DESIGN OF SPRINT-POINT BASED ESTIMATION TECHNIQUES FOR AGILE SOFTWARE

THESIS

submitted in fulfillment of the requirement of the degree of

DOCTOR OF PHILOSOPHY

to

YMCA UNIVERSITY OF SCIENCE & TECHNOLOGY

by

RASHMI POPLI

Registration No: YMCAUST/Ph17/2010

Under the Supervision of

DR. NARESH CHAUHAN

PROFESSOR AND CHAIRMAN

DEPARTMENT OF COMPUTER ENGINEERING

YMCAUST, FARIDABAD

Department of Computer Engineering

Faculty of Engineering and Technology

YMCA University of Science &Technology

Sector-6, Mathura Road, Faridabad, Haryana, INDIA

MARCH, 2015

DECLARATION

I hereby declare that this thesis entitled "**DESIGN OF SPRINT-POINT BASED ESTIMATION TECHNIQUES FOR AGILE SOFTWARE"** being submitted in fulfillment of requirement for the award of Degree of Doctor of Philosophy in the Department of Computer Engineering under Faculty of Engineering and Technology of YMCA University of Science and Technology, Faridabad, during the academic year May 2011 to March 2015,is a bonafide record of my original work carried out under the guidance and supervision of **DR. NARESH CHAUHAN, PROFESSOR & CHAIRMAN, DEPARTMENT OF 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.

(RASHMI POPLI)

Registration No: YMCAUST/Ph17/2010

CERTIFICATE

This is to certify that this thesis entitled **"DESIGN OF SPRINT-POINT BASED ESTIMATION TECHNIQUES FOR AGILE SOFTWARE"** by **RASHMI POPLI,** submitted in fulfillment of the requirement for the award of degree of Doctor of Philosophy in Department of Computer Engineering, under Faculty of Engineering and Technology of YMCA University of Science and Technology Faridabad, during the academic year May 2011 to March 2015, 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.

DR. NARESH CHAUHAN
Professor and Chairman,

Department of Computer Engineering,

Faculty of Engineering and Technology

YMCA University of Science and Technology, Faridabad

Dated:

ACKNOWLEDGEMENTS

First of all, I would like to thank **God**, the Almighty, for having made everything possible by giving me strength and courage to do this work.

I would like to express my sincere gratitude to my thesis supervisor, **Dr. Naresh Chauhan**, for his continuous guidance, valuable advice, constructive criticism and helpful discussions. I am very grateful to him for his continual encouragement, motivation and long hours spent throughout the completion of my work. He always offered wisdom, insight and a skilled hand in overcoming the hindrances faced. I greatly value his timely and valuable advices. He gave me the opportunity to learn various new things, and taught me a lot about research, teaching, and life. Without his warm encouragement, I would not have been able to accomplish this thesis.

I am grateful to other teaching staff of the department for having been imparted with knowledge. Although it is not possible to name individual, I cannot forget my well-wishers for their persistent support and cooperation. I am also thankful to all my students who helped me directly or indirectly in completing my research work. I am thankful to my parents for their love, encouragement, and support throughout my education. I am also thankful for all the support received from my mother-in-law **Ms. Darshana Devi**.

Finally I would like to express my gratitude to my husband **Rajesh Popli** for providing the constant encouragement, financial and unconditional moral support to enable me to come up to this level in my life. Without his friendship and love this thesis would not have been completed. Finally, I like to thank my dear kids **Aarushi and Jeevesh** for the encouragement they have given to me, probably without knowing it.

Thanks to all of you!

(RASHMI POPLI)

ABSTRACT

The process of software development in software industry is going through a seismic shift, from traditional, heavy-weight software development methodologies towards simple, light-weight Agile software development (ASD) methodologies. ASD delivers superior and high-quality software products in small and rapid iterations with flexibility and adaptability to changing business conditions. ASD can be summarized as iterative and incremental development methods that have the adaptability to change throughout the systems development life cycle. In Agile environment, the requirements and results or working software develops through association among self-organizing, self-motivating and cross-functional team. It promotes adaptive planning approach, incremental development and delivery, and a time-boxed approach. Agile supports quick and flexible response to changes. Agile methods have the potential to provide a higher level of customer satisfaction, lower bug rates, a shorter development cycle, and a quicker adaptation to rapidly changing business requirements.

Traditional software development has been widely used in industry. Most of the software industries are drifting from traditional software development life cycle models to Agile environment for the purpose of attaining quality and for the sake of saving cost and time. It is very difficult for anyone to forget previous traditional practices and shift towards Agile as transitioning from traditional to Agile cannot be done at once or in a particular single step. This work presents a model of transitioning from traditional software development life cycle model to ASD model. ASD introduces changes in work habits. This challenge encourages Agile software developers to establish a development process that enables them to cope successfully with changes introduced during that process, while keeping the high quality of the product. The proposed Agile model and mapping function focuses on change introduction into organizations that plan to transit, or that are already in transition, to ASD.

As Agile is highly dynamic in nature so estimation of cost and time in Agile is a critical task. Estimation and structure of Agile methodologies are very different from those in traditional ones. A framework for estimation in Agile is proposed in this research work. By critical look of the literature survey, it has been observed that the modern Agile methods depends on historical data of project or past experience for estimation of cost, size, effort and duration. If the historical data is not present then these methods are not efficient. Therefore, there is a need of an efficient algorithmic method and an estimation tool, which can compute duration, cost and effort of the Agile project. In this research work some project-related factors, people-related factors, resistance factors and regression-testing efforts are proposed that affect estimation in a project. The research also proposed a Sprint-point based estimation framework that calculates more accurate release date, cost, effort and duration for the project by considering proposed factors. As the work attempts to propose an algorithmic estimation method based on the proposed factors, it would enhance efficiency of Agile software development methodology.

As Agile projects are of small duration so the team does not have so much amount of time to apply the mathematical algorithms**.** For implementing the Sprint-point based estimation framework a new Sprint–point based estimation tool(SPBE) is designed.The proposed SPBE tool for estimation place major emphasis on accurate estimates of effort, cost and release date by constructing detailed requirements as accurately as possible. The effectiveness and feasibility of the proposed technique has been shown by considering a case study which has been implemented on SPBE tool.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter I

INTRODUCTION

1.1 AGILE SOFTWARE DEVELOPMENT

The Agile Software Development (ASD) approach has been applied extensively during the mid-nineties of the 20th century. Although there are only about ten to fifteen years of accumulated experience using the Agile method, it is presently conceived as one of the mainstream methodology for software development [7,8,12]. The word "Agile" by itself means that something is flexible and responsive, so the Agile methodology implies its ability to survive in an environment of rapid change. Agile methodologies for software development take a new, lightweight approach to most aspects of designing, coding and producing applications. ASD is an iterative and incremental development method and its basic concept is customer – centered and it acknowledges that requirements can change. ASD offers a professional approach to software development that encompasses human, organizational and technological aspects of software development processes. The iterative and incremental based software development methods in Agile methodologies are powerful ways to deliver high quality software on time.

1.2 MOTIVATION AND GOAL

Although many positive benefits of the Agile methods have been published, there have been few empirical field studies on the negative aspects of various Agile methods [25,32,50]. The negative side of the Agile methods imply that there are problems, challenges and issues faced in developing high-quality software products using these methods. The existing models are being designed based on theoretical assumptions and have not been validated in real business situations. There are several challenges in the small scale as well as large scale development based on ASD. In this thesis, it has been identified that little practical research exists on present Agile estimation processes. This

work deals with the actual problems faced by companies in Agile environment, where the main challenge is estimation of cost, effort and time of a software project. By analyzing the problems in Agile, a framework has been created to perform release planning and estimation more accurately. Scrum [46,63] which is a popular and widely adopted Agile methodology, is chosen as the target of the proposed estimation model.

The objective of this research is to improve the reliability of estimation in Agile so as to calculate the release date of a project more efficiently. To achieve this objective, the work on following goals have been performed in this thesis:

- A framework of transitioning of traditional software development method in the context of adopting Agile.

- To design an efficient estimation model in Agile, which would provide higher accuracy and visibility in planning and estimating in Agile environment.

- To design an estimation tool in Agile that aims to identify how estimation can be conducted in a manner that complies with the need of real business environment.

1.3 CHALLENGES OF ESTIMATION IN AGILE ENVIRONMENT

As the complexity of Agile project grows, it becomes more difficult to estimate it. The researchers have focused on various techniques of estimation. A critical look at the available literature [57,76,92] indicates that the following issues need to be addressed towards building an effective estimation model.

Transitioning of traditional software development methods to Agile methodology: The issue is how to perform the transition when everybody in the company has expertise in a particular traditional model and that model is the heart of the company.

***Solution**: In order to perform the transitioning from traditional to Agile, a mapping model has been proposed by considering the existing traditional model of the organization. A general Agile life cycle model and a mapping function has been presented for transitioning to Agile environment.*

Factors affecting the release date estimation: The release planning is the activity to calculate the actual release date so that the final product is handed over into use for the customer. In scrum estimation technique a release plan is prepared but it doesn't consider any mathematical approach to calculate actual release date.

***Solution**: In order to have an efficient release date estimation, three types of factors i.e. project-related, people-related and resistance have been proposed. The project and people-related factors can increase or decelerate the velocity of project. But the resistance factors always decelerate the velocity and affect on productivity, thereby increasing the duration of the project.*

Uncertainty: The uncertainty of story-points is a big problem. Due to changing nature of Agile, size of user-story is not certain. Moreover, new user-stories can be added or existing user-stories can be modified or removed. This creates uncertainty in Agile project that leads to poor estimation of time and cost.

***Solution:** In Agile uncertainty cannot be eliminated completely but when estimating work, some steps can be taken to reduce it. The proposed technique reduces uncertainty by reducing the size of the user-story to be estimated. The fewer the elements that must be considered when producing an estimate, the more reliable will be the estimate.*

Estimation in Agile environment: Since the ASD is highly dynamic in nature so the issue is how to estimate size, cost and time.

***Solution**: To resolve this issue, a sprint-point based estimation technique in Agile has been proposed where a sprint-point is a unit time which a team member spends in sprint-*

related work in that iteration. The sprint-point estimation is done by using various proposed Agile estimation factors. A mathematical algorithm for release planning has been proposed.

Sprint-point based estimation tool: As Agile projects are of small duration so the team does not have so much amount of time to apply the mathematical algorithms**.**

Solution*: To resolve this issue and to automate the sprint-point based estimation framework a new Sprint –point based estimation tool(SPBE) is designed and developed. The proposed SPBE tool for estimation place major emphasis on accurate estimates of effort, cost and release date by constructing detailed requirements as accurately as possible. This tool is used as a vehicle to validate the feasibility of the project.*

1.4 ORGANIZATION OF THESIS

The thesis has been organized in the following chapters:

Chapter 2: The basic concepts of traditional software development methods and their lifecycles have been briefly discussed in this chapter. The history and various principles of Agile software development methods as well as the Agile manifestos are discussed. A comparative study of traditional software development method with Agile is analyzed and tabulated. Further, a detailed review of Agile software development and its life cycle has been provided. Further to that, Scrum, a popular Agile method is studied in detail. The review also considers the various Agile software development methods along with an overview of Agile estimation techniques.

Chapter 3: A general transitioning model in Agile is proposed for transitioning from traditional to Agile environment. A mapping has been presented so that transitioning of traditional software development methods to Agile can be achieved with convenience of Agile team and upper management. In this mapping model, different components are

proposed which can be helpful to accomplish the more quality standards which are preferred by the end-user.

Chapter 4: In the light of survey performed for Agile software development and issues discussed for estimation in above chapters, the need of a new estimation technique in Agile has been identified. A sprint-point based estimation framework has been designed on the proposals made in this work. This framework with full algorithmic details, implementation details and performance benefits has been presented in this chapter.

Chapter 5: This chapter identifies the need to automate the process of sprint-point based estimation. A sprint-point based estimation tool has been designed and validated through the analysis of a case study.

Chapter 6: 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 REVIEW

2.1 EARLY STAGE OF SOFTWARE DEVELOPMENT

The early stages of software development can be summarized as “build and fix model and code-some-more model”. This is a very simple scheme and it can be considered as the first generation in the history of software development. The fundamental idea behind this scheme is to firstly write code and do not put much effort on pre-planning and pre-designing, and fix bugs later if any are found at any stage. This illustrates that the first generation of software development processes did not include any structured and disciplined approach. This approach of software development worked very well for small-scale and relatively simple projects. However, as the size of projects increased, developers realized that they spent more time on fixing bugs than writing code. This led to a dramatic decrease in efficiency and predictability of software development.

The more software developers worked on large projects, the more they recognized that there is a need of a disciplined approach for software engineering in software development [3,29]. The first generation development methods were replaced by methods which place heavyweight on precise planning. Engineering-discipline-based development methods can be viewed as plan-driven methods, where the documentation of a complete set of requirements precedes architectural and high-level design, development, and implementation. The plan-driven software development methods require extensive planning, full documentation, and accurate reuse. The plan-driven methods also work best when developers know all of the requirements in advance and when requirements are relatively stable. These kinds of methods came to be known as heavyweight methods and are also considered as traditional software development methods. But the common problem is “that the customers or end-users change their minds frequently”

2.2 TRADITIONAL SOFTWARE DEVELOPMENT LIFE CYCLE MODELS

Traditional methodologies try to be predictive as they generate a schedule or plan of the project at the initial stages and follow this schedule for the whole life of the project. By traditional methods, complex software systems can be built in a chronological manner [30,53,69]. In these systems all the requirements are collected at the beginning, then all the design is completed and finally the master design of the system is implemented by the traditional software development life cycle model [97,103,104]. The complex software systems can be built in a single step, without changing requirements according to changing business or technology conditions. But the problem is that the end-users change their minds frequently [78]. In the software world requirements need to be fixed and rigid because it becomes very costly to make changes in the system after a certain point because of expensive construction phase.

2.2.1 Linear Sequential Model (Classic Waterfall Model)

W.W. Royce proposed the classic waterfall model in 1970. In this model firstly requirements analysis is done, after that designing is completed. When requirements freeze then coding, testing, integration, and maintenance are done as shown in Figure 2.1.This model is used in many software frameworks. This is heavy-weight document driven model in which heavy documentation with proper sequence is maintained[111].

The main problem of this classic approach is inflexibility as changes are not welcomed in this model because of its static nature. As bugs are identified after testing, so bugs keeps on increasing from one phase to another. Waterfall model is most suitable in situations where all the requirements are well known and well documented, there is no ambiguity of requirements and all the necessary resources with required expertise are available. The Table 2.1 lists out the pros and cons of classic waterfall model. ASD solves the inflexibility problem of waterfall model, *as Agile is dynamic in nature so requirements can be changed as per the customer need.*

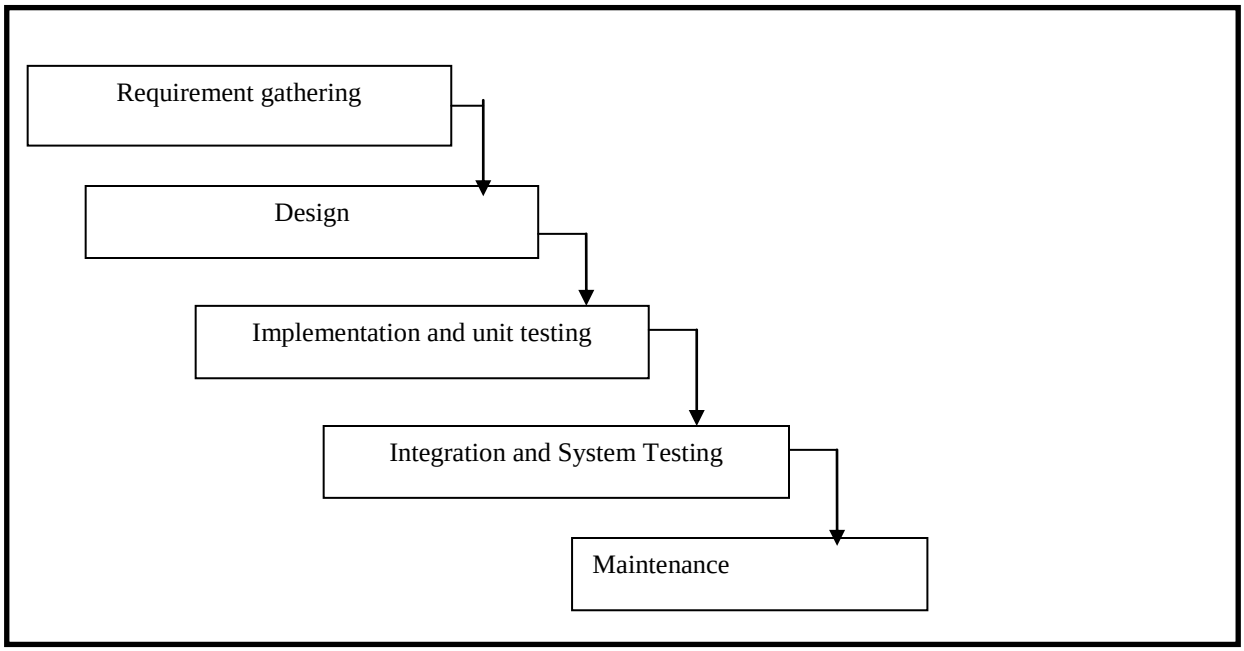


Figure 2.1: Classic Waterfall Model

Table 2.1: Pros and Cons of Classic Waterfall Model

S.No	Pros	Cons
1.	The model follows simple approach. It is easy to understand and use.	Working software is produced only at later stages.
2.	It is easy to manage this model due to the inflexibility of requirements. Every phase of this model has review process and specific deliverables.	There is high amount of risk and uncertainty.
3.	The phases of this model are processed and completed one at a time.	This is not a good model if the requirements are complex or if the project is object-oriented.
4.	It works well for projects where requirements are very clear and well understood.	It is a poor model for long and ongoing projects.
5.	Clearly defined stages and well understood milestones.	This model is not appropriate for projects in which requirements are complex and dynamic.
6.	In it process and results are well documented.	It is difficult to measure progress of the software within stages.

2.2.2 The Prototype Model

The various activities involved in this model are identification of initial basic requirements, developing the preliminary prototype, evaluating and enhancing of prototype as shown in Figure 2.2. There are two types of prototyping including throwaway prototyping or close ended and evolutionary prototyping [77].

In the close-ended prototyping, the prototype according to the customer requirements is created but this prototype never becomes a part of the delivered software as it is discarded. The Table 2.2 shows the pros and cons of prototyping model. The main goal of evolutionary prototyping is to create a robust prototype at initial stages in structured manner and constantly refining the prototype in further stages.

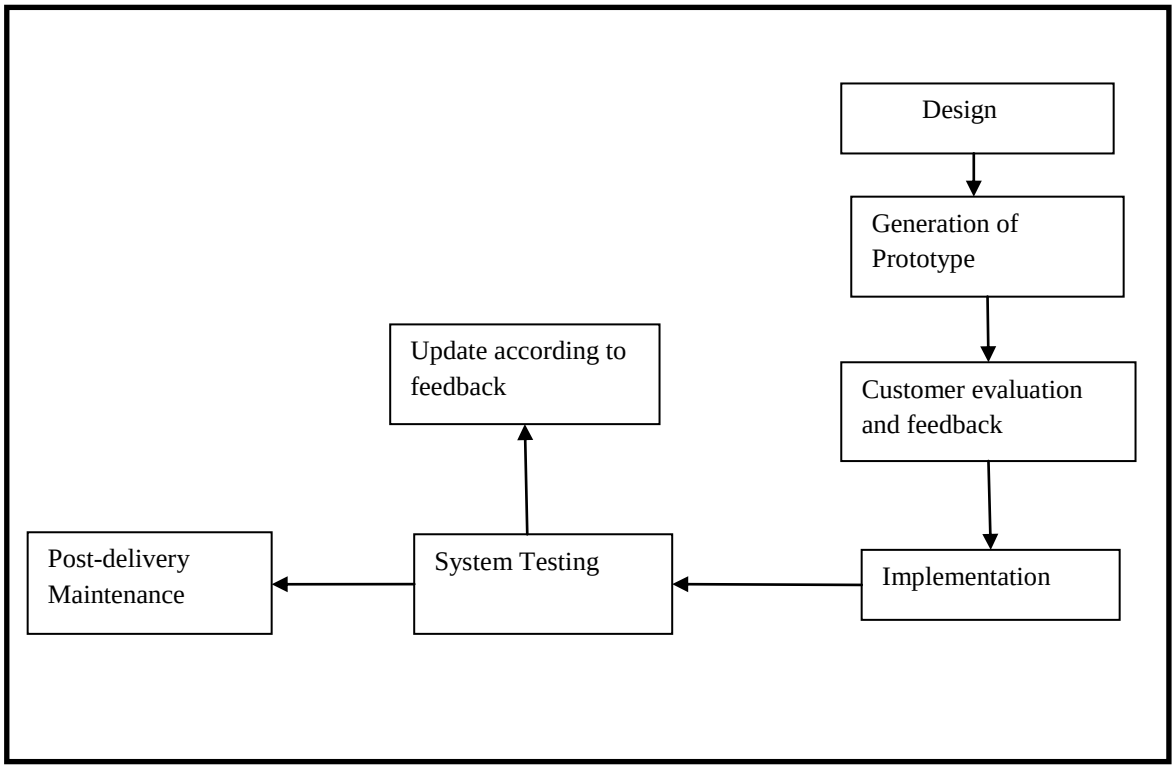


Figure 2.2: Prototype Model

In Agile way of software development, customer feedback and interactions are more important. In ASD one of the representative of customer is present with the team

members so that he can provide feedback for the improvement at any time and requirements can be modified according to his need. Communication between team members is the very important to deliver a good quality software. Also, prototype in case of Agile is not created rather user-stories are developed according to the requirements and working software is shown to the customers at the end of each iteration.

Table 2.2: Pros and Cons of Prototyping Model

S.No	Pros	Cons
1.	The prototype is a usable program	The prototype is not suitable as the final software product.
2.	Experience gathered from prototype is used to develop an actual system.	The code for prototype is thrownaway.
3.	Product is developed according to customer feedback	The development of prototype involves extra cost and time.

2.2.3 Incremental Model

Incremental model [38,106] is an evolution of waterfall model. After every cycle a useable product is given to the customer. The product is planned, designed, implemented, integrated and tested as a series of incremental iterations as shown in Figure 2.3. The Incremental software development model may be applicable to the following projects:

- Requirements are clearly defined even at initial stages.
- No confusion about functionality of the final product.

The small working software is needed early in the project. In incremental model the working software is delivered frequently. This model is more flexible and less expensive. As the working software is the result of a small iteration so it is easier to test and debug. Also, it is easier to manage risk because risky pieces are identified early. This model is mostly used when the requirements of project are clearly documented, defined and

understood. However, some details can evolve with time. And also there is a need to get a product to the market early. The Table 2.3 lists out the pros and cons of Incremental SDLC Model.

