable over comprehensive documentation. Here, displaying of information is preferred as compared with keeping piles of documents. A comparison [19] chart is shown below in Table 2.1.

## 2.3 AGILE DEVELOPMENT APPROACH

Agile [64] is light weight style of software development as it is document free implementation model. Also, it does not carry with itself complexities of large system. In Agile, small iterations/sprints are executed by following Agile Methodologies such as XP, Scrum [62, 75], Crystal Clear [26] etc. Agile software project management is not just departure from traditional development models or adopting Extreme Programming [21] (XP) approach of coding. In fact, it is a holistic approach starting from formation of project team, analysis of customer requirement (Story Telling), adopting iterative model for design, coding, testing of small modules and frequent delivery for mutual confidence building and easy acceptance by the customer. Agile requires motivated and trustworthy people where team composition is of cross functional and self organizing, relying on face to face communication when co-located and video conferencing for off shore projects. Agile minimizes risk by developing software product through multiple deliveries in short time unit [83] (Time Boxes) and remain flexible but stop when customer is satisfied.

Table 2.1 Comparison Chart

Parameters	Spiral	Agile
<b>Risk</b>	For medium to High Risk Projects/large scale projects	Every kind of risk is handled.
<b>No. of team member</b>	Dynamic	Static during sprint
<b>Documentation</b>	More	Very less or no documentation
<b>iteration</b>	Longer span	Small
<b>Customer Presence</b>	Is not must	Is mandatory
<b>Requirements</b>	System	Sprint
<b>Time</b>	More	Less

Agile methodologies [42, 76] promote project management process that encourages stakeholder involvement, feedback, objective metrics and effective controls. Agile presence in an organization is useful for maintaining flexible response by accommodating last minute changes, reducing cost and enhancing quality. Although, Agile exists since 2001 yet there is a great need of setting up standards and best practices.

Agile development [78] is the ability of developing a software sub-system quickly and offer to the customer for using and providing feedback, while other modules/ sub-systems are still in progress. This necessitates flexible design, commitment, ownership, different work culture, team discipline, proper and continuous feedback from the customer. The salient feature of Agile development is that this is an interactive process where whatever you deliver is useful to the customer and every successive module is well tested, efficiently integrated so that performance of earlier module does not degrade in any way. This ensures that customer starts getting a feel of a new system very early. The customer start getting feel of the future product/service quite early and he is quite excited about his contribution of keeping the project on –time and in-line with the projected requirement. The aim of Agile environment is not just to satisfy the customer but to delight him by co-operating and accommodating his expectations’ which are always higher than initial story/requirement.

### **2.3.1 History And Basic Principles Of Agile**

Agile programming started in early 1990’s where concept like Extreme Programming were adopted. It had lots of road blocks and difficulties in changing work culture and organizational ethos. It had problems when dealing with large projects and multi location development teams. With globalizations, number of Multi National Company MNC want software project to be developed overseas by countries like India, Ireland, China, Philippines and Israel.

In 2001, 17 prominent professionals/ practitioners in the field of Agile development (then called "light-weight methods") came together at the Snowbird ski resort in Utah , USA to discuss frameworks of creating software in a lighter, faster, more people-centric way. They created a manifesto for Agile Software Project Management whose principles [5] are given below:

- Customer satisfaction to be ensured by fast and continuous delivery of useful software
- Working software is delivered frequently within weeks rather than months
- Working software is the principle measure of progress
- Even belated changes in requirements are accommodated by the developer
- Close, daily cooperation between customer representative and developer
- Face-to-face conversation is the best form of communication
- Agile projects are built around motivated and disciplined individuals, who should be trusted
- Continuous attention be paid to technical excellence and flexible design
- Simplicity in all stages ensure easy understanding among team members as well as with customer
- Project team be cross functional and self-organizing which can to rotated for developing various modules
- Regular adaptation to changing circumstances

These principles emphasize on face-to-face communication, and their measure of progress is working sub-modules delivered as per time block of the iteration/sprint. Agile methods usually produce less written documentation than other traditional methods like Waterfall or Spiral model where customers are asked to prioritize deliverables. In Agile, at the end of the current iteration, customer generally prefers to see working product. It refers to a group of software development methodologies (See Figure 2.3) that promote development iterations, open collaboration, and process adaptability throughout the life-cycle of the project. It chooses to do things in small increments, with minimal planning, rather than plan at length.

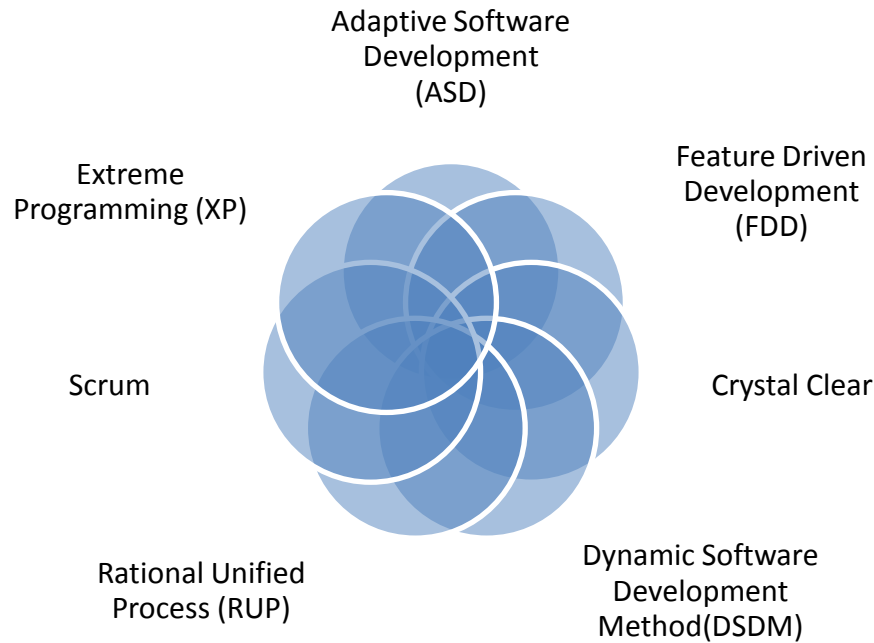


Figure 2.3 Agile Methodologies

### 2.3.2 Guidelines For Creating Agile Environment

#### Agile Terminology

Terminology and tools [32] in Agile is different from traditional terminology. Some of the important terms which are used in Agile context are shown in Table 2.2. Some of the terms used are:

- **Sprint/Iteration** A Sprint (Scrum) or iteration (XP) is a fixed period of time (Typically 2-4 week period).
- **User Story [31]** Any piece of work that can be accomplished in 1 Sprint/Iteration.
- **Ideal hours** The number of hours of uninterrupted work. For example there are 40 hours of calendar hours in 1 week. Not all 40 hours can be devoted to initiative work.
- **Velocity** The average number of hours a team can devote to initiative work in one sprint. (Calculated by Total number of ideal hours worked in 1 sprint divided by number of team members)

- **Customer/Stakeholder** Anyone who has a stake in the creation or operation of the system.
- **Stand up meetings** typically happen daily where each person describes what they did yesterday, what they are going to do today, and what road blocks they foresee.
- **Velocity Calculation** Tracking work in stories allows team to track amount of work accomplished each sprint and give an insight into Velocity to determine how much work a team can realistically accomplish.
- **Planning Tools** There are tools available to run iterative development that will allow to plan out sprints/iterations. Example Version One, Rally Dev, index cards etc.
- **Agile Project Manager** Agile Project Manager is to move away from the traditional approach of planning, preparing activity network diagram and periodic communication to the stakeholders. A good manager must ensure that he and his team can extract the essence of user requirement (the story told the customer) and evolve a flexible design to remain responsive to the changes due to technology, market trends and customer needs.
- **Agile Team Size** Agile team size is quite small where six to eight members from a team sit close as a pair on one work place. Team size relates to project size, complexity and available resources.
- **Team Composition** Agile teams are cross functional and self organizing where the role of a team could be changed after two or more iterations so that they get varied experience and no one is indispensable.
- **End Date** Iteration should not be allowed to keep recycling endlessly to achieve better results. Instead, end date for every iteration must be fixed which is called “Time Boxing” and the teams must deliver whatever they have completed by the end date. This will bring in discipline, commitment and also a measure of competence.
- **Sprint Lock** Although, there is flexibility for the customer to elaborate their requirement as the project progresses but they should not be allowed

to change the requirement during iteration (time box). Locking of requirement as given during story telling gives the development team sense of comfort that there will be no change of direction during the of the sub-module.

- **Frequent Delivery** Frequent delivery to the user of a fully tested and running code is a must. The frequency of delivery could be two weeks and in some cases it could be twice a week. Frequent delivery promotes mutual confidence between the developer and the customer, which ultimately facilitates easy acceptance of the final software product.
- **Progressive and Frequent Integration** In Agile working environment, integration of various modules (iterations) should be everyday and every week so that the sub-modules are tested progressively to build into a final product. This will require automatic build-and-test script so that the software testing is done easily and before close of the day, well-tested and fully integrated codes are delivered to the team leader.
- **Sense of Accomplishment** It is important that daily or at least twice a week, fully integrated sub-modules are tested jointly with the customers to have a sense of victory, which is also called emotional closer. This will be motivating factor to stay tuned.

Table 2.2 Agile Terminology

S.No.	Traditional Terminology	Agile Terminology
1	Requirement	User Story
2	Iteration	Sprint
3	Software Requirement Specification (Requirements)	Product Backlog List (PBL)
4	Prioritized Requirements	Sprint Backlog List (SBL)

- **Customers Delight** By early and frequent delivery of sub-modules, the customer remains excited about his/ her contribution and assured that the project is on track and their requirements is being met fully. In traditional process, the aim is to satisfy the customer by delivering a fully tested final product as per the delivery schedule while in Agile, aim is to seek delight of the customer.
- **Requirement Priorities** Initially customer will narrate various user stories for which the software is to be developed and will also indicate priorities. Customer is allowed to change the priority of various iterations and developer remains responsive to that. This helps the customer to rush-to-the-market as per their urgency.
- **Customer Participation** One of the important ingredients of the Agile project environment is that customer is co-located and readily available to participate actively during the development process. In case of off-shore project or where customer is far located, he or his representative / consultant must be invited to visit the software company and see the software in action. User viewing the development process is very important to build mutual trust.
- **Reflecting Improvement** It is like dressing up in front of the mirror, which reflects state of face and the viewer applies necessary, make up to look better. Similarly, in the case of Agile environment where team gets together prepare a list of what is working and what is not working and discuss the solutions so that it works better. It allows suitable changes in the next iterations so that when integrated the modules perform as desired by the customer. In other words, this is a process to reflect and improve continuously and remain on track on time and of quality.
- **Close Communication** One of the strength and secret of success of Agile environment is close communication among team members/ teams and with the customers. This is achieved by locating them in the same room so that they continuously discuss and contribute to the project. In case of project where co-locating of all the teams or customer in the same room



may not be possible, use of high speed intranet with web camera, micro phones and chat session are used to exchange questions and answers to various issues. In case of overseas project, video conferencing at least twice a week is very helpful.

## **2.4 AGILE SOFTWARE DEVELOPMENT LIFE CYCLE**

In today's information age with ever growing uncertainty and fast pace of life style, business community wants to stay competitive and rush to the market to maintain winning edge. Obviously, traditional method of project management following Waterfall model or Spiral model will not match expectations of those who want to move fast. It is well established that 80% of the software projects fail to satisfy the customer due to time over-runs, cost over-runs, inflexible design (with little or no scope for any enhancement). Similarly, software developers [45] get often frustrated in modifying their codes and conduct repeated tests to quick-fix re-fix errors to maintain software product after delivery. Such a fiasco often leads to losing existing customers for new projects and loss of credibility in the market. To play safe, certain software companies adopt very cumbersome and heavyweight process, which further complicates the issue. Thus, customers hardly get cost-efficient product on time. The answer lies in being adaptive by creating Agile environment by following Agile software development cycle [24] in the software company.

Today there are many Agile development methods which mitigates software project risks, speed up delivery and ensure flexible design which is responsive to customer changing requirement. This is achieved by developing software in multiple repetitions ('iterations') of short time frames ('time boxes') and delivering very frequently to get customer's timely feedback. In Agile software development process [17, 64, 71], one unit of time (iteration) typically lasts from two to four weeks. Each iteration passes through a full software development cycle including planning, story analysis , design, writing unit tests , coding and finally a working sub-module is demonstrated to customer. The developer and customer both remained excited about their accomplishment and confident to remain on-time and within budget.

Agile is a very recent software development methodology [63] (or more correctly, a group of methodologies) based on the Agile Manifesto (See Figure 2.4). This was developed to solve some shortcoming in traditional software development methodologies. Agile methods [2, 3, 22, 50] are based on giving high priority to the customer participation early in the development cycle. It recommends testing by the customer in every phase and often as possible. Testing is done at each point when a sprint version becomes available. The foundation of Agile is based on starting testing from the beginning of the project and continuing throughout to the end of the project. The Agile methods, concentrate more on (1) individuals and interactions than processes and tools, (2) more on working software than comprehensive documentation, (3) value customer collaboration more than contract negotiation, and (4) focus more on responding to change than following a plan. The Scrum and Extreme programming [54] are two of the most popular variations of Agile methods.

## Individuals and Interactions

### Working Software

### Customer Collaboration

### Responding to change

Figure 2.4 Agile Manifesto

In this section, emphasis is on collaboration among all stakeholders so as to improve quality in product during the sprint. Major stakeholders are *customers* including a direct user, requirement provider, indirect user, head of users, marketing analysts, investor and *development team* including program manager, developers working on other systems that integrate or interact with the one under development, or maintenance professionals potentially affected by the development and/or deployment of a software project, *business people and testers* [46].

These stakeholders are mainly active players in Agile working model. Customer role is to provide requirement and feedback on timely basis and team role is to giving response to the feedback and apply relevant technology and market standards on timely

basis. Agile SDLC including these two factors is given in Figure 2.5. Numerals in the diagram show specific activities performed by manager (M), developers (D), testers (T), marketing professional (MP) and customer (C). Details related with these numerals are shown in Table 2.3.

At the time of production of the code or before producing code, testing is applied by writing failed test cases unlike traditional approach of working. Testing activity begins as soon as stories are finalized and prioritized and testers try to move business logic into lower levels in order to test with lower effort in the last stage. In Agile, quality product is delivered by operational teams and acceptance factor is related with rate at which customer accepted the delivered product. Effectiveness of the team is connected by these two factors (See equation 1). Out of these two, quality is valuable as acceptance is totally dependent on Q. Q is more when less number of backlogs is there and it is less when backlogs items are more.

Backlogs can be decreased when automation is preferred approach over manual way of testing.

$$E = Q * A$$

(1)

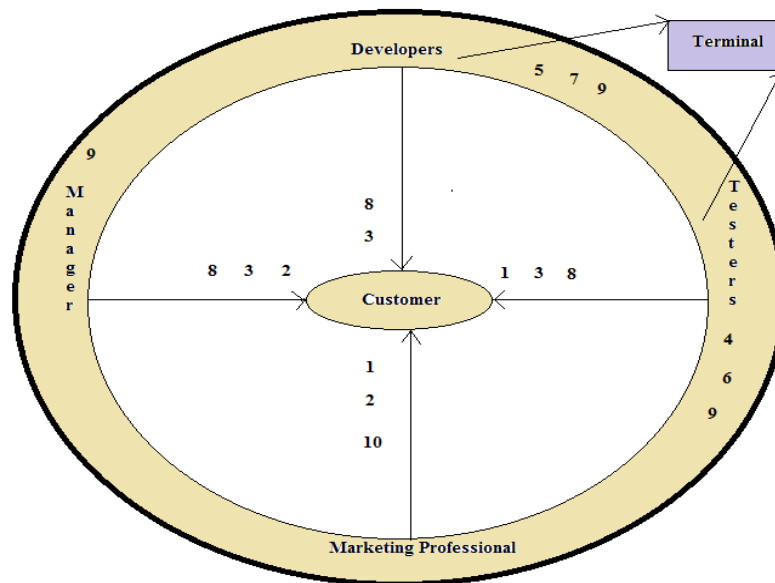


Figure 2.5 Agile Life Cycle

Table 2.3 Actor Activity Chart

S.No.	Actor	Activity
1	C, MP, T	Requirements Gathering
2	M, MP, C	Effort estimation [29], cost, risk, capacity & resource Estimation
3	M, D, T, C	User stories Prioritization
4	T	Designing of Test Cases
5	D	Coding for the user story in the sprint
6	T	Feedback
7	D	Refactoring for the user story
8	C, M, D, T	Review meeting with Demonstration
9	D, T, M, MP	Lesson Learning phase or Retrospective session after the sprint
10	C, MP	Release

## 2.5 AGILE PRACTICES

The existing practices of software project management are process bound for well-defined requirement, time and cost. These are termed as heavy weight processes/ technologies. On the other hand, for rapid development of customer centric software, Agile environment or lightweight technology like Extreme Programming (XP) is recommended (See Figure 2.4). Agile practices are attractive and different so need is to learn them with full dedication and good behavior. The introduction of the extreme programming (XP-Beck 1999b) method has been widely acknowledged as the starting point for the various Agile software development approaches. XP has evolved from the problems caused by the long development cycles of traditional development models. XP is concerned with good engineering practices so, it is a valued driven method. Its core values are simplicity, communication, feedback, respect and courage. The life cycle of XP consists of five phases: Exploration, Planning, Iterations to release, Productionizing, Maintenance and death. There are different roles in XP for different tasks and purposes

during the process and its practices. These roles are Programmer, Customer, Tester, Tracker, Coach, Consultant and Big Boss (Manager).

XP is a collection of ideas and practices drawn from already existing methodologies. XP aims at enabling successful software development despite vague or constantly changing requirements in small to medium sized teams. Short iterations with small releases and rapid feedback, customer participation, communication and coordination, continuous integration and testing, collective ownership of the code, limited documentation and pair programming are among the main characteristics of XP. XP is extreme because of planning game, small releases, simple design, refactoring, collective ownership, continuous integration and pair programming.

### 2.5.1 Pair Programming

Pair Programming [13] is a practice where two Programmers/Testers share the same terminal (See Figure 2.6). One of them does coding while other keep watching and reading the code to find on the spot errors. The important part is that both member intensely interact and are committed to produce first-time-correct code. The pair keep changing role frequently to maintain momentum of rapid development. Pair partnership is changed once a day so that every member of the project team work in two pairs per day. This way over a period say 4 to 6 weeks every member would have worked with every other member of the team to learn all aspects of various iterations/stories. Pairing concept may slightly slow down coding but it significantly reduces errors in the code. In the final analysis, pairing result improves efficiency, quality of code and hence increases overall productivity.

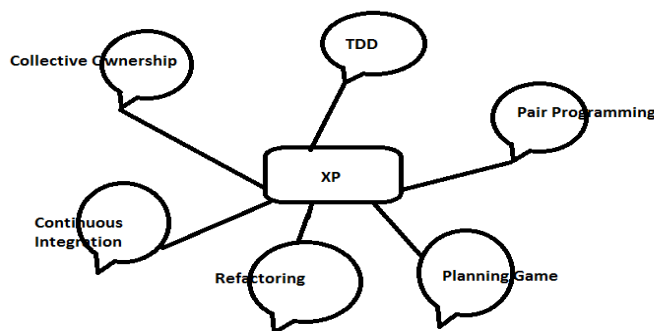


Figure 2.6 Pair Programming

### **2.5.2 Test Driven Development (TDD)**

In TDD [18, 20, 35, 47, 82] Test Case and code are developed side by side. Thus while coding in pair one of the members can run Test case to determine whether code passes the test case. This ensures independent testing of modules/ sub modules. This results in decoupling of modules, which is a concept of Object Oriented Design (OOD). Further, design pattern [9, 38, 53, 56, 68] may be applied to have OOD for the upcoming user stories.

### **2.5.3 Collective Ownership**

A pair has a right to check, cut and improve the module. Every member work on GUI, database, middleware and all have equal responsibility and pride in doing the job. No one is confined to a particular specialty instead everybody knows two or more specialties.

### **2.5.4 Continuous Integration**

As a pair, the programmers continuously keep checking their code and integrating daily. For this purpose, XP team uses non-blooming source control. This necessitate a discipline when one programmer check in a module checked out by another programmer and after modification is required to put it back. In fact, XP team builds a system many times a day.

### **2.5.5 Planning Game**

This relates to the division of responsibility between the customer representative and software Development Team. During this joint exercise, customer decides the priority of the feature to be incorporated while developer decides time and cost of its implementation. The developer indicates the budget at the beginning of each sprint. This helps the customer to select stories, which can be implemented within the budget.

### **2.5.6 Refactoring**

It is the practice of making small transformation to improve the code structure and eliminating degradation, which occurs after bug fixing. After each transformation the team collaborates to ensure consistency. This process continues.

## 2.6 SCRUM

The term “*scrum*” originally derives from a strategy in the game of rugby where it denotes *getting an out-of play ball back into the game* with teamwork. Scrum is named for scrum formation in rugby play.

The scrum approach has been developed for managing the systems development process. It does not define any specific software development techniques for the implementation phase. Scrum concentrates on how the team members should function in order to produce the system flexibly in a constantly changing environment. Scrum is a light weight method to deliver software iteratively in which changes are always welcome. Scrum is concerned with product owner, project lead and team working together in intensive and interdependent manner. Scrum process includes three phases: pre-game, development and post game. There are six identifiable roles in Scrum that have different tasks and purposes during the process and its practices: Scrum master, Product owner, Scrum team, Customer, User and Management. The Scrum does not require or provide any specific software development method/practices to be used. Instead, it requires certain management practices and tools in the various phases of scrum to avoid the chaos caused by unpredictability and complexity. Scrum life cycle is shown in Figure 2.7. Scrum duration may be from 2 to 4 weeks. Scrum practices include Product Backlog list (PBL), effort estimation [33], Sprint, Sprint planning meeting, Sprint backlog list (SBL), Daily scrum meeting (DSM), Test, Code sprint review meeting (R1), retrospective (R2) and release (R3).

## 2.7 COLOR CODING SCHEME FOR BUG TRACKING

Bugs identified within a sprint are displayed on the project wall by Agile team members such as tester using the following steps:

- Tester note down bugs on stickie note and attaches it to the story card.  
(After identification of bugs in the sprint, the tester creates a detailed defect board in collaboration with team members)
- Tester moves card back to in development stage on the story card.

- Product owner representative then add schedule details on the stickie note for the respective defect:
  - a. Schedules the defect for resolution
  - b. Defers the defect for a later release
- Developer in coordination with tester fixes bug and resubmits it to ready for Test on the project wall.

Also, for bug tracking and performance tracking, metrics are involved which act like a communicating metrics. Second option to this is, colored story cards can be placed in the common room (See Figure 2.8). Also burn down/ burn up chart and running tested stories can be used for checking the performance. Burn down chart indicates whether team will finish on time, early or won't finish everything they committed to. Running tested stories shows that how many stories are done.

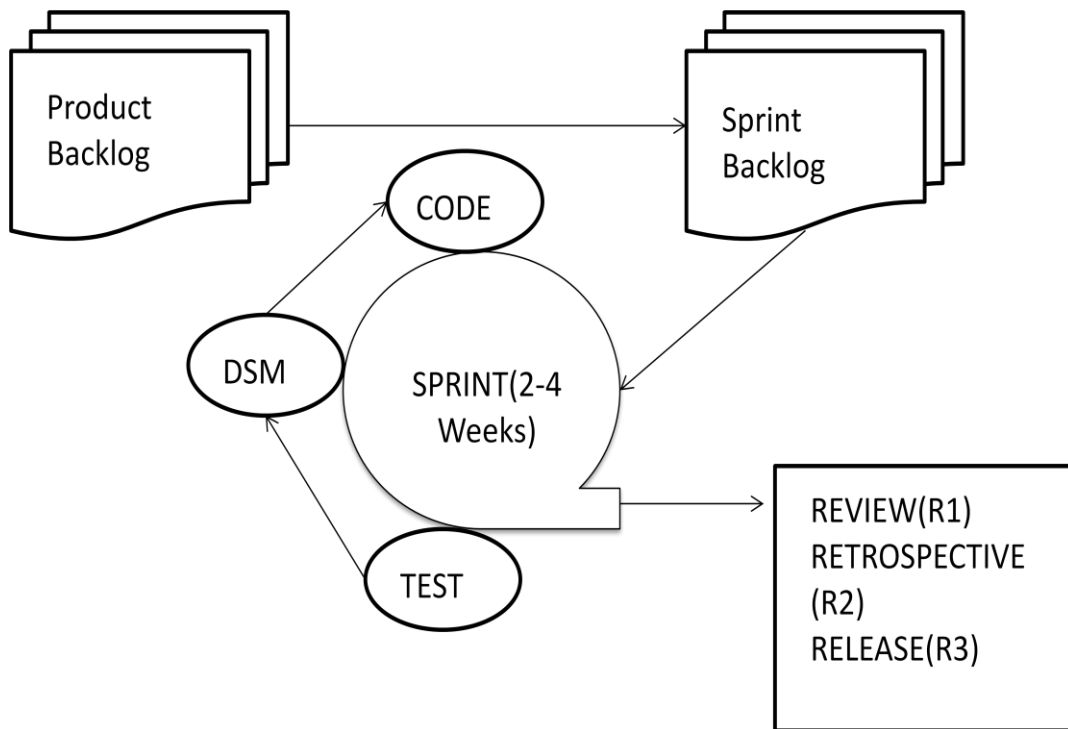


Figure 2.7 Scrum Life Cycle

## 2.8 TESTING QUADRANT MATRIX

Testing quadrant matrix is shown in Figure 2.9. In this matrix, testing types are represented in different quadrants. The testing types present in the first quadrant Q1 are performed using automated tools. More specifically, unit tests may be performed using



Xunit tests or Eclipse or any other automated interface. The testing types in second quadrant Q2 of this matrix may be performed either through automated tools or manual style of testing. In Q3, manual testing is performed. Last but not the least, Q4 quadrant testing may be performed using open source tools. This testing matrix is given in Agile Testing book of Lisa Crispin. These tests are technology facing tests or business facing tests so as to satisfy the customer.


















Product Backlog	SPRINT Backlog List	Defect Backlog	Scheduled Defect Backlog	Completed User Story	Remarks
					
					
					
					

Figure 2.8 Color Coding Tracking Scheme

## 2.9 TRADITIONAL RISK REGISTER

Risk Management in software development is done by maintaining a risk register. The risk register is created for any kind of risk such as low risk, medium risk or high risk. This is a post management technique as after facing risk, any team member may store metadata of any risk. The manual entry of risk involves metadata related to:

- **Description of risk:** A one- or two-line overview of the risk. It should be precise one.
- **Identification date:** Date when the risk was identified.
- **Likelihood:** Estimated probability of occurrence of the risk.
- **Severity:** The severity of the risk is assessed based on impact of the undesired outcome.

- **Priority (optional):** This could be either given an independent value or set as a product of likelihood and severity. A high-severity risk with a high likelihood should receive more importance than a high-severity risk with a low likelihood.
- **Actor:** The person who manages, controls, and takes action in response to the risk.
- **Action:** The response defined to manage/control the risk.
- **Status:** Indicates whether the risk is registered or open or closed or being monitored.

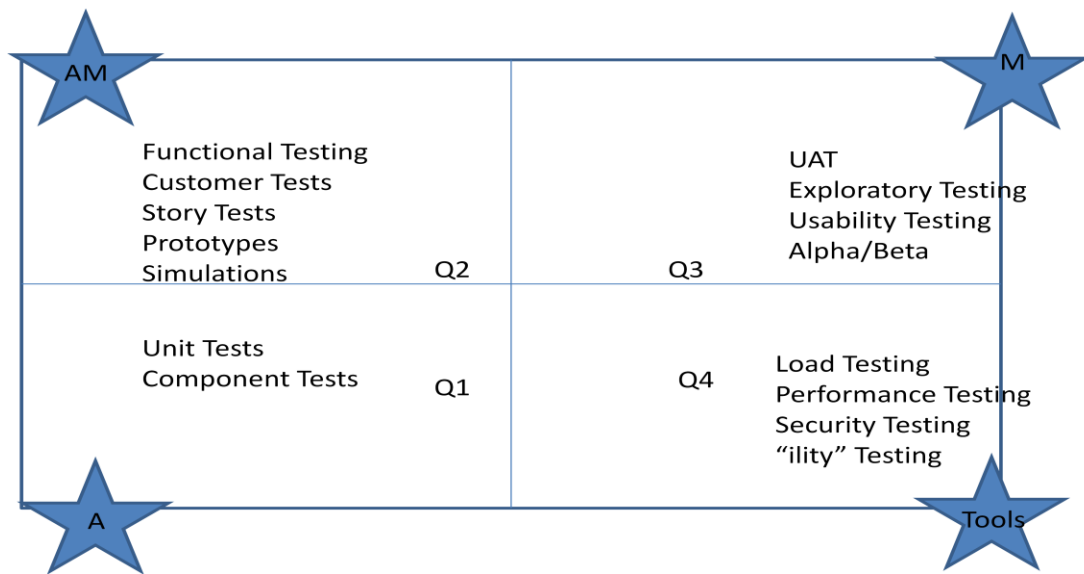


Figure 2.9 Testing Matrix

## 2.10 REGRESSION TESTING

Regression testing aims at ensuring of uncovering any new defects in existing functionality due to changes made to the application. So, any set of test cases that is selected or prioritized should fulfill criteria comprising business need and customer need. Regression testing framework [36, 51] helps in understanding test case management. Regression testing in Agile culture is practiced under two major categories.

**Sprint level regression testing (SLRT)** - focused on testing new functionalities that have been incorporated since last production release.

**End to end regression testing (EERT)** - The regression testing that incorporates all the fundamental functionalities.

Three practical scenarios adopted by different organizations in tackling the regression testing as part of their release cycle are discussed below:

**Normal Agile testing** In this scenario, each sprint cycle is followed by a small span of SLRT (See Figure 2.10). The completed code goes through further regression cycles but is not released into production. After few successful sprint cycles - typically 3 to 4 - the application goes through one round of EERT before being released to production. This is known as normal testing as most of the Agile software companies use this approach for testing their software products. This approach is adopted by many companies.

**Advantages:**

- Allow the team to focus on ensuring functional validity for the sprint without the additional overhead of ensuring EE regression suite completion, and
- There is flexibility in the degree of automation needed, since production release is preceded by an EE regression cycle spanning two weeks.
- Easy approach for transition from traditional to Agile approach.
- Production release happens only once every 10 weeks.

**Disadvantages:**

- Testing team has to spend more time - especially if a third party vendor is doing the testing
- The team deciding to park critical defects with an implicit understanding to take them up on a future date. The future date never comes due to a packed back-log and the team is left to catch up during the end to end regression window.

Sprint 1	Week 1		
	Week 2	Sprint Level Regression	← [ ]
Sprint 2	Week 3		
	Week 4	Sprint Level Regression	← [ ]
Sprint 3	Week 5		
	Week 6	Sprint Level Regression	← [ ]
Sprint 4	Week 7		
	Week 8	Sprint Level Regression	← [ ]
Release preparation	Week 9	End to End Regression	← [ ]
	Week 10		

Figure 2.10 Normal Agile Testing

**Week Plus testing** In this approach, the sprint level regression continues beyond week 2 and extends till the middle of week 3 (See Figure 2.11). This approach is adopted by fewer companies.

**Advantages:**

- This removes the constraint on the team to stop testing abruptly at the end of week 2 and start with the next sprint immediately thereafter.
- Productivity and quality improvement is significant.

Sprint 1	Week 1		
	Week 2	Sprint Level Regression	← [ ]
Week 3			
Sprint 2	Week 4	Sprint Level Regression	← [ ]
	Week 5		
Sprint 3	Week 6	Sprint Level Regression	← [ ]
	Week 7		
Sprint 4	Week 8	End to End Regression	← [ ]
	Week 9		
Release preparation	Week 10		

Figure 2.11 Week Plus Testing

**Sprint Plus testing** In this approach, organizations do not differentiate between SLRT and EERT. Instead, there is a common regression cycle that is extended by a sprint. So, the regression test cases that are employed for sprint 2 execution contains functionality till the stories that are part of sprint 1. This provides the testing team more time to keep their regression test cases current and automated. This approach is adopted by very less companies.

**Advantages:**

- This approach avoids the need for having two separate types of regression test cycles.

**Disadvantages:**

- It is very challenging to maintain the sanctity of regression testing.
- The automation maintenance effort increases as one is still struggling against a two week time window and a relatively less sTable set of requirements.

**Mature Testing for Mature Company** In this approach instant automation and continuous integration are used. This approach is adopted by mature companies.