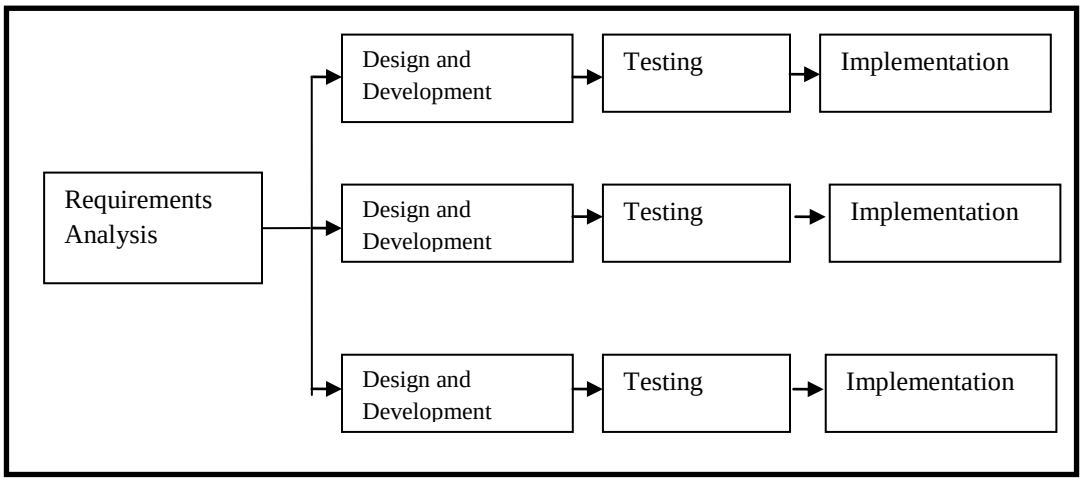


Figure 2.3: Incremental Model

Table 2.3: Pros and Cons of Incremental SDLC Model

S.No	Pros	Cons
1.	Some working functionality can be developed quickly in the life cycle of the project.	More resources may be required in this model.
2.	In it the results are obtained early after each build.	This model is not very suitable for changing requirements.
3.	Due to incremental nature the progress of the project can be measured easily.	More management attention is required because of iterative process.
4.	As iterations are small so testing, risk analysis and debugging is easy.	As all the requirements can't be gathered in the beginning of the entire life cycle so system architecture or design issues may arise.
5.	As high risk part is done first so it is easier to manage risk.	It requires definition of the complete system before defining increments.
6.	operational product is delivered, with each iteration	It is not suitable for smaller projects.

In Agile context, builds or iterations are created gradually and then review is done by presenting demonstrations to the customer. Daily meetings in ASD are done to know how much work is left.

2.2.4. Spiral Model

The Spiral model was first defined by Barry Boehm. He recognized the problem of risk in complex software projects. Important software projects have failed because project risks were neglected and nobody was prepared to manage these risks [10,33,76]. Barry Boehm combined elements of prototyping, evolutionary and incremental models and also tried to incorporate the project risk factor into a new life cycle model.

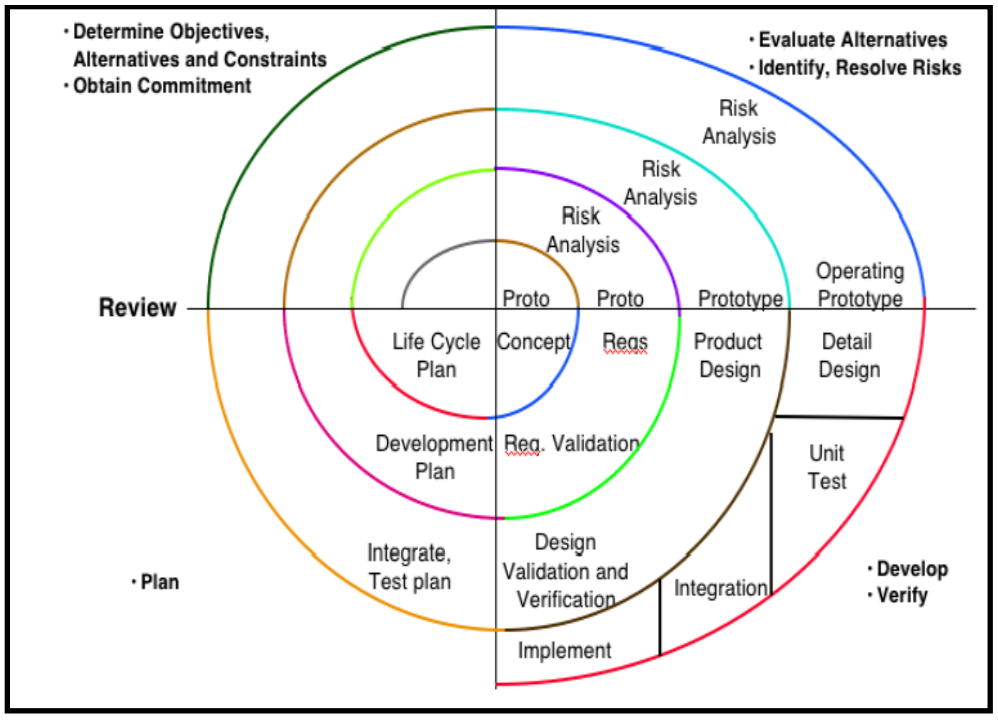


Figure 2.4: Spiral Model

The goal of the spiral model is to identify risk and focus on it early. The term spiral refers to successive iterations outward from a central starting point. According to Boehm, risk is reduced in outer spirals as the product becomes more polished and developed. Each spiral

- starts with initial requirements and design goals

- ends with the client reviewing the progress, giving feedback thus far and future direction

The basic concepts of spiral model are planned system, risk analysis, system modeling and performance assessment as shown in Figure 2.4. Major applications of spiral model are complex projects in which risk is very high and requirements are not clearly understood.

The problem with this model is that its management cost is high and also it is a complex way of software development. This model requires a lot of documentation at the intermediate stages, which is a tough job. Spiral model is a four stage model as described below:

- **Planning:** This phase emphasizes to understand the underlining concept, objectives, alternatives and constraints. As planning phase is the base of the spiral model so approximately 30% of the project time is invested in it. A slightest negligence can adversely affect the complete process.

- **Risk analysis:** This is most crucial stage of spiral model as the main work actually starts from risk analysis. All potential and probable risks involved in the future are analyzed and then measures are taken to overcome the risks. Alternative attempts are identified and evaluated to resolve the future risks.

- **Customer evaluation and assessment:** The customer evaluates the model. If customer give feedback and want changes, then modifications should be done by developers.

- **Development:** After the completion of risk analysis, the subsequent step is the actual development and the verification.

Table 2.4: Pros and Cons of Spiral SDLC Model

S.No	Pros	Cons
1.	In this model the changing requirements can be accommodated.	Due to the complex development process it becomes very difficult to meet budget and time requirements.
2.	It allows for extensive use of prototypes.	To effectively implement this model rules and protocols should be followed properly but doing so, through-out the span of project is tough.
3.	In this model users see the system early.	It is not suitable for small project with low risk as it is expensive for small projects.
4.	For better risk management in spiral model more risky parts can be developed earlier.	Spiral may go indefinitely and as there are large number of intermediate stages so it requires excessive documentation.

2.2.5 V-Model

The V-model is also named as Verification and Validation model. The execution of the various processes is done in a sequential manner in V-model [66,100]. This model is an extension of the waterfall model. In this model for each corresponding development stage there is association of testing phase. It follows a highly restricted and highly-disciplined approach because the next phase always starts only after completion of the preceding phase. The Figure 2.5 describes the different phases of V-Model of software development life cycle.

- **Business Requirement Analysis:** This is the first phase in the development cycle where all the requirements of the product are understood from the end-user or customer perspective. This phase involves extensive communication with the customer for understanding his expectations and exact requirements. Analysis of business requirements is a very important activity because customers are not sure about what unerringly they need. Planning for acceptance test is also done at this stage.

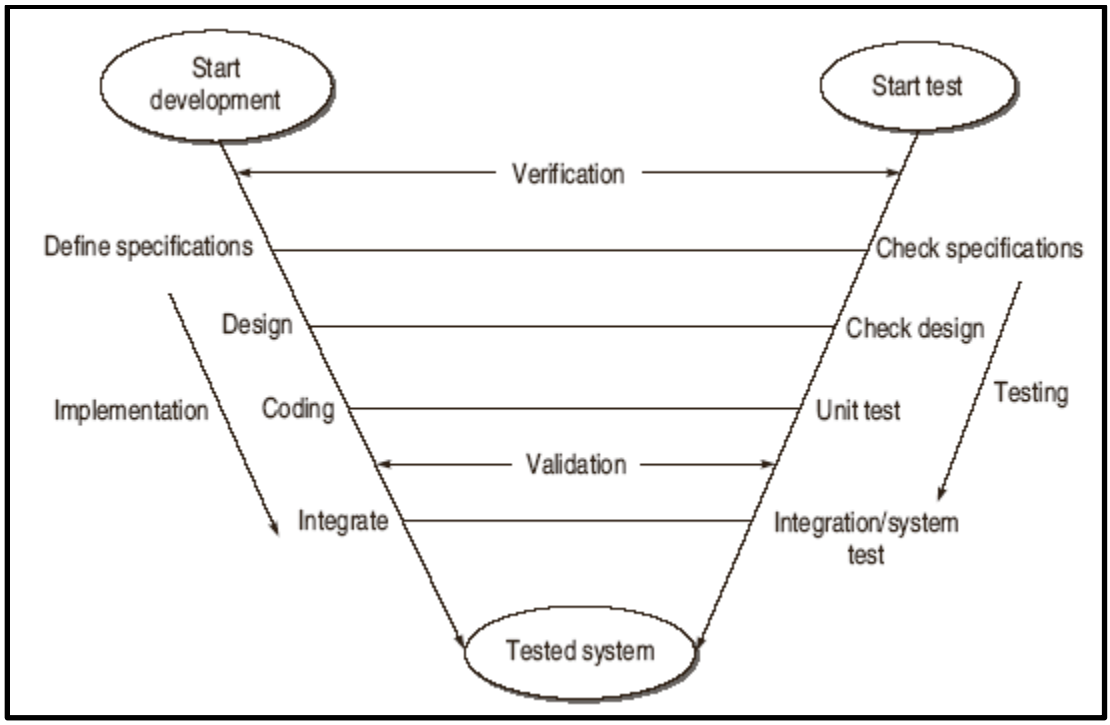


Figure 2.5: V Model

- **System Design:** After requirements are clear the next phase is to design the entire system. System design would encompass understanding the complete hardware and co-ordination or communication setup for the product. Based on the system design, test plan is developed. Performing test planning at the initial stages leaves extra time for execution of test cases.

- **Architectural Design:** During this phase architectural specifications and design documents are produced based on the customer requirements. Generally, more than one approach is planned and depending on the technical and financial feasibility the final decision about design document is taken. The design document is broken down into various small units or modules each having different functionality. The communication and data transfer among the internal units or modules and with the outside world is clearly understood and defined in architecture design.

- **Module Design:** The different modules are clearly designed in this phase. Then each module is named according to the task assigned to the module. It is referred to as Low Level Design (LLD). As unit test removes the faults so for different modules, unit tests are designed at this stage depending upon the design of internal modules.

- **Coding Phase:** The coding of the system is started at this stage. Depending on the requirements of the system, appropriate programming language is decided. Then coding is performed based on the guidelines and standards of that programming language. Previous to releasing the final build, source code goes through frequent code reviews and after that the source code is optimized for most excellent performance.

- **Unit Testing:** The unit tests of the source code are designed in the module design phase and are executed during validation phase. The unit testing eliminates bugs at an early stage and it is the testing at code level. By unit testing all defects cannot be uncovered or removed.

- **Integration Testing:** It is associated with the architectural design phase. In this individual modules are combined and then tested. This testing is performed to check whether the internal nodules within the system communicate with each other appropriately.

- **System Testing:** After all the modules or units of the source code are integrated together, system testing is taken up. System tests check the software and hardware compatibility issues. System testing of software project is testing on a complete and integrated system to validate the system's conformity with its given requirements and objectives. System testing falls under black box testing, as it requires no knowledge of the program code.

- **Acceptance Testing:** Acceptance testing is principally done by the end-user or customer but other stakeholders may also be involved. It involves testing the product in user environment under real conditions. The objective of it is to set up self-assurance in the system. Acceptance tests are done by the customer to uncover the issues regarding compatibility. It also finds out the issues like load and performance problems in the real environment. Table 2.5 shows the pros and cons of V-model.

Table 2.5: Pros and Cons of V- Model

S.No	Pros	Cons
1.	This model is simple and easy to use.	Like waterfall model, the V-model is rigid.
2.	As development is early in the life cycle so there are more chances of success.	To effectively implement this model rules and protocols should be followed properly
3.	It works well for small projects.	Little flexibility.

2.3AGILE SOFTWARE DEVELOPMENT (ASD)

ASD is a collection of software development techniques based upon iterative and incremental development,that gives emphasis to adaptability. In Agile environment the requirements and results or working software develops through association among self-organizing, self-motivating and cross-functional team [7,8,13,34]. It promotes adaptive planning approach, incremental development and delivery, and a time-boxed approach. Agile supports quick and flexible response to change. Agility means to wipe away the heaviness that is present in the traditional software development methodologies and promote fast response to changing requirements. The Agile manifesto [6] introduced the term ‘Agile’ in 2001. The Agile manifesto is discussed below:

- Develop software that satisfies a customer through continuous deliveries of working software and getting feedback from users about it.

- Supports flexibility, accept changes in requirements at any development phase.

- Better cooperation between the developers and the customers and that too on the daily basis throughout project development process.

- Supports development on a test-driven basis, which means writing test prior to writing code.

2.3.1 Principles of Agile Manifesto

The values described above are realized in the principles of Agile manifestos. The principles are the following:

- Customer approval and satisfaction by quick delivery of working software is of uppermost priority.

- Welcome changing requirements, even behind schedule in development.

- Working software is delivered frequently in short release cycle of weeks rather than months.

- The most vital measure of ASD is working software.

- Agile encourage sustainable development, and users should be able to maintain a constant speed indefinitely.

- Daily meetings between customers, business analysts and developers and face-to-face conversation between all the developers.

- Agile projects are built around motivated team members.

- Continuous concentration to technical superiority and good design.

- The success of the project come-outs from self-organizing and self-motivating teams and this success is due to the simplicity and straightforwardness of Agile processes.

- Regular adaptation of team to changing requirements and business needs. The team reflects on how to turn out to be more productive, then adjusts its behavior accordingly.

2.3.2 Definition of Agility

Agility [15,49] is defined as the primary characteristics for business competitiveness which aims at increasing organizational flexibility in new circumstances and opportunities. Different researchers defined agility in different ways. The Table 2.6 below summarizes some definition of agility.

Table 2.6: Definition of Agility

Gunasekaran	In response to growing customer-designed products and services demands, agility refers to the capability for an organization to survive and prosper in a competitive environment, which is signified by unpredictable and continuous changes, by reacting quickly and effectively to changing markets.
Katayama & Bennett	Agility is the ability to satisfy volatile demand and various customer requirements in an economically viable and timely manner.
Sharifi & Zhang	Agility is the ability to master unexpected changes and to take advantage of changes as opportunities.
James	Ability to master change, uncertainty and unpredictability regardless of its source, i.e. customers, competitors, new technologies, suppliers or governmental policies
Ericksson et al.	Agility means to strip away as much of the immensity and heaviness, connected with the traditional software-development methods as possible to promote rapid response to changing user requirements, accelerated project deadlines etc.

2.4 AGILE PROCESS

In order to understand Agile project management [2,55] it is necessary to understand the Agile development process[51,52]. The various Agile processes are shown in Figure 2.7.

- The Agile model put emphasis on the fact that whole Agile team should be a tightly integrated unit and is composed of developers, quality assurance members, testers, project owner and the customer. The key process of Agile is effective communication between all team members. For valuable communication [59] and information exchange daily meetings are held in ASD.

- An important Agile process is iterative delivery. A delivery cycle or an iteration in ASD ranges from one week to four weeks. The delivery cycle are also known as sprints if scrum methodology of Agile is followed [44,45,46].

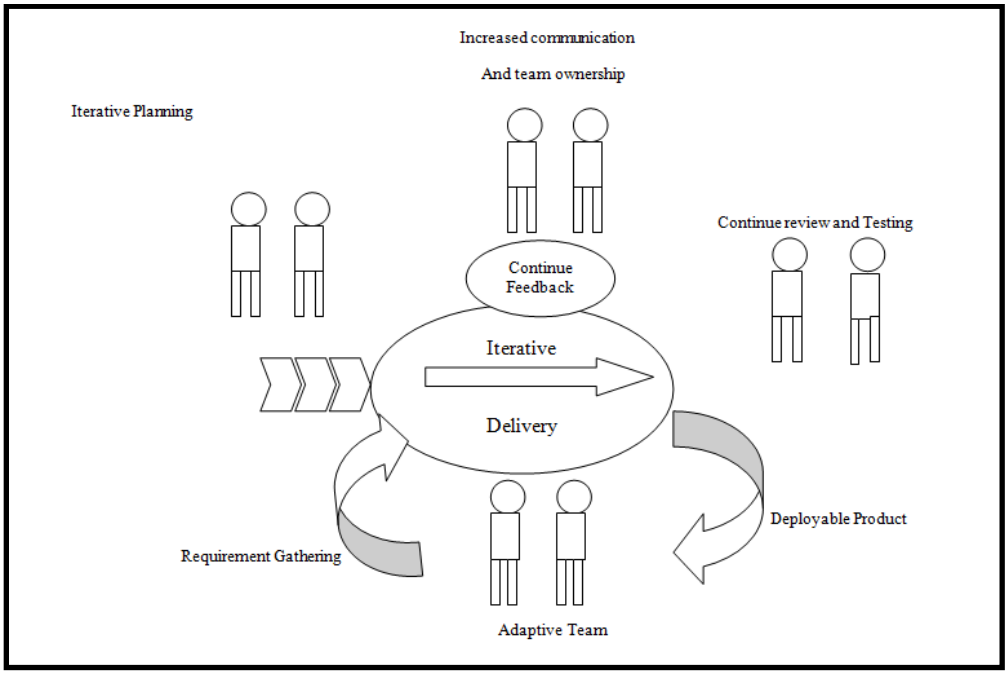


Figure 2.6: Agile Processes

- ASD teams follow various tools for open communication between the team members. These techniques and tools facilitate the team members (including the

end-users) to express their views and feedback. These comments are used while implementing the release of the software project.

2.5 AGILE LIFE CYCLE

Agile methods break tasks into small increments with minimal planning. Iterations typically last from one to four weeks and each iteration is worked on by a team of designer ,tester, sprint master etc. and is developed using a full software development cycle including initial planning, requirements gathering and analysis, architectural and system design, coding, unit, integration, system and acceptance testing. This allows the project to accept changes even in later stages of development and also helps in minimizing the risk. Excessive documentation is not needed, it can be done if required [19]. Agile SDLC consists of six phases i.e. pre-project planning phase, begin phase, construction phase, final release phase and production phase and retirement as shown in Figure 2.7. These phases are described in detail as below:

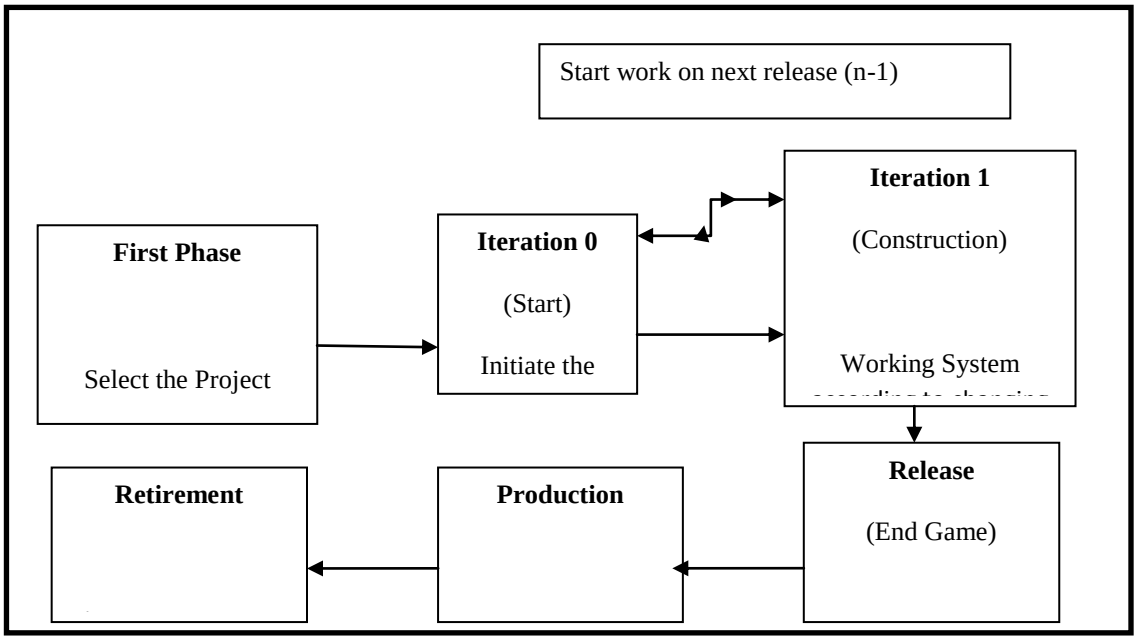


Figure 2.7 :Agile Life Cycle

2.5.1 Pre-Project Planning

The first phase of Agile life cycle is pre-project planning. The various activities performed in pre-project planning phase are:

- **Defining goal and objective of project**: Firstly the goals and objectives of Agile project are defined. This phase investigates how organization's presence in the market get better with the new functionality. This phase also recognize the potential stakeholders of the project.

- **Select best strategy for the project**: The various issues that need to be considered while selecting the best strategy are i.e. the current team members are capable to grip the project or there is a need to increase the size of the team, whether there is requirement of relocation of team members. Which software life cycle model like traditional heavy weight models like waterfall, spiral or prototype model or iterative light weight model like Agile will be good for the project.

- **Feasibility analysis**: The main focus of feasibility study of the project is to determine whether the project is technically or financially feasible.It is defined as the realistic extent to which a project can be executed successfully. The objective of it is to found the reasons for developing the software project that are admitable to users and conformable to standards. At this stage various high level decision about solution strategy are undertaken.

2.5.2. Iteration 0: Project Initiation Phase

The project initiation phase is also called as second phase. It is the first week of an ASD project. The various actions performed during this phase are:

- **Requirement gathering**: Requirement gathering and modeling is done in this stage. Active and full participation of the all the stakeholders is required for gathering the requirements. The main goal of this phase is to understand the problem and solution domain. Very little documentation is done at this stage. The requirements in detail are modeled in brain-storming sessions of stakeholders.

- **Building team according to the need of project**: When the development of a project is started, the team allocation and building is also started in parallel. The team members are identified. ASD team consists of at least two to five developers, the project coach, manager, team lead and a representative of the customer.

- **To construct initial architecture for the project:** This phase doesn't need to write excessive documentation. The only goal of this phase is to identify an architectural strategy. The testers, developers and managers of the project decide a basic architecture for the system. After this phase the team has a general idea of what the system is going to build and how the work will be done.

- **Setting up the environment**: The basic environment setup is done at this stage. There are some basic things which are needed to develop a project like workstations, development tools, work area for the team. Most of these are needed at the start of the project.

- **Estimating the project**: An initial estimation of time, cost and effort is produced based on the initial requirements, the initial high level design, and the proficiency of the team [20]. This estimation is evolved throughout the development of project. Moreover, iteration plan phase possesses iterative estimation activity to estimate size, cost and duration of the project. It also re-estimates efforts depending on team velocity.

2.5.3. Construction Iterations

During this phase, high-quality working software is delivered to the customer incrementally. The working software also meets the changing needs of the customer. The various steps are as below:

- **Communication between stakeholders**: The effective co-ordination and communication between various stakeholders is needed for reducing risk.

- **Implementing functionality by prioritizing requirements**: In ASD the requirements in the form of user-stories are prioritized and a product backlog is maintained.

- **Analyzing and designing:** Every individual requirement or user-story is analyzed before the implementation. The unit testing is performed for every developed requirement. For testing test-driven development [17,18] approach is followed which means that testing is done during all the stages ASD.

- **Ensuring quality delivery**: Quality of the product is ensured by software quality assurance (SQA) group. The SQA group applies various SQA techniques for selecting the best design of the project.

- **Continuous delivery of the working software in every iteration**: After each development cycle or iteration a partial working software is delivered to customer.

- **Final testing (Confirmatory and Investigative testing)**: As Agile follows TDD so a significant amount of testing is required. Final testing is of two types confirmatory and investigative.

Confirmatory testing is described as “the testing against the specification and requirements" it confirms that the software project will work according to the stakeholder requirements. Investigative testing is done by senior testing team. This team find the defects which are missed by developers.