**2.11 COMMUNICATION IN DISTRIBUTED ENVIRONMENT**

Communication among different stakeholders is a real challenge [85] in an Agile environment. Client is one of the stakeholders whose job is to transfer input to the development team. Client may or may not be available at the development site. In other scenario, one of the client representatives may directly involve with the team members. For effective communication among parties, some of the challenges are to be handled beforehand. Communication may start from the source side and end at sink side. Source side initiates the communication by framing a message by passing through a channel. On the other side, sink side receives the message. In between these two sides, all the challenges for effective communication lie. Some of the challenges are as follows:

1. Language barrier
2. Time zone issue
3. Cultural Barrier
4. Channel issue

When message are to be exchanged at global level, first three issues come across. Fourth issue is a technical issue, which is linked with wireless communication for an effective communication, may be resolved easily. In an Agile work culture, different stakeholders may be located at different locations (overseas). In that situation, managing these barriers becomes essential. Understanding and respecting diversity of cultures becomes a necessity. Team has to deal with client of different time zones. Out of all these

challenges, language is the barrier that cannot be managed by the team on their own. In this section, language issue has been dealt and focused by using Adobe Captivate [49] tool for the purpose of understanding requirements in a better way.

Further to add on, communication may have two types: verbal and non verbal. For verbal communication, parties may or may not be present at the same place. For example, Email writing is a verbal communication way, as it involves writing mail with the help of words, phrases, sentences, punctuations and email etiquettes. In addition, source and sink parties may or may not be present for the purpose of communication at same time. Considering the case of telephonic communication, both parties must be present. In this case, communication takes place with the help of words, phrases, pauses and sentences. Here, pauses take the position of punctuation of e-mail writing. Also, language plays major role in verbal communication. If language is different then, both the communication style would be ineffective.

On the other hand, non-verbal communication [48] is very different from verbal communication style. This involves body movements of a speaker for more clarity of subject matter. Different techniques that may be used in non verbal communication such as oculosics, paralanguage and meta-communication, proxemics and kinesics. These terms sound complex but, in simple words, these techniques are related to different expressing styles such as, voice modulation, pitch, tone, volume, eye contact, pronunciation, physical contact, body language etc. This kind of communication is only possible when face to face communication or video conference takes place. Non-verbal communication is better than verbal communication but this type of communication is not always possible.

Specifically, in Agile work environment, non verbal communication is not possible as client is overseas and development team is at different location. Occasionally, client may visit development site or team member may have face to face interaction with the client. If interaction is not non-verbal, then it does not mean that team members may have chance to skip important points of requirements. In that scenario, it becomes the responsibility of every team member to clarify the doubts as onus lies on project manager of specific project. Every business industry give special attention to client but Agile follower company focus on more interactions with the client whether verbal or non

verbal. Selection of these two styles depends on the location of the client. Also, client representative must be present so as to review the current work and for providing feedback. This feedback may be very valuable as this may help in preventing long term issues. Some of the qualities of a client representative are:

- Keen observer
- Trainer
- Technically strong
- Market Analyser
- Feedback Provider
- Secondary requirements provider
- Managerial Skills
- Risk handler
- Decision maker
- Motivation capability
- Optimistic
- Proactive nature
- Multilingual or Bilingual Personality (Optional)

In this section, verbal communication has been focused to ensure that requirements are communicated in proper manner from client/client representative side to the Agile team members. Good communication is desirable in business, as growth of business stays on the pillar of effective communication. Also, at global level, business flourishes, because of effective and productive communication among its stakeholders.

### **2.11.1 Communication**

In this section, communication has been managed effectively in an Agile work culture by incorporating various linguistic components. For getting an understanding on these components one scenario has been considered. That scenario has been shown in Figure 2.12.

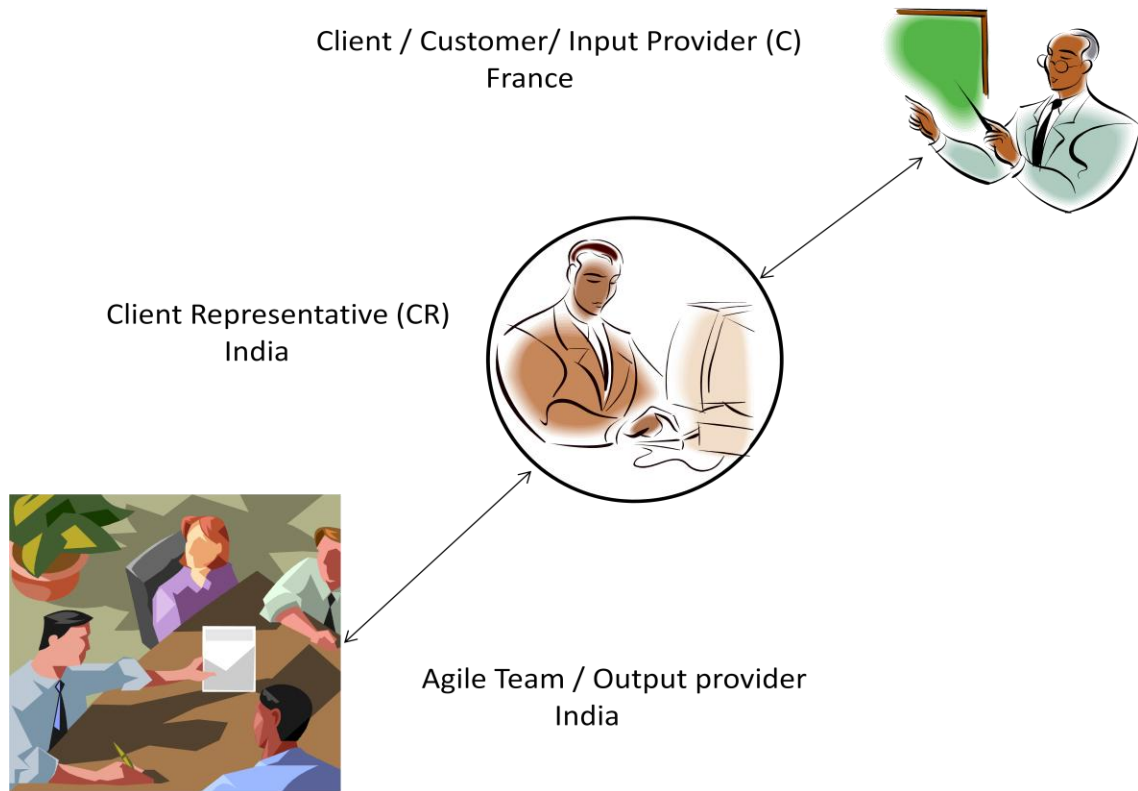


Figure 2.12 Agile Scenario

Figure 2.12, consists of client (C), client representative (CR) and Agile team. These all are the major stakeholders who have keen interest in the project. C is a resident of France and he is fluent in his native language “French”. Agile team is part of one of the renowned multinational company in India, who is famous for delivering software product deliverable with good quality in less time. In addition, all employees are fluent in English Language. CR is representing her client in India and he is bilingual personality with English and French. His main responsibility is translating the requirements provided by C into English language and clarifying the doubts of different team members. C is non-technical person, that’s why she provides requirements in informal way.

Specifically, detailed description about all these are shown in Table 2.4. C and CR job is to provide timely feedback to the team members for enhancing the productivity of the product deliverable. C sends the feedback to CR in French and CR communicates that with team members after translating the French feedback in English. In addition, CR is mostly available at development site. CR translates and communicate corrected version of the requirements to the team.



Table 2.4 Task Chart

Client (C)	Client Representative (CR)	Agile Team
Idea	Communicates with Client & Team	Communicates with CR
Informal requirements	Understands/ Translates Requirements	Implements Requirements
Rarely Available	Mostly Available	Always Available
Monolingual	Bi-Lingual	Monolingual
Feedback Provider	Feedback Provider/ Reviewer	Feedback Receiver

In Agile, less is more approach is used, so, in an ideal team, 7 - 8 members are there for implementing and releasing client requirement. In Figure 2.13, four team members are shown for the simplicity purpose. These four team members are shown on the four corners of the rectangle. This is done to give equal importance to each team member, as there is no hierarchy in Agile. They all can interact with each other and CR, who is shown in the centre of the rectangle. This rectangle is shown to be incomplete as 4 team members are not fixed. There may be more team members as per the requirement of the project. These team members roles may be of a developer, tester, business analyst and language expert such as Java or C plus plus. In SCRUM, one of the methodologies of Agile, scrum master is another team member who manages rest of the four team members. These roles of team members are not static. Depending on the work requirement, roles switch among team members. Open interaction system is followed for the purpose of transparency and easy access of resources. A resource here refers to team members. This open environment helps CR in providing instant feedback for early detection and correction of any current or upcoming issues. A resource may also refer to tools and materials needed for delivering the outcome. A business growth depends on, how strong is the interaction among team members and CR.

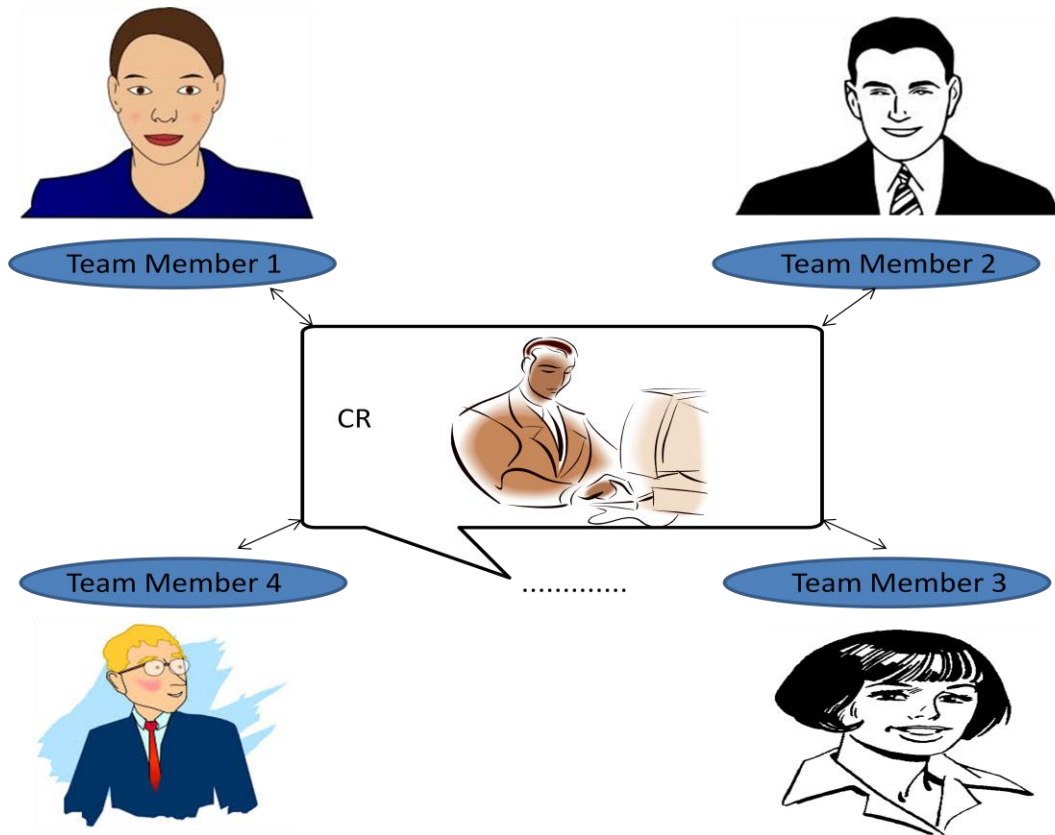


Figure 2.13 Team Communication

Verbal communication, among two major stakeholders C and CR, comprising e-mail and telephonic options is shown in Figure 2.14. Team is shown in this Figure, as team has to interact with the CR for receiving the feedback and input requirement. Also, formal requirements such as user story and ready story are framed by team members, by collaborating with the CR. This is the most important and first step in the user story implementation. Ineffective communication from either side may prove to be very expensive. Also, CR feedback to the team members is very important so as to learn lessons and implement those lessons at an early stage. This early stage lesson may cut down cost to an unexpected level. In some of the earlier proved cases, problem detected at an early stage may cut down cost up to 80%. Also, Pareto principle: 80-20 rule, also says that 80% of time spent in the earlier stages may bring down maintenance cost to 20%. In Agile, requirements keep on changing with time. If correct understanding of requirements would not be there in the beginning, then the maintenance cost may go very

high. In this Figure, different color shows that who started the process. Following is the list of color coding scheme:

- Red: C
- Green: CR
- Brown: Team



Figure 2.14 Verbal Communication

Figure 2.15, represents non-verbal communication among major stakeholders by video conference (VC) or face to face (FTF), which are ways for communicating requirements. Team may get a chance to clarify their doubts in an effective way. Team is part of the conversation, and CR is translating the requirements to C and team. In this case, expression, body movements, gestures of C play a major role in identifying requirements clearly. This type of communication may occur frequently during the project span depending on the availability of the C. After getting the correct version of the requirement through VC or FTF means, team collaborates with CR in framing formal requirement in the form of user story and ready story. Ready story is the expanded version of the user story. Ready story is created by conducting more conversation with CR and confirming the ready story points. These ready story points confirm the acceptance criteria of a user story. After finalizing the acceptance criteria, team may start

implementation of the user story. CR reviews the deliverable prepared by team and reviewed deliverable is delivered to C. This process continues until all the user stories are delivered to the C.



Figure 2.15 Non-verbal Communication

### 2.11.2 User Story

When requirements are finalized, then next step is to frame the requirement in the form of user story. User story has a typical format. A sample format of user story is shown in Figure 2.16. Last clause of this format is optional.

As a research scholar,  
I want to view recent patent filed in Agile software development area, in free patent databases,  
so that, I can improve my search results

Figure 2.16 Story Format

For the shown user story, a ready story may have acceptance points. The CR accepts the user story, when all confirmed points are met. These confirmed points may be written on the back side of the story card. These acceptance points are shown below in Figure 2.17. There may be more number of confirming points. For the simplicity sake, only three are shown here. These may be added as per the CR need. By getting answers of these confirming point's team may start working on the user story. Clarity of confirming points among team members is a must.

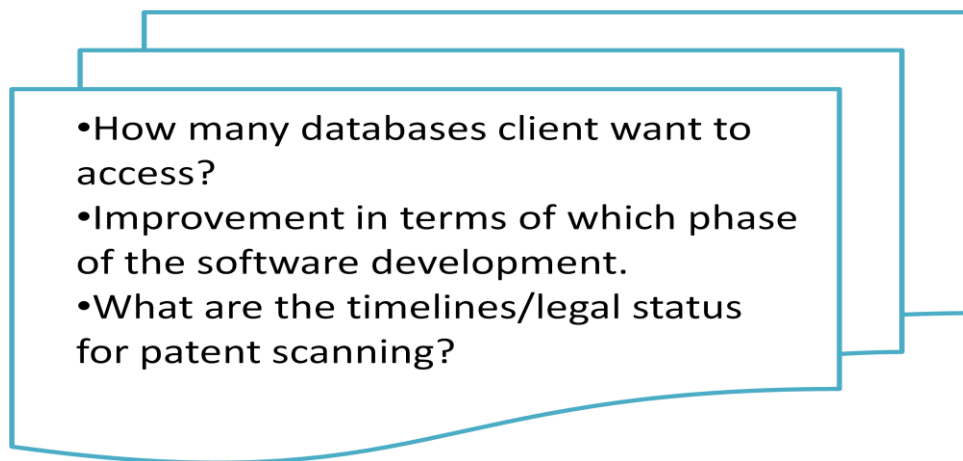


Figure 2.17 Confirming Points

### 2.11.3 Modified User Story

Verbal communication especially, email writing faces different challenges while understanding requirements from C. Adobe Captivate is the tool that is used for understanding the requirements in a better way, stated by C. This understanding is done by using punctuations, with the requirements. The process is used by CR and Team members. Using adobe captivate tool, text may be converted to audio. By using punctuations, pauses may be inserted in the audio. Comma is more frequently used punctuation. Capital letters are used for reading abbreviations. In this way, email writing text can be converted to audio. This audio, when played, will appear as if, C herself is telling her requirements. C may not be expert in captivate tool. If she is, in that case, she can send the audio of her requirements in French language; otherwise, CR will convert requirements into audio files using English Language. Also, CR may collaborate with team in framing user story and ready story using captivate tool. Block diagram showing use of captivate by different stakeholders is shown below in Figure 2.18.

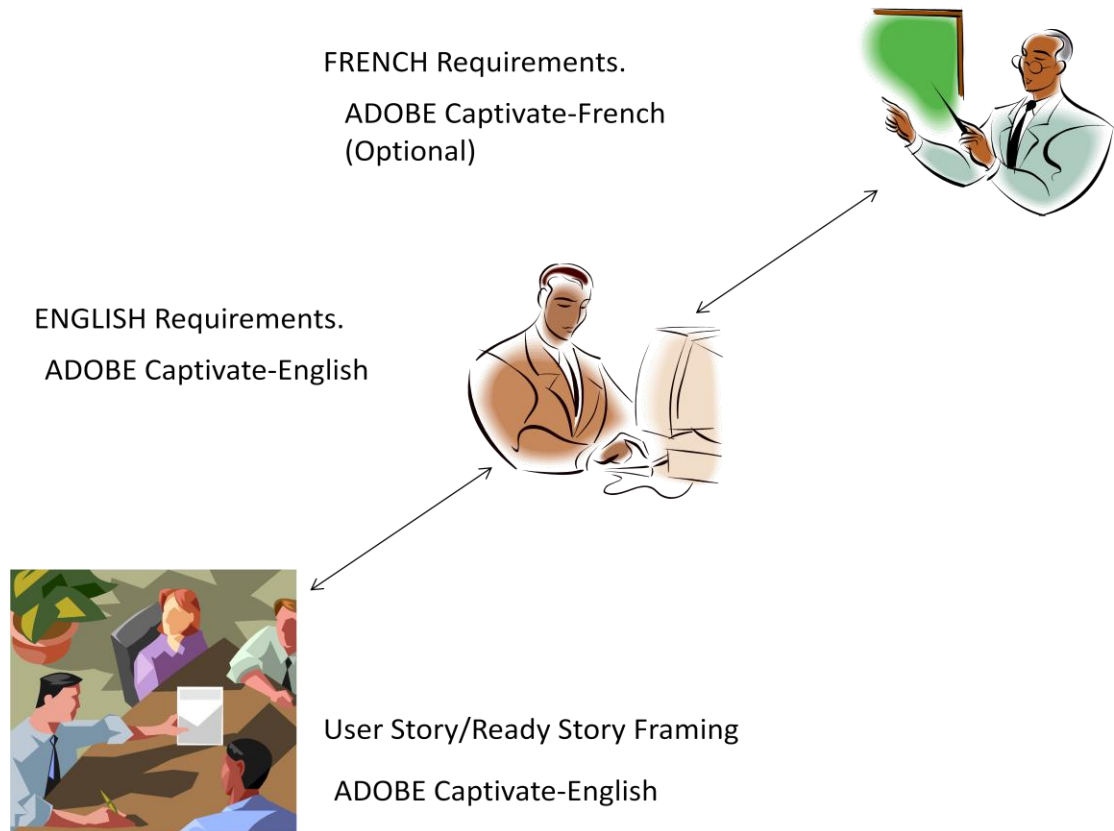


Figure 2.18 Stakeholders Understanding

Considering the case when client give requirement in informal way in the form of Email (see Figure 2.19 for attachment). For the sake of simplicity, requirement is shown here in English (C gives requirement in French to CR). Different work professionals may draw different meanings from this email.

*Hi*

*Database to be created researchers of software domain can access patents of countries us ep india searching is possible by application number patent number keywords class sort possible string formation include other features load management*

*Note please find attachment*

*Thanks*

*John*

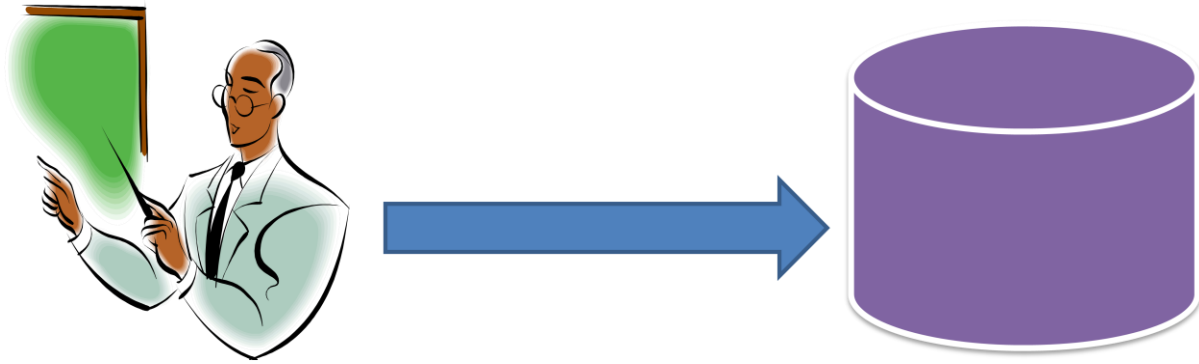


Figure 2.19 Researcher Database

Can anybody work on this kind of requirement?

Can quality would be there in the deliverable?

Can delivery of product is possible on time?

Can client would be satisfied?

Answer to all these questions is NO!

For more clarity, C or CR may further work out on the email text by adding more & more punctuations. In addition, email text may be rewritten in Slides notes section of Adobe Captivate 8 tool in and is converted to audio file that may be played by team members for better understanding. These punctuations would introduce more pauses in the audio file. Also, C or CR may add suitable graphics in the audio file for better clarity. A corresponding screen shot for the requirement is shown in Figure 2.20. Slides notes for the above mentioned email are:

*Hi!*

*"Database" to be created.*

*Researchers of "software domain" can access patents of countries: US, EP, India.*

*Searching is possible by application number, patent number, keywords, class.*

*Sort possible. "String" formation.*

*Thanks,*

*John*

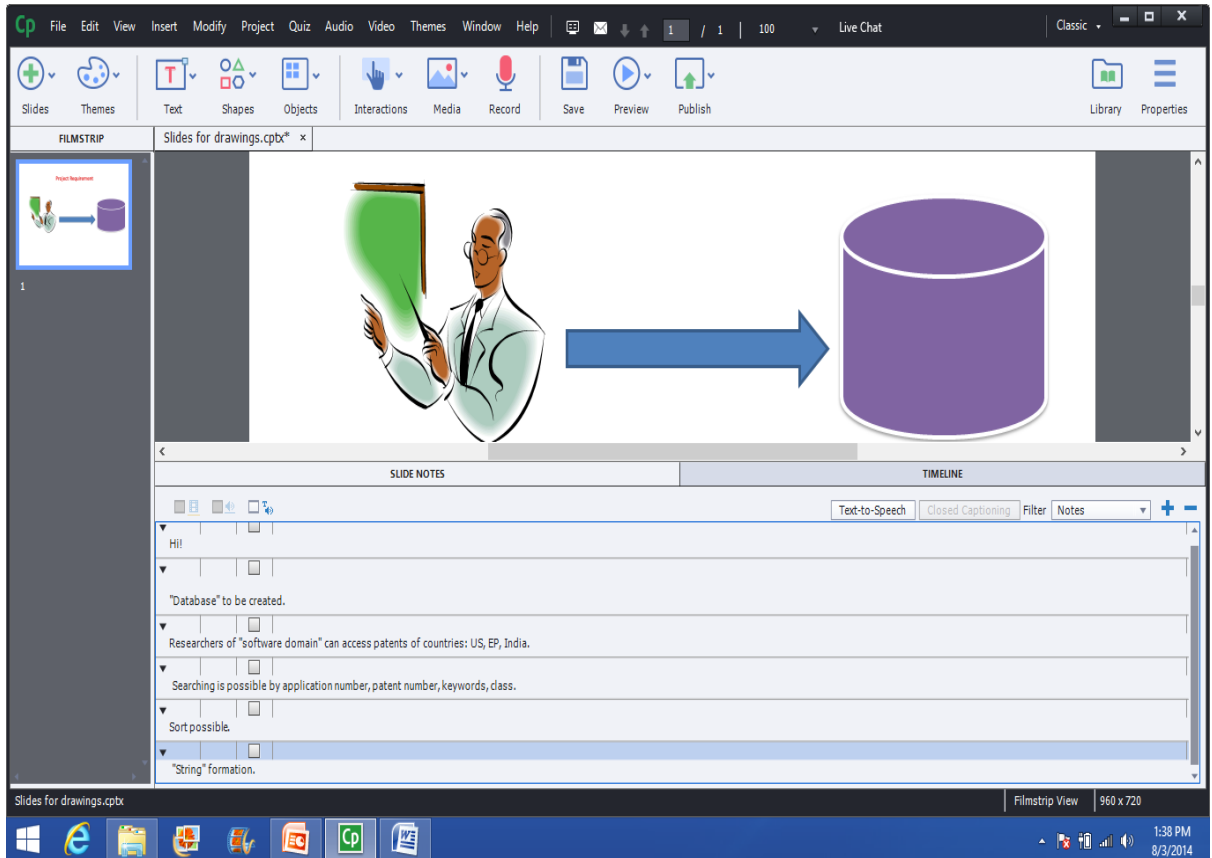


Figure 2.20 Project Requirement

Similarly, slides notes or script is to be written for user story or ready story. Once scripts with graphics (.cptx files) are converted to audio files, these converted audio files are published and uploaded on the central database at the workplace. In this way, less documentation principle of Agile is fulfilled. These uploaded files (having extensions .swf, .avi) may also be accessed by team members who are working at distributed locations.

For the given project requirement, user stories are shown in Table 2.5. CR job is to communicate with C, to get clarity on the following issues after getting informal requirement:

- Who all would be the user of the database?
- What is their job profile?
- How frequently they would access the database?
- Sorting on what factors?
- Whether authentication is needed?
- What kind of String formation is needed in a database?



After getting answers of all these questions from C, CR can start interacting with team. CR job is to explain informal requirement by adding answers received from C.

Table 2.5 User Stories

S.No.	User Story
1.	As a searcher, I want to search patents of US, EP and India, on the basis of, bibliographic details, such as, application number, publication number, priority date, inventor, assignee, date of patent, so as, to perform searches comprising invalidity search, claim mapping, claim charting.
2.	As a lawyer, I want to search legal status of patents, by mentioning publication number, application number, so as to handle infringement suit of different assignee of US, EP or India.
3.	As an Inventor, I want to search, patents of software domain, comprising, Agile software development, software engineering, software quality, software testing etc. so as, to perform “state of the art” search, for countries: US, EP and, India.
4.	As a researcher, I want to search, patents of US, as, US is the software hub of software patents, so as, to improve upon, my findings.
5.	As a drafter of a patent, I want to search, related patents, using software keywords, so as, to learn drafting skills, of software domain.
6.	As a business analyst, I want to search, patents of Countries: US, EP or India, so as, to analyze market trend of patents, in software domain, for different assignees.
7.	As a statistical professional, I want to, search revenue spend on patents, by assignee of countries: US, EP and India.
8.	As an examiner, I want to search, Data Base, for finding prior arts of a given patent by mentioning, its bibliographic details, including application number and, priority date.
9.	As an analyzer, I want to, categorize patent of different classes, such as, United States classification (USC), Cooperative patent classification (CPC), International patent classification (IPC).
10.	As a petitioner, I want to, download patent document, depending upon its legal status: active, pending, inactive, abandoned and revoked, so as, to read patent sections, such as drawings, detailed description, summary, objects of the invention, background, abstract, claims.

Table 2.5, includes the user stories, as per the requirement of every type of user. Also, these user stories are written by incorporating as much punctuation as possible, so as to have clear understanding of the requirements.

One of the samples of script for single user story is shown below in Figure 2.21. The script of the corresponding user story is separated by different punctuations. Each slide note of the script may be selected separately or text to speech checkbox is selected for selecting all slide notes at once. Also, speech management dialog box gives the option

of selecting particular speech agent. After clicking on generate audio button respective text may be converted to audio of selected speech agent. CR may choose English speaking speech agent such as Paul or Kate. On the other side, C may choose French speaking speech agent. For better understanding of CR or C requirement, more pauses in the audio file may be desired. This may be done by using voice text markup language (VTML) tags. These tags are useful when one of the listeners is fast and other one is slow. For example, Japanese speaks fast.

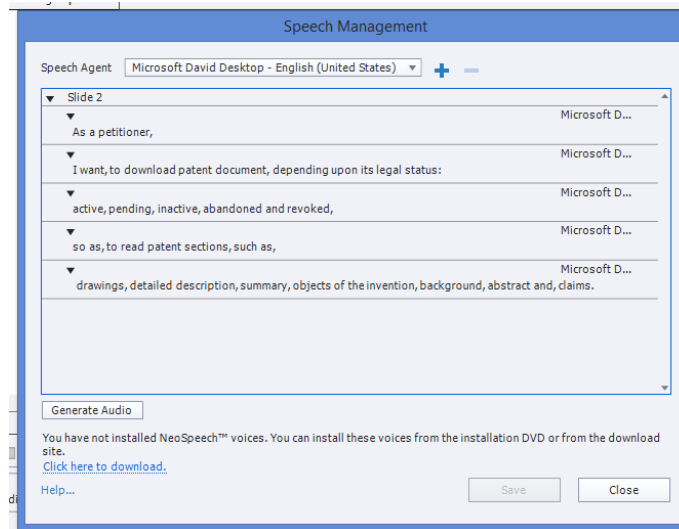


Figure 2.21 Story Script

## 2.12 LINGUISTIC PARAMETERS

The process of story building starts as soon as prioritization step is over. Till now, we have covered understanding C requirements, user story framing and user story prioritization. Next step is to create ready stories for the respective user stories. For user stories 1-10, ready story is written, in collaboration with the CR. Story number 10 is the most risky story, so, this story is considered as seed story. Confirming points for seed story are shown in Figure 2.22.

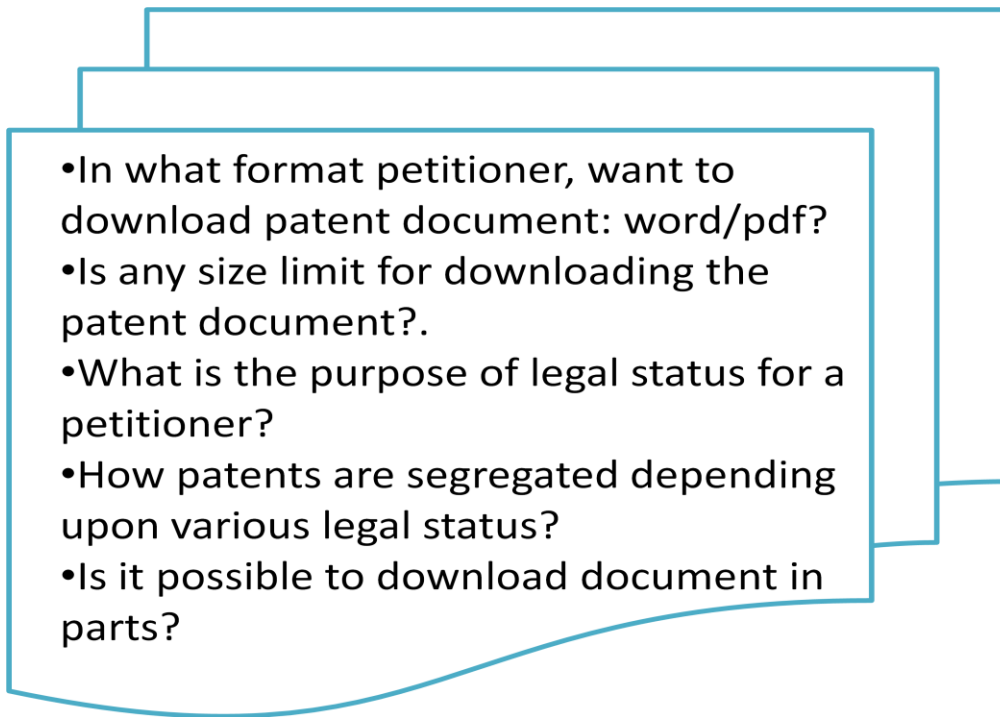


Figure 2.22 Confirming points\_10

After getting answers of these confirming points, client acceptance strategy is set. This strategy helps in preparing definition of done (DOD) for all stakeholders. Suppose CR gives answers of confirming points as:

- pdf
- No
- Usefulness based on legal status
  - Active - very useful
  - Pending - Sometimes
  - Inactive - Never
  - abandoned - Never
  - revoked – On what ground

This means that, if a petitioner login to the system, then inactive and abandoned patent details can be disabled. Also, for revoked patents, case history may be enabled. Active and pending patents are always enabled. For pending patents, only published patents have to be uploaded and for active legal status, complete document should be available in .pdf format.

# **CHAPTER III**

## **AGILE TESTING LIFE CYCLE**

### **3.1 INTRODUCTION**

Requirements are evolved throughout the project in ASD environment from the customers. Customer may bring any change in the requirements as per the market standard. After getting the requirements, team members including tester, market evaluator and many more do the feasibility study which is an analysis activity so as to have more specific and clear version of the requirements by communicating with customer. After including that change in the sprint backlog list (SBL), team along with product owner may start focusing on that particular change. For achieving long term goal of customer satisfaction, good quality software has to be released after every sprint by the development team. As the new requirements (user story in an Agile context) keep on increasing, different kinds of problems like management of people, task, defects, deadlines etc. are faced by team members. Accommodating that particular change in the existing software system also requires different kinds of management such as people management, story management, test case management, communication management, sprint management, backlog management etc. In this work, focus is on test case management which is done using regression testing. In the existing literature, an effective regression testing management mechanism is missing. Therefore, an Agile testing life cycle has been proposed in this work which elaborates various activities which are performed by tester along with other stakeholders to deliver quality product to the customer.

The proposed testing life cycle is based on “more is less” principle which is related with communication management among stakeholders. This principle has been explained for getting quality software product as shown in Figure 3.1. The foremost component of this principle is ‘more interactions’ among all stakeholders. If communication of tester with other stakeholders is very frequent and effective then there would be very less number of doubts or more clarifications. On the basis of more clarity, team focus increases on relevant portion of the user stories. If relevant portion of user stories are covered with high focus then there would be less number of severe defects

during the time span of sprint. This principle ensures that testing activities along with effective communication and collaboration among major stakeholders like

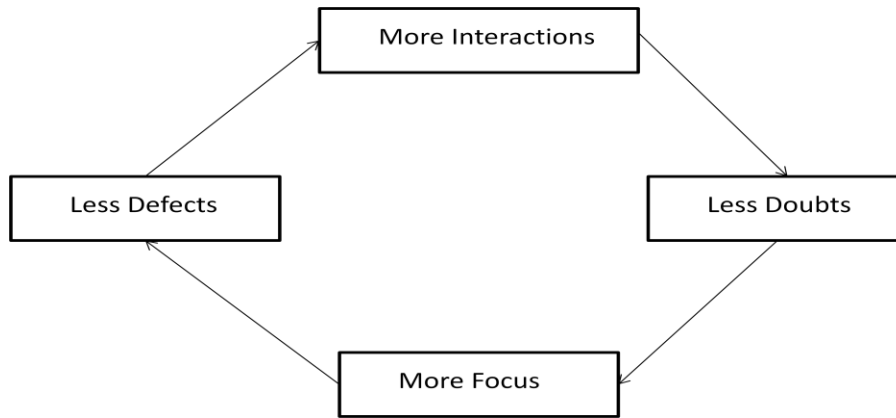


Figure 3.1 More is less

business analyst, market evaluator, customer, developer etc. may result in software product having less defects.

### 3.2 AGILE TESTING LIFE CYCLE

The proposed Agile testing life cycle has been divided in two parts as shown in Figure 3.2. This is done to show the clear cut difference between traditional testing and Agile testing [16, 30, 65]. Moreover the clubbed representation is to show how testing activities (specifically, regression testing for test case management) are performed by Agile tester along with other activities by collaborating with other stakeholders. In the first part of this model, traditional model testing has been shown where tester is involved in all types of testing such as unit testing, functional testing, integration testing, system testing and many more. In traditional style of testing, tester does the testing in isolation with any team member. The development team after completing the coding part in coding phase hand over the code to testing team and testers start testing on that huge code.

In the second part, an Agile tester interacts and collaborates with two circles namely an outer circle and an inner circle. The outer circle is connected to the outside world. In the outer circle, the tester collaborates with customer and market evaluator. The customer job is to provide an informal set of requirements in one specific domain and market evaluator job is to study the

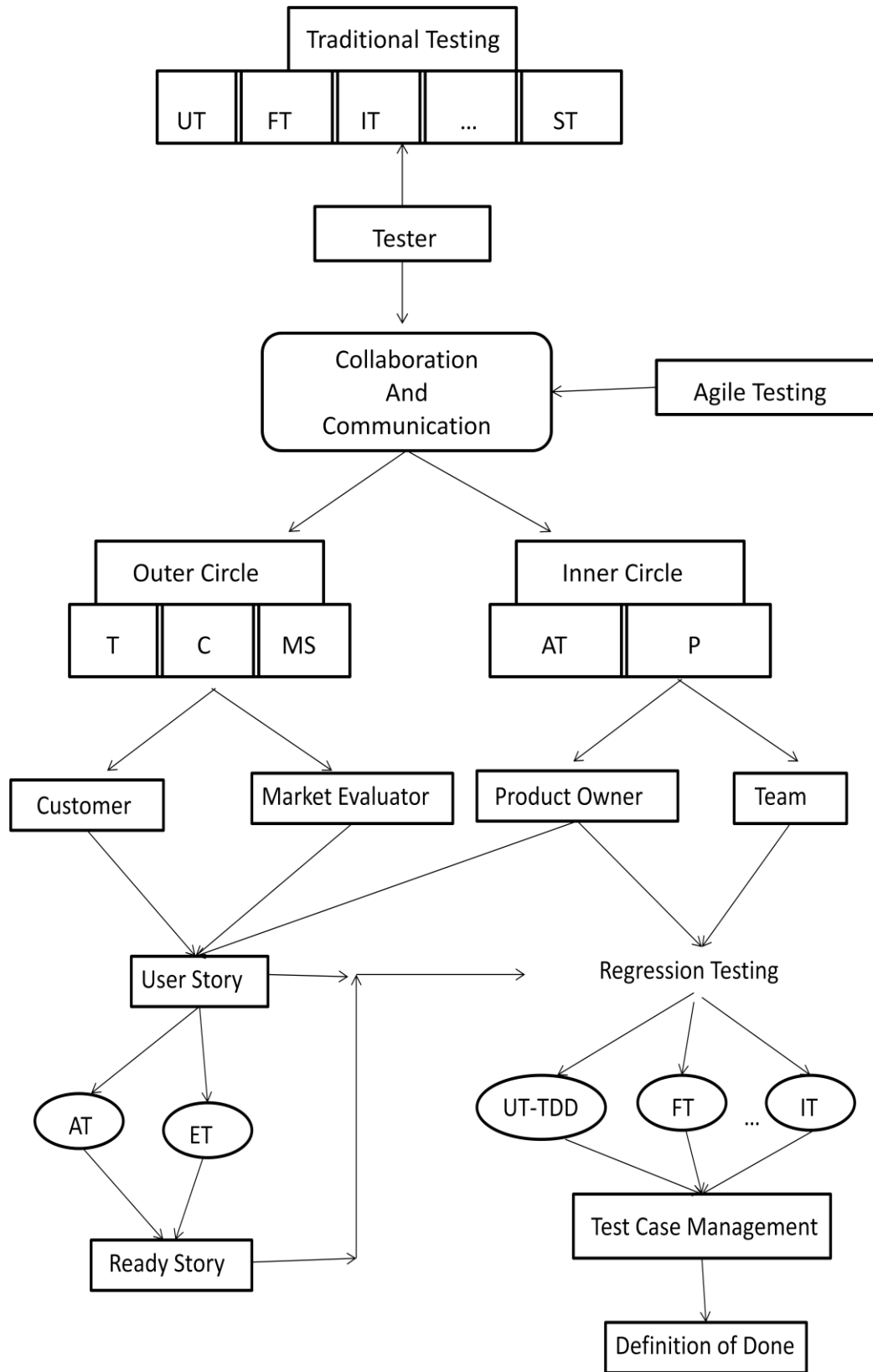


Figure 3.2 Agile Testing Life cycle

Table 1 Agile Testing Abbreviations

<b>Abbreviation</b>	<b>Full Form</b>
UT	Unit Testing
FT	Functional Testing
IT	Integration Testing
ST	System Testing
T	Technology
C	Competitor
MS	Market Standard
AT	Automated Tool
P	Pattern
AT	Acceptance Testing
ET	Exploratory Testing
TDD	Test Driven Development

market trends of the same domain. Further, from the outer circle, a tester may extract latest technology trend, competitor’s software product features and the latest market standard. All these factors are analyzed by market evaluator and an updated set of data is provided during user story finalization meeting in the presence PO. PO role is to satisfy the customer in terms of available bandwidth and expertise of team while converting the informal requirements into formal set of requirements. This formal set of requirements is known as a user story.

After finalizing the user story, tester role is to convert that user story into ready story. This ready story acts like a check list at the time of verification or acceptance of the

user story by the customer. This ready story is the outcome after performing two types of testing. Types of testing which are done for specific user story are:

- Exploratory Testing
- Acceptance Testing

Exploratory testing is the testing in which different possibilities or scenarios are considered as per the market analysis performed by market evaluators having positive and negative limitation. At the same time, risk of user story may be found so that effort estimation may be accurate in terms of effort, complexity and time.

Acceptance testing performed by tester sets the acceptance criteria for a user story. These criteria are also known as verification points. These points are needed to set the complexity level of the user story. Specifically, based on some factors or some expert techniques like planning poker game, complexity level of the user stories is identified. These verification points are then deeply analyzed. In this case also, tester does not work in isolation rather product owner collaborates while finalizing the acceptance criteria of a user story.

The proposed Agile testing life cycle revolves around regression testing as shown in Figure 3.3. In directly, test case management is accomplished by managing test suite of user stories. In the next section, regression testing of the inner circle is elaborated.

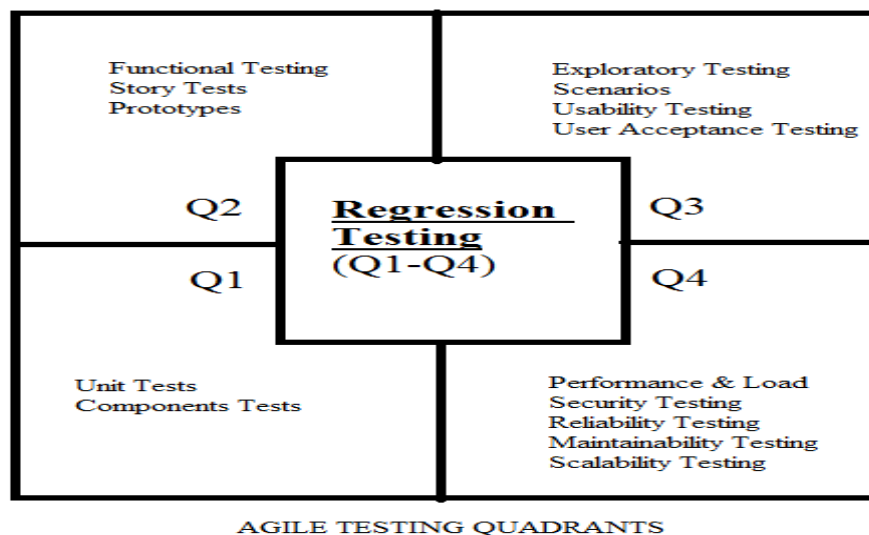


Figure 3.3 Regression Testing in Matrix



### **3.2.1 Agile Inner Testing**

In the inner circle, tester collaborates with team members and product owner. Tester's job is to manage test cases along with delivering quality product to the customer. In Agile, regression testing is important as Agile is based on responding to change over following a fixed plan. Therefore, regression testing is an ongoing activity in Agile. After looking at the verification points of the user story, tester starts with writing failed test cases. For any user story of the sprint, test cases are designed by testers using test driven development (TDD) approach which means failed test cases are written for the upcoming user story. Using this, developers try to convert these failed test cases into pass test cases. This approach of testing comes under white box testing. This TDD approach may be implemented in pair programming style in which on single terminal first failed test cases are written and then code is written by developer. This practice helps in getting the immediate feedback so as to embed quality in the final deliverable.

Further, unit tests for any user story are written by extracting the support from the automated tools like Eclipse for Java application, Xunit for web based applications. This helps in reducing overall time for any sprint. Also, pattern may be utilized so as to handle existing problem with best evolved solution. By following pair programming practice, functionality of the user story is checked that means black box testing is performed as per the verification points of the ready story. During the sprint, integration testing is also performed among user stories of a sprint by considering dependencies among the user stories. Moreover, integration testing is also performed among user stories of the different sprints. Further, to manage test cases, effective regression techniques such as regression test selection (RTS), test case prioritization (TCP) etc. are implemented so as to run only subset of the test cases out of all the test cases. In the subsequent chapters, the proposed techniques for RTS and TCP have been discussed.

Finally, depending upon the feedback cycle of customer, product is released by the operational teams in collaboration with tester by performing all needed testing including usability, scalability etc. Feedback of customer is really a great input for getting great quality product. As per the Pareto Principle, 80% of value is reflected in the software product by 20% of inputs at right time. In this life cycle, customer of outer circle

is interacting with every team member comprising tester. Furthermore, definition of done is declared by customer after matching the verification points of ready story with the actual product. This is an easy way to check the validity of user stories in a sprint.

### 3.2.2 Online Bugs Tracker

Ready story and user story are represented on the online story board which in general is created in Microsoft Excel. This online story board format may vary from one organization to another. To differentiate among status of user stories along with testing information in a sprint, colored notes may be placed on hard board tracker. For example, red color is used to reflect high risk user story, blue color is used to reflect low risk user story etc. or red color is used to reflect complex user story, blue color is used to reflect simple user story or red color is used to reflect a user story with many severe bugs and having high priority, blue color is used to reflect a user story with few number of bugs having low priority etc. These colored sticky notes may be utilized to differentiate among the user stories of a sprint. Further, an alternate coding scheme has been proposed to reflect the bug status for a particular user story in a sprint.

This alternate coding scheme is based on various mathematical sequences such as Fibonacci series, arithmetic sequence, geometric sequence etc. One of the examples for tracking bugs in a sprint using Microsoft Excel is shown in Table 3.1 which is based on Fibonacci series. Each number of the series is a unique representation of the bug status in the user story on particular day in the sprint. Specifically, zero code represents that user story is as per the customer requirements. Similarly, 3 number in the Table indicates that user story 1, on day 2, has

Table 3.2 Online Bug Tracker

ONLINE BUGS TRACKER					
SPRINT	15	Duration	15 days	Customer	pqr
Coding	$F_n = F_{n-1} + F_{n-2}$	Project	xyz	Scrum Master	Tom
	Day 1	Day 2	Day 2	Day 3	
User Story 1	1	3	8	0	
User Story 2	2	8	0		
User Story 3	0				
User Story 4	1'	1	1		
User Story 5	5	0			
Code	0	Green Card	No Bugs		
	1	Single Bug	High Risk		
	1'	Single Bug	No Risk		
	2	Two Bugs	Medium Risk		
	3	Three Bugs	Low risk		
	5	Five Bugs	Unit testing		
	8	Eight Bugs	low risk more effort		

3 low risk defects. Also, for resolving specified bug, a solution may be provided by the tester in the specified cell of online bug tracker by inserting comment, if team members in a pair are changing at different locations having different time zones.

This approach is beneficial when team members are located in different locations. Further, if time zone is different in different locations, then by looking at the comment, team member may start working without any communication. Moreover, this code which is assigned to user story on any particular day is useful in tracking performance of team members and for determining pending work in any sprint.

### **3.2.3 Sprint Flow Diagram**

Scrum methodology of Agile is based upon small duration sprints having small number of user stories listed in Sprint backlog list (SBL). SBL is subset of PBL. After doing effort and complexity estimation by PO, SBL is finalized. This methodology is divided into three phases. The Scrum phases are pre-execution phase, execution phase and post-execution phase as discussed in Chapter 2. However, these Scrum phases do not elaborate or identify testing activities. In this work, all the testing activities occurring before, within and after the sprint have been identified. For the purpose of simplicity, only three sprints namely, S1, S2 and S3 are taken in the sprint flow diagram having three phases (See Figures 3.4 to 3.6), having duration of execution  $W1$ ,  $W2$  and  $W3$  respectively where  $w$  stands for week. In the execution phase, a sprint S1 may be completed having  $n$  number of user stories from (SBL1) as discussed in chapter 2. Similarly, S2 may be completed having  $m$  number of user stories from SBL2.

Figure 3.4 shows testing scenarios in pre-execution phase of Scrum methodology. This phase starts with collaboration among customer, market evaluator, product owner and tester. They sit together to finalize the user story and ready story. The ready story is based upon the confirming points which are as per the market standard, technology and competitor's product features. These confirming points are also known as acceptance criteria. During the sprint, these acceptance criteria's are frequently checked. Also, tester performs exploratory testing so as to check the feasibility of various scenarios. After finalizing the ready story and user story, a list is prepared having all the finalized set of

user stories. The list is known as Product backlog list (PBL) which is input for the second phase that is execution phase.

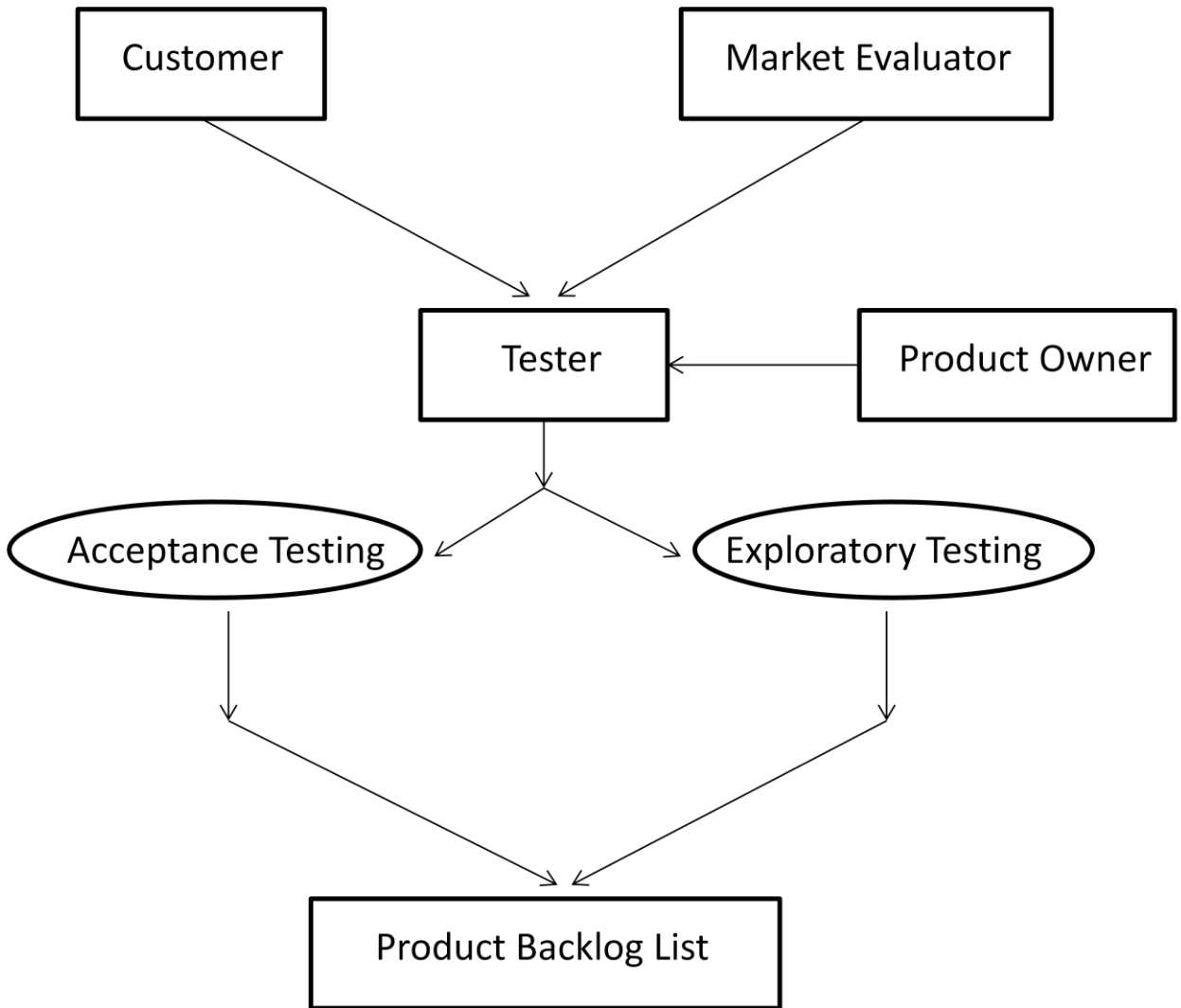


Figure 3.4 Testing Scenario in Pre-execution Phase

After receiving the input from the previous phase, execution phase starts which is shown in Figure 3.5. PBL is analyzed by PO and effort estimation is done for selecting the user stories for SBL1 and SBL 2. SBL1 and SBL2 are executed in sprint S1 and S2 respectively. In S1, tester performs unit testing with TDD or white box testing, functional testing or black-box testing, regression testing, integration testing among dependent user stories and many more depending on the requirements set by customer. The output of S1 is integrated tested set of user stories IT1 with regression test suite during W1 duration. Similarly, in sprint S2, same types of testing are performed and integrated set of user

stories IT2 with regression test suite during W2 duration. Further, these integrated set of user stories IT1 and IT2 are considered as user stories in SBL3. Also, there may be other user stories which need to be developed in W3 duration which are newly added feature in the maintenance time of the product. SBL3 is input for the post-execution phase which is shown in Figure 3.6.

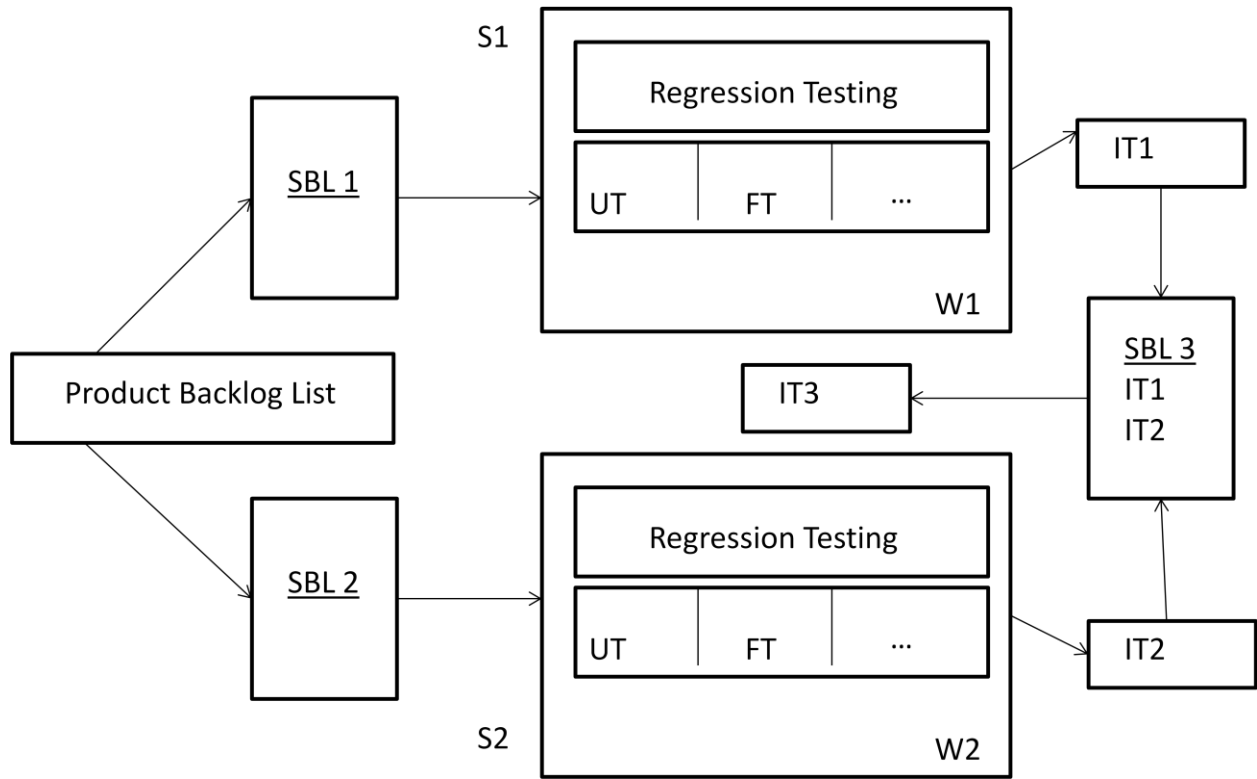


Figure 3.5 Testing Scenario in Execution Phase

In post execution phase, user stories are selected from the SBL3 based on the priority set by the customer or complexity level or risk level or any other prioritization factor. In this phase, different types of testing are performed based on the customer need. Various types of testing that are mandatory in S3 are integration of IT1 and IT2, functional testing, system testing and regression testing depending on the modification suggested by the customer, if any. Other optional testing that may be performed in W3 duration are compatibility testing, security testing, performance testing, usability testing etc. Finally, software product is delivered to the customer.

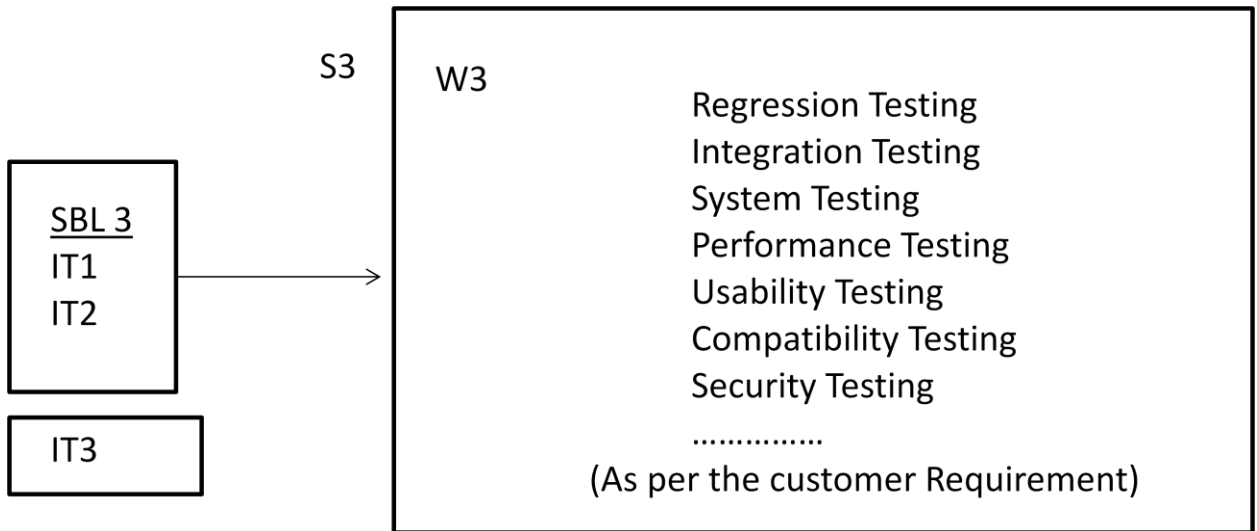


Figure 3.6 Testing Scenario in Post-execution Phase

### 3.2.4 Benefits Of Agile Testing Life Cycle

The proposed Agile testing life cycle defining role of tester before start of a sprint, within a sprint and post sprint is useful for pilot organizations who are about to do transition from traditional model such as waterfall, spiral models etc. to Agile model. An Agile tester collaborates and communicates with other stakeholders mainly customer and team for delivering customer product to the customer. Along with the team, customer is also present to provide feedback for the work done in the sprint. Some of the benefits of the proposed testing life cycle are listed below:

- The regression testing which has to be started from very beginning of the sprint helps in managing test cases in an efficient manner.
- The outer circle and inner circle concept used in formulating detailed testing activities is also helpful in getting big picture of the market scenario in line with the customer requirements.
- The acceptance criteria set by tester in collaboration with market evaluator helps team and customer in verifying the user story which is the output of the overall effort put by an organization.
- The regression testing techniques like regression test selection and test case prioritization helps in reducing time when frequent changes are introduced by customer.

### **3.3 CONCLUSION**

In this chapter, interaction of a tester with other stakeholders along with testing activities has been explained in the proposed Agile testing life cycle, which revolves around regression testing. Moreover, quadrant has been defined for regression testing which covers all quadrants. Further, for Scrum methodology, a sprint flow diagram has been discussed by mentioning all testing activities before the sprint, within the sprint and post sprint. In the next chapter, a framework has been proposed for performing effective testing in distributed environment.

# **CHAPTER IV**

## **AGILE TESTING IN DISTRIBUTED ENVIRONMENT**

### **4.1 INTRODUCTION**

Software market is booming at a tremendous pace, but long term goals of customer satisfaction, risk management etc. are still a problem. One of the reasons behind all this is quality factor. Maintaining and attaining the quality of software projects is tedious task but important also. Many of the IT projects are outsourced in the global market for the purpose of getting the cheap rate professionals or sometimes to get the expertise in the specific field. Cooperation, coordination, communication and collaboration are basic pillars for successful Agile software project. This outsourcing in distributed environment is troublesome as face to face communication cannot take place which is the backbone of an Agile culture. Further, this task is even more complex as pair programming like best practices are difficult to follow in the distributed environment.

Further, ASD is based upon the rapid feedback cycle during the sprint. This cycle comprises of feedback from two parties. Firstly, it is given by the team members and secondly by customer. Improvements are incorporated in the next sprint by keeping in mind these feedbacks. Also, quality strategies are improving a lot by considering different scenarios and experiences of the Agile experts. It means quality strategies are also dynamic but too much flexibility sometimes becomes a hurdle in the path of distributed environment. Quality can not be achieved until and unless software is tested in any environment for the purpose of detecting and resolving bugs. Now, the problem is more severe when teams are working at remote locations in diversified culture. In this culture, timezone may be different, language may be different, working style may be different and many more differences which may create problem while following pair programming/testing practice as an effective communication is not possible. Therefore, a framework has been proposed for the purpose of attaining, maintaining and improving quality in the distributed Agile context by considering the various research challenges.



## 4.2 AGILE TESTING IN DISTRIBUTED ENVIRONMENT

Agile testing does not emphasize on traditional testing procedures and manual test case format, rather it is built upon the strategy that testers need to adapt to rapid deployment cycles of testing. Agile testing involves testing from the customer perspective as early as possible and as often as possible, since working increments of the software are released frequently in ASD. This is commonly done by using *automated testing tools* to minimize the amount of manual effort involved and *pair testing* in which single terminal is shared by two members for the purpose of providing and implementing that feedback for good quality.

For globally distributed web applications, DAD (Agile Framework for Globally Distributed Development Environment) model is published in 8th WSEAS International Conference on Applied Informatics And Communications (AIC'08) Rhodes, Greece, August 20-22, 2008. This model concentrates on full time communication between customer and team. Also, this model talks about phases that can be used for the distributed product delivery. However, this model does not teaches the scenario when team members are separated at different locations.

In the distributed pair programming (DPP) tracking system, two persons work together for common goal when their locations are different. In DPP, sharing of terminal from different locations has been discussed by mentioning its challenges. Further, a conversation model with commitments is presented based on language/action perspective as a framework for understanding communication within DPP processes. This pairing model is effective to use when navigator and driver both follows the holistic approach of tracking and updating the progress information in explicit manner. More specifically, DPP tracking system focuses on pair programming/testing member's performance tracking at different locations. However, this DPP model does not discuss the mechanism for forming a pair among team members of different locations for pair programming/testing practice. Also, simplification of code (refactoring) is required for getting better understanding of it when time zone differ and no communication can take place. For implementing these practices in diversified culture from scratch is a big challenge that's why some way is needed which may work out for solving present issues.

That's why, this chapter discusses refactoring and pair programming practices from testing viewpoint.

In next section, a framework for distributed environment has been proposed which follows pair programming and refactoring practices. Further, a novel method has been discussed for forming pairs for distributed environment.