2.5.4. Release Iterations: The "End Game"

This phase, also known as the "end game" phase. In this phase the working software system is transited into production. The various actions performed in this stage are as below:

- **Final testing of the system**: Although the majority of testing is done during construction phase of the Agile life cycle, but final system and acceptance testing are done in this phase. Beta testing of the system can be performed in presence of end users or customers.

- **Finalizing the system and documentation manual**: If the stakeholders are willing to invest in documentation, then a document manual of the project is written.

- **Training:** Proper instructions and training is provided to customers, operations staff, and support staff for working effectively and efficiently with working system.

- **Deploy the system**: The system is installed and deployed after this phase. Deployment of the system includes various operations to prepare a system for transferring it to the customer site. Various versions of the deployed software may be installed in a test environment at the customer site.

2.5.5. Production

The goal of this phase is to keep software system productive and useful after the product is finalized and deployed. The fundamental goal of this phase is to keep the system in running form. This process is applied differently in different organizations.

2.5.6. Retirement

The retirement phase is also known as sun-setting phase of the system or system decommissioning. The goal of it is the removal of a system release and occasionally the complete system from production. The retirement of the system is a serious issue faced by many software industries as legacy systems are detached and replaced by new systems.

2.6 WORKING WITH TRADITIONAL AND ASD METHODS

Traditional methodologies are heavy-weight processes. In these a schedule is created at the beginning of a project and it is needed to conform to this schedule for the life of the project. The reason for failure of the traditional projects is that the users keep changing their minds and changing requirements are not welcomed.

Agile software development methods solve these problems by incorporating changes even in later stages [4,5]. In the case of a normal heavy weight traditional software management & development process, the different activities and tasks are completed in an orderly sequence as below:

- Meet with the end-user or customers and find out the processes required.Get the requirements to be signed-off to guarantee that the customer not changes their minds.

- Generate a project plan in detail for the entire project, and assign the various resources to tasks. After the planning phase, designing phase is started. When project progresses, different individuals working on different pieces may contact customer on different issues.

- Testing is done at the end. When the project is completed then customer may request many modifications. Project becomes a manufacturing project rather than a development project.

As Agile is highly dynamic in nature. It welcomes requirements change during the development process. The team can regularly adapt changing requirements and business needs. So, in the case of working with Agile software development, the various activities are done in the following manner:

- Meet with the customer; create a high level list of features get sign-off on the requirements. Ask the customer for prioritizing the list of requirements. Create a product backlog.

- Select the items for the first iteration. Implement the plan for the first iteration. Deliver the working software to the customer and get feedback from customer.

- After first iteration is delivered, and approved by the customer successfully, then the time the first iteration took to deliver working software is used as a baseline to predict the size of future iterations. The various steps followed in ASD are as shown in Figure 2.8.

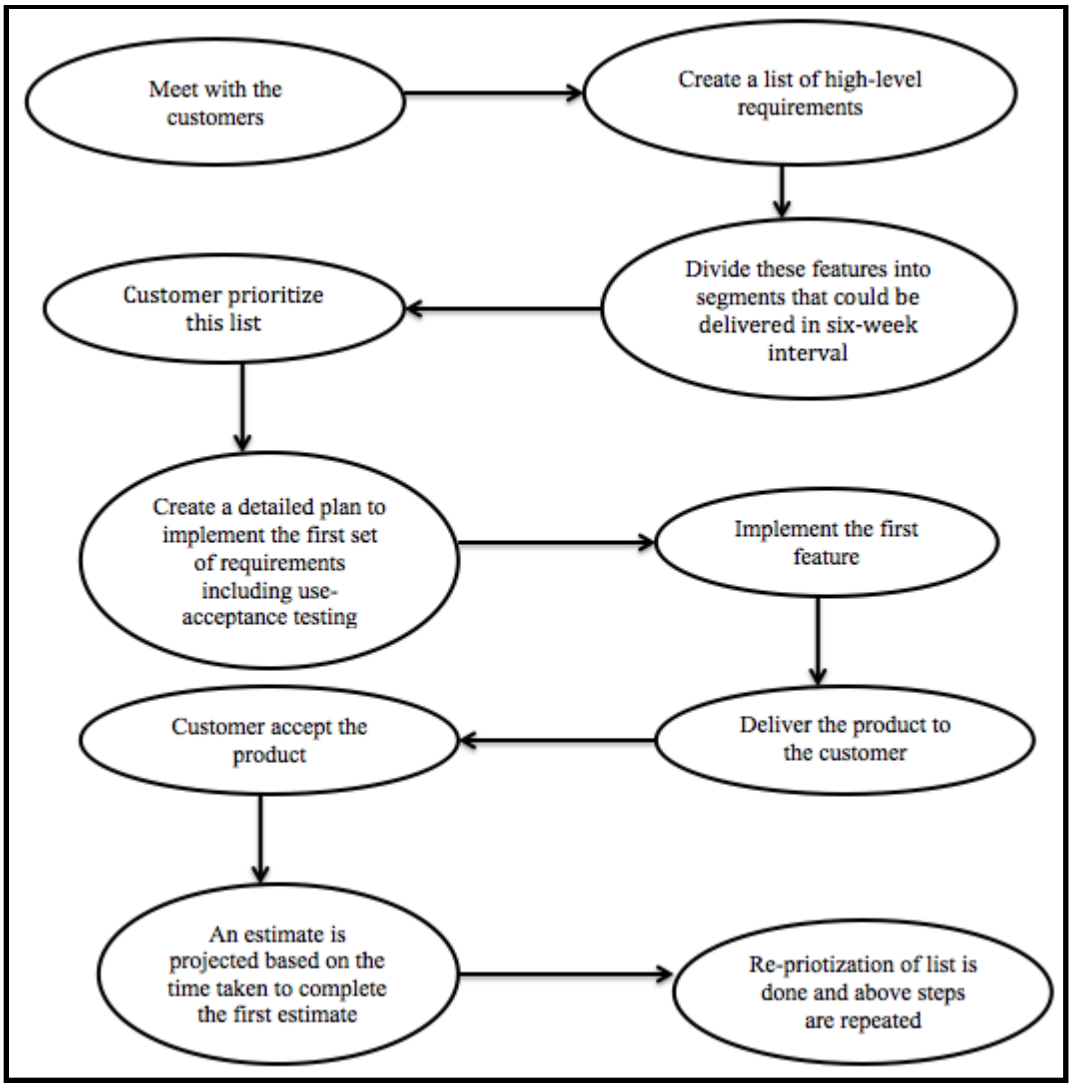


Figure 2.8: Steps followed in ASD

The traditional methodologies aim at making software development more predictable and more efficient. They do not support changes of requirements and the complete system has to be known at the beginning of the project. Thus these are also called as heavyweight methodologies. These methodologies are well documented and thus are quite complex to apply. In traditional methods project objectives are clear and progress of system is measurable. Table 2.7 summarizes the difference between traditional and Agile perspective.

Table 2.7: Difference between Traditional and Agile Perspective

Task	Traditional Perspective	Agile Perspective
1. Design Process	Linear sequence of steps	Iterative and exploratory
2.Type of environment	Stable, predictable	Turbulent, difficult to predict
3.Goal	Optimization, low tolerance to changes	Flexibility, High tolerance to changes
4.Development Principle	Development is based on fixed order	Principle of freer co-operation of development team
5.Problem-solving Process	Selection of best means to a given and through well planned and formalized activities	Learning through experimentation and introspection
6.Type of learning	Single-loop/adaptive	Double-loop/generative
7.Type of management	Directive management	Emphasis on team communication
8.Team-size	Large teams	Small teams(2-10 developers)
9.Customer role	Customer role is only at the initial and final stages of the project	Customer is involved at each and every stage
10.Testing	Testing is done at the end of development	Testing is done throughout the course of development

2.7 ASD METHODS

ASD methods constitute a set of practices for software development that have been created by experienced practitioners [25,32,50,60]. These methods can be seen as a reaction to plan-based or traditional methods. The various methods of ASD are crystal methodologies, dynamic software development method, feature driven development, lean software development, scrum and extreme programming. All of these methods focus on best practices of ASD. These are described in Table 2.8.

Table 2.8: Description of ASD Methods

1.	Crystal methodologies	Crystal methods are family of methodologies discovered by Cockburn in 1998. It includes a number of methodologies and best methodology is selected for a given project. The crystal approach also includes the principles for alternating the methodologies according to the circumstances of different project. Crystal methods are lightweight methodologies. Each of the crystal family member is marked with a color shade indicating the heaviness of the method which means darker the color heavy the methodology is. This model suggests choosing the appropriate color depending on the size and criticality of the project. The word crystal in this methodology refers to the degree of hardness and various colors in the same as crystal comes in various colors and different hardness. The more value of hardness means that more documentation is required and darker color describes the heaviness of the project.
2.	Dynamic software development method [31,107]	Dynamic software development method (DSDM) is an iterative and incremental Agile approach that uses the features of Agile including the involvement of customer at every step. The basic idea of DSDM is to fix the functionality of the product and then adjusting the time and resources to meet this functionality. DSDM consists of five phases: feasibility study, business and economic study, functional iteration, designing and building and implementation. The feasibility study, business and economic study are sequential phases and these are done only one time while the next three phases are iterative and incremental. All the iterations are time-bounded and last for a predefined amount of time. The time given to a particular iteration is decided beforehand along with the results that it will produce. Usually the duration is between few days to few weeks. Nine principles underlie DSDM: user participation, authorizing the project team, frequent and normal delivery, addressing, modern business needs, iterative and incremental development, consent for any change in requirement, high-level scope being preset before starting the project, test driven development[47], and effective communication.
3.	Feature-driven development[26]	Feature driven development is an iterative and incremental Agile and adaptive approach developed by Jeff De Luca and is used for developing systems and this approach does not cover the entire software process but rather focuses on the design and development project[73,74]. The FDD approach expresses iterative development with the best practices found to be effective in industry. In FDD quality aspect is very important. It

		includes frequent deliveries along with the perfect monitoring of the project progress. Combines model-driven and Agile development with emphasis on initial object model. A feature iteration consists of two phases: design and development.
4.	Lean software development	It is an adaptation of principles from lean production and the Toyota production system to Agile software development. It consists of seven principles: eradicate waste, magnify and amplify learning process, make a decision as late as possible, frequent delivery, empowering team, building integrity and reliability.
5.	Scrum	Focuses on project management in circumstances where it is difficult to plan in advance and where to consider feedback of customer is utmost important[70,71]. The working software is developed by a self-organizing and self-motivating team in iterations of 3-4 weeks called sprints. Each sprint starts with planning phase and ends with a review. All the features to be implemented in a particular iteration are registered in a backlog. All the team members communicate and coordinate with each other in daily stand-up meetings. Scrum master, solves the problems that prevent the team members to work effectively.
6.	Extreme programming[24,27,28]	The Extreme programming (XP) focuses on best practices for software development. It consists of twelve practices: the preparation and planning game, small and frequent releases, metaphor, simple and easy design, test driven, refactoring, approach of pair programming[15,65,75], combined ownership, 40-42 hr week, continuous integration, customers involvement, and pre-defined coding standards. In XP coding is done in pairs on one workstation, and pairs are changed continuously. The code should be collectively owned, and each programmer is allowed to change the code, one programmer at a time[35,36,40,72]. The code is refactored continuously to improve its quality and to make it as simple as possible without making any changes into its functionality or its features.

2.8 SCRUM: AN AGILE FRAMEWORK

Scrum is an iterative and incremental Agile software development method that was developed by Jeff Sutherland and his development team in early 1990s and is used for building software products [61,62,63,64]. Scrum does not define any specific

development techniques; instead it concentrates on how a team member should perform his/her job in order to meet flexibility in a constantly changing environment. Scrum improves the existing engineering practices. The life cycle of scrum is shown in Figure 2.9.

Its principles are consistent and are used to guide development activities. The scrum incorporates the following framework activities: requirements analysis, designing, evolution, testing and delivery. Within each framework activity, work tasks occur in a specific pattern called sprint. The number of sprints required in a project depends on the size and complexity.

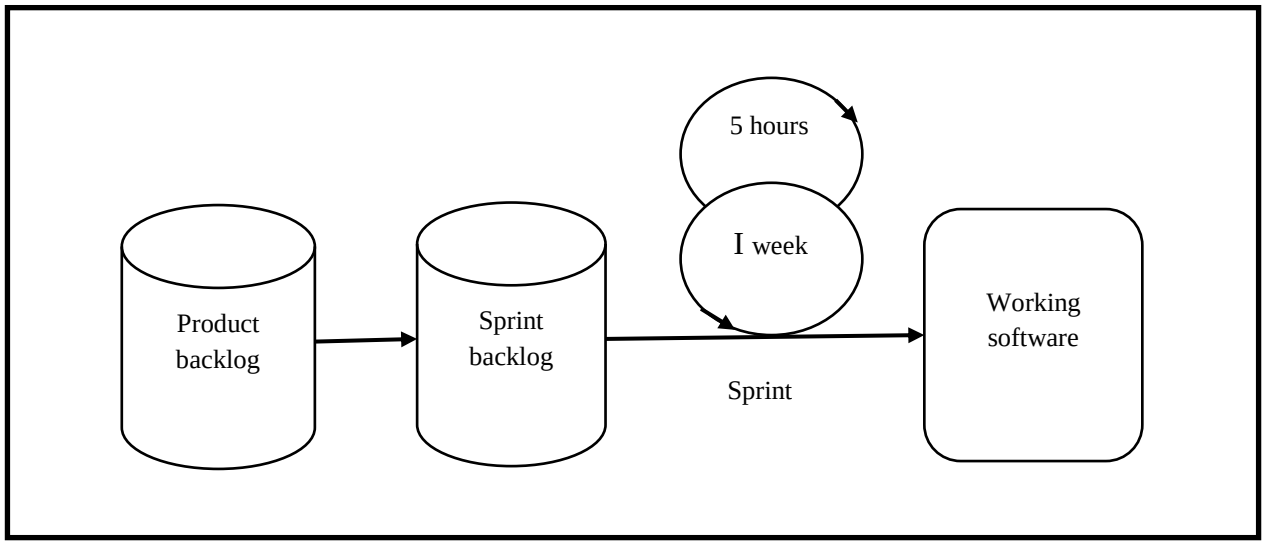


Figure 2.9: Scrum Life Cycle

2.8.1 SCRUM PHASES

Scrum process includes three phases: pre-game, development and post-game.

2.8.1.1 Pre-game Phase

The pre-game phase consists of two sub-phases: planning and high level design. In planning phase, the definition of system is developed. A product backlog list is created

that consists of the requirements. The requirements can be originated from the customers, customer support, sales and marketing division. After getting all the requirements, prioritization is done and the effort is estimated. The new items are also added to product backlog as and when needed. At every iteration, the scrum teams review the updated product backlog so as to gain commitments for next iteration. In architecture phase a high level design of system is planned depending upon the current items present in the product backlog. In case of changes in the existing system, the changes needed are identified and also the problems that may arise while implementing these changes. A design review meeting is conducted to discuss the proposals and decisions are made accordingly.

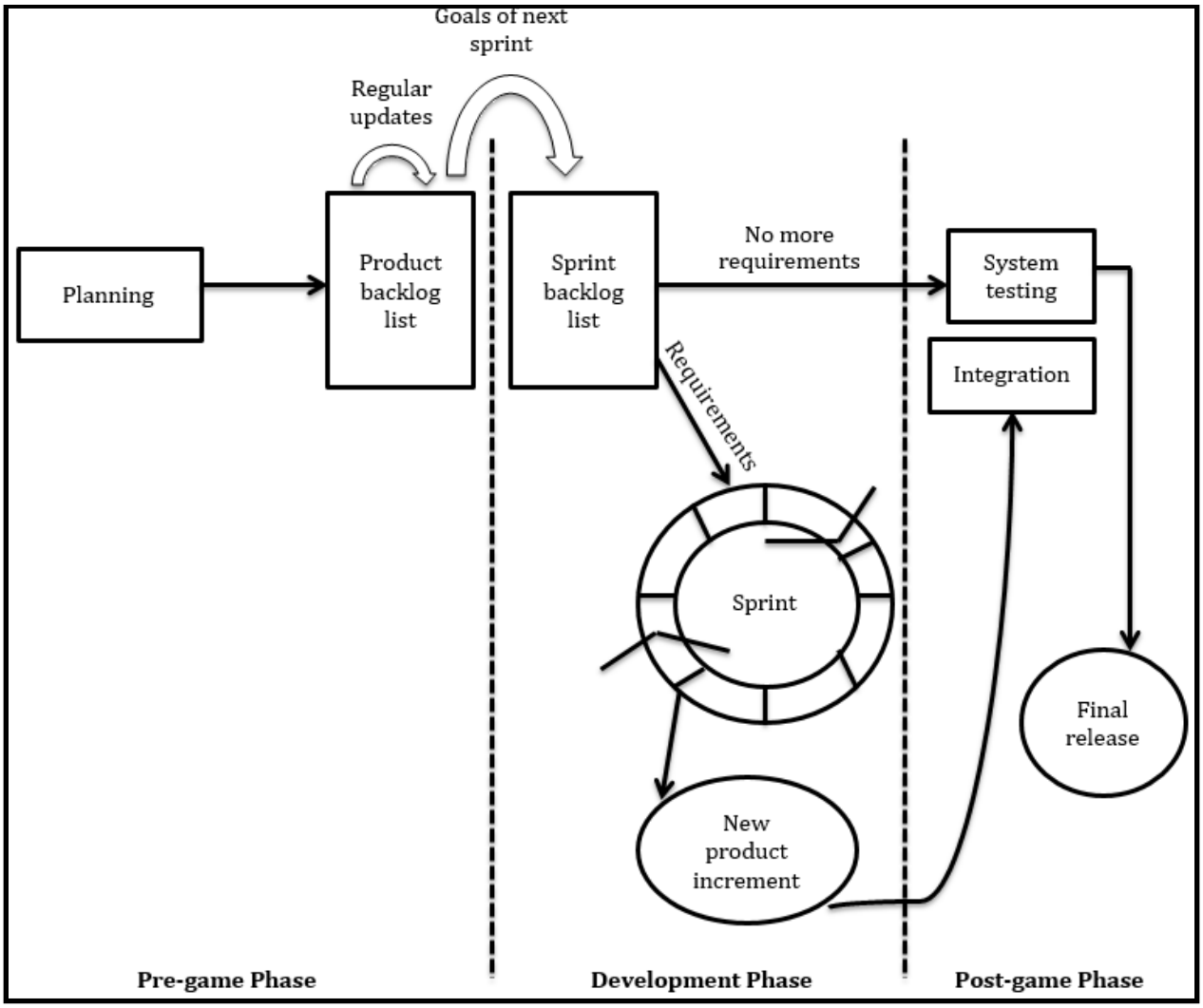


Figure 2.10: Phases of Scrum

2.8.1.2 Development Phase

The development phase also called the game phase is the phase that is Agile part of the Scrum. This is treated as the black box in which unpredictable changes happens. In development phase the system is actually developed in Sprints, which is an iterative cycle where functionalities are developed or enhanced to produce new sprints. All the traditional phases of the software development life cycle like requirement analysis, design, evolution and delivery are present in the sprints. The different variables are identified in the Scrum that change their value, are controlled through various Scrum practices during the sprints. The time to develop one sprint is usually between one week and one month.

2.8.1.3 Post-game Phase

Post-game phase includes the release of the project. In this phase the system is ready to be released and preparation for its release is done such as integration testing, system testing and documentation manual.

2.8.2 The Scrum Teams and Associated Roles

The scrum consists of the scrum team and their roles, time-boxes, artifacts and the rule. Scrum is designed to optimize flexibility and productivity and they work in iteration. The Scrum team consists of three roles 1) Scrum Master 2) Product Owner 3) The Team. These together form the Scrum Team.

2.8.2.1 Product Owner

The responsibilities of product owner include identifying product features, converting them into a prioritized list. The product owner also decides which item should prioritized first in the product backlog. He or She is responsible for managing, controlling and making visible the product backlog list. The scrum master, customer and the management

select the product owner. The product owner is responsible for profit and loss of the project. The product owner takes the final decisions of the task related to the product backlog and also participates in estimating the effort of the backlog items.

2.8.2.2 Scrum Master

The scrum master helps the team learn and apply scrum to develop project. The scrum master interacts with the project team as well as with the customer and the management during the project. The scrum master is not the manager of the team; instead it educates and guides the team and the product owner to carry out the project according to the practices, values and rules of the scrum. It is the duty of scrum master to resolve the threats arising. There should be a full-time scrum master but sometimes a team member plays this role.

2.8.2.3 The Team

The team is the project team that builds the product that the project owner indicates. The team in the scrum has two main features: first it is cross functional which means that it includes all type of expertise necessary to deliver the shippable project in each sprint and the second is that the team should be self-organizing which means that the team is self-managing. The team decides what to commit in a sprint and accordingly decides how to fulfill the commitment. The team in scrum should have around seven people plus or minus two people and the team should include people skilled in analysis, development, testing, architectural design, database design, documentation, etc. The stable team helps achieve higher productivity so the team members should be kept fixed. Although the team may be changed after every sprint but by this the productivity of the team is diminished, so care should be taken while changing the team composition. Teams are also called as feature teams because one team does all the work i.e. requirement analysis, planning designing, unit and integration testing etc.

2.8.3 MEETINGS IN SCRUM

The important element of the scrum is a sprint and all the sprints in a project follows same framework and next sprint starts immediately after the completion of current working sprint. Scrum enforces effective communication and co-ordination. So various types of meetings in scrum are needed which are as below:

2.8.3.1 Sprint and Sprint Planning Meeting

Sprint is the basic development unit in scrum and is a time-boxed, which means that the duration of a sprint is fixed. The time period of sprint is never extended even if the requirements are not completed. The duration is decided before the start of the sprint and is usually between 2 to 4 weeks. Each sprint starts with a planning meeting in which goals of sprint are decided and estimates of the goals of sprint are made. A review meeting in which progress is reviewed and goals for the next sprint are discussed follows the sprint completion.

Sprint planning meeting: Every sprint starts with the sprint planning meeting that is of approximately 8 hours for a 1-month sprint. For the smaller sprint the time period of this meeting should be approximately 5% of the total sprint length. The meeting is divided into two distinct meetings.

Sprint planning meeting part one: It takes place for first four hours. In this meeting, the product owner, the customer and the scrum team decide upon the goals and objectives of the next sprint. The input to the part one meeting is product backlog, the performance of the team in previous projects, the latest increment of product and the capacity of the team.

In this meeting, the product owner and the team, review the high priority items present in the backlog that the product owner wants to implement in this sprint. The goals and the context of these high priority items are discussed so that the team gets an idea of thinking

of the product owner. After the selection of the product list items, a sprint goal is established to meet the desired objectives. The team keeps in mind the sprint goal, in order to satisfy this goal, it implements the sprint. If the sprint turns out to be tougher than the team expects, then the team collaborates with product owner and then implement that sprint partially. The product owner and the team also reviews the definition of done which means the product is fully coded, reviewed, implemented, tested, integrated and documented. This part one meeting basically focuses on what the product owner wants to implement.

Sprint planning meeting part two: The sprint planning meeting part two takes place in between scrum master and the team and it focuses on how the implementation should be done so that the goals can be achieved. In the next four hours the team decides how they will convert the product backlog items selected in the first meeting into a complete increment. Agile team usually starts with the design and identifies the tasks. These tasks are the detailed work that should be done to achieve working software. The team will decide how many sprint backlog items will be done in a single sprint rather than the product owner assigning them to do the task. This makes a reliable commitment because the team is deciding the task that they will complete on the basis of their planning.

The sprint planning part two usually begins with the estimation of how much time each member has for sprint related work and by this the capacity of the team is determined. Once the capacity of the team is decided, the product owner also decides how many items the team can complete in the given sprint and how they will complete the task. Sprint backlog is the work that the team identifies as necessary to meet the one sprint goal. As the team divides each item into individual tasks, it may find that the more or fewer tasks are needed or the given task may take less or more time than it was expected to take. When tasks are found unnecessary, they are removed from the backlog. Only the team can change the sprint backlog during a sprint.