### 4.3 PROPOSED FRAMEWORK FOR DISTRIBUTED ENVIRONMENT

The framework varies from collocated environment to distributed environment. Here, distributed environment is elaborated by considering pair working where one is the driver and other is the follower. Also, both are sitting at different locations. But they are collaborating, coordinating, supporting and communicating despite of the culture and time zone differences. In Figure 4.1, a framework for the distributed environment has been proposed. In this proposed framework, main components are:

- **Teams members at Location 1** The hardware needed for the pair working is a first complete system of video conferencing and Live meeting at location  $m$ .
- **Team members at Location 2** The hardware needed for the pair working is a second complete system of video conferencing and Live meeting at location  $n$ .
- **Central Repository** It is a database [58] (DB) of information related to the user stories, team, market, customer, competitor, timelines etc. in the proposed framework. This DB may be accessed by team members of both the locations at any time. Team information in this repository may include attributes of different team members. Also, some additional details may be entered in the repository which may be useful for taking some important decisions in future. The practices like pair programming and refactoring are already present in some of the methodologies of Agile but using these practices in distributed environment is cumbersome, so, for the purpose of implementing these best practices require significant amount of effort before using practices. That's why to decrease that effort the pair programming has been elaborated by proposed buddy identifier technique

and refactoring has been elaborated by simplifying code using object oriented principles. The central repository comprises components like online story board, web based query tracking tool, code base and test suites, open source software and manual and many more. For the purpose of understanding of this repository, the listed components are explained as:

- a) **Online Story Board** It is the consistent way of maintaining and managing story cards.
- b) **Web based query Tracking Tool** It is the quick and secure way of tracking, submitting and storing any query.
- c) **Codebase and Test suite** Code of any module can be accessed by any team member of any location for the purpose of change request in the sprint. For keeping check on updates in code, versioning mechanism is needed, so that, latest copy is stored in the central repository. With it, all check out and check in are tracked so that updated copy can have the proper version number or release number.
- d) **Open Source Software and Manual** Without automation, it is tough to proceed in distributed culture. Access to the entire open source software's should be there to every stakeholder. Along with it, online user manual copy should be available so that anybody can read it from any location.
- e) **Team members attribute** In the central repository, the next component is database of team member's attributes. These attributes may be accessed by any team member of any location. A pair may be identified by product owner or by following "self organizing principle" among team members so as to work in collaboration for the purpose of delivering product to customer.

In this framework, team members at location 1 and 2 are shown who are also pair members for pair testing/programming practice. They are also using refactoring practice for doing simplifying code. Each team member has her own video conferencing setup for the purpose of communication. These two members have access to central repository

which is stored on the server. They are able to access components (a)-(e) which are mentioned in the Figure 4.1. For the purpose of maintaining consistency among code or test suite which is one of the components of central repository, versioning system may be adopted by each team member. In this framework of distributed environment, focus is on identification of buddy identifier for a pair. A rigorous analysis may be performed for this work before starting any sprint. Further, refactoring by this pair may be executed by following some standard object oriented principles.

In the next section, buddy identifier techniques have been proposed which are based on attributes of team members which are present in the central repository of the discussed framework.

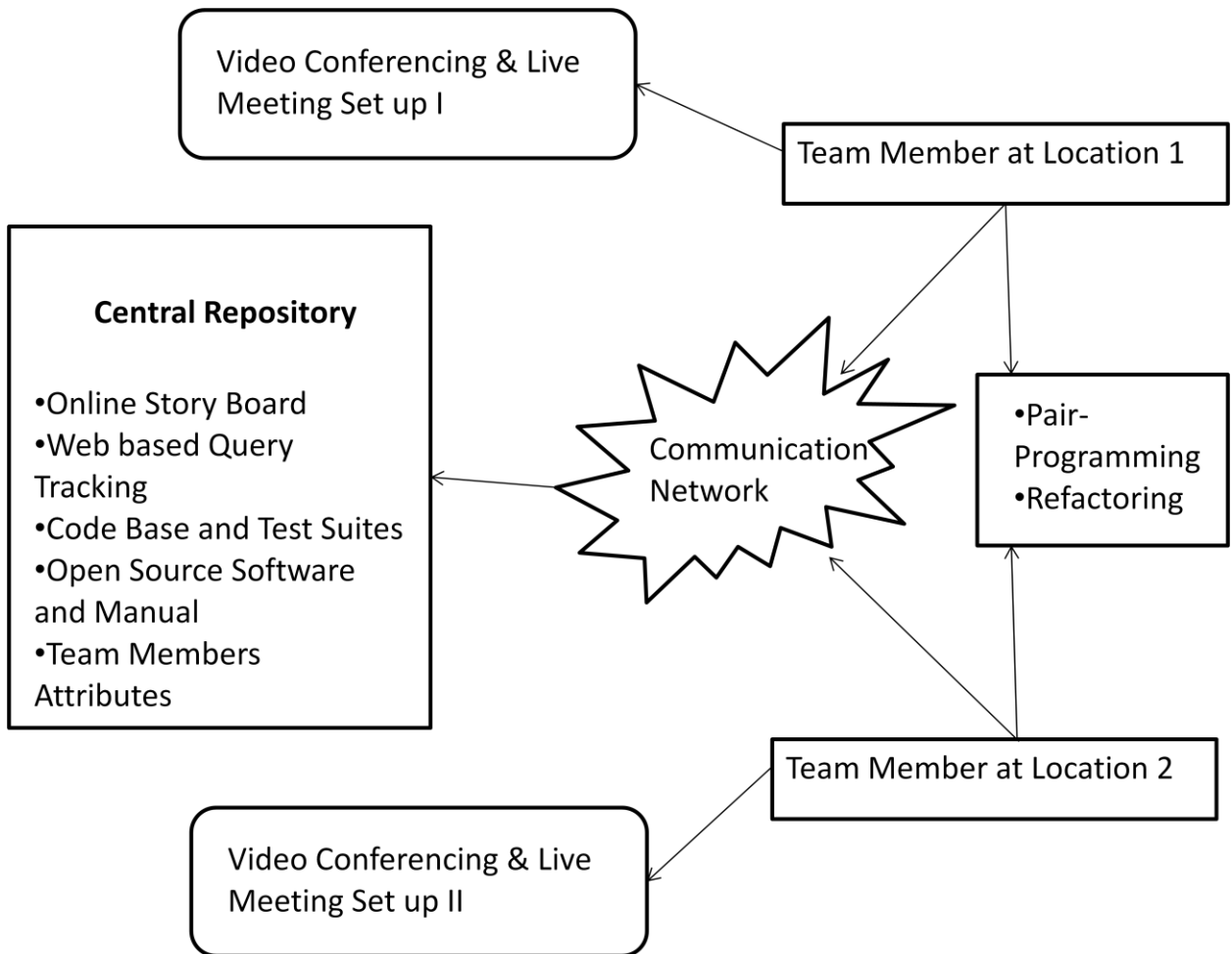


Figure 4.1 Framework for Distributed Environment

## 4.4 PAIR PROGRAMMING IN DISTRIBUTED ENVIRONMENT

The execution of a sprint in the distributed environment is troublesome for the team members of the pair. The foremost issue is how to make pair among separated team. In the existing literature, there are few methods for making pairs in a team such as an experienced with novice, novice with novice and experienced with experienced. These methods of pairing has problem of compatibility among team members. For said issue, a buddy identifier technique, which is based on self centric approach, has been proposed that analyzes the common attributes of the team members which are maintained in a log of the central repository. Then, PO makes strong pairs among team members based on the data collected in the log of the central repository.

### 4.4.1 Proposed Buddy Identifier

In this subsection, a novel approach has been discussed to overcome this compatibility issue. This new approach is based on a self centric network. This may also be known as personal network or ego network. It is a type of network which is based on the personal preferences of a buddy. Personal preferences may be in the form of personal interests, personal hobbies, personal values [60, 61] etc. A graph  $G = (V, E)$  is an undirected graph in which links between vertices are present when there is some personal interest between two nodes. The personal network in the form of a graph  $G$  having vertices  $V=3$  and edges  $E=3$  is shown in Figure 4.2. In this Figure, starting node which is A, is known as a *seed/source node*. If A is connected with two more nodes B and C then B and C are known as *sink nodes*. In other words, A has two interest's e1 and e2, further B and C has e3 interest.

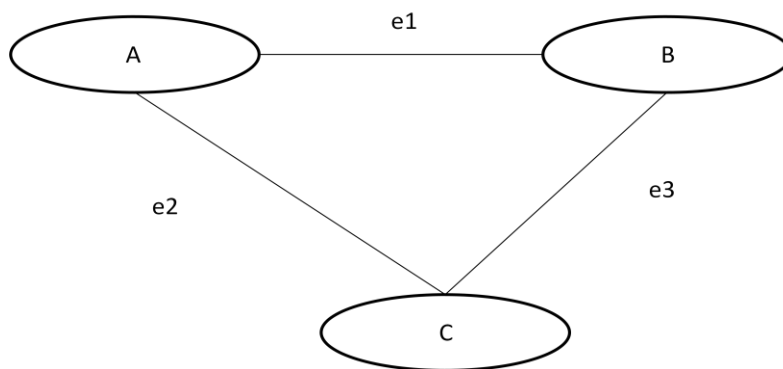


Figure 4.2 Ego centric graph

More specifically, source node A and sink node B which are connected by an edge  $e$  is shown in Figure 4.3. There may be  $n$  number of nodes that may be connected to seed node. This is the general philosophy that a connection is established for the purpose of specific reason. After some time, strength of that connection may be measured and a change can be accommodated for growth and stability, depending upon the need of the system in which personal network is to be implemented.

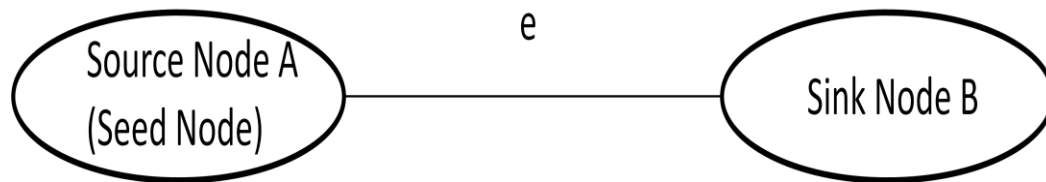


Figure 4.3 Common Habit Representation

The identification of a buddy in a pair is done by analyzing the behavioral features of the team member. This is the human tendency that similar minded people are good friends. For example, common aspects of friendship are nature, interest or taste, regional background, habits, skill sets etc. Similar minded people try to spend more time with people who know moral values, who love rugby, who belong to same area, who love to collect retro style pictures, who are expert in Java etc. For identifying buddy for the pair programming, these commonalities can be suitably taken care of. Before starting any new sprint with a team, PO can distribute one questionnaire for all the team members. This distribution of questionnaire may be through various means like email, meeting, group chat, etc. After analyzing the questionnaire of all the team members, the PO may apply clustering techniques to form cluster of similar minded people. As per Agile, responding to change is a good practice over following a plan. Therefore, rotations may be planned as human behavior changes with time.

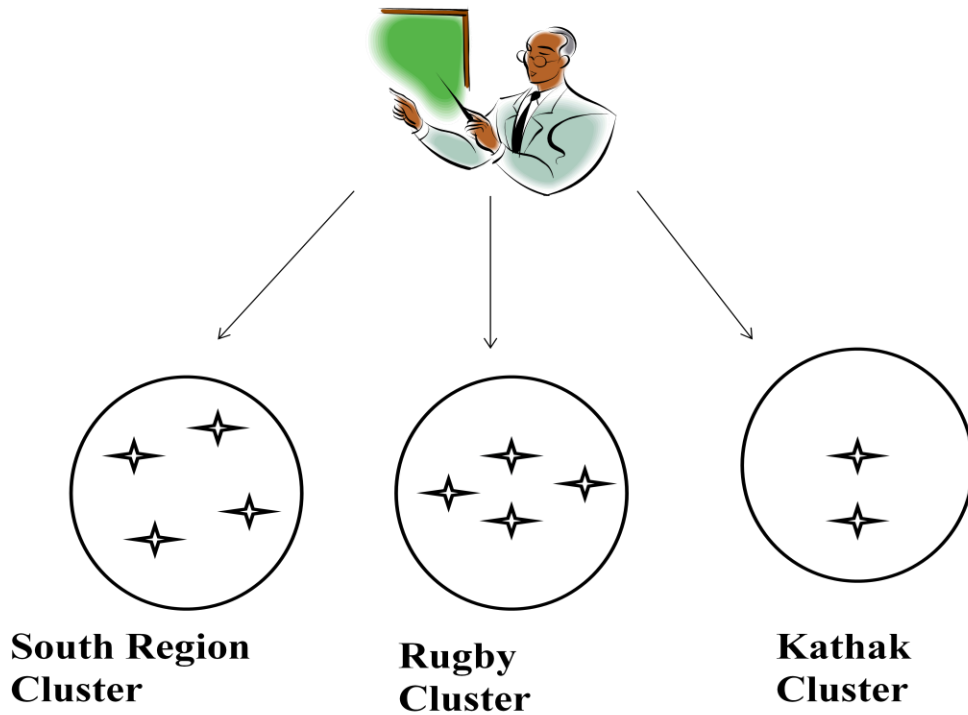


Figure 4.4 Common cluster

The Figure 4.4 discloses that PO has 3 clusters. Each cluster is special in its own commonality. For example, 1<sup>st</sup> cluster is representing members from South India. 2<sup>nd</sup> cluster is depicting members who love to play rugby. Last cluster is cluster of Kathak dancers. These clusters help in buddy pairing in a better way.

The South region cluster has been shown separately in Figure 4.5. In this cluster, 4 team members namely 1, 2, 3 and 4 are shown who belong to south cluster having good compatibility among themselves as compared with the rest of the members. If one cluster has more than two members, in that case rotation may be permitted within the cluster. So, in this south cluster, pairing is possible within the cluster. For example, team member 4 may be paired with 1 or 2 or 3 team member, if 4 is the seed node. If (4, 1) pair is not producing high productivity then rotation may be done within the cluster by doing pairing among (4, 2) or (4, 3).

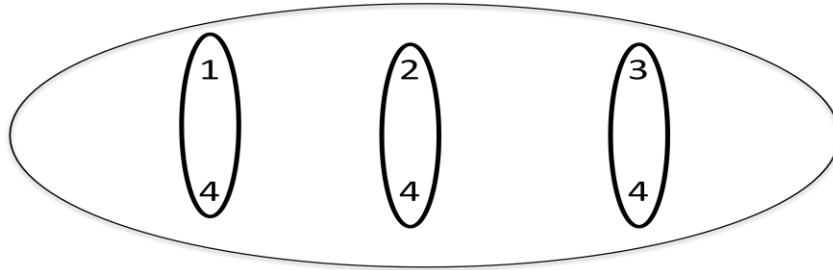


Figure 4.5 South Cluster

In other scenario, one member may be part of more than one cluster, in that case, rotation may be permitted across the clusters. These clusters may be called as an *overlapping cluster* (See Figure 4.6). For example, one member may be interested in two clusters such as music cluster and drama cluster. Then, that member can have buddy from either of these clusters. In that case, decision of pairing a buddy may be taken by any team member or PO depending upon the process followed in an organization. Largely, the organizations follow Self Organizing principle [], so, team members may decide among themselves for the pairs. In Figures 4.4 and 4.6, decision of pairing is taken by the PO. Further, rotation may be permitted depending upon the feedback from both sides. For example, during 1<sup>st</sup> cycle of buddy programming, A (Driver or Leader) is buddy of B (Learner) (See Figure 4.7). During review session, PO notices that productivity is less than threshold mark in an average scenario, then PO may suggest rotation of A with her overlapping buddy so that in next sprint or cycle, productivity level may be enhanced among team members. (See Figure 4.8)

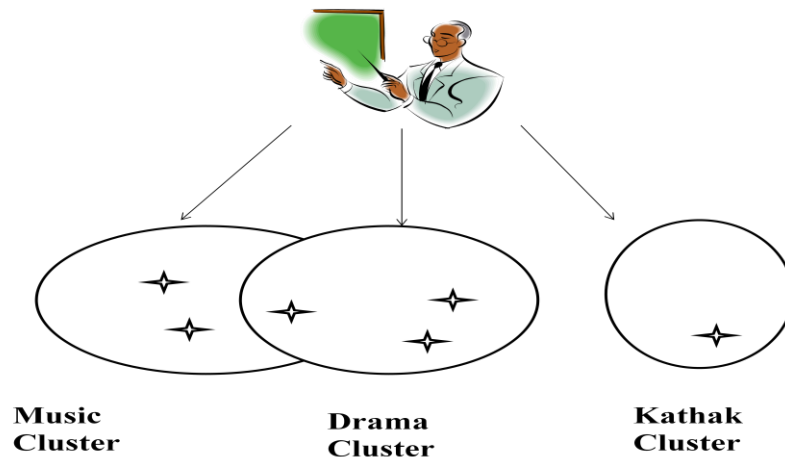


Figure 4.6 Overlapping cluster



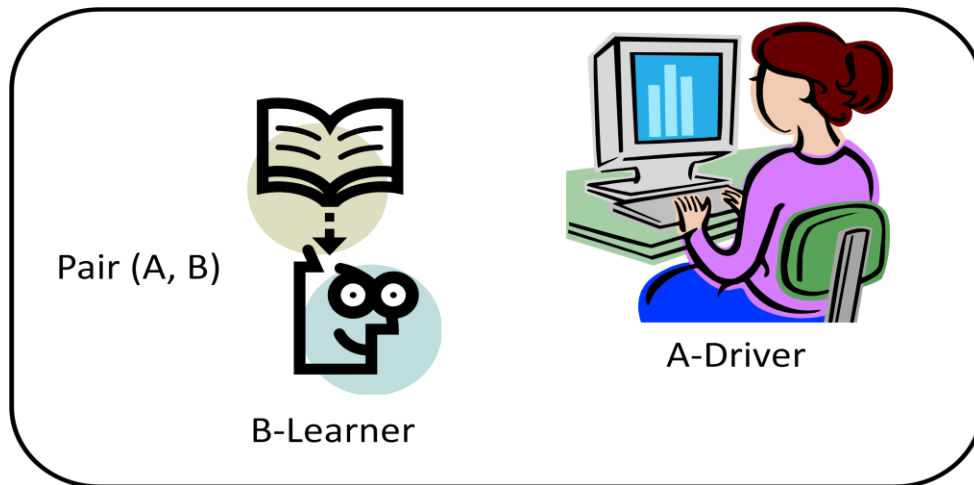


Figure 4.7 A pair

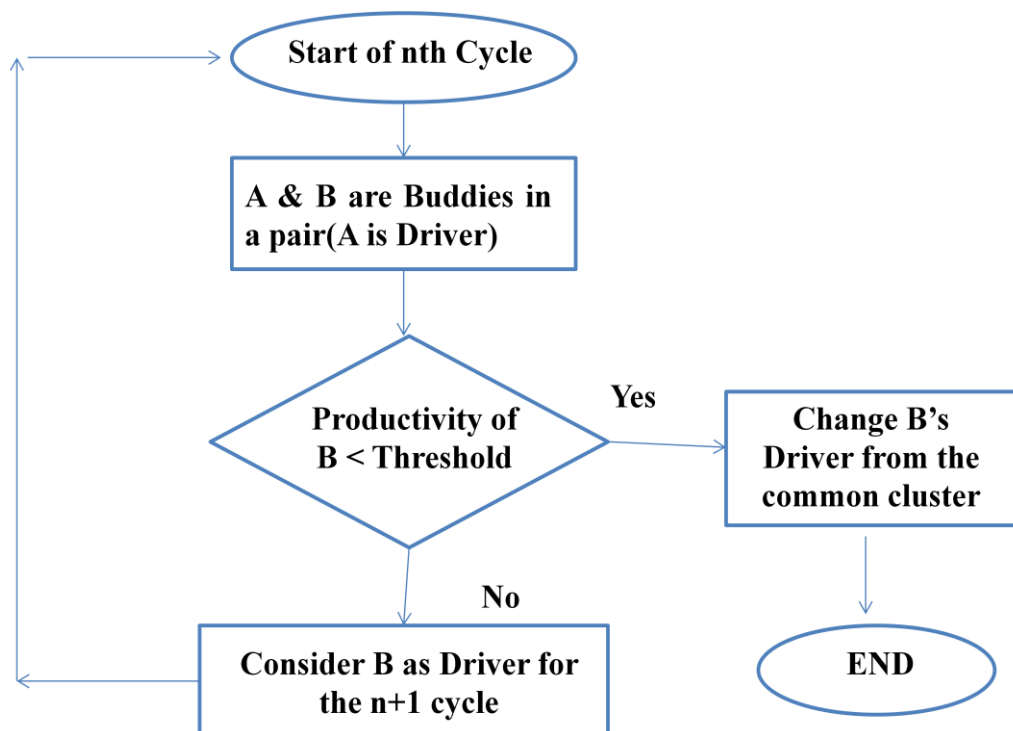


Figure 4.8 Buddy Rotation

The proposed buddy identifier technique discussed in this section is useful when team members are present in different locations having diversified culture. The team member information stored in the central repository may be utilized for making a good pair having compatible team members. By using buddy identification technique,

compatibility among team members is increased when their location is different. A compatible pair working results in fruitful outcome in less time as compared with any pair. Accordingly, there would be less defects in the sprint and time to delivery of software product would be less.

#### **4.5 REFACTORING IN DISTRIBUTED ENVIRONMENT**

Design is the crucial step while working on any user story during the sprint in an Agile culture of software development. A good design can generate good code and moving further in the journey, a good code would have less or minimum bugs by utilizing benefits of various principles of Agile like simplicity, pair programming, less is more approach etc. to its fullest.

In this section, a step wise source code design approach has been proposed for the purpose of obtaining improved code from rotten code (having bad design) using regression testing and refactoring/rewriting methodology. Definition of improved code is proposed on the basis of various design principles like Open Close Principle, Dependency Inversion Principle, Interface Segregation Principle, Single Responsibility Principle and Liskov's Substitution Principle. This improved design of code executes the same behavior irrespective of the change in the design feature of the original code. The presence of critical errors in the previous steps slows the performance of refactoring process. That's why, a regression test need to be performed at every step of the sprint and product need to be tested for better performance.

The syntactic and semantic checks may be performed so as to ensure consistent behavior of the user story after refactoring. With this approach scalability/extensibility chances are higher in distributed environment as team members follow simple design with refactoring practice and continuous regression testing.

The process flow diagram (See Figure 4.8) for the step wise conversion of the object oriented source code into improved code (rewriting or refactoring) is comprised of various components having source code for the user story, test suite of unit tests (UT) and acceptance tests (AT) for the user story created by the tester, regression test suite from previous sprints and finally acceptable and rewritten/refactored code based on the object oriented principles.

In one of the scenarios, source code may be rewritten when change request is requested by the customer for the existing user stories, when code review is performed by the pair programming members and when new defect or problem of high severity is detected. Refactoring is applied on the original source code by applying object oriented principles so that there can be escape from the bad design for future sprints. After doing refactoring of the source code, same test suite having unit tests and acceptance tests are applied to the new code after applying syntactic and semantic check. Also, behavior remains consistent by restructuring the statements of the original source code. This correctness check is precondition before the release of the sprint's output to the customer. As Agile is iterative and incremental, that's why regression testing is also incremental.

Regression testing in this section is limited to unit tests and acceptance tests but it is not only limited to these two tests rather it can include all types of testing which are covered under sprint flow diagram discussed in chapter 3. It is an ongoing process in Agile. The bulky size of test suite makes the testing task cumbersome. So, to release the deliverable on time, regression testing techniques may be applied which are discussed in subsequent chapters of this thesis. This technique will save time, resources and quality check on refactored code would be performed by retaining the original behavior of the source code. Regression testing may be performed manually but to speed up the process it can be done by using different automated open source tools. Similarly, refactoring may be implemented using tools such as *xrefactory*.

#### **4.6 REFACTORING EXAMPLE**

Simple design approach using object oriented principles has substantial effect in programming models. Specifically, simple design is useful in distributed Agile environment while following pair programming practice in which locations of pair mates is different. In that scenario, team member of a pair is able to understand the code written by other pair member. In the suitable time zone, strategies are discussed and implemented using simple design practice. One of the bad example is discussed below for reference (Interface Segregation Principle).

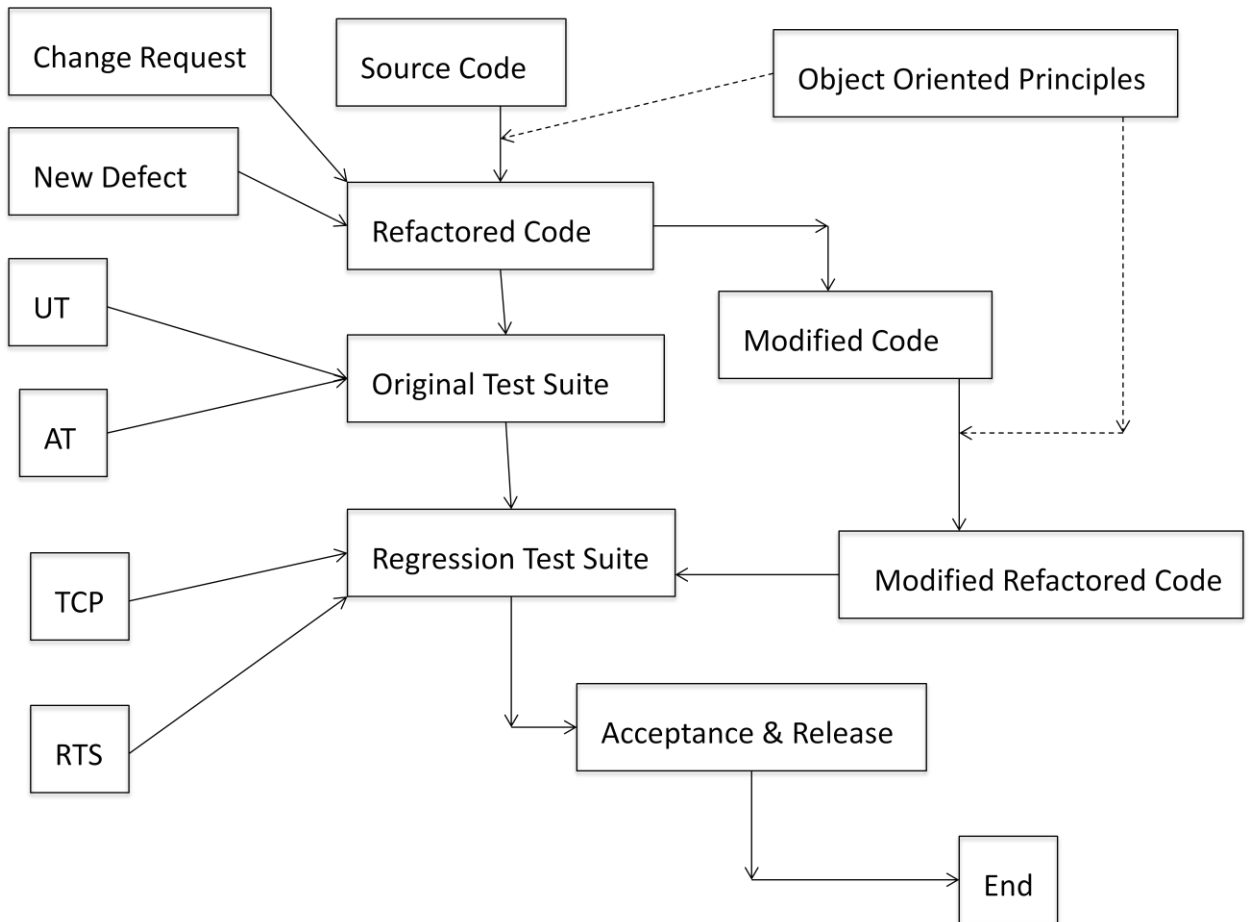


Figure 4.9 “Refactoring using Object Oriented Principles in Distributed Environment”

Table 4.1 Testing Abbreviations

Abbreviation	Full Form
UT	A set of Unit Tests
AT	A set of Acceptance Tests
TCP	Test Case Prioritization technique
RTS	Regression Test selection technique

```

class Example
{
    public:
        virtual void e1() = 0; //pure virtual function
        virtual void e2() = 0; //pure virtual function
};
class A : public Example
{
    public:
        void e1() //Class A e1 method
            { ... }
        void e2() //Class A e2 method
            { ... }
};
class B : public Example
{
    public:
        void e1() //Class B e1 method
            { ... }
        void e2() //Class B e2 method
            { ... }
};
class My
{
    Example *e;
    public void setvalue(Example *w)
    {
        e=w;
    }
    public void try()
    {

```

```
        e->e1();
    }
};
```

This is an example of rotten code (bad design) as object *e* of class *my* is referring to only first method *e1*. Nowhere object *e* is calling *e2* method. So, unnecessary both the methods are present in single place. There is a need to simplify this design by using separate interface for *e1* method and *e2* method.

#### **4.7 BENEFITS**

Communication and coordination would scale up as central repository is there to provide the needed tools and support. Also, by using Twist automation tool with this framework can incorporate growth while working in different locations as this testing tool works in natural language of the specified culture. This paid tool supports data driven testing and handles complex changes fast. Other supporting solutions in distributed environment are:

- Using VSS (Microsoft visual source safe tool) or any other versioning management tool for managing and accessing the latest version.
- By sharing the design pattern for similar kinds of problems in different locations (onshore and offshore). Notification/Alert tools can be beneficial when something fishy is expected because of non overlapping hours.
- Client interaction model for multi way feedback to multiple locations is suggested for enhancing the quality.
- Selection of right open source tool for right kind of job and sharing among all is important.
- This ensures that bad design is converted into consistent code having simple design that is the fundamental requirement in XP methodology of ASD.
- This model ensures that precondition of syntactic and semantic correctness is achieved after refactoring the code. For this, test suite comprising of unit tests and acceptance test is run iteratively so as to have the good design.

- Refactoring are mentioned by way of xrefactory plugin for faster delivery and to save time.

#### **4.8 CONCLUSION**

A distributed framework has been proposed for Agile environment. Specifically, pair programming and refactoring like practices has been discussed for handling different challenges of distributed environment. The pair members for a pair can be identified using proposed buddy identifier approach. The next subsequent chapter is related with one of the regression testing technique regression test selection.

# CHAPTER V

## AGILE REGRESSION TEST SELECTION TECHNIQUE

### 5.1 INTRODUCTION

Agile software development (ASD) [15, 28] has attracted major players of the software industry. This development approach has brought significant changes in the organizations in terms of fast delivery, less documentation, more satisfaction and more interactions. One of the important changes in ASD is acceptance of frequent changes introduced by the customer. An effective handling of frequent changes during development is one of the important motives for software professionals. These frequent changes cause aggregation of test cases in the test suite and may affect the time to delivery of software product to the customer. To manage this large test suite, an effective test case management is needed so that past user stories does not regress. In this chapter, regression test selection (RTS) technique is proposed which is based on the optimality of path in a weighted story graph.

Todd L. Graves et al. [84] has discussed analytical and empirical evaluations of the existing RTS techniques. They conducted an experiment to examine the relative costs and benefits of several regression test selection techniques. The experiment examined five techniques for reusing tests, focusing on their relative abilities to reduce regression testing effort and uncover faults in modified programs. Their results highlight several differences between the techniques, and expose essential tradeoffs that should be considered when choosing a technique for practical application. In said work, optimal connection approach in an Agile environment for an undirected graph is missing which is less prone to faults.

Emelie Engstrom et al. [40] recites an efficient fix cache RTS technique. It makes use of information that already is collected and stored in different databases. Setting it into use involves mainly connecting these databases together. The empirical evaluation is used in this technique. The set of test cases that were selected and executed found significantly more defects per test case. The technique selected a small set of test cases, so the number of faults found is very small compared to the number selected by the



manual method. The size of the cache is a factor that impacts on the number of selected test cases. Future evaluations include varying the cache size, and evaluating the efficiency for various sizes of the cache. The issue with this approach is to use expensive cache which may have significant effect on the overall budget of the sprint.

The proposed RTS approach takes into consideration story point of the user stories in an undirected graph and optimal nature of this proposed method removes other risks of the development comprising potential edges of the undirected graph.

## 5.2 PROPOSED APPROACH FOR REGRESSION TEST SELECTION

RTS approach selects significant test cases from existing test suite depending upon some factor so as to save time and effort involved in early defect detection during or after the sprint. The proposed approach is based upon optimal connections [80] in the weighted graphs. The methodology of the proposed work is shown in Figure 5.1. It consists of following four components. Description related to components is given below:

- **Story graph creator:** Depending upon the existing and new user stories of the system, a graph can be plotted so as to depict the relationship between the stories and an edge from a source node to destination node represent the relationship among the stories. Value on the edge is the effort involved for travelling from source story to destination story.
- **Path strength calculator:** Story graph creator is input for finding strength of any edge of the undirected graph. By using average path value (APV) and average path length (APL) measures for direct and indirect edges, path strength is found in an undirected graph. A measure of average path value between nodes  $n_i$  and  $n_j$  is the ratio of path value to distance and a measure of average path length between nodes  $n_i$  and  $n_j$  is the ratio of total value to distance.
- **Direct/ Indirect paths merger:** These two path strength measures i.e APV and APL, for undirected weighted graphs may identify optimal connections between pairs of nodes that do not necessarily have the middle user stories node which are connected in the graph. More specifically, this component

merges the optimal values corresponding to the specific user story. An optimal connection in an undirected graph involves a combination of the middle nodes with the most intense interactions

- **Comparator and Selector:** Last component of the proposed RTS Technique is concerned with the optimal connections of the user stories. It helps in selecting the significant test cases from the test suite of the user stories.

A case study to demonstrate the proposed RTS technique is discussed in the next section. This case study is based on the user stories defined for the designations used in a law firm. More specifically, user story graph is the basis for implementing the proposed RTS technique.

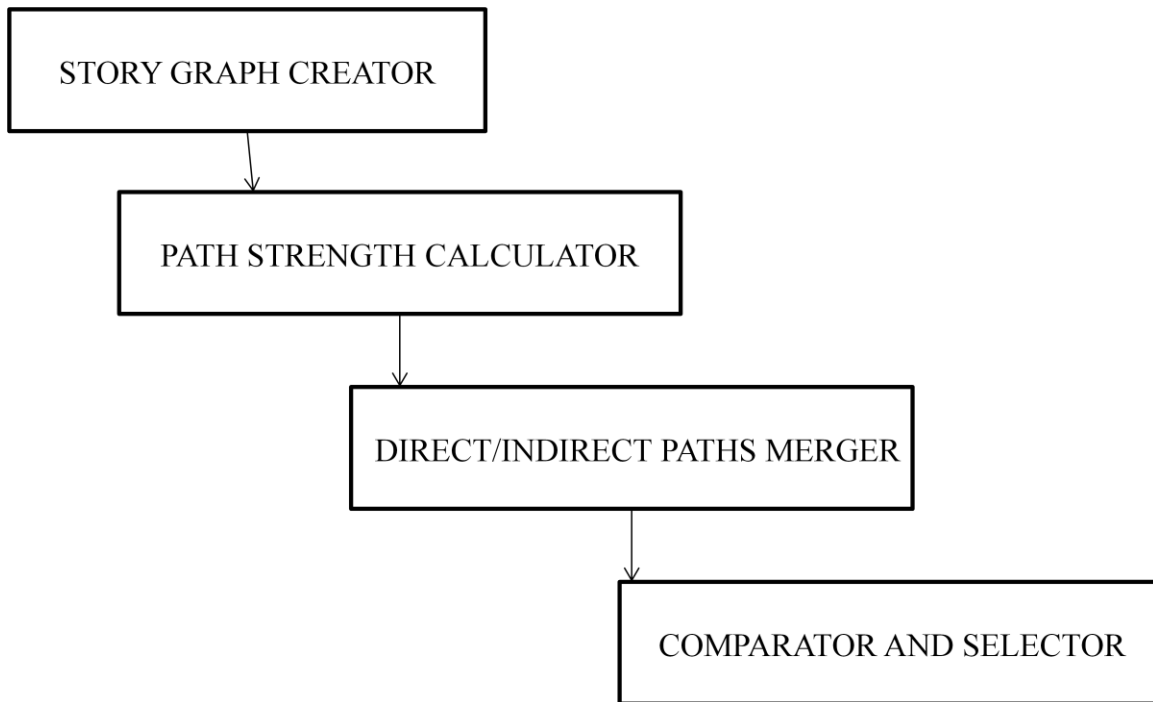


Figure 5.1 A Path Strength based RTS Technique

### 5.3 CASE STUDY

Hierarchy for XYZ law firm starting from low level is secretary, associate, senior associate, partner. Software product to be developed in an Agile environment for XYZ law firm is to analyze the patents granted by the Indian patent office to different

applicants. For achieving successful developed product, first step is to dividing the epic into different user stories. Stories are given in Table 5.1.

In the user stories mentioned below, associated roles are secretary, associate, senior associate and partner. Partner is the person who can directly fetch data from secretary; secretary submits report to the associate; associate submits report to the senior associate and senior associate submits report to the partner. During data generation span, secretary can interact with each other. Communication among different roles is assumed to be dual in nature. Undirected graph corresponding to the specified user stories is shown in Figure 5.2. This task is performed by Story Graph Creator component of the RTS methodology shown in Figure 5.1. This component shows the linkage between the user stories. Two way linkages among stories are represented by undirected graph. In this graph, nodes are user stories and edges are dependency among user stories. For example, user story 3 is dependent on user story 1, 2 and 4 and similarly user story 5 is dependent on 1 and 4.

TABLE 5.1 REQUIREMENTS

Story #	User Story for XYZ Law Firm (Client)
1	As a secretary1, I want to search list of patents granted by Indian patent office to Y Applicant so as to find the technology trend of the software market.
2	As a secretary2, I want to search list of patents granted by Indian patent office to Z Applicant so as to find the technology trend of telecommunication market.
3	As an Associate, I want to do analysis of data of Applicant so as to identify the major players of the field and their significant contribution.
4	As a senior associate, I want to view summary report for all participants.
5	As a partner, I want to attract other players of the same market depending upon their contribution so as to have more clients with maximum benefit.

Respective weights on the edges are story points for the user story. Story points represent effort involved/complexity factor for movement among any source node to destination node. In short, story points shown on the edges indicate how output of one

story is considered as input for another user story. Story points are taken to be odd series 1, 3, 5, 7....., Lowest number represent lowest complexity and higher number represent higher complexity. For the simplicity reason, user story 1 to 5 is represented by alphabets P, Q, R, S, and T.

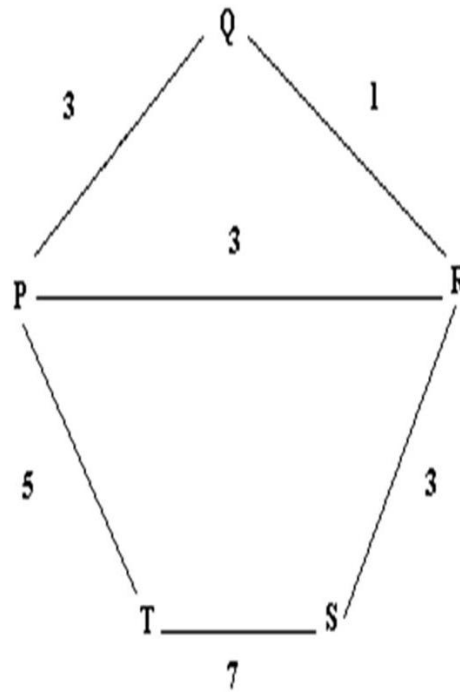


Figure 5.2 Weighted Story Graph

Weighted story graph can be converted into path story Table by considering different paths (direct and indirect between every pair of nodes) using two parameters namely APV and APL. This path story Table contains indirect paths for existing and non existing edges of weighted story graph. Depending upon values of said parameters, optimality can be identified by setting some criteria. A criterion of optimality for APV and APL is finding the maximum entry in the relevant column of the path story Table. For finding said parameters, initially distance between nodes, minimum value and total path length is to be identified. Distance between nodes is calculated by taking into account total number of links that exist between any two nodes of the weighted story graph and minimum value is calculated by considering the path with intermediate nodes

but with minimum weight of the edge that exist between that path. Last parameter which is total path length is calculated by adding weight of all the edges that exist between the paths. For example, if path AB exists with intermediate nodes C and D in the undirected graph, then use equations (1) - (3) for finding the value of equations (4) - (6). Refer Table 5.2 & 5.3 for finding optimality of path for indirect and direct paths of Figure 5.2.

$$AC = 5 \quad (1)$$

$$CD = 6 \quad (2)$$

$$DB = 3 \quad (3)$$

$$\text{Distance between nodes (d)} = 3 \quad (4)$$

$$\text{Minimum value (m)} = 3 \quad (5)$$

$$\text{Total path length (t)} = 14 \quad (6)$$

Each entry in the respective Table shows some significance in relation with the respective story. Any new change in the user story may have significant effect on its related edges. Out of all the related links of that particular user story, a selection has to be made that would make the effect more severe in terms of its story point. In Tables 5.2 and 5.3, this effect is shown by highlighting the specific cell. Maximum entry in the APV and APL column with respect to the existing and non existing edge represents that relation in respect of that effect. Maximum value of APV for example says that a binode like QT with minimum value of  $d = 2$  (less links between nodes) has optimal calculation (APV=1.50). Similarly, binode SP has  $d=2$  and APV=2.50. Exceptionally, QS has  $d=3$  (which is not minimum in this case) and APV=1. Rule for APV optimality calculation is “APV is optimal for the case when  $d < \max (d)$ ”

Similar rule applies for APL of the “Nonexisting Edges” Table 5.2.

“APL is optimal for the case when  $d < \max (d)$ ”

“Existing edges” Table 5.3 shows different optimality rule for different binodes. For example, QP has  $d=1$ (minimum in this case) and APV=3. APV has maximum value for the path QP with minimum link 1. So, rule for APV for nonexisting edges is

“APV is optimal for the case when  $d < \max(d)$ ”

But for APL, after examining the TABLE 5.3, values are optimal for the maximum entry for the specific binode with  $d = \text{maximum number of links}$ . For example, binode RS has  $APV = 5$  and  $d = 1$ . On the other hand binode QP has  $APV = 4$  and  $d = 1$ . This peculiar nature of optimality makes the rule little bit different. Rule for optimality of existing edges says that

“APL is optimal for the case when  $d \leq \max(d)$ ”

TABLE 5.2 NON-EXISTING EDGES

<b>NONEXISTING EDGES (Indirect)</b>						
<i>Binode</i>	<i>Path</i>	<i>d</i>	<i>m</i>	<i>t</i>	<i>APV=m/d</i>	<i>APL=t/d</i>
QT	QPT	2	3	8	1.50	4.00
	QRPT	3	1	9	0.33	3.00
	QRST	3	1	11	0.33	3.67
	QPRST	4	3	16	0.75	4.00
QS	QRS	2	1	4	0.50	2.00
	QPRS	3	3	9	1.00	3.00
	QPTS	3	3	15	1.00	5.00
	QRPTS	4	1	16	0.25	4.00
RT	RST	2	3	10	1.50	5.00
	RPT	2	3	8	1.50	4.00
	RQPT	3	1	9	0.33	3.00
SP	SRP	2	3	6	1.50	3.00
	STP	2	5	12	2.50	6.00
	SRQP	3	1	7	0.33	2.33

“Existing edges” Table takes into account direct as well as indirect path for the specific binodes. For binode RP, R is the source node and P is the destination node.

Direct path between R and P is RP and indirect path that exist between R and P are RQP (via intermediate node Q) and RSTP (via intermediate nodes S and T). These nodes which are actually user story may change at any time. To accommodate that change is crucial job. Effect may transfer from one user story to another as all stories are inculcated in closed loop. This effect may disturb direct as well as indirect paths. Also, not only existing edges but non existing edges may even have stronger effect. The new changes have the potential to affect all possible paths of the closed graph. For example there is change in story Q which says that:

“As a secretary2, I want to search list of patents granted by Indian patent office of Z & Z’ Applicant and want to review work of secretary1 so as to find the technology trend of telecommunication market and to have a check on work for future motivation”

TABLE 5.3 EXISTING EDGES

<b>EXISTING EDGES (Direct + Indirect)</b>						
<i>Binode</i>	<i>Path</i>	<i>d</i>	<i>m</i>	<i>t</i>	<i>APV=m/d</i>	<i>APL=t/d</i>
QR	QR	1	1	1	1.00	1.00
	QPR	2	3	6	1.50	3.00
	QPTSR	4	3	18	0.75	4.50
QP	QP	1	3	3	3.00	3.00
	QRP	2	1	4	0.50	2.00
	QRSTP	4	1	16	0.25	4.00
RS	RS	1	3	3	3.00	3.00
	RPTS	3	3	15	1.00	5.00
	RQPTS	4	1	16	0.25	4.00
RP	RP	1	3	3	3.00	3.00
	RQP	2	1	4	0.50	2.00
	RSTP	3	3	15	1.00	5.00
ST	ST	1	7	7	7.00	7.00

	SRPT	3	3	11	1.00	3.67
	SRQPT	4	1	12	0.25	3.00
TP	TP	1	5	5	5.00	5.00
	TSRP	3	3	13	1.00	4.33
	TSRQP	4	1	14	0.25	3.50

As Q is now modified user story, its sub parts's review reveals that involvement of secretary 1 in it makes the task more cumbersome. In it, no new edges or nodes have been added but still test suite would scale. Table 5.2 presented earlier depicts that Q is attached with T & S user stories whereas Table 5.3 reveals that Q is attached with R & P user stories. Optimal path information in terms of APV and APL for Q user story is represented in Table 5.4.

TABLE 5.4 OPTIMAL PATH

NONEXISTING/EXISTING EDGES						
<i>Binode</i>	<i>Path</i>	<i>d</i>	<i>m</i>	<i>t</i>	$APV=m/d$	$APL=t/d$
QT	QPT	2	3	8	1.50	4.00
	QPRST	4	3	16	0.75	4.00
QS	QPRS	3	3	9	1.00	3.00
	QPTS	3	3	15	1.00	5.00
QR	QPR	2	3	6	1.50	3.00
	QPTSR	4	3	18	0.75	4.50
QP	QP	1	3	3	3.00	3.00
	QRSTP	4	1	16	0.25	4.00

This Table presents paths which are optimal in two categories namely when binode exists and when binode does not exist in the weighted story graph. In both categories, indirect or direct path is chosen depending upon the APL value of specified binode. For first binode QT of Non Existing type, APL value for two paths QPT and



QPRST is same (i.e. 4). In this specific case, out of these two indirect paths QPT would be selected as corresponding APV value is 1.50 and number of links ( $d=2$ ) are minimum. For first binode QR of existing type, optimal APL value is 4.50 but number of links is 4, so, in this case, path QPR would be selected which has APV 1.50 and corresponding value of  $d$  is minimum i.e 2.

Second level of optimization is presented in Table 5.5. This is clear from Table 5.5 that when Q is changing then its effect would be on binodes QT, QS, QR and QP. Corresponding paths of these binodes consist of direct as well as indirect paths. Optimized weighted story graph for Table 5.5 is shown below in Figure 5.3. In it, bold lines represent edges that exist and nonbold represents edges that are nonexistent and have potential to be affected by change in Q user story.

TABLE 5.5 SECOND LEVEL OPTIMIZATION

Optimized Selection						
<i>Binode</i>	<i>Path</i>	<i>d</i>	<i>m</i>	<i>t</i>	<i>APV=m/d</i>	<i>APL=t/d</i>
QT	QPT	2	3	8	1.50	4.00
QS	QPTS	3	3	15	1.00	5.00
QR	QPR	2	3	6	1.50	3.00
QP	QP	1	3	3	3.00	3.00

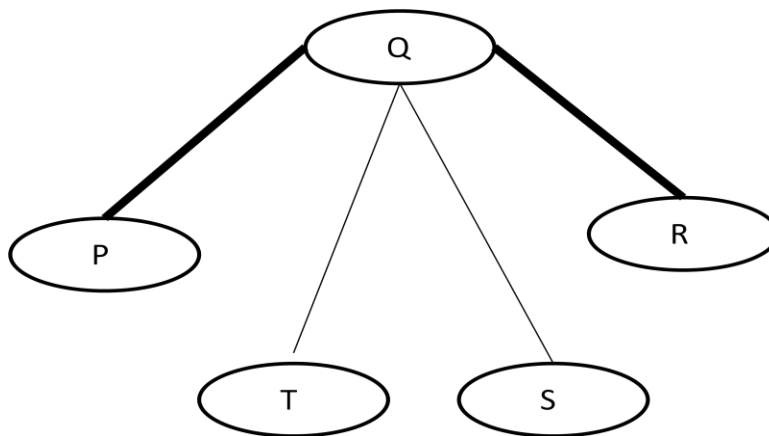


Figure 5.3 Optimized Weighted Story Graph

In an Agile environment, to accommodate any new change, team has to concentrate on many types of testing so as to satisfy the customer and by making optimized use of resources. Regression testing is that testing which consume most of the resources. Running complete test suite during or after the sprint is tedious job. This RTS technique helps in selecting only those test cases which are most relevant to the specific binode. Also, further selection can be made if time is less. In this case, only existing edge paths can be taken into account for finding the bug. For simplicity reason, considering only QR and QP paths of the story graph. For example, each user story has 500 test cases. Then, regression test suite would contain 2500 test cases in all. Running these test cases in short span of one or two day seems to be impossible task when any change is encountered. That's why RTS way of temporarily selection is feasible approach to opt for.

If QR and QP binodes are to be tested, then corresponding paths to be tested are QPR and QP. After QP is fully tested, then for path QPR only half of the job is to be left as QPR has two links namely QP and PR. Left link is PR only. PR edge weight is 3 story point. It means after facing the complexity level of 3 for P user story, R user story can be started as R is dependent on P user story output.

Similarly, for nonexisting edges, QT and QS binodes have path QPT and QPTS. It is clear that QPT is subset of path QPTS. If QPT is running fine then one third of the work is done and left part is only link TS.

#### **5.4 RTS TOOL**

The proposed RTS technique has been implemented in Microsoft Excel. The snapshots for the proposed technique have been shown in Figure 5.4-5.8.

#### **5.5 CONCLUSION**

This chapter discusses a RTS technique for selecting optimized user stories so as to utilize resources to its fullest. It makes use of two important parameters namely average path length and average path value. Optimized results are obtained by considering APL and APV values. Validity of the technique is done by using the velocity metric which is measured for a sprint. Velocity is a metric that predicts how much work

an Agile software development team can successfully complete within a two-week sprint. A work is said to be successfully completed only when it is tested and executed satisfactorily. With this technique, optimized selection of user story is done in less time resulting in more productivity in terms of satisfaction and quality. The next chapter discusses test case prioritization techniques for doing effective test case management.

	A	B	C	D	E	F	G	H	I	J	K	L
1	P	Q	R	S	T				Enter User Stories			
2												
3	PQ	QR	RS	ST	TP	PR						
4	3	1	3	7	5	3			Enter Existing Edges			
5												
6												
7												
8	<b>Binode</b>	<b>Direct/Indirect Path</b>	<b>d</b>	<b>m</b>	<b>t</b>	<b>APV</b>	<b>APL</b>					
9	PQ	PQ	1	3	3	3	3		Direct & Indirect Path Calculator for Existing Edges			
10		PRQ	2	1	4	0.5	2					
11		PTSRQ	4	1	16	0.25	4					
12	QR	QR	1	1	1	1	1		Calculate distance between nodes, minimum weight and total weight in path for Existing Edges			
13		QPR	2	3	6	1.5	3					
14		QPTSR	4	3	18	0.75	4.5					
15	RS	RS	1	3	3	3	3					
16		RPTS	3	3	15	1	5					
17		ROPTS	4	1	16	0.25	4					

Figure 5.4 Snapshot 1

	A	B	C	D	E	F	G	H	I	J	K
27											
28	QT	QS	RT	SP					Enter Non Existing Edges		
29											
30	<b>Binode</b>	<b>Indirect Path</b>	<b>d</b>	<b>m</b>	<b>t</b>	<b>APV</b>	<b>APL</b>		Indirect Path Calculator for nonExisting Edges		
31	QT										
32		QPT	2	3	8	1.5	4		Calculate distance between nodes, minimum weight and total weight in path for non Existing Edges		
33		QRPT	3	1	9	0.33333	3				
34		QRST	3	1	11	0.33333	3.66667				
35		QPRST	4	3	16	0.75	4				
36	QS										
37		QRS	2	1	4	0.5	2				
38		QPRS	3	3	9	1	3				
39		QPTS	3	3	15	1	5				
40		QRPTS	4	1	16	0.25	4				
41	RT										
42		RST	2	3	10	1.5	5				
43		RPT	2	3	8	1.5	4				

Figure 5.5 Snapshot 2

	A	B	C	D	E	F	G	H	I	J	K
49											
50	Q										
51											
52											
53											
54	QT	QS	PQ	QR							
55											
56											
57											
58	<b>Binode</b>	<b>Path</b>	<b>d</b>	<b>m</b>	<b>t</b>	<b>APV</b>	<b>APL</b>				
59	QT										
60		QPT	2	3	8	1.5	4				
61		QPRST	4	3	16	0.75	4				
62	QS										
63		QPRS	3	3	9	1	3				
64		QPTS	3	3	15	1	5				

Figure 5.6 Snapshot 3

	A	B	C	D	E	F	G	H	I	J	K	L
72												
73	<b>Binode</b>	<b>Path</b>	<b>d</b>	<b>m</b>	<b>t</b>	<b>APV</b>	<b>APL</b>					
74	QT											
75		QPT	2	3	8	1.5	4					
76	QS											
77		QPTS	3	3	15	1	5					
78	PQ											
79		PQ	1	3	3	3	3					
80	QR											
81		QPR	2	3	6	1.5	3					
82												
83												
84	T											
85	S											
86												
87												

Figure 5.7 Snapshot 4

	A	B	C	D	E	F	G	H	I	J	K
81		QPR	2	3	6	1.5	3				
82											
83											
84	T										
85	S										
86											
87											
88	P										
89	R										
90											
91											
92											
93											
94											
95											
96											
97											

Changed user story has Potential to affect user story

Changed user Story will Affect user story

Figure 5.8 Snapshot 5

## **CHAPTER VI**

### **AGILE REGRESSION TEST CASE PRIORITIZATION**

#### **6.1 INTRODUCTION**

The Agile project management is an approach used in the software industry so as to attract more customers. In this approach, processes and principles are dynamic in nature. Here, team members and customers are the major parties involved during the business deal. One of the representatives of the customer is always present on the development site so as to give the instant feedback and for any future improvement in the sprint. In ASD, work to be delivered to customer is frequent and response is also frequent from the customer side. This response may comprise of improvement in the existing system, new requirement, new work style, scalability of the existing system etc. At the same time, customer may furnish new requirement that may disturb the original functioning of the existing system. These later introduced changes may have several unnoticed effects in the working system. These effects must be controlled in a planned manner by the team members so as to deliver the quality deliverable to the customer on time. Controlling of the existing system is the first priority as per the definition of the regression testing which says that original modules should not regress by introduction of new functionality/modules/user stories. Although, unit testing and acceptance testing is an ongoing activity during the sprint as a part of regression testing, still some bugs may go unnoticed due to lack of risk measure of any new requirement disclosed by the customer.

In this chapter, an approach is discussed which is based on complete user story matrix that helps in evaluating the overall design measure of the user story. This chapter discloses prioritization of user stories on the basis of risk factor of the different user stories of the system. Further to add on, this predicts the risk factor of any user story that may have substantial effect on the existing stories. This measurement is done by considering the story point of the user story and complete user story matrix. These results are used to provide answer to question like if one user story is changing what other user stories of the system should be examined. In other words, what other modules have

potential to change when any new module is introduced by customer after the sprint or closure of the project. Also, this approach can be used for doing testing of the most suitable module/user story of the system.

Regression testing is a way to do test case management in an efficient way. This management is implemented by first performing user story prioritization and then performing test case prioritization. TCP is implemented by proposing following three techniques.

- Proposed Risk based TCP [11] technique which is based on user story graph,
- Proposed Pattern based TCP [9] technique which is based on object oriented dependencies, and
- Proposed Linguistic based TCP [8] technique which is based on Linguistic parameters such as nouns and verbs present in the user story.

## **6.2 PROPOSED RISK BASED TCP TECHNIQUE**

The Agile designs are dynamic in nature as these designs emerge over time. Design is dependent on the requirements framed by the client at the time of conception of an idea. As requirements keep on adding, design may evolve. The requirements in the Agile environment are known as *epic* (overview of the major task in 2-3 lines). Further, epic is splitted into small user stories. The next step following this splitting is the estimation of the story points in context with the complexity of the user story. The block diagram for the above process is represented in the Figure 6.1. In this Figure, story splitter is one component that helps in splitting the epic into number of stories which are further processed or executed in the following sprints. Estimation for user stories may be done in many ways. One of the ways is planning poker in which many experts sit together and depending upon their experience they assign some story point to the user story.

The proposed approach for prioritizing the user stories is based on the story points of the user story and depending upon these story points, a risk measure matrix has been proposed that is the outcome from complete user story matrix's values. The disclosure of

the proposed work comprises user story graph, complete user story matrix, design matrix and risk measure matrix. These components are shown below in the Figure 6.2.

### 6.2.1. User Story Graph

First and foremost component is user story graph. User story graph shows connections among splitted user stories and their dependence in terms of user story point. A user story point is a measure to estimate the effort involved in implementing a user story. Table 6.1 represents the user story point estimation for the splitted stories. In this Table, four user stories are estimated to have story point 1, 8, 3 and 2 respectively. These estimations are based on planning poker game. Its primary downside has been that all participants/experts need to be sitting in the same room with a physical deck of cards in their hands. This story point estimator is an abstract value as it is based on the baseline that is chosen by the experts. This story point value is further used in drawing user story graph. The links between user stories are represented by some weights. This weight is sum of individual story point of the connected story. For example, edge PQ has weight 9, which is sum of individual story points, as P story has connection with Q and S. P has story point 1 and Q has story point 8. In total, their combined effect is 9. Similarly, other weights are calculated.

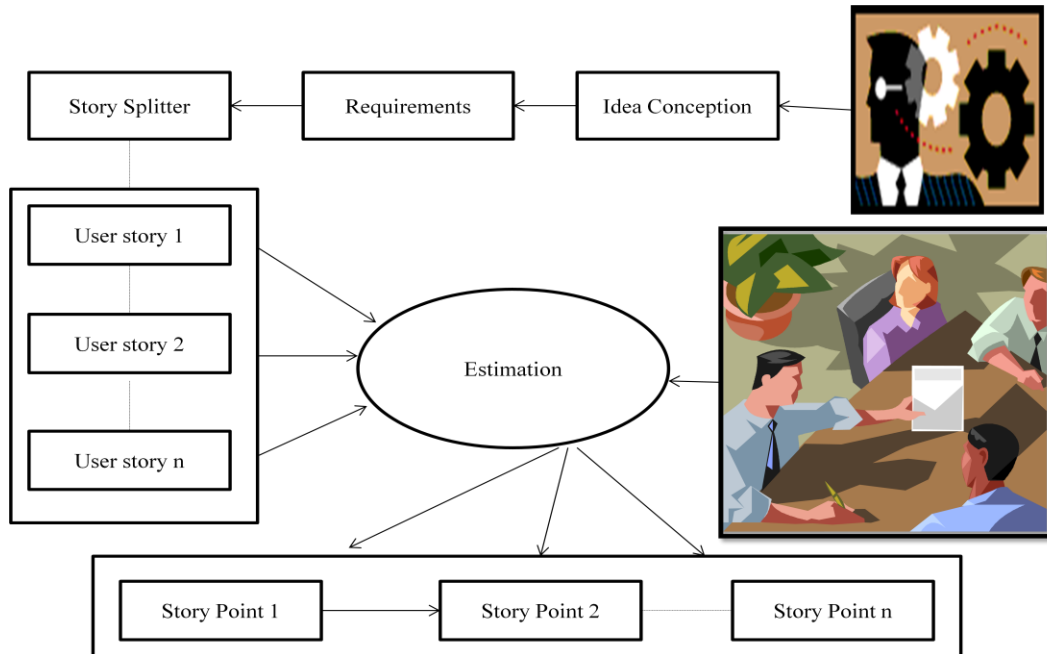


Figure 6.1 Agile Environment



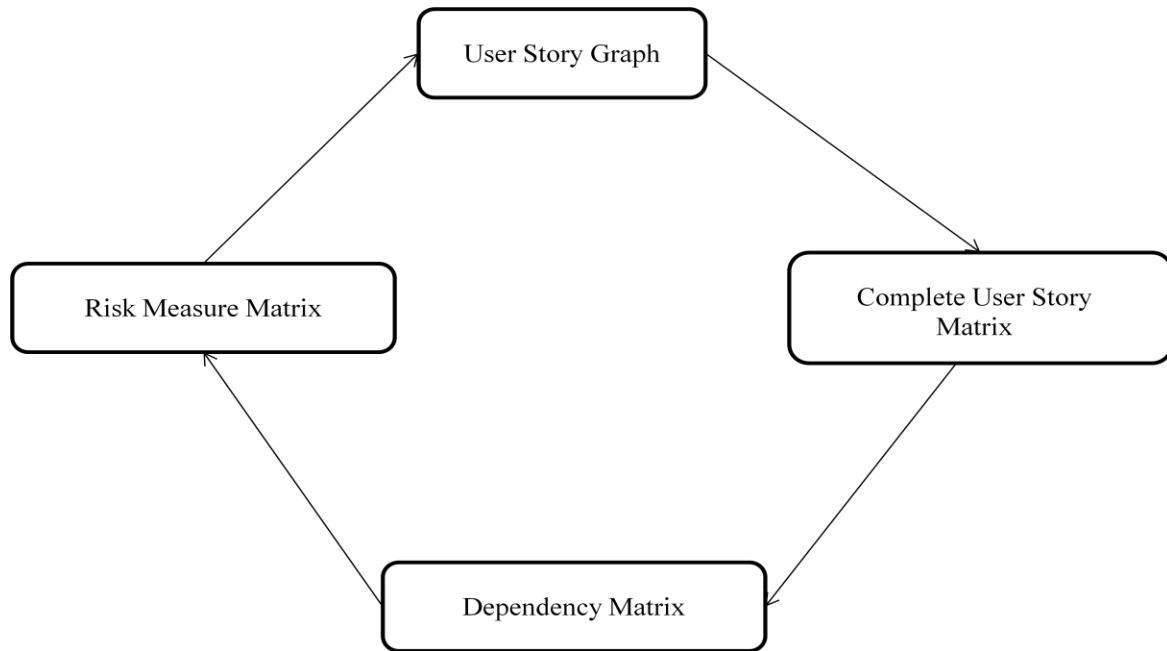


Figure 6.2 Risk Based Model

Table 6.1 Story Effort

S.No.	Story Name	Story Point
1.	P	1
2.	Q	8
3.	R	3
4.	S	2

On this basis, a weighted graph is drawn as shown in Figure 6.3. Links of the graph are shown to be undirected. Thus, P and Q user stories are dependent on each other. It means P and Q are required for moving to next step. That's why combined effort is calculated. The first level dependence matrix for Figure 6.3 is shown in Table 6.2. Diagonal entries in the first level dependence matrix are shown to be 0 as no link exists between self-user stories. Rest of the non-zero entries is as per the user story graph. For example, in Figure 6.3, link PR does not exist, so, in Table 6.2, PR and RP cell entry is filled with Not Applicable (NA).

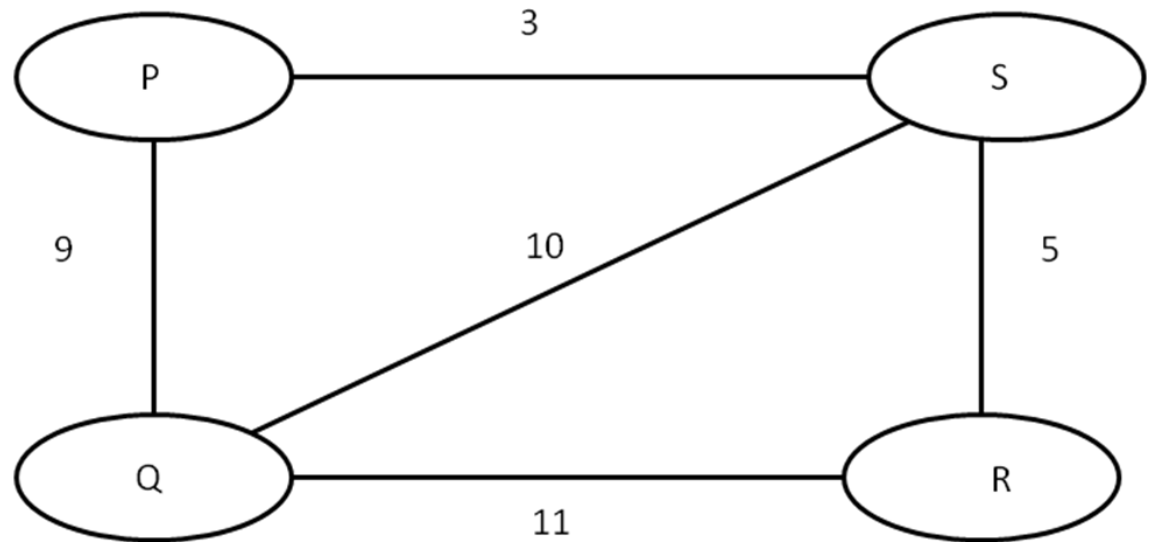


Figure 6.3 User Story Graph

Table 6.2 First Level Dependence Matrix

	<b>P</b>	<b>Q</b>	<b>R</b>	<b>S</b>
<b>P</b>	0	9	NA	3
<b>Q</b>	9	0	11	10
<b>R</b>	NA	11	0	5
<b>S</b>	3	10	5	0

### 6.2.2 Complete User Story Matrix

The complete user story matrix is the matrix showing effective effort for user stories using indirect and direct links weights in user story graph. For example, if client has introduced change in user story P of user story graph then definitely user story S's effort would change as there is direct link between these two user stories. On the other hand, rest of the user stories effort will change as indirect paths exist from source P to destination S. D represents direct link and I represents indirect link. Considering the same example, P to S gives one direct link and two indirect links. Their combined weights is

proposed to be taken as sum of three factors namely story points, volatility rate and implementation dependency. These factors are discussed below:

- Story Point Weight** – This weight is based on the edge weight of the nodes in the user story graph. During estimation process, story point is estimated for every user story using Planning Poker technique or any other technique. In this Table 6.3, from source to destination, indirect link story point weight has been calculated by multiplying intermediate edges weight. For example, from P to S, indirect link is PQS. In this path, PQ edge weight is 9 and QS edge weight is 10. Accordingly, story point weight for indirect link PQS is 90. Similarly, this weight may be calculated for other indirect links between same source and destination nodes.
- Volatility Rate** – Since in Agile changes are always accepted, there may be changes in the existing user stories. Therefore, volatility rate of user stories must be considered as a risk factor. Consequently, there may be change of an indirect link in the user story graph over time. This measure is connected with a chance of occurrence of any new node in the existing user story graph or a chance of occurrence of any change in the existing node. This measure is based on the input given by Product Owner, Developer, Tester or Market Analyst of the team. Each of the team members may be provided with a questionnaire. In the questionnaire, questions are designed to measure volatility rate. Types of questions (See Table 6.3) in the questionnaire may be related to:

Table 6.3 “Questions-Volatility Rate”

1.	How many users can be scaled up in future for the usage of specific user story?
2.	How many more internal parameters may be added to accommodate that change?
3.	On what scale, competitors are upgrading to new versions?
4.	Is this long term goal or short term goal?
5.	Whether real time data is involved in the execution or not?