2.8.3.2 Daily Scrum Meeting

Each team meets on every workday for a 15-minute status meeting called the daily scrum meeting and it usually happens at the same time and same place throughout the sprints. The goal of daily scrum meeting is to improve communications, improve knowledge, identify and remove impediments to development and promote quick decision-making. It is the duty of the scrum master to ensure that the scrum team should conduct the daily scrum meeting. The scrum master guides the team to keep the meeting time as 15-20 minutes. It is not a status meeting and is not for anyone but for the people who are responsible for that particular sprint. It is just a check of the progress of sprint towards the required goal. In this meeting each team member reports three things:

- Tasks they have completed since last meeting.
- Task they are likely to complete before the next meeting.
- The problems that are coming till now.

A team member is responsible to keep in account all the blocks and it is the duty of the scrum master to help resolve the problems. No discussion is held during the daily scrum, only reporting answers to these questions are done. If there is a need for discussion then immediately after the daily scrum a follow-up meeting is conducted. Although it is mandatory for all the team members to attend the daily scrum, but to attend the follow-up meeting is optional. This follow-up meeting is a common event in which the team can change to the information that they heard in the daily scrum. It is recommended that managers and the other persons in similar position should not attend the daily scrum because it risks in making the team feel monitored and the team might avoid reporting problems that they are facing.

2.8.3.3 Sprint Review Meeting

At the end of the sprints, a four-hour time-boxed meeting is held for one month sprint. This meeting is termed as sprint review meeting and for the sprints having duration less than one month, 5% of the total sprint time is kept for sprint review meeting. This meeting is often confused with demo. Sprint review meeting is inspect and adapt process for the product and here the team and the product owner reviews the product. The most important feature of the review is the conversation that takes place between the product owner and the team to learn new things, to take advice and to resolve problems. The meeting also includes a demo of what the team has built but the main focus of the meeting should be conversation rather than the demo of the product.

The role of the scrum master is to make sure that all the group members are aware of the definition of done. He prevents the team from discussing the items that are not done and he also makes sure that all such items go to the product backlog and their re-prioritization is done. This maintains a transparency of the quality of product, as the team cannot fake the quality by presenting the software that appears to work well.

The meeting includes the following elements:

- The scrum master identifies which backlog items have been done and which hasn't been done.

- The team also discuss about what it has completed during the sprint and what problems it faced, and how team will solve these problems. The team members also shows the work that is done.

The product owner, team members, scrum master along with the customers, stakeholders, experts and any other person who is interested, attends the sprint review meeting. After the sprint review meeting has completed, new items may be added on the product backlog and it may change the direction of the system being build.

2.8.3.4 Sprint Retrospective Meeting

In between the sprint review meeting and the next sprint planning meeting, there is a meeting called as sprint retrospective [21,22]. It is a three-hour meeting in which the scrum master encourages the team to change their framework, practices and development process so as to make the next sprint more effective. The sprint review meeting involves inspect and adapt regarding the product whereas the sprint retrospective involves inspect and adapt regarding the process. The team and the scrum master are required to attend the meeting but the product owner may or may not attend the meeting. This meeting provides an opportunity to team members to discuss about what's working and what's not working. The principle of retrospective meeting is to check how the previous sprints went with respect to procedures, processes and tools. This identifies the major items that went well and prioritizes them. It also identifies those items that could have done better.

There are several techniques used in the sprint retrospective meeting and one such method involves drawing two columns on a whiteboard and label them as what's working well and what's not working well and all the members add one or more items to this list. Then the team decides to make small amount of changes to try in the future sprints with the commitment to review the result in the next sprint retrospective meeting. At the end of this meeting the team labels items listed in the two columns in three ways:

- It is labelled as 'C' if it is caused by scrum
- It is labelled as 'E' if it is exposed by scrum
- It is labelled as 'U' if it if unrelated to scrum.

There will be lot of C's on what's working well part whereas there will be a lot of E's on the other one. By the end of this meeting the Team identifies the improvement it has to do for the next Sprints.

2.8.3.5 Sprint Release Planning Meeting

During a sprint development cycle, there are two main discussions: New product in first release, an existing or presented product in a later release. In case of a new product or an existing product, there is a need to do initial product backlog refinement before the first scrum where the product owner and the team shape a proper product backlog. This takes a few days or a week and involves detailed requirement analysis and estimation of all items identified for the first release. The purpose of release planning is to establish a plan that the scrum team can understand. The release plan establishes the objective of the current release, prioritized product backlog, major risks, and overall functionality that the release will contain. The organization can then inspect progress and make changes to this release plan on a sprint-by-sprint basis.

The product owner and the team continuously do the product backlog refinement in every sprint so as to prepare for the future. During the initial product backlog refinement and during the continuous backlog refinement, the team and the product owner do release planning, refining the estimates. Working software is built iteratively using scrum methodology [67]. Each product of the sprint is a potentially shippable portion, when sufficient increments have been created for the product and the customer is satisfied with it then the product is released.

2.9 ARTIFACTS OF SCRUM

In Scrum there are four artifacts: Product backlog, Sprint backlog, Sprint burndown and Release burndown.

2.9.1 Product Backlog

The first step in the scrum is for the product owner to express the vision of the product. The product backlog describes the refined and prioritized list of user-stories. It defines everything that is needed in the final product based on the present knowledge and also

defines the work to be done in the project. It contains the prioritized and updated list of features of a process that is currently being built or enhanced. Multiple actors like customer, customer support and project team, etc. can generate the items in the backlog. Only a single product backlog exists and it includes items such as features, functions, technology upgrades, errors, bugs fixes, etc. Scrum includes the task of creating a product backlog and maintaining it consistently during the product development by adding, removing, updating and prioritizing the requirements in the product backlog. In scrum project the product owner is accountable for maintaining the product backlog.

Team tracks how much work it can do in a sprint and with this information a release date of the project is estimated. The items in the product backlog vary significantly in size; due to this large items are broken into smaller items during the sprint planning meeting whereas the smaller ones are combined together. The product backlog items for the upcoming several sprints should be small enough so that the team can easily understand them.

Product backlog refinement : The valuable guideline in scrum is that five to ten percent of each sprint time must be dedicated to the team for grooming the product backlog items which includes detailed requirement analysis, splitting large requirements into smaller, then estimation of new requirements and re-estimation of existing requirements. A technique that is used for product backlog refinement is that a workshop should be held at the end of each sprint so that the team and the product owner do their work without any interference. This activity is not for the backlog items selected for the current sprints but for the items that will be used in the next one or two sprints. The sprint planning meeting is simple because the product owner starts the meeting with a set of well-analyzed items. If this meeting does not happen, it indicates that the sprint planning meeting will involve significant questions and confusions.

2.9.2 Sprint Backlog

The sprint backlog composed of all the necessary tasks for a particular sprint.The team in the scrum is self-managing. On each day every team member update his estimate of the amount of time remaining to complete his current task in the sprint backlog. After this some other team member adds up the number of hours remaining for the team as a whole.

2.9.3 Sprint Burn down Chart

Sprint burn down graph is the amount of sprint backlog work remaining in a sprint across time in the sprint. Every day this graph shows the estimate of the work remaining until the team's tasks are finished. This is a downward sloping graph and it should be zero on the last day of the sprint. Hence it is called the burn down chart. This graph shows the progress of the team by showing them how much time is left to achieve their goal. If the burn down chart is not sloping downwards towards the end of the sprint then the team needs to adjust so as to reduce the scope of the work and to find a way to work more efficiently.

This graph is created by determining how much work remains by adding the backlog estimates every day. The amount of work left over for a sprint is the sum of the work remaining for all of sprint backlog and keeping track of these sums by day.

2.9.4 Release Backlog and Burn down chart

The part of the product backlog which is planned for the current release is known as release backlog. The primary focus of the product owner is release backlog. The team provides the product owner with the estimates of the effort that is required for each item present on the product backlog. The product owner assigns a business value estimate for each item and it is done with the help of scrum master. On the basis of two estimates namely effort and value, the product owner prioritizes the backlog so as to maximize the return on investment [14]. This effort and the value estimates may be refreshed after each

sprint. Scrum does not have any specific techniques for prioritizing items in the list instead it has a common technique in which estimation is done using a unit of story points. By the end of sprint retrospective meeting, some items from the product backlog must have been finished, some new items are added, some items have revised estimates and some items are dropped from the product backlog list. The product owner ensures that all these changed are updated in the release backlog. In addition to the sprint burn down chart, sprint also includes a release burn down chart. This chart shows the progress towards the release date. It is similar to the sprint burn down chart but it is made for the higher level of requirements rather than the fine-grained tasks.

2.9.5 End of Sprint

The most important feature of the scrum is that it must finish on the allotted date regardless of whether work has been completed or not. A team typically over-commits in its first few sprints and fails to accomplish its commitments. Later it overcompensates and finishes early. By the third or fourth sprint a team figures out what they are capable of delivering and they will then meet their sprint goals more reliably. Teams are encouraged to pick one duration for the sprint period and do not change it.

2.9.6 Release Sprint

The vision of scrum is that at the end of every sprint there is a shippable product that can be delivered to the customer. At this point no work such as testing, documentation is required. This implies that everything is finished and after the sprint review the product can be deployed. This means that each increment is a part of the final product and gives an idea to the customer of where he is after every sprint. However in many organizations due to lack of proper tools this vision cannot be achieved, so in this case some work will be remaining work such as integration testing, and for this release sprint is required. It handles the remaining work. The need for sprint release shows weakness of the team. The sprints continue until the product owner decides that the product is ready to be released then there will be a release sprint to launch the product.

2.9.7 Starting the Next Sprint

After the sprint review, the product owner updates the product backlog. At this point the team and the scrum master are ready to start the next sprint. There is no gap between two sprints. As soon as the sprint retrospective meeting of one sprint stops, the other day sprint planning meeting is conducted for the next sprint.

2.9.8 Test Driven Development In Scrum

Test Driven Development [47] is an important practice in scrum which combines test-first development where the team writes a test first, and then just enough code to fulfill that test.

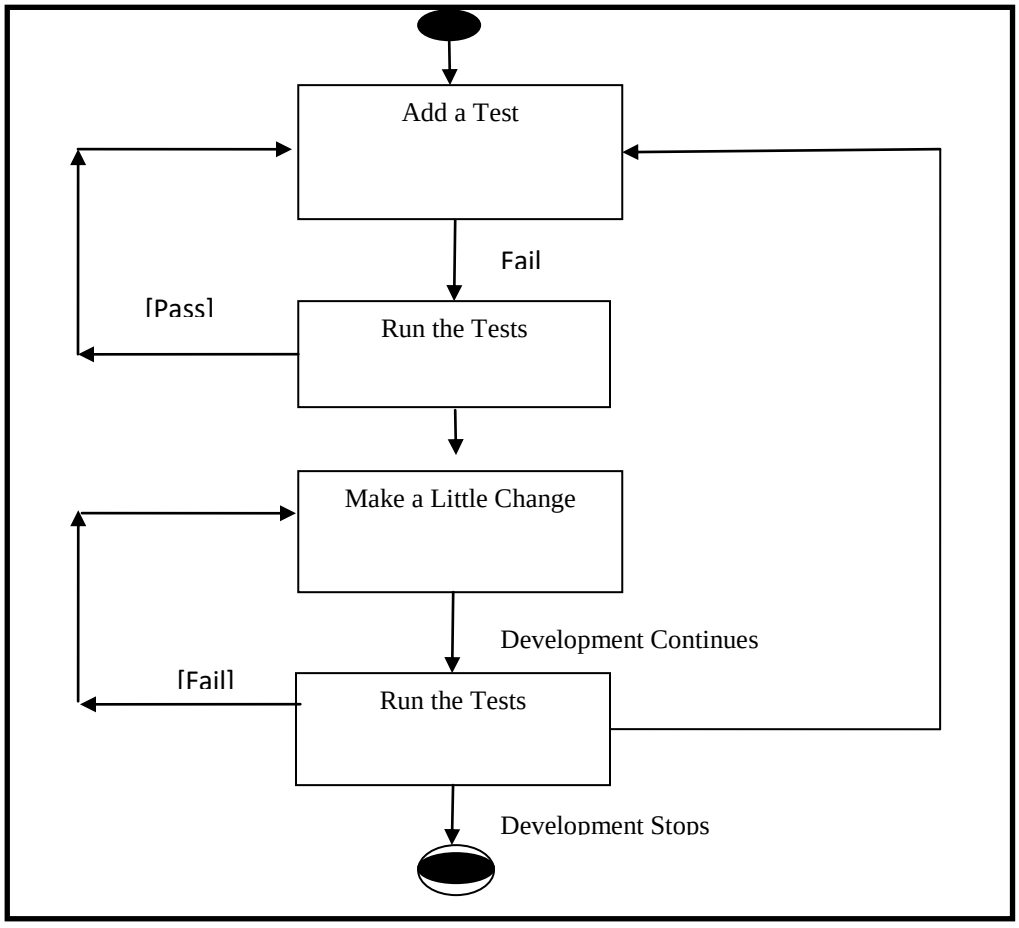


Figure 2.11: Test Driven Development

TDD allows the team to start with uncertain requirements and relies on the feedback loop between the development team and the customers or end-users for input on the requirements. It consists of the following steps:

- **Create the test**: Start with an automated framework to create the test. With TDD, the teams do not need a well-defined architectural design before starting the development phase. The test drives the development of functionality.

- **Write/Modify the code:** Write the code for the application block so that it can pass all test cases written for building the required functionality. The first iteration involves developing new functionality, and subsequent iterations involve modifying the functionality based on the failed test cases.

- **Create additional tests:** Develop additional tests for testing of the code.

- **Test the code.** Test the code based on the test cases developed.

2.10 AGILE REQUIREMENT SPECTRUM

In software development, there is a spectrum of application requirement specification and design. The endpoints of the spectrum are “Nothing Defined” and “Everything Defined” as shown in Figure 2.12. The more the requirements are defined, the less investigation, research and exploration will be needed, and hence more accurately the project size and schedule can be defined. However, the newer (and perhaps more interesting) the project, the less well defined requirements. If the requirements are well identified at early stages of software then the prioritization can be done according to importance of client. In Agile the requirements are always taken in the form of user-stories. A user story is an self-regulating, unfixed, precious, estimable, little, testable requirement. User Stories are great for development teams and product managers as they are simple, easy to understand and prioritize.

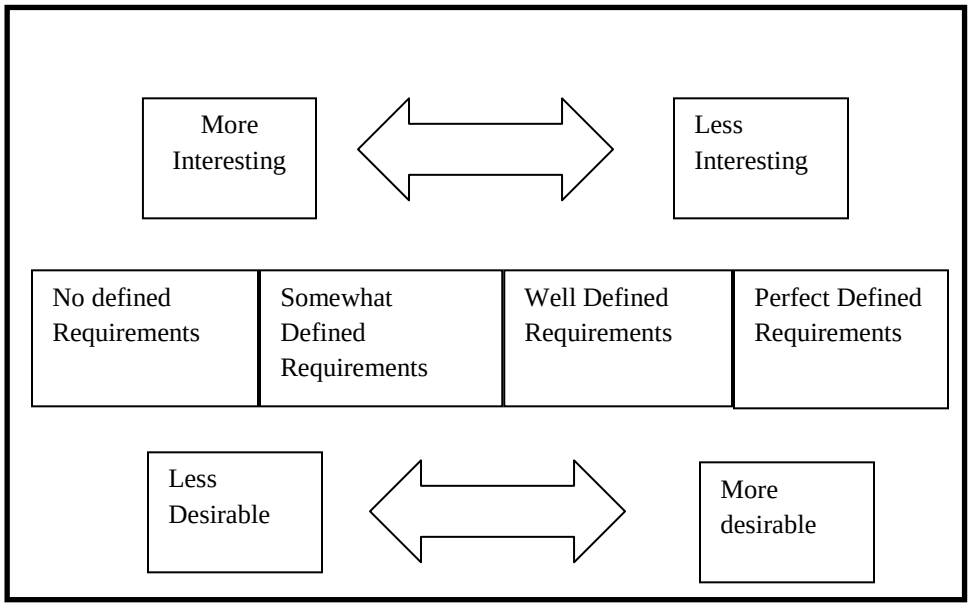


Figure 2.12: Agile Requirement Spectrum

The user-stories are used at sprint-level. These have three critical aspects which are card, conversation, and confirmation as shown in Figure 2.13. In all three aspects either scrum team and product owner are much closely related with each other.

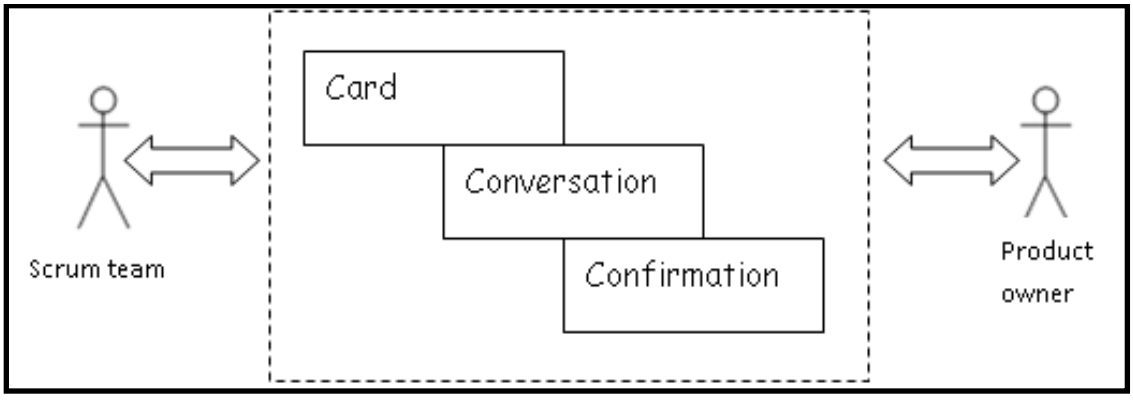


Figure 2.13: User-story

2.10.1 Card

Usually user stories are written on cards but these cards does not include every information that makes up the requirement. The card has just an adequate amount of text to recognize the requirement. The card is a token which represents the requirement. It's

used in planning. All the important information reflecting priority and cost is also written on it. The cards are handed to the programmers when they start implementing the story, and given back to the customer after the story is completed.

2.10.2 Conversation

The requirements in the form of user-stories are communicated from customer to programmers through conversation. It is an exchange of judgements, thoughts, ideas and feelings. This conversation takes place, predominantly when the story is estimated or scheduled for implementation. The conversation provides more information about the feature.

2.10.3 Confirmation

When the user-story is completed, then at the end of iteration developers show the client that the user-story is successfully finished, and the development team authenticates its success by a presentation that the acceptance tests for the user-story run properly.

2.11 PRIORTIZATION OF USER-STORIES

Agile software development methodologies become increasingly popular. A key characteristic of any Agile approach is its explicit focus on creating business value for the clients. Essentially, in Agile software projects, the development process is a business value creation process that relies on active client participation. The business value creation is ensured both through the final product as well as through the process itself. Prioritization of user-stories is a difficult task in Agile environment because of its volatile nature [39,54]. Ignorance of criticality of user-stories will result in several problems like unsatisfied client, poor quality of product.

2.11.1 TRADITIONAL PRIORTIZATION METHODS

The process of prioritization start from planning phase and refined throughout the project. The traditional method emphasizes on few factors like risk, cost, and duration and customer requirements [68,96]. Some of the traditional methods of prioritization are discussed below:

- **Moscow Prioritization**

  MoSCoW stands for **M - MUST** have this which means that the requirements are non-negotiable, if these requirements are not delivered then it is project failure,**S - SHOULD** have this if at all possible, **C - COULD** have this if it does not affect anything else, In SHOULD and COULD category nice to have features are classified. **W - WON'T** have this time but would like in the future. The requirements which are marked as "Won't" are as important as the "Must" requirements category. Classifying requirements as "Won't" acknowledges that it is important, but can be left for a future release.

- **Business Value Based**

  Value-based Prioritization is the core principle that drives the structure and functionality of the whole scrum framework. Scrum aims at delivering a valuable product or service to the customer iteratively and incrementally.

  Prioritization is done by the product owner when he or she prioritizes user stories. The prioritized product backlog contains a list of all the requirements given by the customer. Once the product owner has received the business requirements from the customer and written these down in the form of feasible and effective user stories, he or she works to find out which business requirements provide maximum business value and benefit.

The product owner firstly understand what the customer actually wants and then he prioritize product backlog items or user-stories by relative importance to the customer. In some cases a customer requires that all of the user stories to be of high priority but still, a list of high-priority requirements or user stories needs to be prioritized inside the list itself. Prioritizing a product backlog is must because it finds out the criticality of each user story. In product backlog the high-value requirements are recognized and moved to the top of the product backlog. Value-based prioritization is based upon the principle of prioritizing the product backlog and grooming the prioritized product backlog. Simultaneously, the product owner must work with the scrum team to understand the project risks and uncertainty as they may have negative consequences associated with them. Prioritization is based on a subjective estimate of business value and profitability, and is not limited to, customer interviews, brainstorming sessions, surveys, reviews, financial models and analytical techniques.

The product owner translates the inputs and needs of the project stakeholders to create the prioritized product backlog. Product owner considers business value, risk or uncertainty [41,42] and user-story dependencies while prioritizing the user stories. Thus prioritization results in deliverables that satisfies the requirements of the customer with the objective of delivering the maximum business value in the least amount of time.

- **Waling Skeleton Prioritization**

A walking skeleton is a small implementation of the system that carry outs a small end-to-end function. It does not necessitate the use of the final architecture, but it links together the major architectural elements. The architecture and the functionality in this prioritization method evolve in parallel. It also works as a synchronizer if various teams are working on similar products. The walking skeleton provides a basic structure and way to perform various tasks in product where independent teams are working.

- **Validate Learning**

  Validated learning is a process in which individual learns by trying an initial original idea and then evaluating it to validate the effect. Validated learning is particularly accepted on the web, where analytical software can follow visitor behavior and produce perfect statistics and insight on how website features work in actuality. Typical steps in validated learning:

  - Specify a goal
  - State a metric that signify the goal
  - Act to achieve the goal
  - Analyze the metric - did you get closer to the goal?
  - Improve and try again

2.12 ESTIMATION

Estimation is a process of finding approximate value of a result that determines the amount of time, cost and effort required to complete a software project [20,23]. Success or failure of a project depends on the successful estimation of the effort and time of that project so accurate estimations are very critical for both customer and developer. Ignorance or lack of proper estimation methods may cause effects like poor quality of software, exceeding the budget, not delivered on time and sometimes product functionalities also get affected. So, estimation of software is an important task to calculate cost of the project (in rupees), effort (in man-hours, man-days and man-years) and time required to complete the project (in months) efficiently. It is very difficult to calculate estimations if requirements are changing. In case of traditional methodologies, requirements are static but in Agile methodologies requirements may change during the development process. Estimation of software project is a complex task. It involves following steps as shown in Figure 2.14.

- Identify the aim of the project and its requirements, estimate size of each component.

- After that estimate complexity of each component.

- Estimate effort and resources required.

- Estimate cost of the project.

- Schedule the project on the basis of estimation.

- Compare the result and refine estimations.