Based on answers of questions from different team members, a scaling factor may be assigned for every subsequent link between source and destination, whether direct or indirect. For the sample case, a scaling of 1-5 has been considered where 1 is more volatile and 5 means less volatile.

- **Implementation Dependency** – Implementation dependency is also a measure which is connected with technical dependency of different user stories as per the developer or market analyst. In this case, questions (See Table 6.4) may be related to:

Table 6.4 Questions-Implementation Dependency

1.	How frequently any technology is changing which is used for developing user story?
2.	How fastly quality standards are changing?
3.	How frequently team members are switching an organization?
4.	How frequently competitor is launching product in the market?

Based on answers of questions from different developers and market analyst, a scaling may be assigned for every subsequent link between source and destination, whether direct or indirect. For the sample case, a scaling of 1-5 has been considered where 1 is more dependent and 5 means less dependent.

After calculating these three factors, a total weight is calculated by adding these three factors as shown in Table 6.5. Since these events are not mutually exclusive, the following equation is used to track the cumulative effect C(s) when there is any change in the existing user story S.

$$C(s)=P(X) + P(Y) + P(Z) - P(X).P(Y) - P(Y).P(Z) - P(X).P(Z) + P(X).P(Y).P(Z)$$

(1) [79]

Where P(X) is Weight of direct path

P(Y) is Weight of indirect path 1

P(Z) is Weight of indirect path 2

Using equation 1, cumulative effect,  $C(s)$ , of changing user story  $S$  is 378481. Similarly, other values may be calculated using equation 1. The method for finding paths from any source node to destination in user story graph is shown in Figure 6.4.

Table 6.5 Effort Data

Links	Type of Link	Story Point Weight (SPW)	Volatility Rate (VR)	Implementation Dependency (ID)	Total Weight=SPW+VR+ID
P-S	Direct	3	4	2	9
P-Q-S	Indirect	$9*10=90$	1	5	96
P-Q-R-S	Indirect	$9*11*5=495$	3	1	499

In this Figure, P is the source node and S is the destination node. The direct and indirect paths between P and S may be calculated by calculating first order dependence matrix which is given by Myers [69] and adjacency matrix which may be found by looking at the connections between nodes of the user story graph. For example, in the adjacency list P is connected to Q and S. Similarly, Q is connected with P, R, and S node of the user story graph. For finding direct path from P to S, adjacency list is analyzed deeply. Source node P has two more connecting nodes such as Q and S. The direct path exists in the user story graph from P to S as S is the destination node in the first row of the adjacency list. Further, P is connected with Q and Q is connected with P, R and S. So, indirect path can be found by passing through vertex Q. Hence, first indirect path is PQS. Furthermore, Q is connected with R and R is connected with S. Thus, next indirect path in the user story graph is PQRS.

In this user story graph, self loops are not considered. More specifically, first indirect link has three nodes or two links. The second indirect link has four nodes or three links. These links are shown in Figure 6.4 in Links column name.

Breadth First Search BASED IMPLEMENTATION						
Enter name of modules	P	Q	R	S		
First order dependence matrix						
		P	Q	R	S	
	P	0	9	NA	3	
	Q	9	0	11	10	
	R	NA	11	0	5	
	S	3	10	5	0	
Source	P					
Destination	S					
Direct Path		P	Q	R	S	Links
	P	0	9	NA	3	1
	Q	9	0	11	10	
	R	NA	11	0	5	
	S	3	10	5	0	
Indirect path 1		P	Q	R	S	Links
	P	0	9	NA	3	1
	Q	9	0	11	10	2
	R	NA	11	0	5	
	S	3	10	5	0	
Indirect Path 2		P	Q	R	S	Links
	P	0	9	NA	3	1
	Q	9	0	11	10	2
	R	NA	11	0	5	3
	S	3	10	5	0	
Adjacency List	P	Q	S			
	Q	P	R	S		
	R	Q	S			
	S	P	Q	R		
Direct path	P	Q	S			1
PS	Q	P	R	S		
	R	Q	S			
	S	P	Q	R		
Indirect path 1	P	Q	S			1
PQS	Q	P	R	S		2
	R	Q	S			
	S	P	Q	R		
Indirect Path 2	P	Q	S			1
PQRS	Q	P	R	S		2
	R	Q	S			3
	S	P	Q	R		

Figure 6.4 BFS Based Indirect Path Implementation

Depending upon number of indirect paths, equation 1 can be extended accordingly. Indirect paths in an undirected graph may be determined by using breadth first search implementation which is based on adjacency list. For the discussed case, complete user story matrix is shown in Table 6.6. Sixth Column of the matrix is sum of the column entry in individual row of the matrix. This column is known as dependency value. This value is determination of the risk involved for the user story. Maximum value in the Table is for user story S and minimum value is for user story Q. Maximum value in dependency matrix is representation of high risk story. Therefore, planning and estimation efforts are more for user story S as compared to Q, R, and S user story. That's why, focus is more on S user story by team members so that risk can be tackled easily.

Table 6.6 Complete User Story Matrix

	<b>P</b>	<b>Q</b>	<b>R</b>	<b>S</b>	<b>Dependency Value=P+Q+R+S</b>
<b>P</b>	0	64329	-1817	378481	440993
<b>Q</b>	64329	0	151801	23791	239921
<b>R</b>	-1817	151801	0	347301	497285
<b>S</b>	378481	23791	347301	0	749573

### 6.3 OUTCOMES

The proposed model shows that user story S is high risk story. The cumulative results shown in Table 6.7 predicts that P or any other story is directly or indirectly linked to rest of the stories and change in respect of other story may affect P or any other corresponding user story. Nowhere, in the prior art it is accomplished that story point, volatility rate and implementation dependency are linked with the risk factor of any user story. So, shown results are useful for prioritization of existing user stories after calculating complete user story matrix. Risk matrix reveals the order of user story prioritization as S, R, P and Q. Q user story is least risky story.

Table 6.7 Risk Measure Matrix

<b>STORY</b>	<b>DEPENDENCY VALUE</b>	<b>DEPENDENCY-RISK</b>
<b>P</b>	440993	
<b>Q</b>	239921	Least Dependent- Less Risk
<b>R</b>	497285	
<b>S</b>	749573	Most Dependent-More Risk

After finding the most risky user story, testing efforts may be calculated from the beginning of the sprint. A high risky story of the sprint is always center of attraction. Every reviewer, of that user story, has to focus more for accomplishing bug free deliverable as compared to less risky story. It does not mean that less risky story may be ignored. Both have equal importance. Only difference is high risky story demands more reviews as compared with less risky story. Also a high risky story demands experienced professional as compared with a less risky story. In light of this, a reviewer may concentrate on following points before releasing deliverable (software) to customer from her end:

- (a) Minimum usage and maximum usage of deliverable is checked.
- (b) Optimum resources needed for running software are verified.
- (c) Ready story of user story is mapped with the deliverable.
- (d) Previous deliverable mistakes are not repeated if customer is same.
- (e) Standard set by customers are fulfilled whether she is pilot or non pilot customer.
- (f) Criteria such as quality and timely delivery of outcome is fulfilled.
- (g) Manual has to be released along with the deliverable for handling exceptional conditions.



After prioritizing the user stories of the SBL, test cases are prioritized based on the respective risk factor of the user stories. More specifically, high risky user story test cases are executed prior to the less risky user story of the SBL. Accordingly, TCP is executed based on the risk measure of the respective user story.

#### **6.4 PROPOSED PATTERN BASED TCP TECHNIQUE**

Emerging technologies and developments are explored in software development domain by different software professionals so as to deliver the user stories of customers. Temporal changes proposed by customers for her project may give birth to new technologies and developments. Also, in some cases market standards forcefully demands change in existing principles and policies. There may be two problems along with the introduced change so as to fulfill specific needs. First and foremost problem is acceptance among team members for adapting to new technology, new environment or new solution for existing problem. Secondly, risk involved while adapting to new change is cumbersome. An exact risk measurement for user story may be implemented using first proposed technique discussed in section 6.2 which is based on user story graph. After identifying most risky user story from the SBL, test cases may be managed using any of the proposed regression TCP techniques. This section focuses on pattern based TCP technique.

In [72], a comparative analysis was performed among different patterns of object oriented software to find out which design pattern is more error prone. For this purpose, researchers had taken five open source systems and among those, Java files, which contain bugs, were observed for pattern occurrence. Four design patterns were considered for evaluation; *Singleton*, *Factory*, *Composite* and *Adapter*. Different hypotheses were established and statistical tests were performed. It had been evaluated that Adapter pattern is more error prone as compared to other patterns. One other reason of being more error prone is that it is excessively used in all the projects especially Hibernate Project.

When a pattern will be excessively used then its tendency of producing errors would be less, when selection of pattern is accurate, or more, when pattern is not executed correctly on the desired application. This analysis may also vary based on the type of project on which it is implemented. In object oriented applications, there is a high

chance of extensibility or scalability. Therefore, errors might rise to an unacceptable level. That's why in this section, a technique to evaluate object oriented connections in user stories is proposed and further, on the basis of the object oriented connections, TCP has been implemented by following rules of the proposed test pattern which are linked to the internal design of the object oriented components.

#### **6.4.1 Proposed "Test" Pattern**

In this section, focus is on more severe module user stories after introducing minute change in the original user story. The identification for the most severe module having strongest intensity is done using first order dependence matrix concept that matrix was introduced by Myers.

The test pattern for the common problem of frequent change in an Agile context is the reason for its existence. As customer or one of his representatives is always at the development site for supplying instant feedback, so, by this specific activity his contribution is more demanding for considering him in many other related tasks that are significant for user story enrichment.

Customer may or may not be technical sound but his instant feedback is useful for detecting and removing defects at an early stages (at the time of unit testing, integration testing, incremental testing or regression testing during sprint) of the sprint. Customer cannot frame good test cases for defect detection but skeleton of user story is specified by him/her. The test pattern is a kind of pattern which is simple and effective to use for customer satisfaction. The *Template* for test pattern is shown in Table 6.8.

The proposed technique for prioritizing the test cases is based on following four components (See Figure 6.5). The components are:

1. **First Order Dependence Matrix Calculator:** In this component, call graph is created using first order dependence matrix (introduced by Myers) for the user story. This call graph is applicable for one user story at a time and consists of linkages of nodes for each task of the user story. Different tasks (Class) of the user story are connected by coupling mechanism and one class is tightly connected with its data and methods by cohesion principle.

2. **Dependency Finder:** This component finds the dependencies among the various tasks by considering the output from the first component. More specifically, directly dependent and indirectly dependent class are identified for the assigned user story. Finally, objects of the changing task in relation with the dependency are identified.
3. **Test Case Generator based on Dependency diagram and Relationship diagram:** This component generates the test cases for the identified classes and objects based on the relationship diagram between objects.
4. **Test Case Prioritizer:** This component prioritizes the generated test cases.

The first component has been explained here for the reference as output of first order dependence matrix is used as input for the proposed TCP technique. The rest of the components of this model are explained in detail by considering one case study which is discussed in the next subsection. This case study is related with one user story having multiple tasks and user of the user story introduces new task in the existing user story. The next subsection focuses on said issue of dependencies among existing tasks for the purpose of prioritizing test cases for the said user story.

Table 6.8 Test Pattern

<b>Name</b>	Test Pattern.
<b>Intent</b>	To Prioritize Test Cases of Object oriented Application.
<b>Motivation</b>	To save time and to have good quality for the user story.
<b>Apply</b>	within the sprint (when there is a change in the user story).
<b>Participants</b>	story point, story effort index and customer's priority for specific user story.
<b>Collaborations</b>	through daily meeting for short duration (online in case of distributed teams).
<b>Consequences</b>	less defects for user story in small sprint.

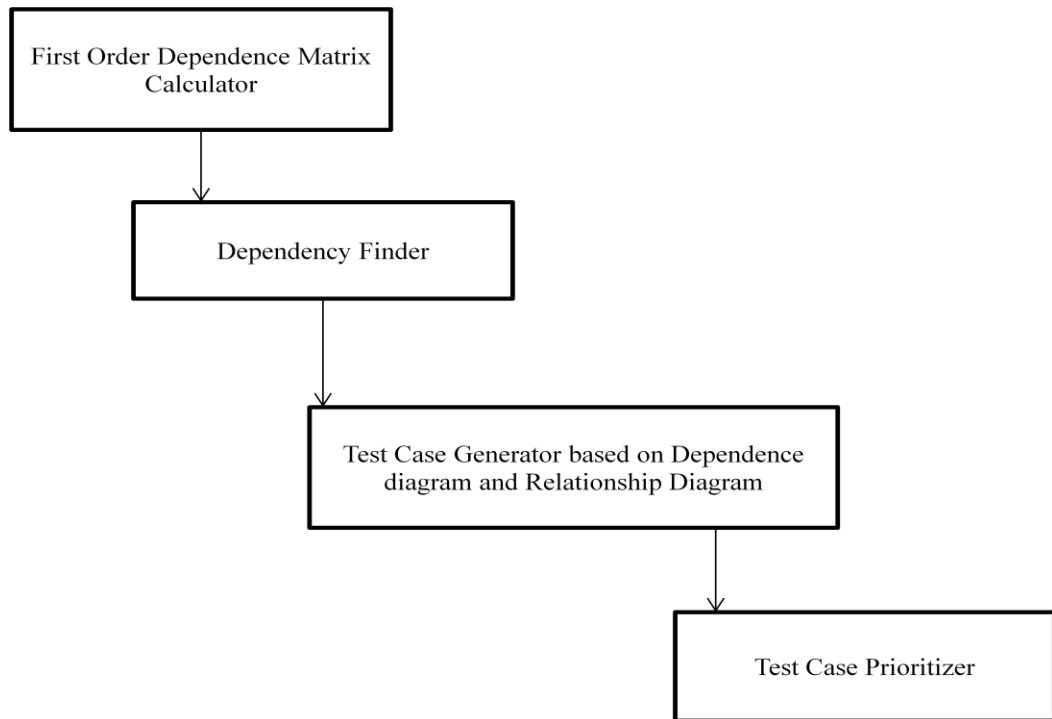


Figure 6.5 Working Model for pattern based TCP

#### 6.4.2 Case Study

Considering, for example, a user story, “*getting the patent to be granted by government of India to have hold on all the rights in the specified territory by any patentee*” (S). In this user story, primary parties are patent office and applicant. On the other hand, secondary party is public and it is not always active. Tasks for the user story S are:

- Filing the application with complete specification in the patent office by applicant for his invention. (c1)
- Invention is published online in the patent office’s official journal (weekly issue) after 18 months. (c2)
- Request for examination for the said invention within 48 months. (c3)
- First examination report by patent office to applicant. (c4)
- Office Action/Office Action Reporting is generated by patent office/applicant until satisfaction level is reached for putting an application in order for grant within 12 months. (c5)

- Pre grant opposition done by opponents/public to patent office. (c6)
- Response sent by applicant to patent office. (c7)
- Grant of patent for 20 years by patent office. (c8)

So, three classes namely patent office, patentee and public are interconnected with each other with respect to the single user story S by applying first component (See Figure 6.6). Thus, it is clear that classes are dependent on each other especially primary one. Secondary, one is active only in case of pre grant opposition after publication.

Suppose there is a change (c9) in the user story which says that *opposition from public is allowed after grant of patent also or after c8*. Now, situation is little bit complex as public class is becoming active.

Adding c9 will introduce some ambiguous linking between primary class's methods. In this case, the proposed test pattern may be applied, where direct as well as indirect class dependency exists and need is to prioritize the test cases so as to gain quality.

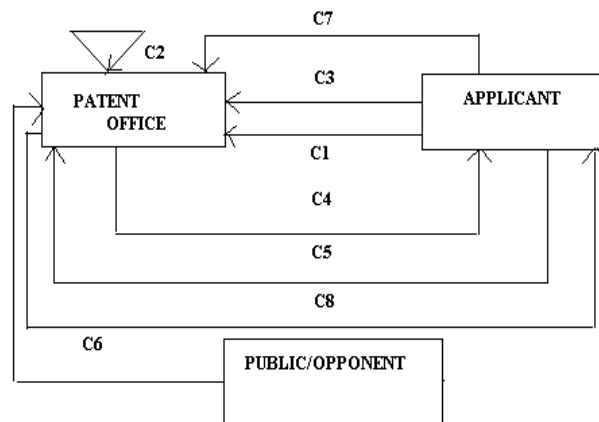


Figure. 6.6 Patent Grant Procedure

After applying the first component, next component is Dependency Finder. More specifically, in this case, objects of classes are not directly dependent on each other rather these objects may be indirectly dependent based on c9. Steps for component Dependency Finder are:

- Find the directly dependent class and indirectly dependent class from the Figure 6.6.
- Identify the objects of the changing task's with respect to the dependent classes (directly and indirectly).
- Find the relationship of objects of all the related classes in specific methods.

After applying steps of Dependency Finder, relationship of classes for the specified user story is shown in Figure 6.7. In this Figure, dotted lines represent indirect relationship and solid lines represent direct relationship between classes for the user story S. Also, for every class in the relationship diagram, object list and method list is there. Complexity of the change is dependent on number of classes present in the user story. But in Agile culture, less is more concepts is used, so, small user story would have small number of classes in one sprint and relationship would be strong.

Time of delivery of user story to customer is very frequent in nature. So, there is a need to find defects at an early stage of development. The subsequent components are linked to the prioritization of the test suite for S.

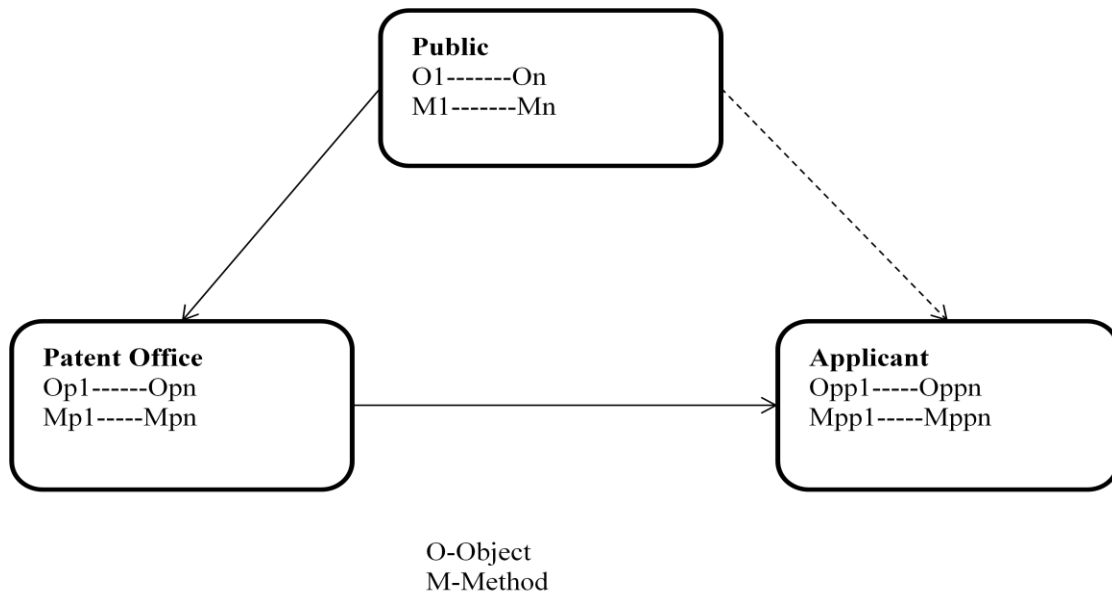


Figure. 6.7 Relationship Diagram

Steps for prioritization are:

- Create the dependence diagram; it is a smart way for representation of class linkages in a user story, (see Table 6.9) and is to be displayed along with the story board (see Table 6.10).
- When a significant change is encountered in the user story, observe the dependence diagram and relationship diagram as discussed earlier to identify the intensive classes and objects (with relevant data members and methods) of the user story. (Third Component)
- Sort the records of the test suite on the basis of dependent and indirectly-dependent objects by considering relationship diagram and dependence diagram (Checking non null entries in the test suite Table).
- Apply second sort now on the basis of maximum number of non-null entries for existing objects in the classes for one particular user story.
- Frame new test suite Table after doing prioritization (Fourth Component).

Table 6.9 Dependence Diagram

<b>DEPENDENCE DIAGRAM FOR S1</b>		
<b>CLASS #</b>	<b><i>DIRECTLY DEPENDENT</i></b>	<b><i>INDIRECTLY DEPENDENT</i></b>
1	4, 5	2, 3
2	....	.....
.		
N	.....	.....

### 6.4.3 Results & Analysis

The purpose of this analysis is to prove that quality component is optimal one as compared with non-prioritization scheme of regression testing. Generated test suite (TS) is given in Table 6.11 and after considering dependence and relationship diagram, new test suite is shown in Table 6.12 in which sorting is applied.

Table 6.10 Story Board

US#	Ana 5	Dev 2	Test 3	Rel	Done	DepDiag
S1	Task1	Task1	⇒	⇒	Task1	
	Task2	Task2			Task2	
	Task3				Task3	
	Task4				Task4	
	Task5				Task5	
S2	---	---	---			

**Ana**-Analysis

**Dev**-Development

**Test**-Testing

**Rel**-Release

**Done**-Done

**Dep Diag**-Dependence Diagram

**US**-User Story

Suppose O1-O4 are objects of public class, Op1-Op3 are objects of patent office and Opp1-Opp2 are objects of applicant class from Figure 6.7. Non zero entries in Table 6.11 for test cases 1,2,3,4,5 are 7,6,5,5,4. Out of all, find the nonzero entries for direct class objects. Directly dependent class is patent office and indirectly dependent class is applicant. Objects of directly dependent class are Op1-Op3. After doing sorting on this basis, series of test cases would be by excluding 5 number test case as all the entries are null for specified objects (Op1-Op3). The rest of four test cases are needed to be sorted.

Ascending order series for four test cases for non-zero entries is 1,2,2,3 for test cases number 4,1,2,3. Out of these results, maximum number is 3. So, priority would be high for 3 number test case. Out of tc 1, 2 which one will take priority depends upon non zero entries for indirectly dependent classes. Test case 2 has higher priority as compared with test case 1 as Op1 and Op2 has potential to affect Opp1 and Opp2. If there is no such difference then either of two test cases may be considered.



#### 6.4.4 Phases in Test Pattern

This test pattern is based upon the observer pattern under behavioral category. In it, test cases are prioritized for the purpose of saving time in an Agile context by considering dependence diagram of the story stories which is updated from time to time. Main phases of this pattern (See Figure 6.8) using components of Figure 6.5 are:

Table 6.11 Generated Test Suite

TC #	O1	O2	O3	O4	Op1	Op2	Op3	Opp1	Opp2
1	1	2	3	4	2	-	3	1	-
2	1	2	-	-	3	4	-	4	5
3	-	-	2	3	5	5	5	-	-
4	1	2	3	4	5	-	-	-	-
5	-	-	3	4	-	-	-	1	2

Table 6.12 New Test Suite

TC #	O1	O2	O3	O4	Op1	Op2	Op3	Opp1	Opp2
3	-	-	2	3	5	5	5	-	-
2	1	2	-	-	3	4	-	4	5
1	1	2	3	4	2	-	3	1	-
4	1	2	3	4	5	-	-	-	-

- (1) Call graph and test case generator-*Generator*
- (2) Dependent and indirectly dependent class identification-*Finder*
- (3) Prioritization of test cases using dependence diagram-*Prioritizer*

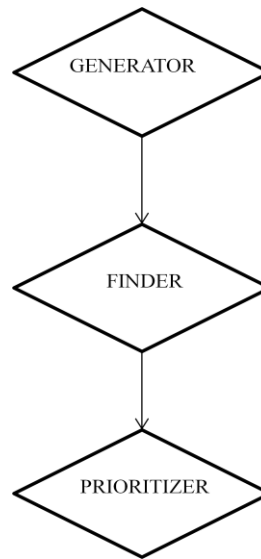


Figure 6.8 Test Pattern

## 6.5 LINGUISTIC BASED OBJECT ORIENTED TCP TECHNIQUE

TCP techniques are customer requirement-based techniques, coverage-based techniques, cost effective-based techniques, chronographic history-based techniques etc. Customer requirement-based techniques are methods to prioritize test cases based on requirement documents. Coverage-based techniques are methods to prioritize test cases based on coverage criteria, such as requirement coverage, total requirement coverage, additional requirement coverage and statement coverage. Cost effective-based techniques are methods to prioritize test cases based on costs, such as cost of analysis and cost of prioritization. Chronographic history-based techniques are methods to prioritize test cases based on test execution history. In this section, an alternate technique based on linguistic analysis of the user stories has been explained. This technique identifies noun and verb present in any user story. Further, identified nouns and verbs are used to find the sentence priority score of the user story by matching identified nouns and verbs with the nouns and verbs of the new user story. The block diagram for this technique is shown in Figure 6.9. In Figure 6.9, a technique for TCP has been shown using various components which are:

- **Story Prioritization:** This component prioritizes the existing user stories which are formal requirements framed by the team members and customer. The prioritization is based on the number of punctuations present in the user stories. For achieving this goal, the total number of punctuations is counted in each user

story and a user story with higher no. of count will be treated as user story which require more effort. These prioritized are then developed during the sprint.

- **New User Story:** In this component, user introduces new user story in the existing working system. Due to this addition in the existing system, there might be some effect on the existing user stories which are developed previously. That effect may be calculated using next component which is based on the linguistic parameters such as noun and verbs.
- **Noun & Verb Identifier:** This component identifies linguistic parameters such as noun and verb present in the existing user stories, ready stories of existing user stories, ready story of the new user story and the new story.
- **Matching User Stories:** This component performs the matching between noun and verb of the new user story with noun and verb of the existing user stories. Also, noun and verb of the ready story of new user story are matched with noun and verb of the ready story of the existing user stories. After matching the user stories, dependency among new user story and existing user stories is identified.
- **Sentence Priority Score:** On the basis of the identified dependencies among various user stories, sentence priority score of each test case is calculated by considering the number of nouns and verbs in the existing test suite of the existing user stories. The priority of a sentence over the other sentences is calculated using story point (SP) of the user story, customer priority (CP) and effect (Eu) where Eu is the product of no. of nouns and verbs present in any sentence.

$$SPS = \sum_{i=1}^n SP*CP*Eu \quad (2)$$

- **Prioritized Test Suite:** This component finally prioritizes the existing test suite by sorting the values of the sentence priority score. A score with higher value rates high priority to test case as compared with a score with lesser value.

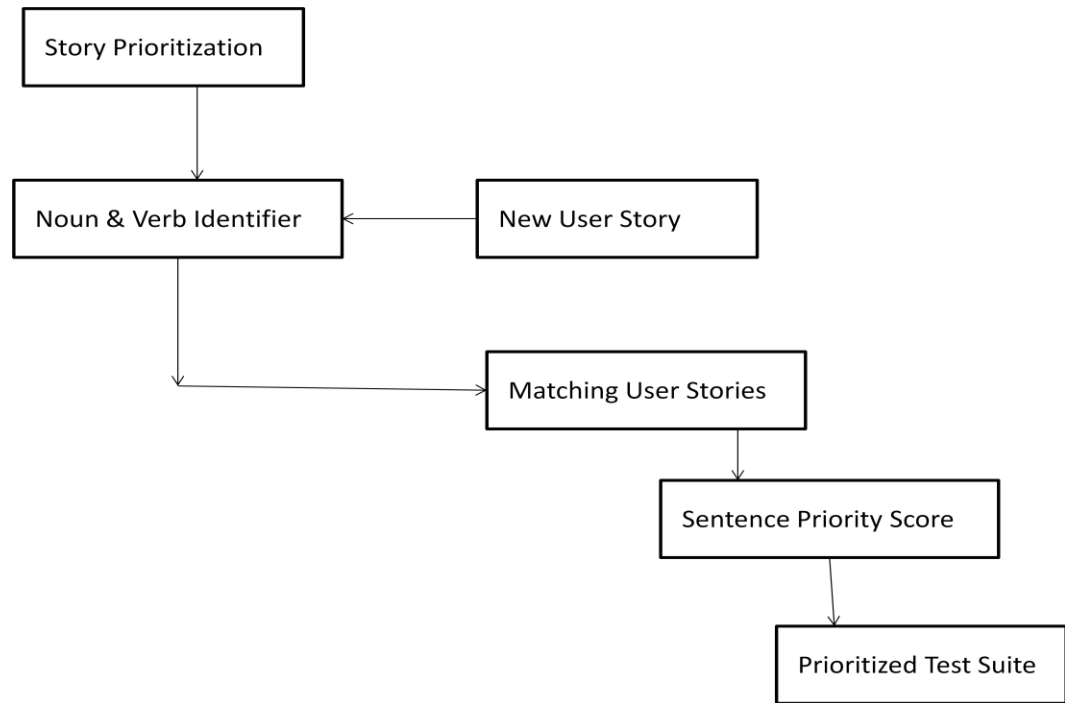


Figure 6.9 Linguistic TCP Technique

### 6.5.1 Story Prioritization

The proposed work includes two levels of prioritization, namely, story prioritization and test case prioritization. In this section, story prioritization has been described in detail. Story prioritization may be performed by a method in which punctuation marks are of great importance. For the purpose of attaining quality, punctuated user story has to be reviewed two times. One review may be performed by scrum master, if SCRUM methodology is followed, or team member and other review by Client Representative (CR). Freezing of user story is the important step, as further work is to be implemented, on this final user story. Next step is to prioritize user stories by using excel formulas. Len and Substitute formulae have been used to count the number of individual punctuation in the user story. For example, following combined formula has been used. (See Figure 6.10)

=LEN(B2)-LEN(SUBSTITUTE(B2,"",""))

Where, B2 represents any cell in the excel sheet, and 2<sup>nd</sup> argument of the Substitute function is the specific punctuation that is to be search in the B2 cell. Also,

sum function is used to calculate total number of occurrences of punctuations in a cell. Respective formula for the same is:

=SUM(C2:K2)

Where, C2 to K2 range is selected for performing addition of specific values. After applying these functions, sort was performed so as to get the most risky user story. (See Figure 6.11)

- More punctuations in a user story signifies more breaks,*
- More breaks in user story signifies more relations/linkages/coupling,*
- More relations in a user story signifies more dependencies,*
- More dependency in a user story signifies more risk,*
- More risky user story means more effort required.*

S.No.	User Story	Count (,)	Count (.)	Count (:)	Count (!)	Count (())	Count (;)	Count (?)	Count (-)	Count ("")	Total
1	As a searcher, I want to search patents of US, EP and India, on the basis of, bibliographic details, such as, application number, publication number, priority date, inventor, assignee, date of patent, so as, to perform searches comprising invalidity search, claim mapping, claim charting.	15	1	0	0	0	0	0	0	0	16
2	As a lawyer, I want to search legal status of patents, by mentioning publication number, application number, so as to handle infringement suit of different assignee of US, EP or India.	5	1	0	0	0	0	0	0	0	6
3	As an Inventor, I want to search, patents of software domain, comprising, agile software development, software engineering, software quality, software testing etc. so as, to perform "state of the art" search, for countries: US, EP and, India.	11	2	1	0	0	0	0	0	0	14
4	As a researcher, I want to search, patents of US, as, US is the software hub of software patents, so as, to improve upon, my findings.	7	1	0	0	0	0	0	0	0	8
5	As a drafter of a patent, I want to search, related patents, using software keywords, so as, to learn drafting skills, of software domain.	6	1	0	0	0	0	0	0	0	7
6	As a business analyst, I want to search, patents of Countries: US, EP or India, so as, to analyze market trend of patents, in software domain, for different assignees.	7	1	1	0	0	0	0	0	0	9
7	As a statistical professional, I want to, search revenue spend on patents, by assignee of countries: US, EP and India.	4	1	1	0	0	0	0	0	0	6

Figure 6.10 Total Punctuation

S.No.	User Story	Count(,)	Count(,)	Count(:)	Count(!)	Count()	Count(;)	Count(?)	Count(-)	Count(")	Total
2	As a lawyer, I want to search legal status of patents, by mentioning publication number, application number, so as to handle infringement suit of different assignee of US, EP or India.	5	1	0	0	0	0	0	0	0	6
7	As a statistical professional, I want to, search revenue spend on patents, by assignee of countries: US, EP and India.	4	1	1	0	0	0	0	0	0	6
5	As a drafter of a patent, I want to search, related patents, using software keywords, so as, to learn drafting skills, of software domain.	6	1	0	0	0	0	0	0	0	7
8	As an examiner, I want to search, Data Base, for finding prior arts of a given patent by mentioning, its bibliographic details, including application number and, priority date.	6	1	0	0	0	0	0	0	0	7
4	As a researcher, I want to search, patents of US, as, US is the software hub of software patents, so as, to improve upon, my findings.	7	1	0	0	0	0	0	0	0	8
6	As a business analyst, I want to search, patents of Countries: US, EP or India, so as, to analyze market trend of patents, in software domain, for different assignees.	7	1	1	0	0	0	0	0	0	9
9	As an analyzer, I want to, categorize patent of different classes, such as, United States classification (USC), Cooperative patent classification (CPC), International patent classification (IPC).	6	1	0	3	3	0	0	0	0	13
3	As an Inventor, I want to search, patents of software domain, comprising, agile software development, software engineering, software quality, software testing etc. so as, to perform "state of	11	2	1	0	0	0	0	0	0	14

Figure 6.11 Story Prioritization

Figure 6.11 shows the priority order for the user stories, in terms of risk, of a given project. Order is as follows:

$$2 < 7 < 5 < 8 < 4 < 6 < 9 < 3 < 1 < 10$$

User story 10 is the most risky story and user story 2 is the least risky story. In this way, story point marking for a user story becomes easy and also, effort estimation for completing any user story can be calculated. In addition, more risky story would have more number of confirming points.

The rest of the components are further explained in detail by considering one case study. The case study has been discussed in the next subsection.

### 6.5.2 Case Study

Considering the scenario in which online shopping store is managed by an administrator and following is the list of prioritized user stories which are existing in the system (See Table 6.13). In respect of Table 6.13, noun and verb are framed (See Table 6.14 & 6.15 for the same).

Table 6.13 User Requirements

<b>STORY #</b>	<b>User Story</b>
1	As an administrator, I want to add items in the women clothing section so as to have updated catalog
2	As an administrator, I want to add items in the kids clothing section so as to attract kids
3	As an administrator, I want to add items in the kitchenware section so as to attract hotel management fellows
4	As an administrator, I want to add items in the electronic section so as to attract the users.
5	As an administrator, I want to see frequent users of online shopping store so that I can send them new offers
6	As an administrator, I want to see daily sales of items so as to maintain the account details
7	As an administrator, I want to attract users so as to increase sales
8	As an administrator, I want to maintain stock so that no shortage of items could be there

Table 6.14 Noun Table

<b>STORY #</b>	<b>Noun 1</b>	<b>Noun 2</b>	<b>Noun 3</b>	<b>Noun 4</b>	<b>Noun 5</b>
1	Administrator	Items	Women	Clothing Section	Catalog
2	Administrator	Items	Kids	Clothing Section	
3	Administrator	Items	Kitchenware Section	Hotel Management Fellows	
4	Administrator	Items	Electronic Section	Users	
5	Administrator	Users	Online Shopping Store	New Offers	
6	Administrator	Sales	Items	Account Details	
7	Administrator	Users	Sales		
8	Administrator	Stock	Items		

Table 6.15 Verb Table

<b>STORY #</b>	<b>Verb 1</b>	<b>Verb 2</b>	<b>Verb 3</b>
1	Want	Add	Updated
2	Want	Add	Attract
3	Want	Add	Attract
4	Want	Add	Attract
5	Want	See	Send
6	Want	See	Maintain
7	Want	Attract	Increase
8	Want	Maintain	

Stories written in Table 6.13 are not clear in terms of preciseness means what needs to be done by the respective story. So, Slicing of the respective stories is needed. The 'definition of done' tells us when a feature is completed and is ready for release. On the other side, the 'definition of ready/Confirm Story/Story with Acceptance criteria' is the criteria a user story has to meet before it is ready to be passed to the team and taken

into a sprint. The definition of ready and the definition of done are two points in the sprint life cycle – one defines when a user story is ready to go in, and the other defines when a user story is ready to come out. Ready story for each of the existing requirements (1-8) are given below (See Table 6.16-6.23):

Table 6.16 Ready Story 1- Acceptance Criteria

A	Administrator can see the new catalog
B	Items are added in the exact Price range
C	Items are added in the exact Color range
D	Items are added in the exact Brand types
E	Items are added in the exact size range
F	Items are added in the exact Dress section (Formal, Casual, Ethnic, Fashionable.....)

Table 6.17 Ready Story 2- Acceptance Criteria

A	Administrator can see the new catalog
B	Items are added in the exact Price range
C	Items are added in the exact Color range
D	Items are added in the exact Brand types
E	Items are added in the exact age range
F	Items are added in the exact Dress section (Party Wear, Daily Wear, Night Wear .....)
G	Items are added in exact Gender section

Table 6.18 Ready Story 3- Acceptance Criteria

A	Administrator can see the new catalog
B	Items are added in the exact Price range
C	Items are added in the exact Color range
D	Items are added in the exact Brand range
E	Items are added in the exact Category (Serving, cooking, Storing, Cutlery ....)
F	Items are added in the exact size range

Table 6.19 Ready Story 4- Acceptance Criteria

A	Administrator can see the new catalog
B	Items are added in the exact Price range
C	Items are added in the exact size range
D	Items are added in the exact Brand range
E	Items are added in the exact category (Kitchen Related, Utility, Entertainment....)
F	Items are added in the exact color range

Table 6.20 Ready Story 5- Acceptance Criteria

A	Administrator can see the list of frequent users
B	Users are selected on the basis of criteria (Monthly, Biweekly, weekly, daily .....)
C	Users are provided with different kinds of offers
D	Offers are valid for selected category (Kids, Women, Men, Hotel Management Fellow....)
E	Frequent Users are of two types (Maximum amount range, Maximum number of items.....)

Table 6.21 Ready Story 6- Acceptance Criteria

A	Administrator can calculate the total daily sales of items
---	--



B	Administrator can bifurcate payment by cash/credit/net banking /Cash on delivery
C	Administrator can calculate the daily sales of items for any section
D	Administrator can calculate the daily total profit earned
E	Administrator can calculate the daily total profit earned for specific section
F	Account Reports are generated on different basis (weekly, monthly.....)

Table 6.22 Ready Story 7- Acceptance Criteria

A	Administrator has different promotional schemes for different sections of store
B	Administrator can emphasize on specific section for increasing sales
C	Policies are flexible enough
D	Employee/customer Referral is allowed and subsequent schemes are there.

Table 6.23 Ready Story 8- Acceptance Criteria

A	Administrator can order new items by checking the stock catalog
B	Administrator can find the total shortage of items in the online store
C	Administrator can do the payment online after giving order of shortage items
D	Administrator can generate report for less items in the stock.

In respect of all the ready stories of original stories (1-8), noun and verb Table are given below (See Table 6.24 & 6.25).

Table 6.24 Ready Story-Noun Table

STORY #	Noun 1	Noun 2	Noun 3	Noun 4	Noun 5	Noun 6	Noun 7
1	Price	Color	Brand	Size	Dress	New	
2	Price	Color	Brand	Age	Dress	New	Gender
3	Price	Color	Brand	Size	Category	New	
4	Price	Color	Brand	Size	Category	New	
5	List	Criteria	Types	Category			
6	Payment method	Section	Profit	Report	Basis		
7	Schemes	Section	Online Shopping Store	Employee Referral	User Referral		
8	New	Online Shopping Store	Shortage items	Online Payment	Order	Report	

Table 6.25 Ready Story-Verb Table

STORY #	Verb 1	Verb 2	Verb 3	Verb 4
1	See			
2	See			
3	See			
4	See			
5	Select	Provide	Valid	
6	Calculate	Bifurcate	Generate	
7	Emphasize	Allow		
8	Order	Find	Do	Generate

Suppose customer and business analysts have felt the need of updating the existing software application of online store through several meetings so as to face the

competitive world. As per them, new requirement is related to have satisfaction for all users. (See Table 6.26-6.28 for the same)

Table 6.26 New Requirement

STORY #	User Story
9	As an administrator, I want that users should be able to search items in the specific category so as to satisfy users

Table 6.27 New Requirement- Noun Table

STORY #	Noun 1	Noun 2	Noun 3	Noun 4
9	Administrator	Users	Items	Category

Table 6.28 New Requirement- Verb Table

STORY #	Verb 1	Verb 2	Verb 3
9	Want	Search	Satisfy

Similarly, ready story is created for new requirement and for corresponding noun and verb Table See Table 6.29 to 6.31.

Table 6.29 Ready Story 9- Acceptance Criteria

A(3/5)	User can search an item in the specific category (Women, kids, Kitchenware, electronic etc)
B(3/4)	User can search items as per the different criteria (brand, color, price, age, size, gender etc)
C(1/2)	Satisfaction level of user should be high by giving on time delivery of products
D(1/5)	User can login

**Note:** Numbers in bracket represent story point and customer priority of the story

Table 6.30 New Ready Story-Noun Table

STORY #	Noun 1	Noun 2	Noun 3	Noun 4	Noun 5	Noun 6	Noun 7
9	Women	Kid	Kitchenware	Electronic	Brand	Color	Price
	Noun 8	Noun 9	Noun 10	Noun 11	Noun 12	Noun 13	
	Age	Product	Satisfaction Level	Time	Size	login	

Table 6.31 New Ready Story-Verb Table

STORY #	Verb 1
9	Give

Now, next step is to map the new story noun Tables (6.27 and 6.30) with the Tables (6.14 and 6.24)

Table 6.32 Noun Dependent Story Table\_a (6.24 & 6.30)

New Requirement No.	Dependent Story #
9	1,2,3,4

Table 6.33 Noun Dependent Story Table\_b (6.14 & 6.27)

New Requirement No.	Dependent Story #
9	1,2,3,4,5,6,7,8

After comparing Table 6.32 and 6.33, we note that story 9 is dependent on 1, 2, 3 and 4 which are common entries in both the Tables.

Moving next, map the new story verb Tables (6.28 & 6.31) with the Tables (6.15 and 6.25) so as to identify the dependent user stories.

Table 6.34 Verb Dependent Story Table\_a (6.25 & 6.31)

New Requirement No.	Dependent Story #
9	No match

Table 6.35 Verb Dependent Story Table\_b (6.15 & 6.28)

New Requirement No.	Dependent Story #
9	1,2,3,4,5,6,7,8

After comparing Table 6.34 & 6.35, we note that story 9 is not dependent on other stories which are common entries in both the Tables. From this analysis, we come to a conclusion that story 9 is dependent on stories 1, 2, 3 and 4. In case, some data would have been generated from 6.34 & 6.35 Tables then we would have added that number to previous Tables' result (6.32 & 6.33). It means out of the total regression test suite of all the stories we have to select test cases of stories 1, 2, 3 & 4. This phenomenon is known as regression test case selection (RTS). Suppose test suite for each user story consists of 1000 test cases, thereby, after RTS, test suite would consist of 4000 test cases.

**Organization:** XYZ

**Application:** Online Store

**Assumption:** Normal Agile Testing (A)

If new user story need to be tested within a single sprint of two weeks then running these 4000 test cases is very tedious task. One option is to adopt automation which is a costly affair but simple and fast. Second option is to do prioritization of test cases so as to have fast and safe testing. Number of verbs for selected user stories (1, 2, 3 & 4) is 4 and number of nouns corresponding to 1, 2, 3 & 4 are 11, 11, 10 & 10. Effect

(Eu) of noun and verb on user story 1, 2 3, and 4 is 44, 44, 40 & 40. Equation for the same is:

$$\text{Effect on User Story (Eu)} = \text{Number of verbs for story (v)} * \text{no. of nouns for story (n)}$$

(3)

The test suite for all the selected user stories (1, 2, 3, 4 & 9) is given in Table 6.36-6.39. Notation used for test cases is N1.N2TCN3.N4:N1-Sprint #, N2-Story #, TC-Test Case, N3-Test Case Heading, N4-Test Case Subheading. Story number 1, 2, 5 & 6 are done in sprint 1 and story number 3, 4, 7 & 8 are done in sprint 2. Two parameters are considered for doing some planning. First is, story points are taken as per the Fibonacci series like 1, 2, 3, 5, 8..... . As the number keep on increasing, complexity would start to increase. Also, other parameter used is customer priority. Higher number represent higher priority over others user stories. Range for customer priority is 1, 2, 3, 4, 5.....so on.

Table 6.36 Test Cases Story 1

<b>Story #</b>		<b>1 (As an administrator, I want to add items in the women clothing section so as to have updated catalog)</b>
<b>Story Points/Customer Priority</b>		3/5
<b>1.1TC0.0</b>	<i>Administrator can enter his login and password in the login screen and can perform operation in Women section</i>	
	<b>TC0.1</b>	Administrator can add items for women clothing with five colors
	<b>TC0.2</b>	Administrator can add items for women clothing with ten brands
	<b>TC0.3</b>	Administrator can add items for women clothing in fifteen different sizes
	<b>TC0.4</b>	Administrator can add items for women clothing with twenty different price range
	<b>TC0.5</b>	Administrator can add the clothing in exact Dress section (Formal, Casual, Ethnic, Fashionable)
	<b>TC0.6</b>	Administrator can see the newly added items in tabular form on daily basis
<b>1.1TC1.0</b>	<i>Administrator can see login page with floral background on home page</i>	
	<b>TC1.1</b>	Login text field should be of 30 characters
	<b>TC1.2</b>	Password text should be of 15 alphanumeric characters
	<b>TC1.3</b>	Display of password should be in asterisk character
	<b>TC1.4</b>	Two buttons with text submit and cancel
	<b>TC1.5</b>	Password retrieval option should be there

Table 6.37 Test Cases Story 2

<b>Story #</b>		<b>2 (As an administrator, I want to add items in the kids clothing section so as to attract kids)</b>
<b>Story Points/Customer Priority</b>		3/4

<b>1.2TC0.0</b>	<i>Administrator can enter his login and password in the login screen and can perform operation in Kids Section</i>	
	<b>TC0.1</b>	Administrator can add items for kids clothing with five colors
	<b>TC0.2</b>	Administrator can add items for kids clothing with ten brands
	<b>TC0.3</b>	Administrator can add items for kids clothing up to twelve years
	<b>TC0.4</b>	Administrator can add items for kids clothing with twenty different price range
	<b>TC0.5</b>	Administrator can add the clothing in exact Dress section (Party, Daily, night wear)
	<b>TC0.6</b>	Administrator can see the newly added items in tabular form on daily basis
	<b>TC0.7</b>	Administrator can add items in proper gender section(boy or girl)
<b>1.2TC1.0</b>	<i>Administrator can see login page with floral background on home page</i>	
	<b>TC1.1</b>	Login text field should be of 30 characters
	<b>TC1.2</b>	Password text should be of 15 alphanumeric characters
	<b>TC1.3</b>	Display of password should be in asterisk character
	<b>TC1.4</b>	Two buttons with text submit and cancel
	<b>TC1.5</b>	Password retrieval option should be there

Table 6.38 Test Cases Story 3

<b>Story #</b>	<b>3 (As an administrator, I want to add items in the kitchenware section so as to attract hotel management fellows)</b>	
<b>Story Points/Customer Priority</b>	2/2	
<b>2.3TC0.0</b>	<i>Administrator can enter his login and password in the login screen and can perform operation in Kitchenware section</i>	
	<b>TC0.1</b>	Administrator can add items for Kitchenware with five colors
	<b>TC0.2</b>	Administrator can add items for Kitchenware with ten brands
	<b>TC0.3</b>	Administrator can add items for Kitchenware in three different sizes
	<b>TC0.4</b>	Administrator can add items for Kitchenware with twenty different price range
	<b>TC0.5</b>	Administrator can add the kitchenware items in exact Category (Serving, cooking, Storing, Cutlery )
	<b>TC0.6</b>	Administrator can see the newly added items in tabular form on daily basis
<b>2.3TC1.0</b>	<i>Administrator can see login page with floral background on home page</i>	
	<b>TC1.1</b>	Login text field should be of 30 characters
	<b>TC1.2</b>	Password text should be of 15 alphanumeric characters
	<b>TC1.3</b>	Display of password should be in asterisk character
	<b>TC1.4</b>	Two buttons with text submit and cancel
	<b>TC1.5</b>	Password retrieval option should be there

Table 6.39 Test Cases Story 4

<b>Story #</b>	<b>4 (As an administrator, I want to add items in the electronic section so as to attract the users)</b>	
<b>Story Points/Customer Priority</b>	2/3	
<b>2.4TC0.0</b>	<i>Administrator can enter his login and password in the login screen and can perform operation in electronic section</i>	
	<b>TC0.1</b>	Administrator can add items for electronic with three colors
	<b>TC0.2</b>	Administrator can add items for electronic with five brands
	<b>TC0.3</b>	Administrator can add items for electronic in three different sizes

	<b>TC0.4</b>	Administrator can add items for electronic with five different price range
	<b>TC0.5</b>	Administrator can add the electronic items in exact Category (Kitchen Related, Utility, Entertainment)
	<b>TC0.6</b>	Administrator can see the newly added items in tabular form on daily basis
<b>2.4TC1.0</b>	<i>Administrator can see login page with floral background on home page</i>	
	<b>TC1.1</b>	Login text field should be of 30 characters
	<b>TC1.2</b>	Password text should be of 15 alphanumeric characters
	<b>TC1.3</b>	Display of password should be in asterisk character
	<b>TC1.4</b>	Two buttons with text submit and cancel
	<b>TC1.5</b>	Password retrieval option should be there

In this way, further moving on color, brand, price, size and for others many test cases can be written in user friendly linguistic style. Now, test cases are written for new requirement (9) when all the stories are developed and delivered in first two sprints.

Table 6.40 Test Cases Story 9

<b>Story #</b>		<b>9 (As an administrator, I want that users should be able to search items in the specific category so as to satisfy users)</b>
<b>Story Points/Customer Priority</b>		8/5
<b>3.9TC0.0</b>	<i>User can enter his login details and password in the login screen and can perform operation of searching in different sections</i>	
	<b>TC0.1</b>	User can search items for women with different options in different categories of dress section
	<b>TC0.2</b>	User can search items for kids with different options in different categories of dress section for boy/girl
	<b>TC0.3</b>	User can search items for hotel management fellow with different options in different categories of kitchenware
	<b>TC0.4</b>	User can search items with different options in different categories of electronic section
<b>3.9TC1.0</b>	<i>User can see login page with green background on home page</i>	
	<b>TC1.1</b>	Login text field should be of 30 characters
	<b>TC1.2</b>	Password text should be of 15 alphanumeric characters
	<b>TC1.3</b>	Display of password should be in asterisk character
	<b>TC1.4</b>	Two buttons with text submit and cancel
	<b>TC1.5</b>	Password retrieval option should be there
<b>3.9TC2.0</b>	Administrator can evaluate the satisfaction level	
	<b>TC2.1</b>	User can fill the questionnaire for satisfaction evaluation
	<b>TC2.2</b>	Administrator can see the report of satisfaction level in each category
	<b>TC2.3</b>	User feedback can be used to improve the policy of the online store
	<b>TC2.4</b>	Administrator can see the report in graph form

**Note:** For each test case of the user story positive and negative scenarios are considered as per the requirement. Negative scenario means what will happen if something

unexpected is entered in the text field. For example constraint for age is numeric entry and user enters alphabets. Then, accordingly warning messages would be displayed.

SLRT is used before the end of sprint 1 and 2 of the current scenario in the form of unit tests and acceptance tests. Unit testing means suite of white box tests that is the base of the regression tests and acceptance tests means suite of black box tests that is needed for checking the definition of done of a story. After every sprint, application is delivered to the customer by checking the functionality. So, for the defined scenario stories 1 to 8 are assumed to be operational. Now, in sprint 3, story 9 that needs to be implemented is dependent on stories 1, 2, 3 & 4 of the existing working system.

Sentence Priority Score (SPS) can be identified for any sentence. The priority of a sentence over the other sentences is calculated using story point of the story, customer priority and effect Eu. SPS is calculated for Table 6.40 using equation 4 and is shown in Table 6.41 for story 9.

$$(4) \quad SPS = \sum_{i=1}^n SP * CP * Eu$$

Table 6.41 Sentence Priority Score Story 9

Story #	Sentence Priority Score (SPS)	
9	<b>3.9TC0.0</b>	
	<b>TC0.1</b>	$(3*5*4*11) + (6*9*17*4) = 660 + 3672 = 4332$
	<b>TC0.2</b>	$(3*4*4*11) + (6*9*17*4) = 528 + 3672 = 4200$
	<b>TC0.3</b>	$(2*2*4*10) + (6*9*17*4) = 160 + 3672 = 3832$
	<b>TC0.4</b>	$(2*3*4*10) + (6*9*17*4) = 240 + 3672 = 3912$
	<b>3.9TC1.0</b>	
	<b>TC1.1</b>	$(3*5*4*11) + (3*4*4*11) + (2*2*4*10) + (2*3*4*10) + (1*5*17*4) = 660 + 528 + 160 + 240 + 340 = 1928$
	<b>TC1.2</b>	$(3*5*4*11) + (3*4*4*11) + (2*2*4*10) + (2*3*4*10) + (1*5*17*4) = 660 + 528 + 160 + 240 + 340 = 1928$
	<b>TC1.3</b>	$(3*5*4*11) + (3*4*4*11) + (2*2*4*10) + (2*3*4*10) + (1*5*17*4) = 660 + 528 + 160 + 240 + 340 = 1928$
	<b>TC1.4</b>	$(3*5*4*11) + (3*4*4*11) + (2*2*4*10) + (2*3*4*10) + (1*5*17*4) = 660 + 528 + 160 + 240 + 340 = 1928$
	<b>TC1.5</b>	$(3*5*4*11) + (3*4*4*11) + (2*2*4*10) + (2*3*4*10) + (1*5*17*4) = 660 + 528 + 160 + 240 + 340 = 1928$
	<b>3.9TC2.0</b>	
	<b>TC2.1</b>	$(2*1*17*4) = 136$
	<b>TC2.2</b>	$(2*1*17*4) = 136$
	<b>TC2.3</b>	$(2*1*17*4) = 136$
	<b>TC2.4</b>	$(2*1*17*4) = 136$

Table 6.42 Sentence Priority Score Sorting

TC	SPS
TC0.1	4332
TC0.2	4200
TC0.4	3912
TC0.3	3832
TC1.1	1928
TC1.2	1928
TC1.3	1928
TC1.4	1928
TC1.5	1928
TC2.1	136
TC2.2	136
TC2.3	136
TC2.4	136

After sorting results are shown in Table 6.42. TC0.1 has the highest priority as compared with rest of the test cases, TC0.2 has higher priority as compared with TC0.4 and TC2.4 has lowest priority. This sentence priority score which is based upon number of nouns and verbs is less time consuming as compared with random ordering of test cases. The core measure for effectiveness of build is velocity. If velocity measure is showing tested part is following the definition of done then acceptance criteria is met. Another metric is defect removal efficiency.

Defect Removal Efficiency is one of testing metric in Agile which says the number of defects that are removed per time unit (hours/days/weeks).It indicates the efficiency of defect removal methods, as well as indirect measurement of the quality of the product. It is computed by dividing the effort required for defect detection, defect resolution time and retesting time by the number of defects. This is calculated per test type, during and across test phases. Here effort is assumed to be time.

$$DRE = \frac{\text{Time (Defect Detection + Defect Resolution + Defect Retesting Time)}}{\text{No. of Defects}}$$

Here, Time calculation is dependent on story points and customer priority and unit of time is assumed to be hours.



### 6.5.3 Implementation

In broad sense the proposed TCP is shown in Figure 6.12. In section 6.5.2 “*A Linguistic Approach for Test Case Prioritization*” was performed on the basis of sentence priority score. In this section some of the steps of the prioritization using Linguistic parameters have been implemented. The proposed TCP technique is started by finding noun and verb of the ready story of new requirement and then performing mapping with the existing user stories noun and verb Tables. Finding of noun and verb is performed by making use of inbuilt formulae of Excel like:

- IF
- ISNUMBER
- SEARCH
- SUM
- COUNT

For reference, most risky user story 10, has been considered as the new user story for performing test case prioritization and rest of the user stories 1-9 are the existing user stories which are delivered to the C after proper quality check. User story 10 has the following elaborated requirements based on the confirming points (ready story) which are output of communication among team members and CR. One important point is that framing ready story is mandatory step for getting good hold on the client’s perspective for effective requirement development. Here, CR inputs are of great importance. Following are the confirming points for user story 10.

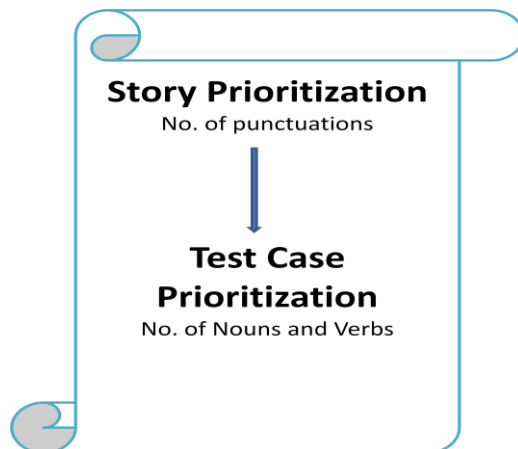


Figure 6.12 “Sequential Approach”

- Petitioner has access to active patents in .pdf format.
- Petitioner has access to revoked patents' case history in .pdf format.
- Petitioner has option of saving patent subsections as per the space available on her system.
- Petitioner has access to pending published applications in .pdf format.
- In the database various types of legal status are searched based on the unique publication number, application number or patent number or petition number.

In implementing test case prioritization, foremost step is to find noun and verb in these confirming points. Confirming points and ready story are used interchangeably. The ready story has following mixed verbs:

- Access
- Save
- Search

In the ready story 10, counTable and common nouns are present. These nouns have less significance as compared with uncounTable nouns. The uncounTable or mass nouns make tasks little bit tricky. Some of the nouns are as follows:

- Patent-----CounTable
- Subsection-----CounTable
- Application -----Common Noun
- Database-----Common Noun
- Publication-----Common Noun
- Petition-----Common Noun
- Legal-----Common Noun

If these nouns or verbs are to be found in the existing user stories 1-9 then following excel formula can be used:

IF(ISNUMBER(SEARCH("Patent", Cell number)), "YES", "NO").....(1)

Snapshot for counting noun using Excel has been shown below in Figure 15. In this case, ready story's nouns such as Patent, Subsection etc. are found in existing user stories 1-9. Using same formula, nouns may be searched in the existing ready story's 1-9 if time permits.

Similarly, verb search may be performed for user story or ready story.

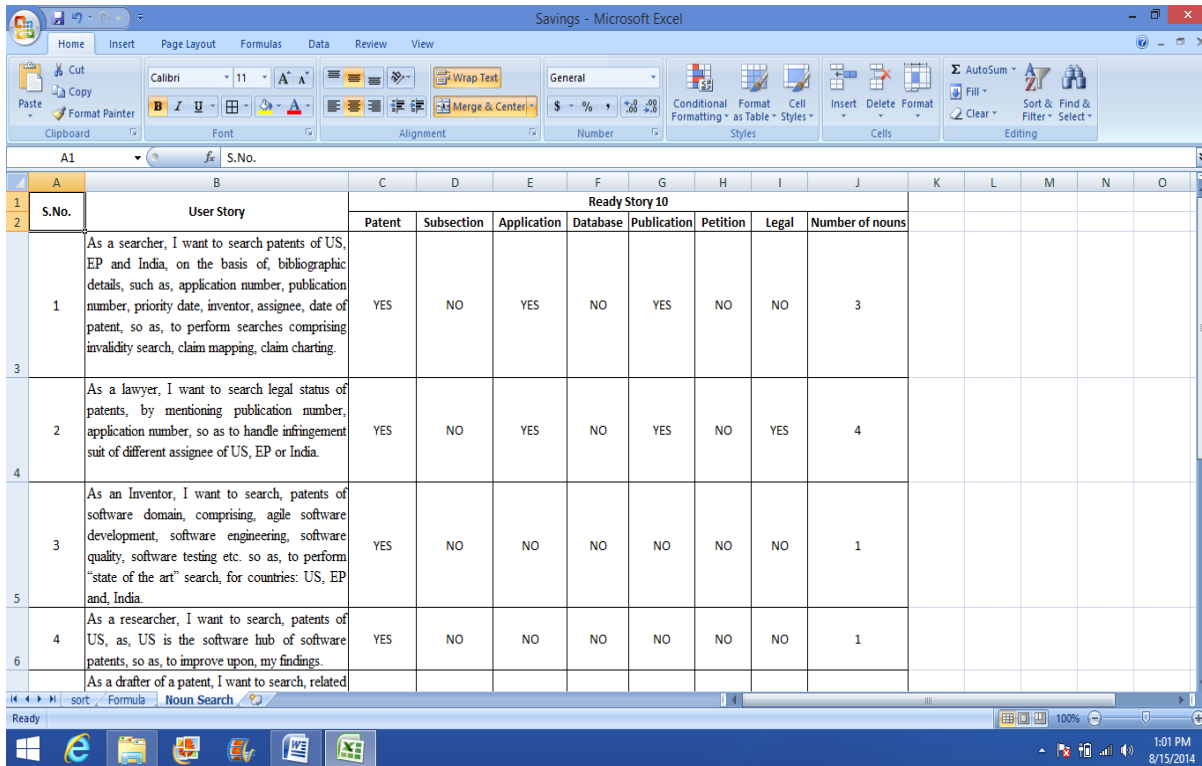


Figure 6.13 Noun Count

COUNTIF(C2:I11,"YES").....

(2)

Further, to count, total number of “YES” for nouns of Figure 6.13, Countif function may be used (See formula F2). This function returns 3, 4, 1, 1, 1, 1, 2 and 1, as number of nouns in respect of user stories 1-9. Similarly, total number of verbs is calculated. Screen shot for the same is shown in Figure 6.14. The countif function returns 1, 1, 1, 1, 1, 1, 1, 1, 0, as number of verbs in respect of user stories 1-9. Using multiplication operator on these count values of noun and verb, effect (see Figure 6.15) is calculated for new ready story in respect of user stories 1-9.

By reviewing, effect values, it is clear that user story's 9 has effect value zero. So, user stories 1-8 as per sorted effect values are:

$$2 > 1 > 8 > 3, 4, 5, 6, 7$$

User story 2 with effect value 4 is the user story that is most adversely affected by the change introduced by C in terms of user story 10. Similarly, user stories 3, 4, 5, 6, 7 with effect value of 1 are of equal importance and are least affected by the change introduced by C in terms of user story 10. Accordingly, SPS may be calculated.

S.No.	User Story	Access	Save	Search	Number of verbs
1	As a searcher, I want to search patents of US, EP and India, on the basis of, bibliographic details, such as, application number, publication number, priority date, inventor, assignee, date of patent, so as, to perform searches comprising invalidity search, claim mapping, claim charting.	NO	NO	Yes	1
2	As a lawyer, I want to search legal status of patents, by mentioning publication number, application number, so as to handle infringement suit of different assignee of US, EP or India.	NO	NO	Yes	1
3	As an inventor, I want to search, patents of software domain, comprising, agile software development, software engineering, software quality, software testing etc. so as, to perform "state of the art" search, for countries: US, EP and, India.	NO	NO	Yes	1
4	As a researcher, I want to search, patents of US, as, US is the software hub of software patents, so as, to improve upon, my findings. As a drafter of a patent, I want to search,	NO	NO	Yes	1

Figure 6.14 Verb Count

Story No.	No. of Noun	No of Verb	Effect
1	2	4	4
2	1	3	3
3	8	2	2
4	3	1	1
5	4	1	1
6	5	1	1
7	6	1	1
8	7	1	1
9	9	1	0
10		0	0

Figure 6.15 Effect

Highest value in SPS column represents highest priority of the corresponding test case. This SPS is less time consuming as compared with random/no ordering of test cases. The core measure for effectiveness of build is velocity. If velocity measure is showing that the tested part is following the definition of done then acceptance criteria is met. Another metric is (average percentage of fault detection) APFD [69]. APFD can be calculated as follows:

$$APFD = 1 - \frac{(Tf_1 + Tf_2 + \dots + Tf_m)}{mn} + \frac{1}{2n} \dots \dots \dots F4$$

where, n be the no. of test cases and m be the no. of faults.