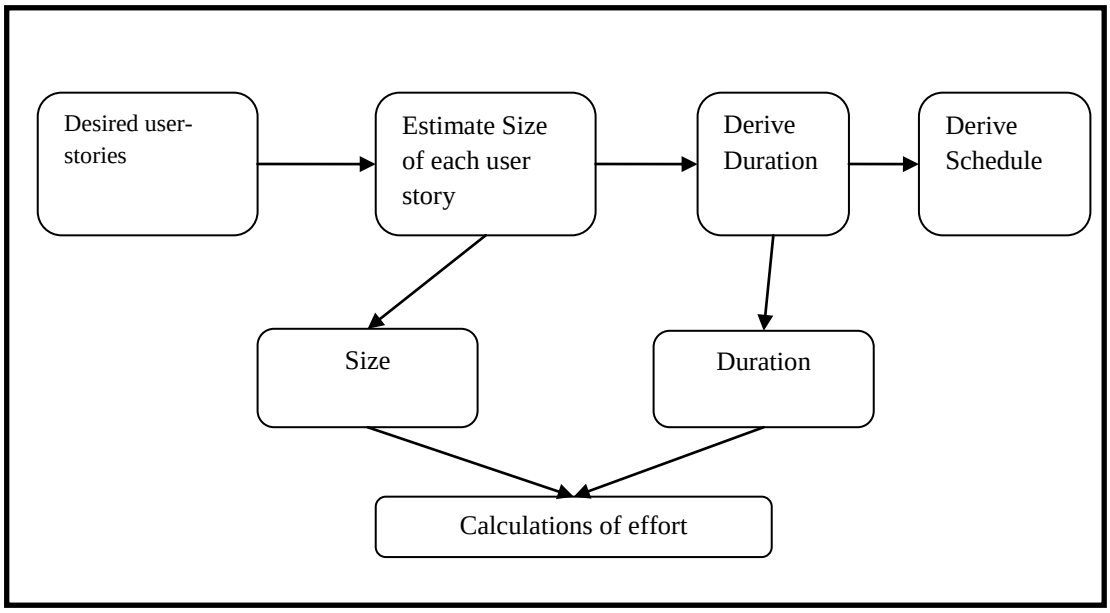


Figure 2.14: Estimation

2.12.1 Types of Estimation

- **Effort estimation**: It is the process of predicting the effort required to develop or maintain the software on the basis of uncertain input. Effort estimates may be used as an input to project plans, budgets, investment analyses and pricing

processes. Expert estimation is considered as the dominant strategy in case of calculating the software development effort.

- **Cost estimation:** The cost of software development is determined by the cost of developing the software plus the cost of equipment and supplies. Cost estimation is needed to establish a budget for the project and to set the price for the software for the customer. If the organization is not confident of their cost estimate, this may increase their price by some possibility over and above its normal profit. Actual cost must not exceed the estimated cost. Cost estimation of a software product is the most difficult and error-prone task in software engineering. It is difficult to make an accurate estimate during the planning phase as many factors are not known at that time. Estimation should be upgraded regularly.

- **Time estimation:** Time estimation tells how much time it will take to complete the project. Time estimation is not an easy task as there are lots of unknown factors.

2.12.2 Approaches of Estimation in Agile

In the Agile methods the estimation is done mostly by expert opinion, analogy, disaggregation and planning poker. No mathematical model is present for estimation of cost and effort.

- **Expert Opinion**: This method seeks advice from various experts of a particular domain. The experts present estimates using their knowledge, own method and experience [58]. In an expert opinion-based approach to estimating, an expert is asked about how long some task will take to finish and how big it will be. The expert on the basis of his experience or intuition provides an estimate. However, this approach is less useful in Agile projects. This approach requires a variety of skills, which are normally performed by more than one person and it is not a easy

process to find appropriate experts who can judge the effort in various disciplines. The main benefit of this technique is that it doesn't take very long.

- **Analogy Costing:** When estimating by analogy, the story being estimated is compared with one or more other user-stories. If a user-story is two times the size, it is given an estimate twofold as large. This approach requires the data of previous one or more completed projects, estimation is done by analogy using the these past projects.

- **Disaggregation:** It will be very difficult to estimate a single story that is large. Disaggregation is defined as dividing a user-story into smaller but easy-to-estimate parts. However, in this technique level is also decided up to which the big story point is broken into smaller story points.

- **Planning Poker:** Planning poker is the best way in Agile projects to estimate. Planning poker combines expert opinion approach, estimation by analogy, and disaggregation into a pleasant approach of estimation which results in quick and trustworthy estimates. The participants of planning poker comprise all of the developers, all testers, designers, database engineers, business analysts and so on. In an Agile project, the size of team should not exceed ten to twelve people. If it does, it is generally best to divide the team into two teams. Then each of the team estimates independently. The product owner participates in this activity but does not provide his estimates. When planning poker starts then a deck of cards is given to each of the estimator. Each card has one valid estimates printed on it. For example estimates are 0, 1, 2, 3, 5, 8, 13, 20, 40, and 100. The cards should be arranged before the planning poker meeting. For estimation of each user story product owner reads the description of it after that he answers various questions of the estimators. Each estimator confidentially selects a card representing his estimate. After each estimator has made selection, cards can be shown. All participants can see each other estimate at the end. At this point the estimates will differ significantly. The estimators with high estimates and with low estimates

explain the reason. After the discussion, each estimator re-estimates again by selecting a card. Again the cards are kept private until everyone has estimated, again the same process is repeated until estimates of most estimators match. In most of the cases, the estimates will come together or converge in second round or third round. But if it is not so, repeat the process. The goal of this complete process is to converge on a single estimate by different estimators.

- **Velocity Measurement:** The requirements normally exist in the form of user-stories. A user story describes some feature or other piece of work. Story point allots a relative size to each of the user story and is used to estimate the velocity of project. Velocity is the developed story points per iteration. This approach starts with time boxing – in a time-box the duration of each iteration is fixed. The ideal day is the time that is fully devoted to the task without any interruption. This approach has the statistical problems.

- **Price-to-win:** The software cost is estimated as the best value to win the project. The estimation is done on the basis of the customer's finances instead of the software functionality. Consider an example, suppose a practical estimation of a project costs 120 person-months. Suppose the customer can afford 80 person-months then in such a case the estimator is asked to alter the estimation to fit in 80 person-months effort for winning the project. This is not a good quality software engineering practice since development team is forced to work overtime.

- **Parkinson:** According to Parkinson's principle “work always expands to fill up the available volume”, cost of a project is judged by available resources rather by objective estimation. If 6 people are available in a project and the project has to be delivered in 12 months, then the effort is estimated to be 72 person-months. This approach occasionally gives good estimation, but it is not recommended as it may present very impractical estimates and does not encourage high-quality software engineering practice.

- **Bottom-up:** In bottom-up approach [57], every component or module of the software system is estimated independently and after that the cumulative results are used to produce the overall estimate of the entire system. For implementing this approach an initial design of the requirements must be clear which specifies that how the overall system is decomposed into modules.

- **Top-down:** This approach is the contradictory to the bottom-up approach of estimation. This approach is more appropriate when cost estimation is needed at the beginning stage. In this approach the overall estimate for the system is derived by using either algorithmic or non-algorithmic methods. The total cost can then be split up among the various modules or components of the system.

2.13 RECOMMENDATION FOR SUCCESSFUL ESTIMATION

Successful estimation in Agile projects depends on various factors and conditions. Based on Mike Cohn book "Agile Estimating and Planning" [62], some recommendations are as below:

- **Entire team will perform estimation:** Every team member must be involved while estimating a Agile project. Most excellent estimates are given when each of the team member is engaged in estimating**.**

- **Planning at various levels:** In Agile projects there is a need of more planning than in traditional software development approaches, but the planning is different. In traditional approaches planning occurs at the start of a project but in Agile planning is done repeatedly through the life of an Agile project during sprints, during release etc. The plans in Agile are however the most detailed plans. These plans sets out exactly how the team will work and how the goal will be accomplished. As the plan only looks at the next few weeks it is generally not needed to change. In the iteration plan the scrum master plans about the different iterations.

- **Use unique estimation unit for each type of estimate:** The most excellent way for estimation is that there should be a different unit for each type of estimate. So that there is no confusion. The size of a user-story is measured by using story-points.

- **Estimate Again if there is a need:** When starting the next sprint then on the basis of the previous sprint estimates, re-estimation can be done.

- **Tracking the projects:** Sometimes the customer wants that after every sprint he should know how much the project is completed then in this case tracking of project can be done by making various types of project tracking methods used in Agile**.**

- **Use User-stories of the right size:** If the sub-stories are very small then it will become difficult to analyze the stories, and delay the project completion. There is no use of reducing the size of stories less than a certain level where the relative uncertainty does not improve. So always use user-stories of the right size.

- **Prioritization of user-stories:** Prioritization of user-stories is very important. As Agile is people-centered, so consider the importance of user-stories for client and effort for each user-story of developers for effective prioritization.

- **Plan at different levels:** It's quite important not to skip iteration plan while doing release plan. The release and iteration plans each cover a different time horizon with a different level of precision, and each serves a unique purpose.

- **Keep estimates of size and duration separate by using different units:** The best way to maintain a clear distinction between an estimate of size and duration is to use separate units that cannot be confused. It is easy to tell that a feature is 0% or 100% done, but it is very difficult to

measure anywhere in between—is this task 50% done or 60% done? Because that question is so hard, stick with what it's known: 0% and 100%.

- **Leave some slack:** Especially when planning / estimating iteration, do not plan on using 100% of every team member's time. Just as a highway experiences gridlock when filled to 100% capacity, so will a development team slow down when every person's time is planned to full capacity.

- **Coordinate teams through look ahead planning:** On a project involving multiple teams, coordinate their work through rolling look ahead planning. By looking ahead and allocating specific features to specific upcoming iterations, interterm dependencies can be planned and accommodated.

2.14 REGRESSION TESTING

Regression testing is applied to code immediately after changes are made. The main goal of regression testing is to assure that the changes have no unintended effects on the behavior of the software. These effects may be either in the software being tested, or in another related software component. It is often necessary to ensure software quality.

Test cases are implemented which help the tester to detect bugs in the system. These are the well documented procedure to test the functionality of the system. Main purpose of these test cases is to find errors in the system. For designing the test cases, test data or set of inputs and their corresponding expected outputs need to be provided.

2.14.1 Need of Regression Testing

Good regression testing gives us the confidence that changes can be made while maintaining the intended behavior and quality of the software.

- If there is change in some requirements i.e. user-stories and code is modified according to the changed requirements then to ensure that modification does not have adverse effect on the software regression testing is needed.

- If new features are added to the software then it may happen that addition of these new features may have adverse effect on the previous implement features.

- If there are some defects in the software then it is required to detect the source of defects and as well as to fix them.

- If software is not performing properly then to check and fix the issue regression testing is required.

- Sometimes customer demands quality of the important requirements of the software. In such cases regression testing is required to check the quality of the software.

- To ensure that software is implemented according to intended requirements, if all the intended requirements are not implemented then customer may refuse to accept the software.

2.14.2 Regression Testing Techniques

Software maintenance comprises of enhancements, error correction, improvements, optimization and removal of existing features. Due to these modifications sometimes the system starts working incorrectly. To uncover bugs in existing functional areas of the existing system, regression testing becomes necessary[108]. Various regression test selection techniques are described as below:

- **Coverage technique:** In this coverable program parts are selected that have been modified. Then test cases are selected which works on these parts. Then selected test cases are executed.

- **Minimization technique**: Minimization-based regression test selection techniques, attempt to select minimal sets of test cases from test suite that provide coverage of modified or affected portions of software.

- **Safe technique**: Most of the regression test techniques, including minimization technique and coverage technique are not designed to be safe. Techniques which are not safe, might fail to find out all the errors of the modified programs. On the other hand, when a specific set of safety conditions are fulfilled, safe regression test can successfully select the subsets that cover all the test cases from original test suite and reveal errors of the program.

- **Ad-hoc/ random technique**: If there is time constraint for retesting all of the programs and no selection tool is available, developers often select test cases based on loose association of test cases with functionality. One simple approach is to randomly select the predetermined number of test cases.

- **Prioritization technique:** This technique of regression testing prioritizes the test cases to increase the rate of fault detection of a test suite. In this test cases with high priority need to be executed first, because they have high probability of finding the errors. If there is enough time available only then test cases with low priority are executed.

2.15 UNCERTAINITY IN AGILE

In ASD there is more planning than in traditional software development approaches, but the planning is different. In traditional approaches planning occurs at the start of a project but in Agile planning is done repeatedly through the life of an Agile project so Agile

environment is dynamic or has uncertainty [37]. In ASD the requirements are taken in the form of user-stories, then user-stories are prioritized. After that scrum master creates a plan for the project by building a Gantt, or Pert type chart. Project tracking is done by measuring against the plan. In Agile there are three levels of planning i.e. iteration plan or sprint plan, release plan and roadmap as shown in Figure 2.15.

Iteration or Sprint plan: At the start of each sprint the scrum team decides what they will perform in the next iteration. Since sprints typically are of short-duration so these plans are not changed in future. The plans are however the most detailed plans, which sets out exactly how the team will work and how the goal will be accomplished. As the plan only looks at the next few weeks it is generally not needed to change. In the iteration plan the scrum master plans about the different iterations.

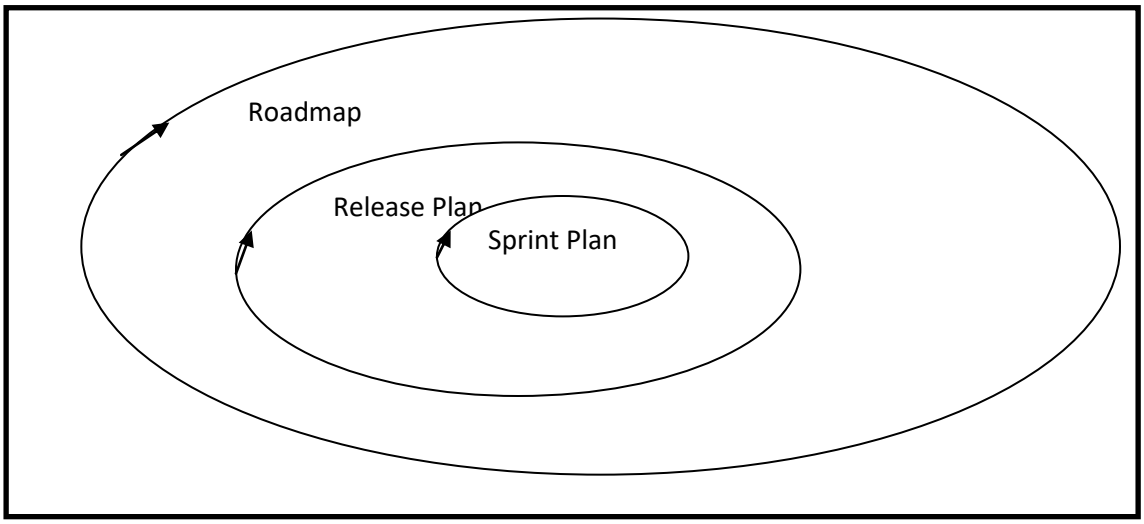


Figure 2.15: Agile Release Planning

Release Plan: It consists of several sprints or iterations as shown in Figure 2.16. Release plans can help coordinate team routine and provide early warning about what is coming up in the respective release. These plans are also detailed plans.

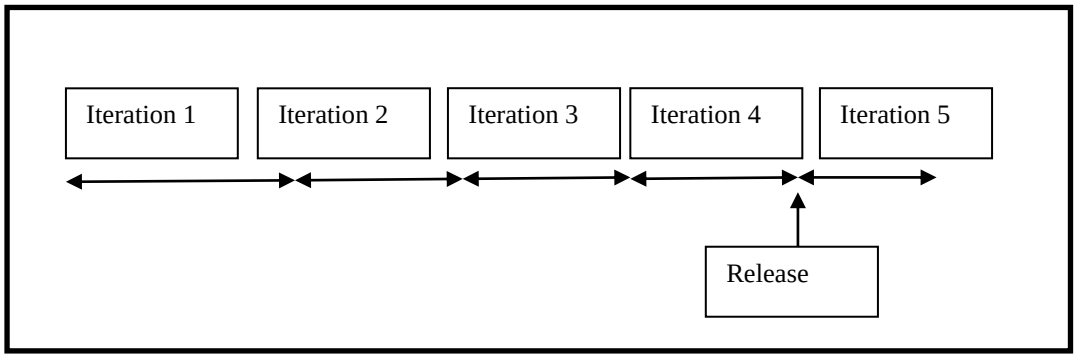


Figure 2.16: Release Plan

Roadmap: Roadmaps do not look weeks ahead but years ahead. Most of the times the roadmaps are separated by quarter for first year or two years, beyond that these are divided by years. So a roadmap written in summer 2010 tell about what is expected in each quarter of 2011, also it may make rough draft of some ideas for each half 2012, and speculate what will happen in 2013 and 2014.

2.16 RECENT WORK RELATED TO AGILE DEVELOPMENT

Malik Hneif, Siew Hockow[60] presented a review of Agile methodologies in software development. This review starts with a brief background about different approaches in software development. It includes difficulties in software development as development involves more critical and dynamic industrial projects and new difficulties emerged according to the growth of companies like evolving requirements, customer participation, deadlines and communication gaps. In this paper, three Agile approaches including extreme programming, Agile modeling and scrum are reviewed. It describes the differences between these methodologies and recommends when to use them. Agile development aims to support early and quick development of working code that meets the needs of the customer. In this work, some limitations are described which may arise while using Agile methodologies. These methodologies depend heavily on the user involvement, thus the success of the project depends on the cooperation and communication of the user.

D. Dalcher[69] performed an experiment in which fifteen software teams developed comparable software products using four different development approaches (V-model, incremental, evolutionary, and XP). The greatest difference in productivity was between the V-model teams and the XP teams, with the XP teams being, on average, 337% more productive than the V-model teams. However, this productivity gain was due to the XP team delivering 3.5 times more lines of code without delivering more functionality.

Karlstro¨m and Runeson[44] studied how traditional stage-gate project management could be combined with Agile methods. In a case study of three large companies, they found that Agile methods give the stage-gate model powerful tools for micro planning, day-to-day work control, and reporting on progress. They also found that they were able to communicate much more effectively when using the working software and face-to-face meetings of Agile methods than when using written heavy documents. The stage-gate model provides the Agile methods with a way to synchronize with the development teams and to communicate with marketing and senior management. The conclusion of the author was that it is possible to integrate Agile methods with stage-gate project management to improve cost control, functioning of the product, and timely delivery. A central concern for Agile methods is to attend to the real needs of the customer, which are often not stated explicitly in a more or less complete requirements specification.

Dagnino,Tore Dyba[19] compared and contrasted the use of an evolutionary Agile approach with a more traditional incremental approach in two different technology development projects. They showed that by planning in detail only the features and requirements to be implemented in a specific cycle, the Agile team was more able to incorporate changes in requirements at a later stage with less impact on the project. In addition, by delivering in-progress software to the customer more frequently, the Agile team was able to demonstrate business value more quickly and more often than the traditional, iterative team. Combined with continuous feedback by the customer, this lead to a sharp increase in customer satisfaction on the Agile project.

M. Ceschi, A. Sillitti, G. Succi, S. De Panfilis[55] survey of project managers found that companies that use Agile methods are more customer-centric and flexible than document-driven ones, and that companies that use Agile methods seem to have a more satisfactory relationship with the customer.

Balasubramaniam Ramesh, Lan Cao,Richard Baskerville[9] concluded that compared to traditional development, team members of Agile teams are less interchangeable, and more difficult to describe and identify.

Rajlich [78] described Agile development as a paradigm shift in software engineering, which has emerged from independent sources: studies of software life cycles and iterative development. ‘‘The new paradigm brings a host of new topics into the forefront of software engineering research. These new topics have been ignored in the past by researchers.

P. Abrahamsson, J. Koskela [72] demonstrated how to collect software metrics to measure effort estimation, productivity, quality and schedule estimation and cost estimation for a Software project using XP.

L. Williams, W. Krebs, L. Layman, A. Antón, and P. Abrahamsson[52] investigated the usage of a subset of XP practices at a group in IBM. The product developed at IBM using XP was found to have significantly better pre-release and post-release quality compared to an older release. The teams working with XP method reported a great improvement in estimation as well as productivity.

F. Maurer and S. Martel [24] studied the development of a web based system by nine full time employees in a small company that used XP and observed substantial productivity gains compared to their pre-XP timeframe.

Heemstra [23] surveyed 364 organizations and found that only 51 used models to estimate effort and that the model users made no better estimate than the non-model users. Also, use of estimation models was no better than expert judgment.

Ananda Rao and Kiran Kumar [1] have invented a technique of cost-effective regression testing in Agile environment by reducing the test suite. The reduced regression test suite has same functionality as the original regression test suite and also has the same bug finding capability. In this work, two aspects of testing are shown that is testing for functionality and testing for boundary values can be tested with reduced test suite. These two aspects can be tested together simultaneously in most of the situations.

In this work, proposed approach is shown in three phases. In phase1, the reduced test suite is derived from the original test suite and in phase 2, the reduced regression test suite is derived by applying a regression test selection method on the reduced test suite that is derived in the phase. In last phase, a derived testing cost-estimation model is applied on the reduced regression test suite and the cost reduction in regression testing is calculated.

Todd L. Graves, Mary Jean Harroldy, Jung-Min Kimz, Adam Porterx, Gregg Rothermel [108] have done an empirical study of different regression test selection techniques. Regression testing is an expensive process and to reduce its cost, regression test selection techniques were proposed for selecting a subset of the test cases in existing test suite of the program. In this work, five different technique of regression test selection are examined. Two models are constructed for calculating the cost of using a regression test selection technique and fault detection effectiveness of the resulting test suite. This information is captured for every test suite, subject program, and test selection technique. From this information percentage reduction in test suite size is calculated.

Chapter III

TRANSITIONING OF TRADITIONAL SOFTWARE DEVELOPMENT METHOD TO AGILE METHODOLOGY

3.1 INTRODUCTION

According to Agile Manifesto [6] 'individuals and interactions over procedures and tools" is the primary proverb in Agile environment. It means processes; procedures and tools have less value in comparison with individuals and interactions. In traditional approaches like waterfall, spiral, V-model etc. process remains fixed. All the phases are properly described and documented so that anyone can follow this fixed and static approach. The static approach of traditional software development method and dynamic approach of ASD makes this matter debatable as transitioning is taking place from waterfall or any other traditional model to Agile model in most of the software companies.

ASD [71] introduces changes in work habits. When an organization wishes to transition to ASD, a change is required at the organizational level. To ensure this change in a software organization many concerns need to be discussed like upper level management attention, team interest, infrastructure needed and many more [99]. Various approaches have been suggested by different researchers for these organizational changes in general and for transition to Agile software development.

Manns and Rising [54] suggest 48 patterns for change introduction. The patterns are the result of years of documenting observations, investigations from community who have introduced latest ideas, reading various topics of change and finding out how these problems are tackled in history.

Orit Hazzan and Yael Dubinsky [71] proposed an organizational survey for transitioning from traditional to Agile. This survey helps to understand the current situation and status of software development in the organization. It also helps in the decision whether the Agile approach fits for the organization or not. The author also proposed four major categories of cooperation tools in change processes: power, administration, leadership and customs. To choose the right cooperation tool requires assessing the software industry along two significant extents: the degree to which team agree on what they actually want.

D.Leffingwell [15] discussed how Agile methods can be useful to enterprise development. He also discussed seven most excellent practices of agility that are best at the enterprise level. Further his book provides an additional set of seven enterprise capabilities that organizations can master to get the benefits of agility on an enterprise scale.

A critical look at the above literature indicates that the previous approaches of transition do not provide an Agile model with the help of which transitioning can take place. Since most of the organizations are working on a traditional SDLC models, there is a need of mapping of the traditional model into Agile life cycle model so that transitioning can take place between two SDLC models in appropriate manner. An Agile model and a mapping function has been proposed in this research work so that transitioning can be attained with ease of team members and upper management.

3.2 PROPOSED AGILE MODEL

To accept change is the compulsory requirement for the Agile developers. Agile cannot exist in industry without accepting change. But the problem is how to perform the transitioning from traditional model to Agile model when the traditional model is the base of the organization and every team member has expertise in that. For accepting change, in the start, things seem to be very difficult but with the support of top management, scrum master and coach, Agile can be implemented with great success.

This section describes a mapping model for transitioning by considering the existing traditional model of the organization [79,90]. The following are the main components of the proposed Agile model which are shown in Figure 3.1

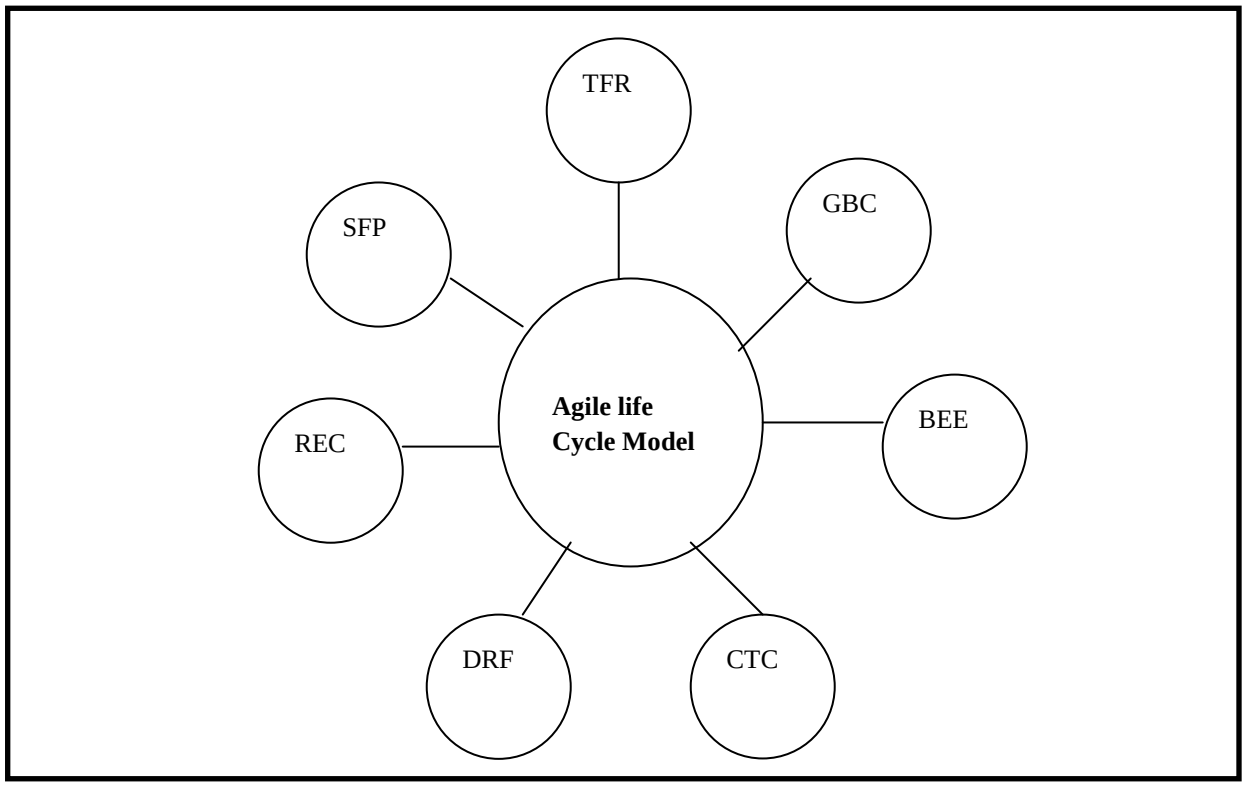


Figure 3.1: Proposed Agile Model

- Team Formation by good recruitment policy and good team interaction (TFR)
- Goal Building cycle with business Analyst, quality assurance analyst and customer (GBC)
- Coding and Testing activities with Communication and co-ordination (CTC)
- Budget and Effort estimation (BEE)
- Satisfaction for all parties (SFP)

- Demonstrations in Review with feedback (DRF)

- Risk evaluation and correction (REC)

These seven components are the base of an Agile model. The description of each component is given below:

3.2.1 Team Formation by Good Recruitment Policy, Good Team Interaction (TFR)

In Agile working environment, good recruitment policies should be followed to find the right person. In the Agile team, there can be experienced team members as well as freshers. The attitude of a team member towards work should be the biggest factor while doing recruitment.

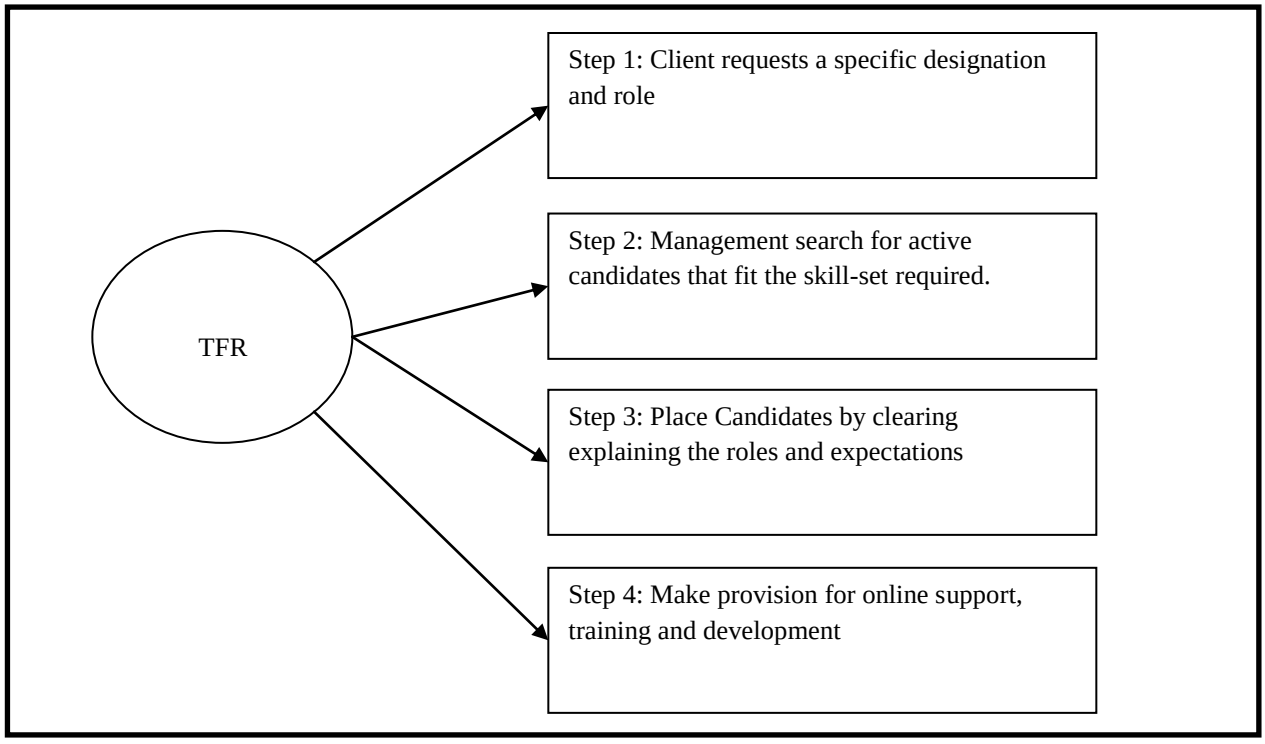


Figure 3.2: Team Formation by Good Recruitment Policy

A experienced Agile team can be formed by upgrading the technical and managerial skills of team by devoting training by trainers, polishing the attitude of team towards work and motivating team time to time.

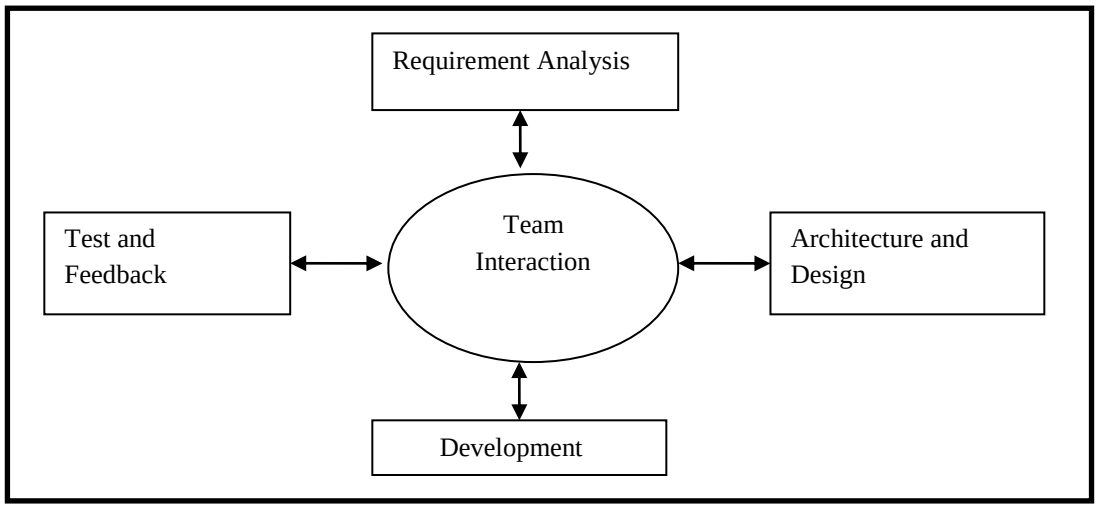


Figure 3.3: An Agile Team Interaction

3.2.2 Goal Building Cycle with Quality Assurance Analyst, Business Analyst and Customer (GBC)

The user-stories [28] on the basis of requirements are identified and approved by client, quality assurance analyst and business analyst by taking into account the return on investment and market demand. An assessment is approved by finding the competitive stage of the existing products. The presence of quality assurance analyst along with customer helps in setting the pattern in mind so that at the time of pair programming he or she can provide the correct feedback to the developer. Also test cases can be designed before development starts.

3.2.3 Budget and Effort Estimation (BEE)

The budget and effort of a user-story is estimated by considering the various requirements for each user-story after prioritization of user-stories. After initial prioritization and estimation of user-stories, two to three weeks cycle of sprint starts. The

effort estimation in ASD can be done by any famous estimation technique like estimation by analogy or planning poker [60,61,62]. Estimation is possible at three levels namely iteration level, release level and project level. The unit of estimation of user-story is story-points [67,68] and ideal time.

3.2.4 Coding and Testing Activities with Communication and Co-ordination (CTC)

The implementation of story starts when estimation is properly done. In pair programming approach two programmers sit together and work together. One person is the leader who performs coding and testing and second person is the reviewer [87] as shown in Figure 3.4. This approach provides immediate feedback with the help of which number of bugs can be reduced. Otherwise the bugs keep on propagating from one phase to another phase. In distributed pair programming or virtual pair programming or remote pair programming the two programmers work together but they are in different locations.

By pair programming as two developers sit together for coding, so knowledge and programming skills are shared. By this approach of pair programming mistakes are also reduced. Test driven development [14,47] (TDD) approach is also used in ASD in which before writing the code for the user-story test cases are written.

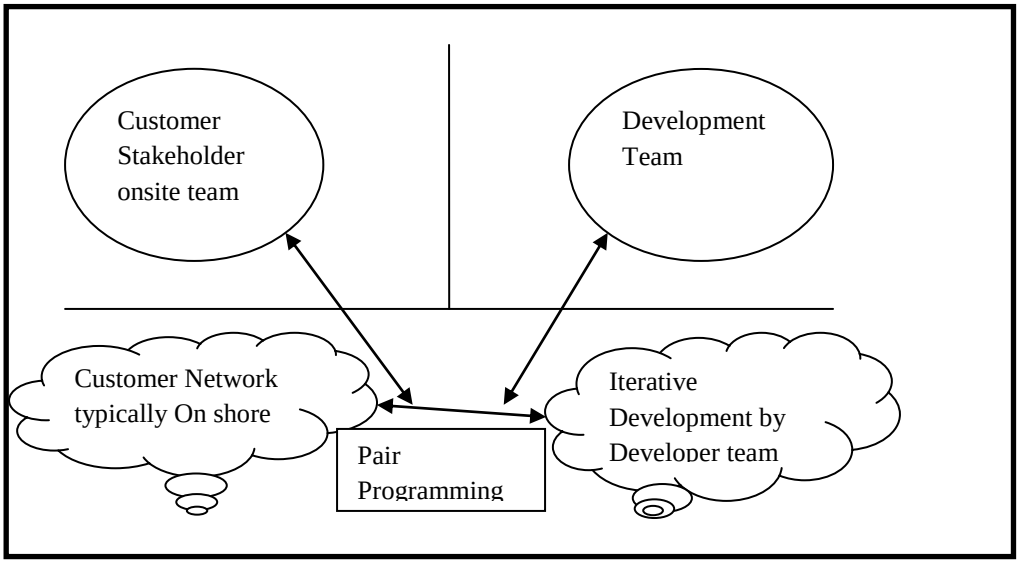


Figure 3.4: Pair Programming

3.2.5 Demonstrations in Review with Feedback (DRF)

At the time of review, Agile team members, upper management, business analyst and customers sit together for demonstrating the software product. Scrum master gives the demo for the product. After demo, the goal matching action is carried out to check that whether story approved is the end product or not. Figure 3.5 shows the feedback system, the feedback can be given by any stakeholder including customer, upper management or any business analyst or existing member of the team. After feedback, a review meeting is done which is an informal meeting between all stakeholders.

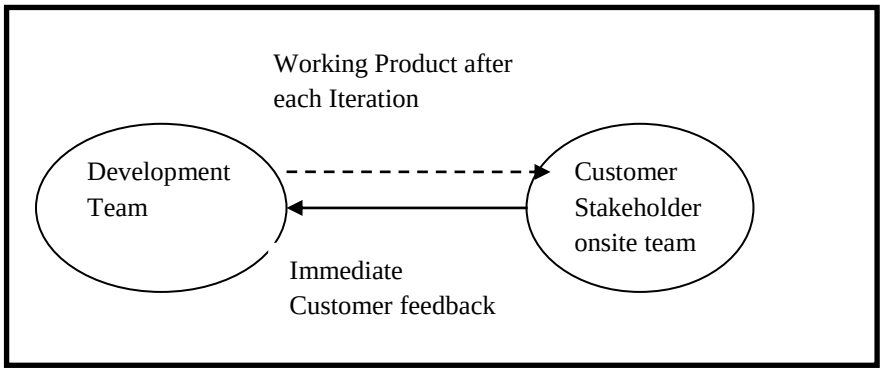


Figure 3.5: Feedback System

3.2.6 Risk Evaluation and Correction (REC)

Further, in the Agile model, risk assessment is performed for the future user-stories so that risk can be reduced or completely eliminated. In fact customer is not only the customer rather he or she is worried about quality, time and also sustainability of the software product in market for long time. In brief, customer is more concerned about return on investment and benefits [14]. In the proposed Agile model high risk user-stories are detected early so that risk is minimized. If high risk stories are not involved then after-effect of it can degrade the quality of the product.

3.2.7 Satisfaction of All Parties (SFP)

All stakeholders whether Agile team, customers or the upper management are satisfied because final product is delivered on time by Agile processes like continuous working software delivery, continuous feedback from customers, continuous integration and testing and continuous return on investment.

When there is a need for transitioning from existing traditional model to Agile model, the issues that may come during mapping are as below:

- Why transition is needed? Is management or customer interested?
- How transitioning from traditional software development life cycle model to ASD model is performed?
- What will be the mapping function to perform transitioning?
- Whether team is of that much caliber or not? Whether new team is required?
- How effort, time and cost estimation will be done?

After resolving all these above issues, management and team both start work for the mapping function from traditional model to Agile. If some software industry is ready for accepting change then in the start, processes would seem to be very complex but after some time with the support of the team members, organization, upper management and scrum master projects can be implemented using ASD with good success rate. Figure 3.6 shows a mapping function which will be applied when the top management takes the decision for transformation in the organization.

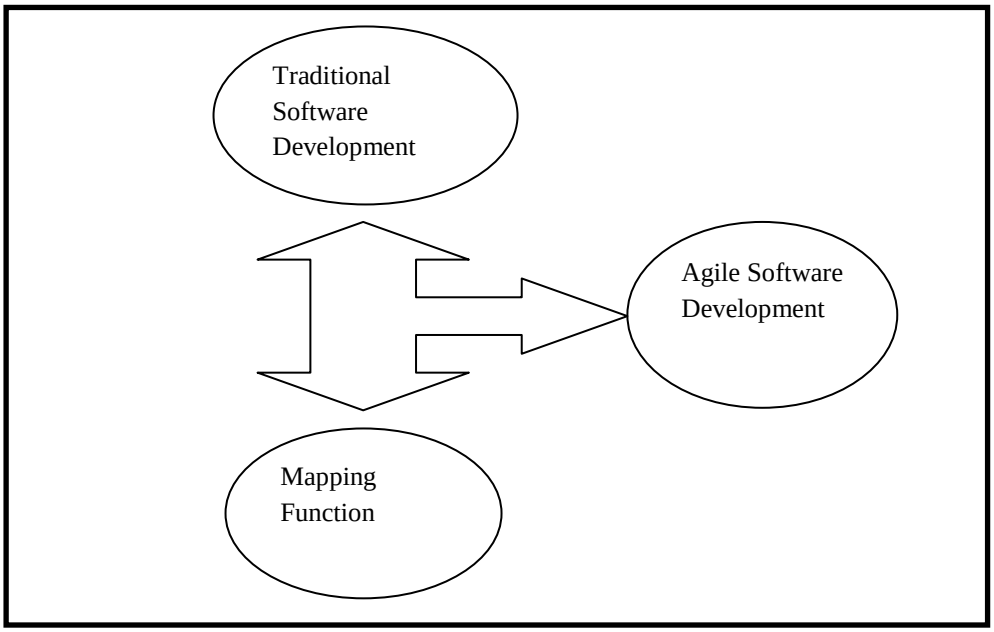


Figure 3.6: Mapping Function

In the expression 3.1 co-ordination effectiveness is proposed which depends upon implicit and explicit factors as shown in Figure 3.7.The expression 3.2, shows the role of mapping function (MF) which is to map the large teams of traditional projects into small and efficient Agile teams (T), long release cycles of traditional models into small sprint (I),large tasks into small stories (J), for pair programming two monitors into one terminal (MO), long feedback cycle into instant feedback (F), estimation in lines of code or functional points into story points (E),extended meetings into daily small meetings (M),late delivery into fast small delivery (D), late testing into test driven testing (TG), and last project manager into no chief or boss approach (B) and self-organized team and co-ordination effectiveness(CE). The implicit factors for co-ordination effectiveness are know why, know what is going on, know what to do and when, know who is doing what. The explicit factors are about right place and right time for doing a particular task.

CE=Implicit factors+ Explicit factors----------3.1

MF = (T, I, J, MO,F, E, M,D, TG, B, CE)---------- 3.2

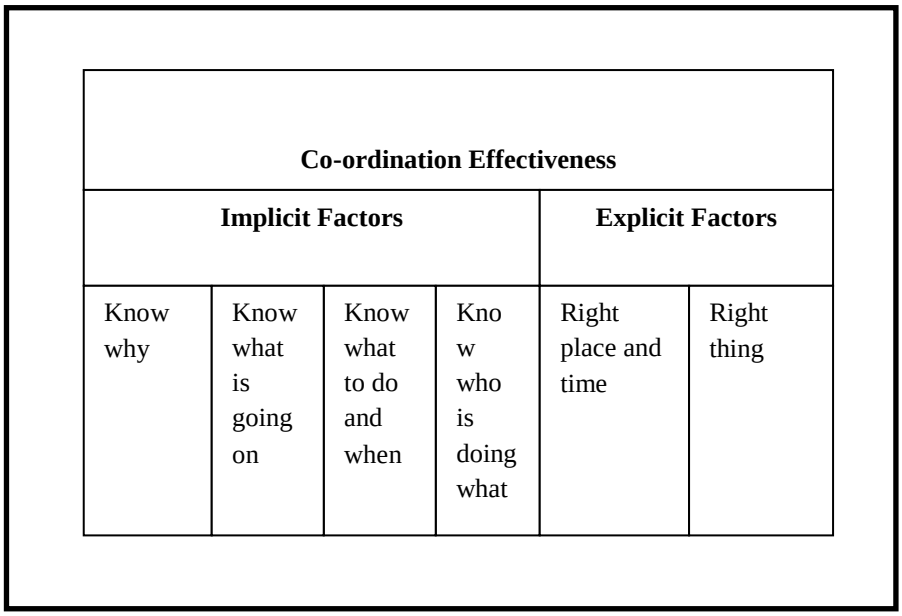


Figure 3.7: Factors of Coordination Effectiveness

Any of the traditional software development models can be transformed in to the Agile model by using this mapping function. In proposed mapping function, ten parameters are there which are must for transitioning to Agile environment in an organization. For team interaction and co-ordination cubicles can be converted into open work surroundings, heavy documentation can be converted into simple story-boards, overtime is converted into 38-40 hrs per week of valuable and effective work. Various automated tools are converted into definite tool for a specific domain. In short, Agile approach is more advantageous with less cost and time.

3.3 STEPS FOR APPLYING MAPPING FUNCTION

- Don’t apply the mapping function all of a sudden. Discover the ways to simplify operational and administrative documentation.

- Start with a low risk, small project and develop user stories and scenarios as the feature units and start initial estimations.

- Break the large release cycle to small iterations called as deployment cycles. Change large projects of more than eight months into several versions released after every two months.

- Under the guidance of Agile coach, form the small Agile teams, with experienced people from the different functional areas. Good recruitment policies should be followed to find the right person. Thus team will work as Agile team.

- Start pair programming in the team because by pair programming as two developers sit together for coding, so knowledge and programming skills are shared.

- Start involving client at every stage to get earlier feedback.

3.4 THE BENEFITS

The proposed mapping function can be well-designed and purposeful when all the parameters are identified in the existing traditional model of the organization and transformation is done by mapping according to the mapping parameters. The major benefits from this mapping function are as below:

- Time consumption would be less because to apply mapping function is very simple.

- Everybody would be happy (team, customers, top management) as the project will be delivered on time and within budget.

- Old resources of the organization would not be unemployed.

3.5 CONCLUSION

An Agile model is proposed for adopting Agile processes in the software industry. A mapping function is also presented for transformation from traditional software development model to new Agile model. It is the base to implement Agile and victory rate of any Agile project can be increased by matching all the parameters of the mapping function.

Once the environment for Agile has been set up, the proper estimation in Agile can be done. The next chapter proposes the estimation techniques.

Chapter IV

A SPRINT-POINT BASED ESTIMATION FRAMEWORK IN SCRUM: PROPOSED WORK

4.1 PROBLEMS IN AGILE ESTIMATION

Based on literature study [57,58] it has been found that most of the existing effort estimation techniques have been developed to support traditional sequential software development methodologies whereas ASD is iterative and dynamic in nature. If these traditional techniques are used for effort estimation of Agile software projects, then the results will be definitely inaccurate.

Various approaches have been suggested by different researchers for estimation which are discussed below:

O.Benediktsson, Dalcher[69] performed a controlled experiment to investigate the impact of software development approach on the resulting product and its attributes by comparing V-model (VM), evolutionary model (EM), incremental model (IM), and extreme programming (XP).The conclusion was that : XP groups spent significantly less time in requirements specification than V-model evolutionary model groups ,XP produced significantly more LOC in general than all other methodologies, XP produced significantly lower number of pages per Person Months(PM) than all other methodologies, XP produced significantly higher pages per PM than VM, no differences in total pages per PM between methodologies.

S. Bhalerao, Maya Ingle [98] introduced an algorithm to calculate the development cost, time and effort. The need for mathematical algorithm arises due to the limitations of the previous work. The authors have considered some of the factors that affect the estimates. The name of the algorithm is termed as Constructive Agile Estimation Algorithm

(CAEA). This algorithm uses the vital factors mainly; project domain, configuration, performance, complex processing, data transaction, operation ease, multiple sites and security to calculate effort, time and cost. This algorithm helps reduce the risks factors.

Buglione [11] investigated the effort estimation activity within Agile software development methodology. He briefly described the estimation approaches used in various Agile software development methods.

P. Abrahamsson, J. Koskela [72] demonstrated how to collect software metrics to measure effort estimation, productivity, quality and schedule estimation and cost estimation for a Software project using XP.

Based on the critical study of various research papers it has been found that there are several problems in existing estimation and tracking methods as discussed below.