(Tf1, ..., Tf<sub>m</sub>) are the position of first test T that exposes the fault.

Table 6.43 TCP Ordering

Faults	Test Cases											
	TC-2.5	TC-2.1	TC-2.2	TC-2.3	TC-2.4	TC-2.6	TC-4.1	TC-4.2	TC-4.3	TC-4.4	TC-4.5	TC-4.6
F1	*									*		
F2			*						*			
F3		*								*		*
F4	*			*			*				*	
F5		*			*							
F6	*					*			*		*	
F7					*		*					
F8		*		*								
F9							*					
F10	*								*			

Table 6.43 is a Table showing occurrence of faults by running test cases. For TCP, APFD comes out to be 83.3 %. Using no ordering method (See Table 6.44), APFD comes out to be 75.8%. Thus the prioritized test cases yield better fault detection than the non – prioritized test cases. A graph has been shown in Figure 6.16 to show the exact comparison between TCP and No ordering for test cases.

$$APFD = 1 - \frac{(1+3+2+1+2+1+5+2+7+1)}{10 \cdot 12 + \frac{1}{2} \cdot 12}$$

$$= 1 - \frac{25}{120 + \frac{1}{24}}$$

$$= 0.833$$

$$APFD = 1 - \frac{(5+2+1+3+1+5+4+1+7+5)}{10 \cdot 12 + \frac{1}{2} \cdot 12}$$

$$= 1 - \frac{34}{120 + \frac{1}{24}}$$

$$= 0.758$$

## 6.6 CONCLUSION

In this chapter, TCP techniques have been proposed for Agile environment. The first technique is based on calculated risk measure of the user story. Further, this risk measure is based on story point, implementation dependency and volatility rate of the links present in the user story graph. The second proposed technique is based on the relationship among objects of the primary and secondary parties. Further, a test pattern has been proposed for similar kind of problems. The last technique is based on the presence of noun and verbs present in the user stories. Further, on that basis, a sentence priority score is calculated for each test case using linguistic parameters and test cases can be managed. The effectiveness of TCP technique has been proved using APFD metric.

Table 6.44 “No Ordering”

Faults	Test Cases											
	TC-2.1	TC-2.2	TC-2.3	TC-2.4	TC-2.5	TC-2.6	TC-4.1	TC-4.2	TC-4.3	TC-4.4	TC-4.5	TC-4.6
F1					*					*		
F2		*							*			
F3	*									*		*
F4			*		*		*				*	
F5	*			*								
F6					*	*			*		*	
F7				*			*					
F8	*		*									
F9							*					
F10					*				*			

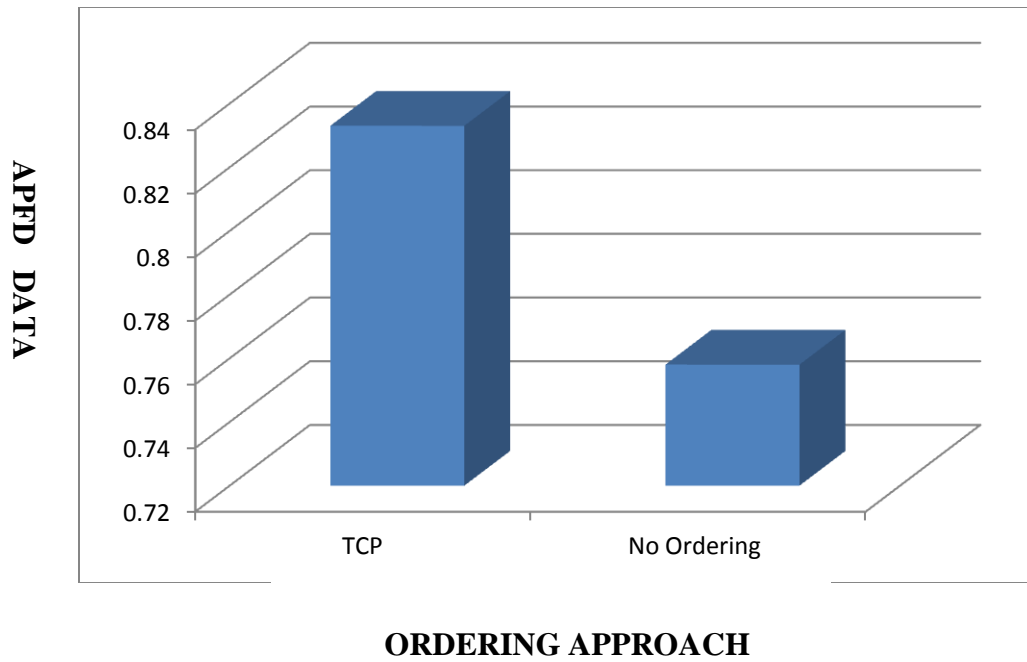


Figure 6.16 Comparison

## **CHAPTER VII**

### **CONCLUSION & FUTURE WORK**

In this work, Agile software development has been explored in the testing direction by discussing Agile testing life cycle for all stakeholders. More specifically, interaction of a tester with other stakeholders along with testing activities has been explained in the proposed Agile testing life cycle, which revolves around regression testing. Moreover, quadrant has been defined for regression testing which covers all quadrants. Further, for Scrum methodology, a Sprint flow diagram has been discussed by mentioning all testing activities before the sprint, within the sprint and post sprint.

Further, a distributed framework has been proposed for Agile software development environment. More specifically, pair programming and refactoring like practices has been discussed for handling different challenges of distributed environment. The pair members for a pair can be identified using proposed buddy identifier approach.

Furthermore, a RTS technique has been proposed for selecting optimized user stories so as to utilize resources to its fullest. It makes use of two important parameters namely average path length and average path value. Optimized results are obtained by considering APL and APV values. Validity of the technique is done by using the velocity metric which is measured for a sprint. Velocity is a metric that predicts how much work an Agile software development team can successfully complete within a two-week sprint. With this technique, optimized selection of user story is done in less time resulting in more productivity in terms of satisfaction and quality.

Finally, TCP techniques have been proposed for Agile environment. The first technique is based on calculated risk measure of the user story. Further, this risk measure is based on story point, implementation dependency and volatility rate of the links present in the user story graph. The second proposed technique is based on the relationship among objects of the primary and secondary parties. Further, a test pattern has been proposed for similar kind of problems. The last technique is based on the presence of



noun and verbs present in the user stories. Further, on that basis, a sentence priority score is calculated for each test case using linguistic parameters and test cases can be managed. The effectiveness of TCP technique has been proved using APFD metric.

The proposed Agile testing techniques comprising regression test case prioritization based on linguistic parameters such as noun and verb may be implemented by considering other parameters such as adjectives, adverbs, accent and many more. In the present work, excel and adobe captivate tool were used to implement proposed prioritization techniques of the distributed environment. Furthermore, macros may be implemented for same functionality using VB script. Also, the proposed regression test selection approach takes into consideration weights of the undirected graph in a module and optimal nature of this proposed method removes other risks of the development of the user story. The new method can be developed which is based on directed graph for user stories.

The proposed Agile model takes into account interest of all stakeholders. For this proposed model, automation central can be created which may link all stakeholders for all kinds of communication during Agile software development.

## REFERENCES

- [1]. Abrahams Faried, “Method And Apparatus For Enabling Agile Development Of Services In Cloud Computing And Traditional Environments”, Publication No. US20140282380.
- [2]. Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J., “New Directions on Agile Methods : A Comparative Analysis”, In Proceeding of 25th International conference on Software engineering 2003.
- [3]. Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J., “Agile Software Methods review and Analysis”, Espoo, Finland: Technical Research Centre of Finland, VTT publication, <http://www.inf.vtt.fi/pdf/publications/2002/478.pdf.2002>
- [4]. Adipat Larprattanakul and Taratip Suwannasart, “An approach for regression test selection using object dependency graph,” 978-0-7695-4988-0/13 © 2013 IEEE DOI 10.1109/INCoS.2013.115.
- [5]. “Agile Alliance. Principles Behind the Agile Manifesto”, [www.Agilemanifesto.org/principles.html](http://www.Agilemanifesto.org/principles.html), 2001.
- [6]. Amit Juyal, Umesh Kumar Tiwari, Lata Nautiyal, Shashidhar G. Koolagudi, “Agile Plus Comprehensive model for Software Development”, In International Journal Computer Technology& Applications, Volume 3 (4), 1378-1383
- [7]. Anita, Naresh Chauhan, “A Framework for Quality Improvement in Distributed Agile Environment”, IEEE International Conference On Research And Development Prospects On Engineering And Technology, March 2013, E. G. S. Pillay Engineering College, Tamilnadu, India.
- [8]. Anita, Naresh Chauhan, “A Linguistic approach for TCP in an Agile Environment”, 13<sup>th</sup> Annual International Software Testing Conference (4<sup>th</sup>-5<sup>th</sup> Dec 2013), Crossing The Chasm: From Assurance To Confirmation, Bangalore, India.
- [9]. Anita, Naresh Chauhan, “A Pattern Based Approach To Prioritize Test Cases For User Stories In An Agile Environment”, International Journal of Advance Foundation and Research in Computer (IJAFRC), Volume 1, Issue 7, July – 2014, pp. 185-194.

- [10]. Anita, Naresh Chauhan, “A Regression Test Selection Technique by Optimizing User Stories in an Agile Environment”, 4<sup>th</sup> IEEE International Advanced Computing Conference, IACC 2014 (21<sup>st</sup>-22<sup>nd</sup> Feb 2014), in ITM University, Gurgaon, India.
- [11]. Anita, Naresh Chauhan, “A Risk Based Story Prioritization Technique In An Agile Environment”, International Journal of Advance Foundation and Research in Computer (IJAFRC), Volume 1, Issue 7, July – 2014, pp. 16-25.
- [12]. Anita, Naresh Chauhan, “A Simplest Agile Life Cycle in an Agile Environment”, CSI Sponsored 7<sup>th</sup> International Conference on Software Engineering CONSEG - 2013 (15<sup>th</sup>-17<sup>th</sup> Nov 2013), Humanizing Software Engineering, Pune, India.
- [13]. Anita, Naresh Chauhan, “Agile Learning Model”, 2nd IEEE International Conference on MOOCs, Innovation and Technology in Education (19-20th December 2014), Patiala, Punjab, India.
- [14]. Anita, Naresh Chauhan, “An Object Oriented Design Approach For Modification of Rotten Code Using Regression Testing & Refactoring”, “An Object Oriented Design Approach For Modification of Rotten Code Using Regression Testing & Refactoring” Volume 4, Number 7, 2014, pp, 681-686.
- [15]. Anita, Naresh Chauhan, “State of the art search-Agile Software development in global market”, International Journal of Advances in Science, Engineering and Technology (IJASEAT), Institute of Research & Journals, Volume-2, Issue-2, April 2014.
- [16]. Anita, Naresh Chauhan, “Testing in an Agile Environment: A Project”, International Conference on Next Generation Communication and Computing Systems (ICNGC2S-10), December 25-26, 2010, Chandigarh, India
- [17]. Aoyamma, M., “Agile Software Process and its Experiences”, In IEEE Transaction 1999.
- [18]. Astels, D., “Test-Driven Development: A Practical Guide”, Prentice Hall: Upper Saddle River, NJ, USA, 2003.
- [19]. Balaji Sundramurth, Ronald S. Cordova, M. Sundara Rajan, “Traditional Vs Agile Methodology: An Analysis on Challenges faced in Testing Perspective”, Proc. of

- the Intl. Conf. on Advances In Computing, Electronics and Electrical Technology-CEET 2014.
- [20]. Beck, K., "Test Driven Development: By Example", Pearson: New York, NY, USA, 2003.
  - [21]. Beck, Kent., "Extreme Programming Explained: Embrace change", Addison Wesley, 2000.
  - [22]. Bhalerao, S., and Ingle, M., "Formalizing Communication Channel in Agile Methods", In Proceedings of International Conference on Trends in Information Science and Computing (TISC07), Dec. 2007.
  - [23]. Bhalerao, S., and Ingle, M., "Mapping SDLC phase with Various Agile Methods", In Proceedings of International conference on Advances in Computer Vision and information Technology, Nov. Aurangabad, pp. 318-325.
  - [24]. Bhalerao, S., D. Puntambekar and Ingle, M., "Generalizing Agile Software Development Life Cycle", In International Journal on Computer Science and Engineering Vol.1(3), 2009, 222-226.
  - [25]. Cao L. and Balasubramanium R., "Agile Software Development: Ad-hoc Practices or Sound Principles ?", IEEE ITPRO, March- April 2007, pp. 41-46.
  - [26]. Cockburn, A., "Crystal Clear: A Human-Powered Methodology for Small Teams", Addison-Wesley, 2004.
  - [27]. Cockburn, A., "Agile Software Development", Pearson Education, Asia, Low Price Edition, 2007.
  - [28]. Cockburn, A., and Highsmith, J., "Agile Software Development: The People Factor", In Computer, Nov. 2001, pp. 131-133.
  - [29]. Coelho Evita and Basu Anirban, "Effort Estimation in Agile Software Development using Story Points", International Journal of Applied Information Systems (IJ AIS)-ISSN: 2249-0868, Volume 3-No. 7, August 2012
  - [30]. Cognizant 20-20 insights, <https://www.cognizant.com/InsightsWhitepapers/The-Case-for-Agile-Testing-codex891.pdf>, June 2014.
  - [31]. Cohn, M. "User stories applied: For Agile software development," Boston, MA: Addison-Wesley, 2004.

- [32]. Cohn, M., and Ford, D., “Introducing an Agile Process to an Organization”, In IEEE Computer Society 2003, pp.74-78.
- [33]. Cohn, Mike, “Agile Estimating and planning”, Prentice Hall, 2005.
- [34]. Crispin, L., & House, T., “Testing Extreme Programming” Addison-Wesley, 2002.
- [35]. De Souza, Ken, “A Tester in Developer’s Clothes”, blog, <http://kendesouza.blogspot.com>.
- [36]. Dhalait Shamim Ahmad Dadamir and TCS Limited, “Agile Unit and regression Testing Framework for domain specific languages”, Publication number US20130159963.
- [37]. Dingsøy, T., Dybå, T., and Abrahamsson, P., ”A Preliminary Roadmap for Empirical Research on Agile Software Development”, In Proceedings of Agile Conference 2008
- [38]. E. Gamma, R. Helm, R. Johnson et al., “Design Patterns: Elements of Reusable Object-Oriented Software”. Addison- Wesley.
- [39]. Elizabeth, K., Dwight, A. et al., “A development environment for distributed synchronous collaborative programming”, In Proceedings of the 13th annual SIGCSE Conference on Innovation and Technology in Computer Science Education, 2008, pp.158-162
- [40]. Emelie Engstrom, Per Runeson and Greger Wikstrand, “An empirical evaluation of Regression testing based on fix cache recommendations,” 978-0-7695-3990-4/10 © 2010 IEEE DOI 10.1109/ICST.2010.40.
- [41]. Fagerholm, F., Ikonen, M., Kettunen, P., Münch, J., Roto, V., Abrahamsson, P., “Performance Alignment Work: How Software Developers Experience the Continuous Adaptation of Team Performance in Lean and Agile Environments”, Information and Software Technology, vol. 64, pp. 132–147.Elsevier, 2015.
- [42]. Fagerholm, F., Pagels, M., “Examining the Structure of Lean and Agile Values Among Software Developers”, In Proceedings of the 15th International Conference on Agile Software Development (XP 2014): Agile Processes in Software Engineering and Extreme Programming. Lecture Notes in Business

- Information Processing, vol. 179, pp. 218–233. Springer International Publishing, 2014.
- [43]. Garg Swati, “Training Technique for Learning Agile Methodology”, Publication number US20080305460.
- [44]. Gerard Meszaros, “Agile regression testing using record and play,” OOPSLA 2003, Oct 26-30, 2003, Anaheim, California. ACM 1-58113-751-6/03/0010
- [45]. Graziotin, D., Wang, X., and Abrahamsson, P., “Understanding the Affect of Developers: Theoretical Background and Guidelines for Psychoempirical Software Engineering” In Proceedings of the 7th International Workshop on Social Software Engineering, SSE 2015, pages 25–32, NewYork, NY, USA. ACM.
- [46]. Hendrickson, E., “Agility for Testers”, Pacific Northwest Software Quality Conference 2004.
- [47]. <http://collaboration.csc.ncsu.edu/laurie/Papers/TDDpaperv8.pdf>
- [48]. <http://humanresources.about.com/od/interpersonalcommunication1/qt/nonverbal-communication-in-the-workplace.htm>
- [49]. <http://www.adobe.com/support/captivate/gettingstarted.html>
- [50]. Iacovelli A. and Souveyer C., “Framework for Agile Method Classification”, Proceedings of Model Driven Information System Driven Engineering-Enterprise, User and System Model (MoDISE –EUS) 2008, pp. 91-10
- [51]. Inoue Tomomi, “Information Processor, Method, And Program For Determining Priority Of Test Case To Be Executed In Regression Test”, 2008129661.
- [52]. Jennifer Dorette, “Comparing Agile XP and Waterfall Software Development Processes in two Start-up Companies”, Master of Science Thesis in the Programme Software Engineering and Technology, Chalmers University of Technology Department of Computer Science and Engineering Göteborg, Sweden, November 2011.
- [53]. K. Beck, W. Cunningham, "Using Pattern Languages for Object-Oriented Program". OOPSLA '87 Retrieved 2006- 05-26.
- [54]. Kniberg, Henrik, “Scrum and XP from the trenches”, Lulu.com,2007
- [55]. Kohl, J., “Pair Testing. Better Software”, Jan 2004.

- [56]. L. Aversano, L. Cerulo, M.D. Penta, “The relationship between design patterns defects and crosscutting and cross cutting concern scattering degree”, *Software, IET*, vol.3, no.5, pp.395-409, October 2009.
- [57]. L. Crispin and J. Gregory, “Agile Testing: A Practical Guide for Testers and Agile Teams”, ISBN-13: 978-0321534460, Edition 1.
- [58]. Labuschagne, A. and Holmes, R., “Do Onboarding Programs Work?”, In *Proceedings of the 12th Working Conference on Mining Software Repositories*, Florence, Italy. IEEE.
- [59]. Larman, C.; Basili, V., “Iterative and incremental developments”, *A brief history. Computer* 2003, 36, 47–56.
- [60]. Lenberg, P., Feldt, R., and Wallgren, L. G., “Behavioral software engineering: A definition and systematic literature review”, *Journal of Systems and Software*, 107:15–37, 2015.
- [61]. Lenberg, P., Feldt, R., and Wallgren, L.-G., “Towards a Behavioral Software Engineering. In *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*”, CHASE 2014, pages 48–55, New York, NY, USA. ACM.
- [62]. M.Paasivaara, S. Durasiewicz, C. Lassenius, “Using Scrum in Distributed Agile Development: A Multiple Case Study, *International Conference on Global Software Engineering*”, pp.195-204, 2009
- [63]. Maria Sagheer, Tehreem Zafar, Mehreen Sirshar, “A Framework For Software Quality Assurance Using Agile Methodology”, *International Journal Of Scientific & Technology Research* Volume 4, Issue 02, February 2015.
- [64]. Microsoft Corporation, “System and method to facilitate manageable and Agile deployment of services in accordance with various topologies”, Patent number US7636782, Year 2009.
- [65]. N. Ganesh and S. Thangasamy, “New Agile Testing modes”, *Information Technology Journal* 11 (6): 707-712, 2012.
- [66]. Orit Hazzan, Yael Dubinsky, “Agile Software Engineering”, Springer International Edition, 2011.

- [67]. Pettichord, B., “Agile Testing Challenges”, Pacific Northwest Software Quality Conference 2004.
- [68]. R. Ferenc, A. Beszedes, L. Fulop et al., “Design Pattern Mining Enhanced by Machine Learning”, 25-30 September 2005, Budapest, Hungary.
- [69]. R. Pradeepa and K. VimalaDevi, “Effectiveness of Test case Prioritization using APFD Metric”, published in International Conference on Research Trends in Computer Technologies (ICRTCT - 2013) Proceedings published in International Journal of Computer Applications® (IJCA) (0975 – 8887).
- [70]. Rajendra Ganpatrao Sabale, A.R. Dani, “Comparative Study of Prototype Model for Software Engineering With System Development Life Cycle”, IOSR Journal of Engineering (IOSRJEN) ISSN: 2250-3021, www.iosrjen.org. Volume 2, Issue 7, July 2012, pp. 21-24.
- [71]. Rajlich, “Agile Software Development-Software change”, Agile 2014” 1916\_Software\_Change\_Agile2014.pdf.
- [72]. Rashmi Popli, Anita, Naresh Chauhan ,”A Mapping Model for transforming Traditional Software Development Methods to Agile Methodology”, International Journal of Software Engineering and Applications (IJSEA) Vol 4, No. 4, July 2013,pp. 53-64.
- [73]. Rashmi Popli, Anita, Naresh Chauhan, “Mapping of Traditional Software Development Methods to Agile Methodology” The Third International Conference on Computer Science and Information Technology (CCSIT- 2013), February 18-20 , 2013, Bangalore, India.
- [74]. S.Balaji, Dr.M.Sundararajan Murugaiyan, “Waterfall vs V-Model vs Agile : A Comparative Study on SDLC”, International Journal of Information Technology and Business Management, Vol.2 No. 1, JITBM, ISSN 2304-0777, 2012.
- [75]. Schwaber, K. & Beedle, M., “Agile Software Development with SCRUM”. Prentice Hall, 2001.
- [76]. Shelly, “Comparative Analysis of Different Agile Methodologies”, International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online) Vol. 3, Issue 1, pp: (199-203), Month: January - March 2015.



- [77]. Shikha Maheshwari, Dinesh Ch. Jain, "A Comparative Analysis of Different types of Models in Software Development Life Cycle", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 5, ISSN: 2277 128X, May 2012
- [78]. Shore, James and Shane Warden, "The art of Agile Development", O'Reilly Media, 2007.
- [79]. Singh Yogesh and Aggarwal K.K., "Software Engineering", New Age International Publishers, Revised second edition 2005
- [80]. Song Yang, David Knoke, "Optimal connections: strength and distance in valued graphs," [www.elsevier.com/locate/socnet](http://www.elsevier.com/locate/socnet), 2001
- [81]. Ståle Amland, "Risk Based Testing of a Large Financial Application", Proceedings of the 14th International Conference and Exposition on TESTING Computer Software, June 16-19, 1997, Washington, D.C., USA.
- [82]. Sukkarieh, J.Z.; Kamal, J., "Towards Agile and Test-Driven Development in NLP Applications", In Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing, Boulder, CO, USA, 5 June 2009; pp. 42–44.
- [83]. "Timeboxing DSDM Consortium, DSDM Agile project Framework", [www.dsdm.org](http://www.dsdm.org) September 2013
- [84]. Todd L. Graves, Mary Jean Harrold, Jung-Min Kim, Adam Porter and Gregg Rothermel, "An empirical study of Regression test Selection Techniques", 0-8186-8368-6198, 1998 IEEE.
- [85]. VersionOne "8th annual state of agile survey", <http://stateofagile.versionone.com/>, 2014.

## **BRIEF PROFILE OF THE RESEARCH SCHOLAR**

She has done M.Tech (CE) from Maharishi Dayanand University in year 2009, B.Tech (CSE) from Maharishi Dayanand University in the year 2004. She has 4 years of industry experience and 7 years of teaching experience. She is a lifetime member of Computer Society of India and Agile Software Community of India. Her research area includes Software Engineering, Software Testing, Software Quality and Agile software development. She has presented her papers in International Conferences and National conferences. Her papers are published in various International Conferences and International Journals. She has knowledge in the field of Intellectual Property Rights such as patent, copyright etc.

## LIST OF PUBLICATIONS

### List of Published Paper (International Journal)

S.No	Title	Name of Journal where Published	No.	Volume & Issue	Year	Pages
1.	<ul style="list-style-type: none"> <li>• “A Pattern Based Approach To Prioritize Test Cases For User Stories In An Agile Environment”</li> </ul> <p>Listed in <u>Thomson Reuters</u> Researcher ID</p> <p><b>Indexing:</b></p> <ul style="list-style-type: none"> <li>• Google Scholar</li> <li>• INDEX COPERNICUS</li> <li>• RESEARCH BIBLE</li> <li>• DRJI</li> <li>• iiS</li> <li>• MENDELEY</li> <li>• ScienceCentral.com</li> <li>• Bing</li> <li>• .docstoc</li> <li>• Slideshare</li> <li>• Scribd</li> <li>• CiteSeer</li> <li>• Knimbus</li> <li>• INNOSPACE</li> <li>• Citeulike</li> <li>• Advanced Sciences Index</li> <li>• Journalindex.net</li> <li>• Q.Sensei</li> <li>• Electronic Journals Library</li> <li>• CiteFactor</li> <li>• Scientific Indexing services</li> </ul>	<p>International Journal of Advance Foundation and Research in Computer (IJAFRC)</p> <p>Directory of Science Score-22.86</p> <p>Scientific Journal Impact Factor 2012 : 3.599</p> <p>ISRA: Journal Impact Factor (JIF) - 1.317</p>	ISSN 2348 - 4853	Volume 1, Issue 5	May - 2014	185-194

	<ul style="list-style-type: none"> <li>• Open Academic Journals Index</li> <li>• DIIF</li> <li>• Issuu</li> <li>• TechRepublic</li> <li>• Indian Citation index</li> <li>• Academia.edu</li> </ul> <p><b>Link:</b> <a href="http://ijafr.org/">http://ijafr.org/</a></p>					
2.	<p>“A Risk Based Story Prioritization Technique In An Agile Environment”</p> <p>Listed in <u>Thomson Reuters</u> Researcher ID</p> <p><b>Indexing:</b></p> <ul style="list-style-type: none"> <li>• Google Scholar</li> <li>• INDEX COPERNICUS</li> <li>• RESEARCH BIBLE</li> <li>• DRJI</li> <li>• iiS</li> <li>• MENDELEY</li> <li>• ScienceCentral.com</li> <li>• Bing</li> <li>• .docst c</li> <li>• Slideshare</li> <li>• Scribd</li> <li>• CiteSeer</li> <li>• Knimbus</li> <li>• INNOSPACE</li> <li>• Citeulike</li> <li>• Advanced Sciences Index</li> <li>• Journalindex.net</li> <li>• Q.Sensei</li> <li>• Electronic Journals Library</li> <li>• CiteFactor</li> <li>• Scientific Indexing services</li> </ul>	<p>International Journal of Advance Foundation and Research in Computer (IJAFRC)</p> <p>Directory of Science Score- 22.86</p> <p>Scientific Journal Impact Factor 2012 : 3.599</p> <p>ISRA: Journal Impact Factor (JIF) - 1.317</p>	ISSN 2348 - 4853	Volume 1, Issue 7	July - 2014	16-25

	<ul style="list-style-type: none"> <li>• Open Academic Journals Index</li> <li>• DIIF</li> <li>• Issuu</li> <li>• TechRepublic</li> <li>• Indian Citation index</li> <li>• Academia.edu</li> </ul> <p><b>Link:</b> <a href="http://ijafr.org/">http://ijafr.org/</a></p>					
3.	<p>“A Mapping Model for Transforming Traditional Software Development Methods to Agile Methodology”</p> <p><b>Abstracting &amp; Indexing:</b></p> <ul style="list-style-type: none"> <li>• ProQuest</li> <li>• Scirus</li> <li>• EBSCO</li> <li>• Google Scholar</li> <li>• CSEB</li> <li>• Scribd</li> <li>• DOAJ</li> <li>• getCITED</li> <li>• pubget</li> <li>• CiteSeer</li> <li>• .docst c</li> <li>• Pub zone</li> <li>• ULRICHSWEB</li> <li>• PKP</li> <li>• WorldCat</li> <li>• Cnki.net</li> </ul> <p><b>Link:</b> <a href="http://www.airccse.org/journal/ijsea/ijsea">http://www.airccse.org/journal/ijsea/ijsea</a></p>	<p>The International journal of Software Engineering &amp; Applications (IJSEA)</p> <p>Academy &amp; Industry research Collaboration center (AIRCC)</p>	<p>ISSN : 0976 - 2221</p> <p>e-ISSN : 0975 - 9018</p> <p>doi :10.5121/ijsea</p>	<p>Vol.4, No.4,</p>	<p>July 2013</p>	<p>53-64</p>
4.	<p>“An Object Oriented Design Approach For Modification of Rotten Code Using Regression Testing &amp; Refactoring”</p>	<p>International Journal of Information &amp; Computation Technology. International Research Publication House</p>	<p>ISSN 0974-2239</p>	<p>Volum e 4, Number 7</p>	<p>2014</p>	<p>681-686</p>
5.	<p>“State of the art search-Agile Software development in global market”</p>	<p>International Journal of Advances in Science, Engineering and Technology</p>	<p>ISS : 23 N 21</p>	<p>Volum e-</p>	<p>April 2014</p>	

	<p><b>Indexing:</b></p> <ul style="list-style-type: none"> <li>• Directory of Science</li> <li>• Google Scholar</li> <li>• DOAJ</li> <li>• JOUR Informatics</li> <li>• BASE</li> <li>• OAJI</li> <li>• DRJI</li> </ul> <p><b>Link:</b>  <a href="http://www.iraj.in/journal/IJASEAT/">http://www.iraj.in/journal/IJASEAT/</a></p>	<p>(IJASEAT)  Institute of Research &amp;  Journals</p> <p>JIFACTOR: 3.15</p>	<p>(Pri –  nt) 89  91</p> <p>ISS 23  N 21  (On : –  line 90  ) 09</p>	<p>2,Issue  -2</p>		
--	---	---	---	------------------------	--	--

## List of Published Paper (International Conference)

S.No.	Title of the Paper along with volume, Issue No., Year of Publication	Publisher	Refereed or Non-Refereed	Whether you paid any money or not for publication	Remarks
1	<p>“A Regression Test Selection Technique by Optimizing User Stories in an Agile Environment”</p> <p>4<sup>th</sup> IEEE International Advanced Computing Conference, IACC 2014(21<sup>st</sup>-22<sup>nd</sup> Feb 2014), in ITM University, Gurgaon, India.</p>	IEEE Explorer	Yes	Yes	
2	<p>“A Linguistic approach for TCP in an Agile Environment”</p> <p>13<sup>th</sup> Annual International Software Testing Conference (4<sup>th</sup>-5<sup>th</sup> Dec 2013), Crossing The Chasm: From Assurance To Confirmation, Bangalore, India</p>	<p>QAI</p> <p><a href="http://www.qaiglobalservices.com/conference/stc2013/PDFs/ANITA.pdf">http://www.qaiglobalservices.com/conference/stc2013/PDFs/ANITA.pdf</a></p>	Yes	No	
3	<p>“A Framework for Quality Improvement in Distributed Agile Environment”</p> <p>IEEE International Conference On Research And Development Prospects On Engineering And Technology (ICRDPET - 2013), March 29-30, 2013, Vol. 4</p>	E. G. S. Pillay Engineering College , Tamilnadu, India	Yes	Yes	
4	<p>“A Simplest Agile Life Cycle in an Agile Environment”</p> <p>CSI Sponsored 7<sup>th</sup> International Conference on Software Engineering CONSEG -2013 (15<sup>th</sup>-17<sup>th</sup> Nov 2013), Humanizing Software Engineering, Pune, India.</p>	<p>Computer Society of India, 2013</p> <p>ISBN 978-1-63041-578-5</p> <p><a href="http://www.conseg.in/pune2013/proceedings.html">http://www.conseg.in/pune2013/proceedings.html</a></p>	Yes	Yes	

5	<p>“Testing in an Agile Environment: A Project”</p> <p>International Conference on Next Generation Communication and Computing Systems (ICNGC2S-10), December 25-26, 2010, Chandigarh, India</p>	<p>Institution of Engineers, Technocrats and Academicians Network (IETAN) Conference Proceedings</p>	-	Yes	
6	<p>“Mapping of Traditional Software Development Methods to Agile Methodology”</p> <p>The Third International Conference on Computer Science and Information Technology (CCSIT- 2013), February 18-20 , 2013, Bangalore, India</p>	<p>Academy &amp; Industry research Collaboration center (AIRCC)</p> <p><a href="http://airccse.org/currentissue14.html">http://airccse.org/currentissue14.html</a></p>	-	Yes	
7	<p>“Agile Learning Model”</p> <p>2nd IEEE International Conference on MOOCs, Innovation and Technology in Education ( 19-20th December 2014), Patiala, Punjab</p>	<p>IEEE Explorer</p>	-	Yes	