The first problem is that if the estimates are unrealistically low, the project will be understaffed and the resulting excessive overtime or staff burnout will cause attrition and compound the problems facing the project. Overestimating a project have the problems like overstaff and over cost. *Thus, an effort metric is needed that calculates the size of the Agile team.*

Second, the story points and velocity are used to estimate the initial size of the project, there are magnitudes of factors which can affect the story points and decelerate the velocity and thus affect on productivity of team. Although S.Bhalerao and Maya Ingle [98] identified some vital factors that affect estimation, there are various other factors that must be considered to check the affect on velocity. *So, there is a need to find out a comprehensive list of various factors which can affect the velocity and thus estimation.*

Third, in Agile environment, at the initial stage of a project, there is high uncertainty about various project attributes. The estimates produced at early stages are inaccurate, as the accuracy depends highly on the amount of reliable information available to the

estimator. Agile estimation methods may lead to the errors in case of inexperienced Agile team. *Therefore, there is strong need of analyzing the uncertainty that may affect the estimation of the Agile project.*

Considering the characteristics of ASD methodology and all the problems discussed above, an effort estimation framework to predict development effort of Agile software project is proposed in this chapter.

4.2 PROPOSED SPRINT-POINT BASED ESTIMATION FRAMEWORK IN SCRUM

When planning about first sprint, at least 80% of the backlog items are estimated to build a reasonable project map. These backlog items consist of user-stories grouped in sprints and user-stories based estimation is done using story-points [80,81,84]. When a team member estimates that a given task can be completed within 8 hours it does not mean that he can complete the task in 8 hrs. Because no one can sit in one place for the whole day and there can be a number of factors that can affect story-points and hence decrease the velocity.

To resolve this problem the concept of Sprint-point is proposed. *A Sprint-point basically calculates the effective story-points. Sprint point is an evaluation unit of the user story instead of story-point. By using Sprint points, more accurate estimates can be achieved. Thus, the unit of effort is Sprint Point (SP) which is the amount of effort, completed in a unit time.*

In the proposed sprint-point based estimation framework, requirements are first gathered from client in the form of user-stories. After requirement gathering, a user-story based prioritization algorithm is applied to prioritize the user-stories. Consequently story-points in each user-story are calculated and uncertainty in story-points is removed with the help of three-types of story-points proposed. Then these story-points are converted to sprint-

points based on the proposed Agile estimation factors. Afterwards sprint-point based estimation algorithm is applied to calculate cost, effort and time in a software project.

As regression testing [94] is a necessary but expensive activity aimed at showing that code has not been adversely affected by changes. So defect data is gathered based upon the similar kinds of projects, which is used to calculate rework effort and rework cost of a project. Finally the sprint-point based estimation algorithm is applied to calculate the total cost, effort and duration of the project.

This **Sprint-point Based Estimation Framework** as shown in Figure 4.1 performs estimation in scrum using below steps:

Step 1: User-stories are prioritized by using User-story Based Prioritization Algorithm (will be discussed in section 4.3).

Step 2: As Agile is highly dynamic so uncertainty cannot be removed completely. Some steps can be taken to reduce uncertainty (will be discussed in section 4.4).

Step 3: Story-points are converted into sprint-points by considering Agile estimation factors like project-related, people-related and resistance factors. These factors affect the user-stories and thus affect the cost, effort and duration of a software project. (will be discussed in section 4.5).

Step 4: Regression testing is done in Agile to make sure that the new incorporated changes should not have side effects on the existing functionalities and thereby finds the other related bugs. Thus regression testing may consume much time, cost and effort. So there is a need to calculate regression testing efforts in Agile.

Step 5: Sprint-point based estimation is done by using Proposed Sprint-point Based Estimation Algorithm including Agile estimation factors and regression testing (will be discussed in section 4.6)

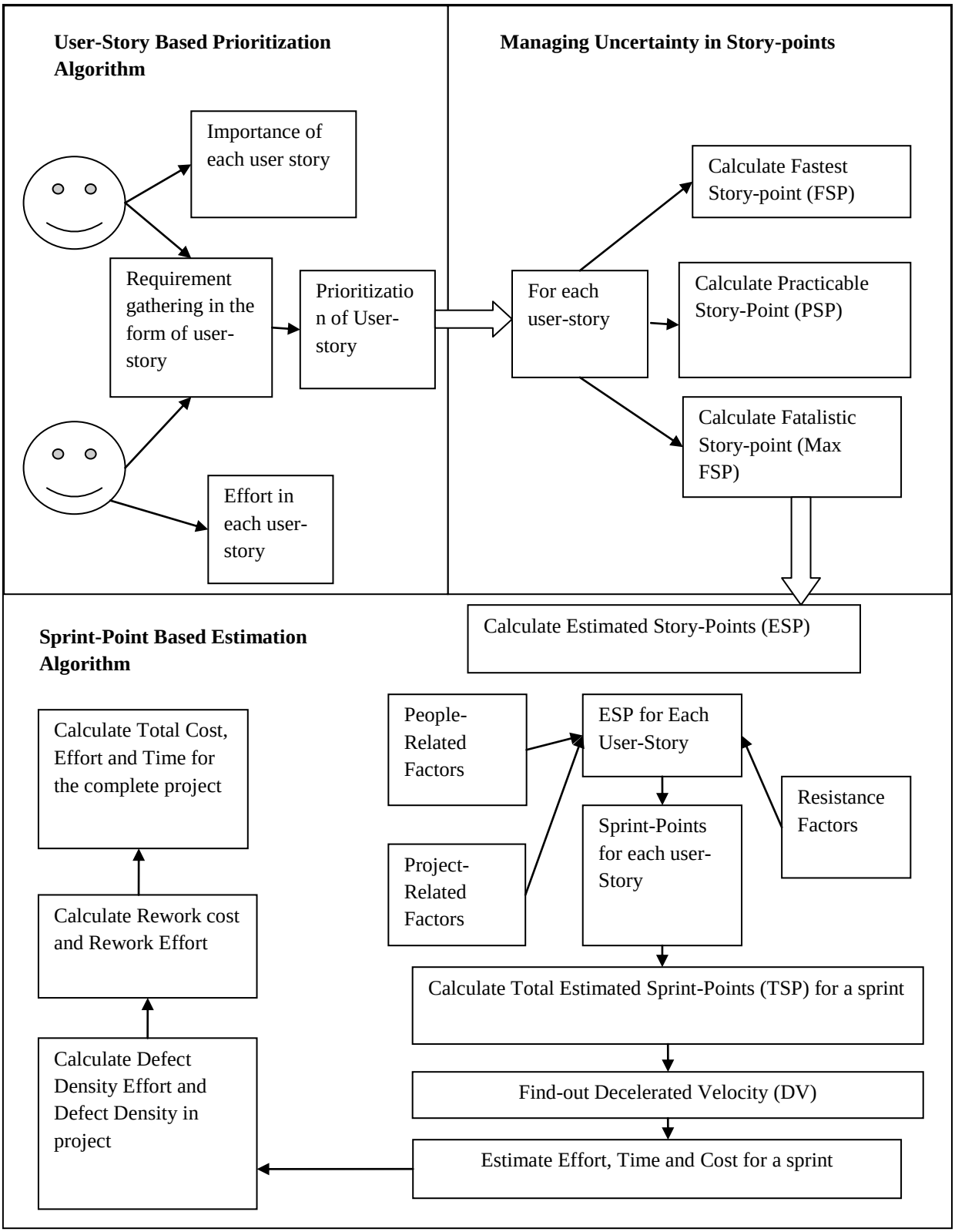


Figure 4.1: Sprint-Point Based Estimation Framework

4.3 PRIORTIZATION OF USER-STORIES

In Agile a client always gives the requirements in the form of user-stories. A user story is autonomous, unfixed, precious, estimable, small and testable requirement as discussed in chapter 2. User Stories are great for development teams and product managers as they are easy to understand, discuss and prioritize. The user-stories are more commonly used at sprint-level. For requirements elicitation and prioritization these user-stories must be prioritized. However there is no algorithm in Agile environment for prioritization of user-stories as per importance of the client. This research work suggests a method of prioritization that helps in choosing the optimal order of user stories.

4.3.1 Problems in Existing Prioritization Methods

If the requirements are well identified at early stages of software then the prioritization can be done according to importance of client. But based on the critical study of various research papers, it has been found that the existing techniques of prioritization have various problems due to dynamic nature of Agile. The problem in MOSCOW method was that the managers are worried that their requirements will fall into "should" or "could", and won't get done, so they make up reasons why their requirement is a "must". This ends up delaying business-critical functionality. In MOSCOW method, a lot of time is wasted in discussing things that "should", "could" or "would" happen, delaying progress on the things which are absolutely essential.

By considering the problems and based on the characteristics of ASD methodology a prioritization rule has been proposed to prioritize user-stories in Agile environment.

4.3.2 Proposed Prioritization Rule

As Agile is people-centred, so considering the importance of user-stories for client and effort for each user-story of developers, prioritization rule has been proposed [83].

The proposed prioritization rule is ***"****To prioritize the user stories such that the user-stories with the highest ratio of importance to actual effort will be prioritized first and skipping user stories that are "too big" for current release"****.***

Consider the ratio of importance as desired by client to actual effort done by project team (I/E) as in Formula 4.1.

$$\textit{Prioritization of user stories} = \frac{\textit{Importance of user stories}}{\textit{Effort per user stories}} \quad \text{---------- } (4.1)$$

The various steps involved in prioritization of user stories in the Agile environment are as below:

- Gather requirements in the form of user-stories.
- Find out importance and effort related factors as in section 4.3.3.
- Calculate I/E per user story to decide the priority.

4.3.3 Proposed Importance and Effort Related Factors

In this work importance related factors such as timely-delivery, dependencies in user-stories etc. and effort related factors such as project domain, technical ability etc. are proposed that impact the prioritization of user-stories.

a) **Timely delivery:** The time constraint is a big issue for client as well as Agile environment. The product release date is given to client according to user story i.e. the client will tell which story is to be done earlier so to start his work as soon as possible. The product which is released early or periodically is best way to satisfy the client needs. If the team is uncertain about the implementation of feature, then early release is the best solution.

b) **Dependencies:** Dependencies of user-stories in the product backlog is always having a vital role in Agile environment. Dependencies between user stories affect the prioritization in Agile environment. Combining several dependent items into large one and splitting the items differently are two common techniques for dealing with dependent user stories.

c) **Business value:** The business value of a user story can be assessed as a combination of user value, revenue, validated knowledge and future return on investment[110].

d) **Risk minimization:** User stories with high risks and high business priority are implemented at early stages of project so that changes in requirement can be detected in early iteration and product can be thoroughly tested in various rounds of testing.

Table 4.1: Importance and Effort Related Factors

Proposed Importance and Effort Factors		
S.No	***Importance Related Factors***	***Effort Related Factors***
1.	Timely Delivery	Project domain and ease of coding
2.	Dependencies	Technical Ability
3.	Business value	Usability
4.	Risk Minimization	Complexity
5.	Cost minimization	Security
6.	Quality	Pre-requisite availability of resources

e) **Cost minimization**: The cost per user-story is also a big concern for a client. The total cost should be in budget of the client.

f) **Quality delivery:** Project quality may be characterized as functionality, reliability, usability, efficiency, maintainability and portability. The outcome product should develop in such manner that it meets the customer's requirements. These features of project increase the cost and duration of the project. For

example, Fault tolerance and recoverability are of primary concern for the interfaces but not for the other parts of the system.

g) **Project-domain and ease of coding:** The type of the project affects the cost, effort and duration of the project. The project can be of web based application, construction, new project development, information system, military project etc. Projects can categorize based on unique characteristics of different types of projects. The ease of code depends upon the type of project. Some projects use the template which is available and designed quickly. This can be called as auto generated code. But in some task, the teams have to develop their own code which requires more time and effort.

h) **Technical ability:** The technical ability of a team member is his expertise in some particular activity like any process or tool related with the ASD. It involves specialized knowledge, analytical ability within that specialty, and facility in the use of the tools and techniques of the specific discipline. Technical skills involve process or technique knowledge and proficiency in a certain specialized field such as engineering, computer and accounting.

i) **Usability:** Usability means how is it easy for user to use or operate the product. It may be in the form of a better GUI, to understand the functionality of the product, minimum user input etc. It increases the quality of the software as well as customer satisfaction. System design and architecture include numerous activities which increase the cost of a software project.

j) **Complexity:** The complexity of task in Agile environment depends upon whether the task is composite or stands alone in nature. Composite task are the task which have many dependencies. It has high cross department effect and extensive consulting activity which requires much effort. Standalone or simplex task is the task which has few or no dependencies. It has intermediate total cost tested technology.

k) **Security:** It is considered as network security, functional security, code security, documentation security etc. based on customer requirements. At the time of project development life cycle security must be taken as higher priority factor.

l) **Pre-requisite availability of resources**: Resources are money, people, material, technology, space and other asset which are necessary for effective operation. The pre-requisite availability has vital role in Agile environment because the task get delayed if resource is not available at that time.

4.3.4 Proposed User Story Based Prioritization Algorithm

The proposed algorithm explains the various steps involved in prioritizing the backlog in Agile environment as discussed in Figure 4.2.

1. *Identify the importance of the user story based on the importance related factor. Suppose I is importance of user stories i.e.* $I_1, I_2, I_3, \ldots I_n$. *Importance is reciprocal of rank of user story i.e. the user story with rank 1 has least important value.*
2. *Identify the effort per user story based on effort related factor. Suppose E is effort of each user story i.e.* $E_1, E_2, E_3 \ldots E_n$. *Effort is calculated in hours or days based on size of project.*
3. *Compute the I/E for all the user stories and takes this value prioritization.*
   i. $P_{i\ 1<=i<=n} = I_i / E_i$
4. *Order the data such that* $I_{[i]}/E_{[i]} >= I_{[i+1]}/E_{[i+1]}$.
5. *Draw a graph between the user stories and I/E user stories and choose user stories from top to down.*

Figure 4.2: Proposed User Story Based Prioritization Algorithm

4.4 MANAGING UNCERTAINITY IN STORY-POINTS

Due to dynamic nature of Agile, size of user-story is not certain. New user-stories can be added or existing user-stories can be removed at any time. This creates uncertainty in Agile project that leads to poor estimation of time and thus cost. Due to this, all estimates

of project schedule in Agile are subject to uncertainty. This research work focuses on the reasons of existing uncertainty and proposes a new technique to reduce this. Various factors for existence of uncertainty are as below.

- There may be incomplete understanding of scope or incomplete understanding of work per scope.

- Sometimes team has imperfect understanding of known work or the team is not able forecast the unexpected work.

Consider an example of the story-point estimation in Agile by comparing it with a physical artifact like 'Remodeling a House'. Then the work will be started by breaking down the above project into a few smaller steps as below: *Remodel Kitchen, Remodel Bedroom, Remodel Living Room.*

Then the total work required for each of these steps is estimated such as "Remodel living room." Unfortunately, estimate will not be exact because of uncertainty. Suppose an estimate of "Paint living room" is 6 days, with an uncertainty of 2. The best case is 3 days, it is 3 days below the estimate, the most likely case is 6 days and the worst case is 12 days which is 6 days greater than the estimate.

4.4.1 Proposed Technique of Reducing Uncertainty in Story-points

Agile developers have to face the problem of release planning because of the dynamic nature of Agile. At any time new user-stories can be added or changes according to the requirements of client. Once the stories are defined, the development team will define the size of story. The size of user story is defined in term of number of days i.e. time needed to complete a user story but this time estimation contains uncertainty [87,88,89]. In this proposed technique of uncertainty management, three types of story-points are defined for reducing uncertainty as below:

Fastest story-points (FSP): is the minimum number of story-points required for an activity to be completed. For minimum number of story-points, supposition is made that all predecessor activities are completed as planning is done for them and also all the essential resources whether software or hardware are available when desired.

Practicable story-points (PSP): Most of the times, project executives are asked to suggest only one estimate. This is the estimate that goes to the upper management.

Fatalistic (Maximum) story-points (Max FSP): The fatalistic is the maximum number of story-points required to complete an activity. In this case, assumption is made that resources are not available when needed. Also the predecessor activities are not completed as planned.

In Agile uncertainty cannot be eliminated completely but when estimating work, some steps can be taken to reduce it. The proposed technique reduces uncertainty by reducing the size of the user-story to be estimated. While producing estimate, if number of items are less, then results will be more reliable. The proposed strategy of decreasing the size or "granularity" of items which are to be estimated improves accuracy and reduces uncertainty.

Consider previous example of remodeling a bedroom. It contains a number of steps. If the estimate of "Remodel Bedroom" is taken directly without breaking it into smaller steps like paint room, remove old carpet etc., then estimates will be uncertain but if the various smaller steps are taken into account then uncertainty can be removed to some extent.

The proposed strategy for reducing uncertainty is to break large specifications or work items into smaller pieces. Thus the work of "Remodel Bedroom" is divided into four smaller tasks: paint room, remove old carpet, interior decoration and install new carpet. An estimate for each of the smaller specification or task is produced. When these story-point estimates are added together, then the uncertainty of estimates will be reduced.

In proposed technique to reduce uncertainty in an Agile project in a better way, the stories must be divided into tasks as called as sub-stories by the development team. The task is smallest parts in which a story can be divided into. Next thing is to determine the fastest, practicable and fatalistic story-points by development team. Then the average story-points will be calculated by proposed Formula 4.2. Time of each task is combined together to estimate the size of the user story.

Then the average story-points are calculated by proposal formula

$$\text{Estimated Story-points} = \frac{FSP + 4 * PSP + MaxFSP}{6} \quad (4.2)$$

4.4.2 Proposed Rules for Breaking Stories into Sub-stories

If the sub-stories are very small then it will become difficult to analyze the stories, and delay the project completion. There is no use of reducing the size of stories less than a certain level where the relative uncertainty does not improve.

The proposed rule is to pick a granularity that

- Enable an acceptable level of uncertainty.
- Produce a set of specifications or tasks to estimate that are small enough to be sensible and practical.

4.4.3 Proposed Algorithm of Managing Uncertainty

The proposed algorithm as shown in Figure 4.3 describes the various steps involved in managing uncertainty with release planning in Agile environment.

1. *Identify user-stories.*
2. *Then divide the user-stories into sub-stories until they overlap with each other according to the proposed rule of granularity.*
3. *Then estimate fastest, practicable and fatalistic number of story-points in each sub-story.*
4. *Calculate estimate number of story-points for each sub-story by using proposed formula*

$$\textit{Estimated Story-points} = \frac{FSP + 4 * PSP + MaxFSP}{6} \text{ ---------4.2}$$

Figure 4.3: Proposed Algorithm for Reducing Uncertainty

- Identify user-stories.
- Then divide the user-stories into sub-stories until they overlap with each other according to the proposed rule of granularity.
- Then estimate fastest, practicable and fatalistic number of story-points in each sub-story.
- Calculate estimate number of story-points for each sub-story by using proposed formula

$$\text{Estimated Story-points} = \frac{FSP + 4 * PSP + MaxFSP}{6} \text{ ---------- (4.2)}$$

4.5 PROPOSED SPRINT-POINT BASED ESTIMATION ALGORITHM USING AGILE ESTIMATION FACTORS

The requirements are taken in the form of user-stories which are grouped in sprints and user-stories based estimation is done using story-points. When a team member estimates that a given task can be completed within 8 hours it does not mean that he can complete the task in 8 hrs. Because no one can sit in one place for the whole day and there can be a number of factors that can effect story-points and hence decrease the velocity.

To resolve this problem the concept of Sprint-point is proposed. A Sprint-point basically calculates the effective story-points. It is an evaluation unit of the user story instead of story-point. By using Sprint points, more accurate estimates can be achieved. The unit of effort is Sprint Point (SP). A Sprint Point is the amount of effort, completed in a unit time. The Sprint-Point Based Estimation Algorithm is as shown in Figure 4.4.

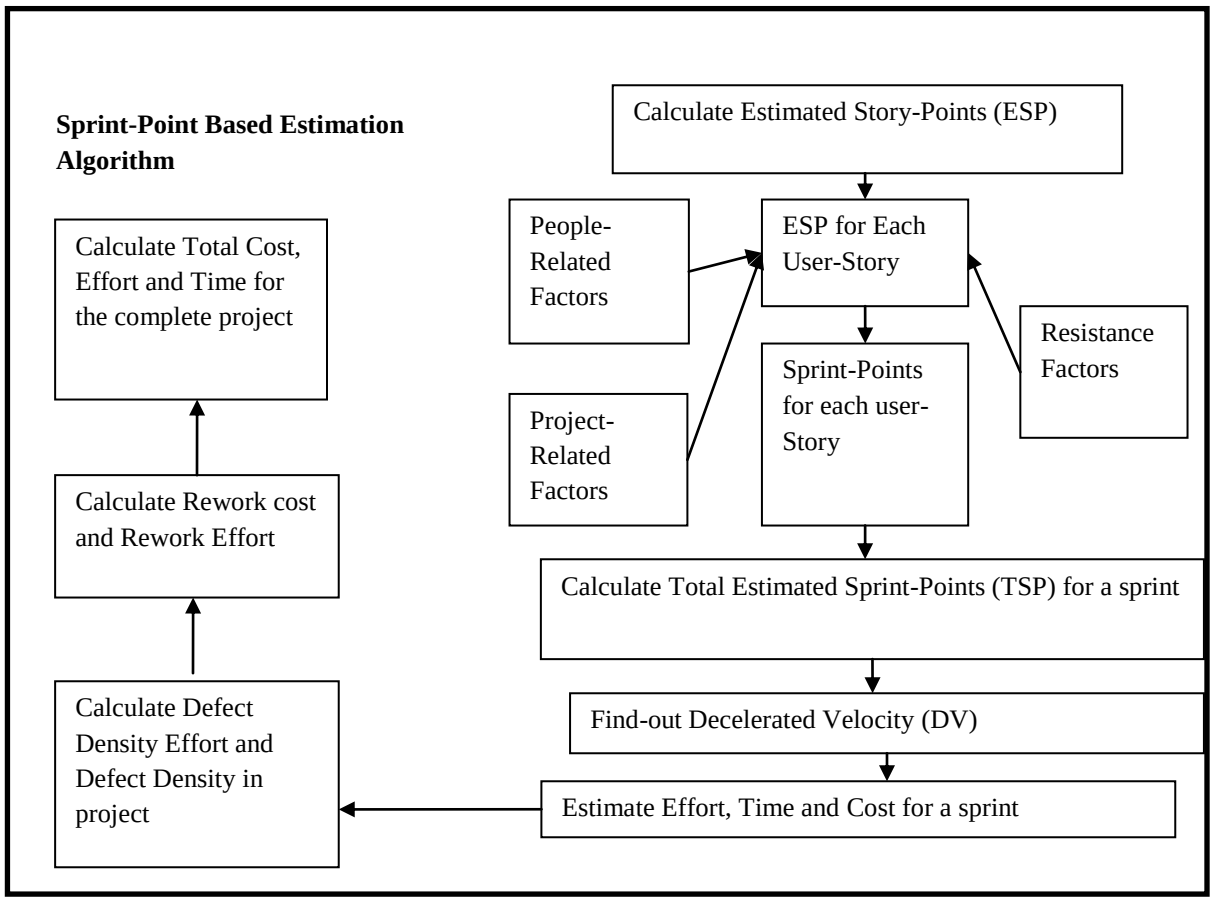


Figure 4.4: Sprint-Point Based Estimation Algorithm

4.5.1 Proposed Agile Estimation Factors

The project and people-related factors can increase or decelerate the velocity of project [80,84]. But the resistance factors always decelerate the velocity and affect on productivity, thus increases the duration of the project. If the duration of the project increases then the costs of the project also get affected. The various Agile estimation factors are shown in Figure 4.5.

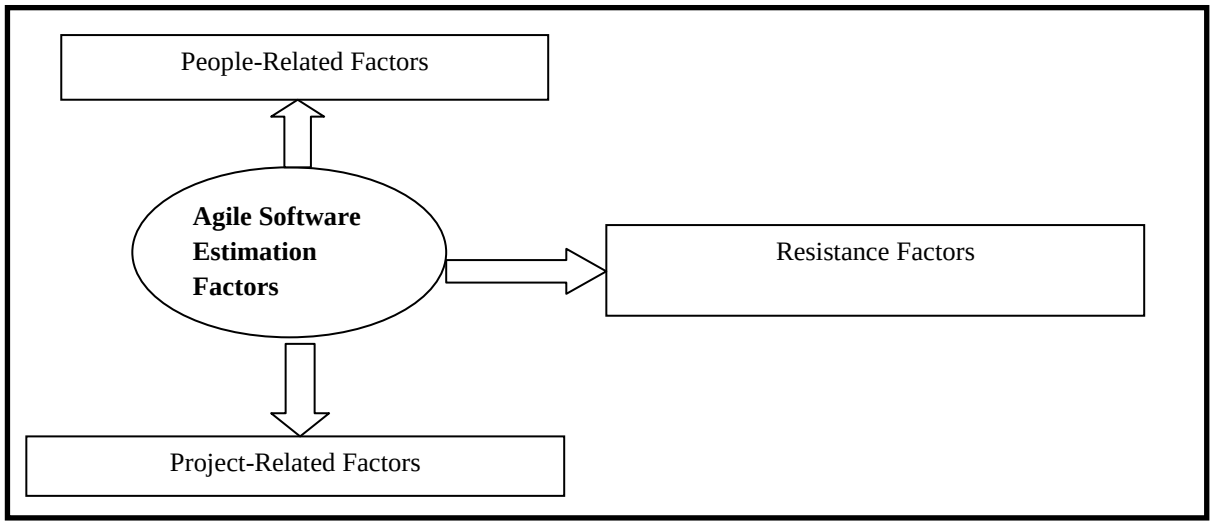


Figure 4.5:Agile Software Estimation Factors

4.5.1.1 Project-Related Factors

These factors are related to project like complexity of project, type of project and quality requirements etc. The various project related factors are shown in Table 4.2.

Table 4.2: Project-Related Factors

S.No	**Project-Related Factors**
1.	Project-domain
2.	Quality requirement
3.	Hardware and software requirements
4.	Operational ease
5.	Complexity
6.	Data transaction
7.	Multiple sites

A. Project domain: The project domain affects the cost, effort and duration of the project. The project can be of web based application, construction, new project development, information system, military project etc. Projects can be categorized based on unique characteristics.

B. Quality requirements: Project quality may be characterized as functionality, usability, effectiveness, maintainability and reliability and portability. The outcome product should develop in such manner that it meets the customer's requirements. These features of project increase the cost and duration of the project. For example, fault tolerance and recoverability are of primary concern if quality is considered.

C. Hardware and software requirements: Hardware and software requirements are basic need for the development of the project. All projects need certain hardware components or other software resources to be present on a computer. These prerequisites are known as system requirements. With increasing demand of new functionalities in newer versions of software, system requirements tend to enhance over time. System requirements depends on the project, the requirement can vary according to the project. These requirements affect the cost of the project. For example to run applet java virtual machine is necessary.

D. Complexity: It refers to how complex is to develop project. Technical complexity includes a number of aspects such as numbers of technologies are involved, number of technical interface. Management complexity includes project staffing and management etc. Complexity is major aspect in estimation of a project, as the complexity of the project increase Cost, Size and duration of project also increase. Military project are more complex as compared to information based system.

E. Operation ease: Operation ease refers to that how it is easy for a user to use and operate the product. It may be in the form of a better GUI to understand the functionality of the project. It increases the usage of the product which increases the quality and thus customer satisfaction.

F. Data transaction: Data transaction refers to the transfer of the data from one machine to another. Data can access from other machine as per requirement. Data transaction affects the estimation of the project, for example if high data transaction required it necessitate high security. That means high cost of the project.

G. Multiple sites: Multiple sites means software or project is developed on one workstation or it is developed on different workstations or sites and integrated later. Big size projects are broken in parts and developed at different sites according to the availability of requirement to develop project. If software runs on multiple sites or many team members work together in distributed environment, cost of the software will increase. Communication delay in distributed environment must be considered in estimation of the duration of project. High communication delay will increase the effort and duration of the project.

4.5.1.2 People Related Factors

People related factors are the factors which are people or team oriented. These factors affect the duration of the project and ultimately the cost of the project. People related factors are described in Table 4.3.

Table 4.3 People-Related Factors

S.No	People Related Factors
1.	Communication skills
2.	Familiarity in team
3.	Managerial skills
4.	Security
5.	Working time
6.	Past- project experience
7.	Technical ability

A. Communication-skills: Communication is an important part of life. Communication skills are essential in all areas of life. People in organizations usually spend 75 percent of their daily time on other activities like documenting, meetings, listening, e-mail checking and speaking etc. Communication skills also have the importance like technical skills in a Agile team. Communication skills are important in reducing the duration and cost of the

project, since if there is good communication within the team it will take less time to understand each other's work and worked efficiently and effectively.

B. Familiarity-in-team: Familiarity in team affects the team performance, namely team errors, i.e. errors that occur in the interactions of team members. It has been observed that there is a U-shaped relationship between team familiarity and team errors. Initially when familiarity in team member increases it reduces team errors; but they increase if team members become too familiar. Familiarity in team reduces the duration and effort up to a point but if team familiarity increased so much then the reverse impact can be on the performance. So familiarity in team is an important factor that affects the estimation of project.

C. Managerial-skills: Management is a tough job. Managerial skill is the ability to communicate with other persons in the department or organizations and the ability to understand their desire and persuade them to work as a team.

D. Security: Security may be considered as network security, functional security, code security, documentation security etc. based on customer requirements. At the time of project development life cycle security must be taken as higher priority factor. But it may increase the complexity of the project development and hence resulting into the increase in cost, size, effort and duration of project. For example, online money transaction software projects require various levels of securities to maintain the integrity of software. Banking system and Military projects also requires higher security; hence cost of these projects is high as compared to others.

E. Working-Time: Working time is the period of time that an individual spends at paid occupational labour. The working Time is defined as the period during which the worker is doing his work, at the employer's disposal and carrying out his or her duties, in accordance with national laws or practice.

F. Technical-ability: The technical skill means implies expertise of a team member in a specific kind of activity like any process or any tool related with the project. It involves knowledge in a specific area, critical ability within that area, and capability of using the tools and techniques of specific discipline. Technical skills involve process or technique knowledge and proficiency in a certain specialized field such as engineering, computer and accounting. It refers to a person's proficiency in any type of process or technique.

G. Experience of previous project: It basically involves specialized knowledge of managers with the previous projects. It refers to a person's past knowledge and proficiency of previous projects.

4.5.1.3 Resistance Factors

The resistance factors always decelerate the velocity of the project [86]. These factors have long-term affect. In Agile environment some resistance factors are shown in Table 4.4.

Table 4.4:Resistance Factors

S.No	Resistance Factors in Agile Environment
1.	Perfect team composition
2.	Working place uncomfort
3.	Drifting to Agile
4.	Team dynamics
5.	Expected team changes, other project responsibilities
6.	Introduction to new technology
7.	Usability
8.	Defects in third-party tools
9.	Stakeholder response
10.	Lack of clarity in requirements
11.	Volatility of requirements
12.	Change in working environment
13.	Prerequisite availability of resources

A. Perfect team composition: The most important feature of Agile teams is that the teams are small and must have the skills like design skills, database skills, testing skills and user interface skills. To compose such a team is a difficult task and it takes lots of time and effort.

B. Working place uncomfort: These factors affect the working place. These include interruptions, noise, poor ventilation, poor lighting, uncomfortable seating and desks, inadequate hardware and software etc.

C. Drifting to Agile: With the introduction of Agile in a organization it is needed to change the complete process of organization which is again a resistance factor.

D. Team dynamics: Team members need to interact frequently with the entire team during meetings, pair programming, or discussions throughout the project. All team members participate in `daily stand-up meetings' using technology-mediated communication. Since Agile primarily encourages open communication and emphasizes effective exchange of thoughts within the team and this takes more time.

E. Expected team change and outside project responsibilities: This factor describes the change in team. Some team members may be added and some team members may leave the project and sometimes the responsibilities of the team members may also change. This factor affects the duration as well as the effort of the project. Switching between the projects may be done and due to this the duration of the project is affected.

F. **Introduction of new technology**: With the introduction to new technologies in software industry, the duration of the project is affected. The members are required to learn these technology which takes more time.

G. Usability: Usability means how is it easy for user to use or operate the product. It may be in the form of a better GUI, to understand the functionality of the product, minimum

user input etc. It increases the quality of the software as well as customer approval and satisfaction. Better GUI include numerous activities which increase the cost of a software project.

H. Defects in third party tools: Various projects require third party tools and software for the implementation as well as design. Some defects may also arise in these tools and software and thus they affect the duration and cost of the project.

I. Stakeholders response: The involvement of a stakeholder is required at almost every stage of the development life cycle. But sometimes they do not respond to the requests for information from the developers and sometimes they are not present at the time of meeting .So certain decisions get delayed due to the absence of the stakeholders. Thus directly or indirectly they affect the duration of the project.

J. Lack of clarity in requirements: Sometimes lack of clarity in the requirements causes the change in duration as well as cost of the project. Requirements are gathered in the beginning of the project and on that basis the task is performed, but if the requirements are not clear no task can be started.

K. Volatility of requirements: Agile promotes volatility, which means that the requirements can be changed at any point of time in the project. Due to change in requirement there may be a situation where more tools are required which affect the cost and duration of the project.

L. Change in working environment: Change in the working environment affects the duration of the project as at new place to install proper hardware or software takes more time thus affects cost of the project.

M. Prerequisite availability of resources: Resources are money, people, material, technology, space and other asset which are necessary for effective operation. The

availability of resource means that the resource should be available when required. Prerequisite availability has vital role in Agile environment because the task get delay if resource is not available at that time.

4.5.2Velocity Factor and Complexity Factor

Agile estimation factor decides the velocity factor and complexity factor of the project.

Velocity Factor: It refers to the velocity of the project. If the factors are taken at low level then it means that the velocity is not very much affected, but if the level of factors is high then velocity will be affected more. The rating of factors is shown in Table 4.5.

Table 4.5:Rating of Velocity Factor

S.No	Level of Factor	Rating	Type of project
1.	Low	0.94—0.98	Project is simple. For example requirements are very straightforward, no volatility of requirements, All business and technical requirements are very clear to the team with no uncertainty, No research required in the project and it requires basic programming skills to complete.
2.	Medium	0.90—0.94	Project is Moderately complex. For example it requires little or no research and team has strong expertise in allotted work.
3.	High	0.85—0.89	Project is extremely complex and demands accurate estimates by consideration of all the factors at a high level. For example the project requires specific expertise or skill set that is important, but missing in the team .The project requires extensive research.

Complexity Factor: It refers to how complex is to develop project. Technical complexity includes a number of aspects such as numbers of technologies are involved and number of technical interfaces. To accommodate all characteristics of Agile software development methodology, the complexity relating to each project is rated; if factors are

at low level then complexity is less, if the level of factor increases then complexity becomes high. The rating of complexity factor is shown in Table 4.6.

Table 4.6: Rating of Complexity Factor

S.No	Level of factor	Rating	Type of Project
1.	Low	1	Project is simple. For example requirements are very straightforward no volatility of requirements, No research required in the project and it requires basic programming skills to complete. All business and technical requirements are very clear to the team with no uncertainty. There is no product uncertainty, process uncertainty and resource uncertainty. Team of right ability and experience is available.
2.	Medium	3	Project is moderately complex. For example it requires little or no research and team has strong expertise in allotted work.
3.	High	5	Project is extremely complex and demands accurate estimates by consideration of all the factors at a high level. For example the project requires specific expertise or skill set that is important, but missing in the team .The project requires extensive research.

4.5.3 PROPOSED REGRESSION TESTING EFFORTS IN SPRINT-POINT BASED ESTIMATION ALGORITHM

Regression testing is done in Agile to make sure that the new incorporated changes should not have side effects on the existing functionalities and thereby finds the other related bugs. Regression testing is applied to code immediately after changes are made.

The main goal of regression testing is to assure that the changes have no unintended effects on the behavior of the software. These effects may be either in the software being tested, or in another related software component. It is often necessary to ensure software quality. Thus regression testing may consume much time, cost and effort in the project and therefore it is considered as an expensive process. So there is a need for a technique to calculate regression testing efforts in Agile.

The proposed scenario of regression testing in Agile shows that the iterations in Agile are of short duration [82,94]. The very first day of sprint is fixed for sprint planning, decision making and for meetings with the developer and tester. Thereafter the actual work starts, the development takes place for one sprint in first iteration and in second iteration it is sent to the tester for testing work. The tester tests the first sprint meanwhile the developer starts his work on the second sprint, when the second sprint is ready for the testing then the regression testing of first sprint is started. When the third sprint is under development then regression testing of first and second sprint is done. This process continues till all the sprints are developed and are completely tested.

In proposed scenario of regression testing as shown in Figure 4.6 the rework effort and cost is calculated by finding out defects in each sprint as given below.

- *Rework effort*= Total number of defects /Defect fixing effort(DFE)

- *Defect Fixing Effort(DFE)*= Number of defects fixed per hour*No of working hours per day *No of persons working

- *Rework Cost*=Rework effort*Cost of one employee*Number of employees.

- *Defect Density of one sprint*=Defect in that sprint / Total sprint points that iteration is covering.

For regression testing in a particular sprint total number of defects fixed per hour are need to be calculated. On the basis of defect fixing effort rework effort is calculated. This rework effort is called as regression effort of the system which is then used to find out the rework cost of the sprint.

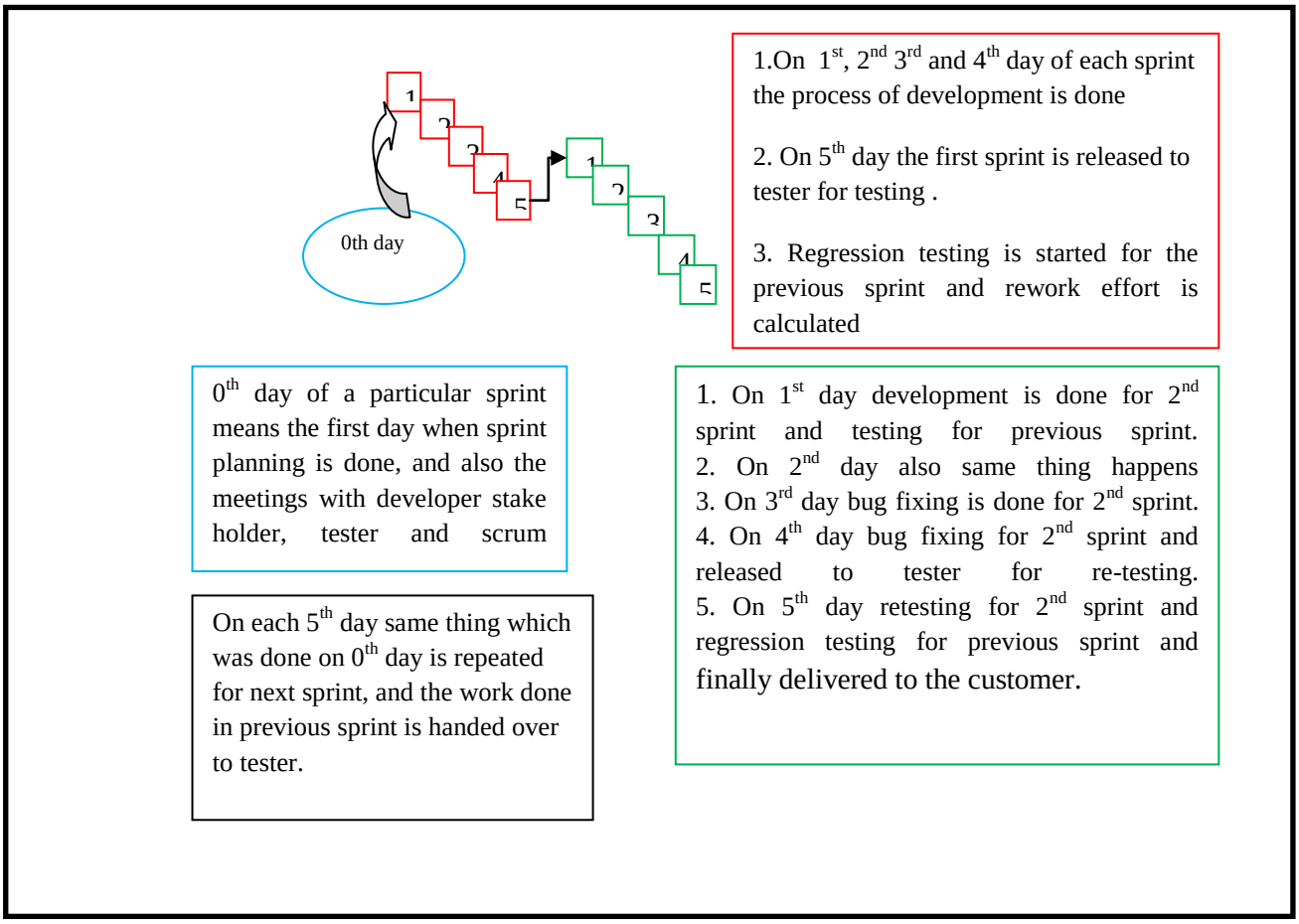


Figure 4.6: Proposed Regression Testing Scenario

4.5.4 Proposed Sprint-point based Estimation Framework Algorithm

The proposed algorithm explains the various steps involved in estimating a project in Agile environment as shown in Figure 4.7.This algorithm calculates total estimated cost, effort and time of the project by using proposed Agile estimation factors like people-related, project-related and resistance factors. Also in this algorithm regression testing effort is calculated.

- *Identify the number of user-stories.*
- *Find out estimated story-points (ESP) by using three types of story-points.*
- *Identify the People-related, Project-related and resistance factors which affects the story points in Agile Scrum environment where P= {$p_1, p_2,....p_i...,p_{14}$ }, where 1< i <=27.*
- *Identify the level set L for all factors where L= {1, 2,3},If L=1 then level of factors is low , If L=2 then level of factors is medium, If L=3 then level of factors is high.*
- *Assign the Unadjusted value of sprint point (UVSP) corresponding to each level,If L=1 then UVSP = 1 ,If L=2 then UVSP = 3 ,If L=3 then UVSP = 5*
- *Compute the Sprint Points(SP) as*
    - *SP=ESP+0.1(UVSP)*
- *Compute the Velocity from first iteration as*
    - *V = Sprint point completed in one iteration/ Sprint point in one user story.*
- *Assign Velocity factor (VF) depending upon the velocity to perform the task and complexity factor depending upon the complexity of factor (CF) in all the cases.*
- *Compute the Decelerated Velocity by considering various factors to optimize the velocity*
    - *DV= V*VF*
- *Compute the Estimated development time required for the Scrum project*
    - *Estimated Development Time (EDT) = SP/Velocity (in Days)*
- *Compute Total Estimated Effort (TEE)*
    - *TEE=SP+ Complexity factor (CF)*
- *Compute Total Estimated Cost (TEC)*
    - *TEC=TEE*Cost per person*
- *Compute Defect Fixing Effort (DFE)*
    - *DFE=Number of defects fixed per hour*Number of working hours per day*Number of persons working*
- *Compute Rework Effort(RE)*
    - *RE=Total number of defects/DFE*
- *Compute Total Estimated Effort using Regression testing (TEERT)*
    - *TEERT=TEE+RE*
- *Compute Rework Cost(RC)*
    - *RC=RE*cost of one employee*number of employees*
- *Compute Total Estimated Cost using Regression testing(TECRT)*
    - *TECRT=TEC+RC*

Figure 4.7: Proposed Sprint-Point Based Estimation Framework Algorithm

4.6 CONCLUSION

The purpose of this research work is to develop an algorithm for estimation in scrum which can calculate accurate cost, effort and duration of the project. Due to dynamic nature of Agile there exists uncertainty which cannot be eliminated completely but when estimating work, some steps can be taken to reduce it. The proposed technique reduces uncertainty by reducing the size of the user-story to be estimated. The fewer the elements or specifications that are to be considered while producing an estimate, the more reliable will be the result. The proposed strategy of decreasing the size or granularity of items to be estimated improves accuracy and reduces uncertainty.

In this work people-related, project-related and resistance factors in Agile environment are proposed that impact the estimation of the project. Sprint-point highly depends on the value of these factors. The approach developed is really simple and easy to understand and can be effectively used for estimation in Agile environment.

Using this sprint-point based estimation framework the estimation of small and medium size project can be calculated efficiently. Further as in Agile projects regression testing is very important. So, the estimation is done by using regression testing cost and effort.

Chapter V

ANALYSIS AND IMPLEMENTATION

5.1 INTRODUCTION

In the previous chapters the following approaches have been proposed for Agile estimation**.**

- **User-story Based Prioritization Algorithm:** The proposed user-story based prioritization algorithm suggests a method of prioritization that helps in choosing the optimal order of user stories.

- **Managing Uncertainty in Estimating User-stories:** The proposed technique of managing uncertainty in Agile reduces uncertainty by reducing the size of the user-story to be estimated.

- **Sprint-Point Based Estimation Algorithm:** Looking towards the various unaddressed problems of estimation, a new sprint-point based estimation framework in Agile has been proposed based on various Agile estimation factors and regression testing efforts. This algorithm helps to estimate the accurate cost, time and effort.

To analyze the efficacy of the proposed approaches a case study has been taken, which is discussed below.

5.2 CASE STUDY

For case study, the user-stories of a software project named as 'enable quiz' are considered.' This 'enable quiz' is a lightweight technical quizzing solution for companies that recruit engineers. This software will allow software companies to screen job candidates in a better way and assess their internal talent for skills improvement. For simplicity product backlog of only ten user stories of this software has been taken. The user-stories are as shown in Table 5.1

Table 5.1: User-stories of the Case Study

S.No	User-Story
1.	As a manager, I want to browse my existing quizzes.
2.	As a manager, I can make sure I'm subscribed to all the necessary topics for my skills audit.
3.	As a manager, I can add additional technical topics to my quizzes.
4.	As a manager, I want to create a custom quiz bank
5.	As a manager, I want to create a quiz so I can use it with my staff.
6.	As a manager, I want to create a list of students from an Excel file so I can invite them to take the quiz.
7.	As a manager, I want to create a list of students online.
8.	As a manager, I want to invite a set of students.
9.	As a manager, I want to see which students have completed the quiz.
10.	As a manager, I want to see how the students scored on the test so I can put in place a skills improvement program.

5.3 USER-STORY BASED PRIORTIZATION ALGORITHM

The proposed user-story based prioritization algorithm discussed in chapter 4 explains the various steps involved in prioritizing the product backlog in Agile environment. This algorithm has been implemented on the case study for prioritization of user-stories. In the case study the importance of user-story from customer and effort of user-story from developers has been taken and then importance to effort (I/E) ratio for each user-story is used for prioritization as shown in Table 5.2.

Table 5.2: User-Story based Prioritization Algorithm

S.No	User-Story	Prioritization Algorithm		
		I	E	I/E
US-01	As a manager, I want to browse my existing quizzes	10	21	0.4761
US-02	As a manager, I can make sure I'm subscribed to all the necessary topics for my skills audit.	4	26	0.1538
US-03	As a manager, I can add additional technical topics to quizzes	1	14	0.0714
US-04	As a manager, I want to create a custom quiz bank	8	17	0.4705
US-05	As a manager, I want to create a quiz so I can use it with my staff.	5	13	0.3846
US-06	As a manager, I want to create a list of students from an Excel file so I can invite them to take the quiz.	2	7	0.2857
US-07	As a manager, I want to create a list of students online.	1	3	0.3333
US-08	As a manager, I want to invite a set of students.	3	32	0.0937
US-09	As a manager, I want to see which students have completed the quiz.	9	22	0.4090
US-10	As a manager, I want to see how the students scored on the test so I can put in place a skills improvement program.	2	25	0.08

After calculation of I/E ratio a graph is plotted between (I/E) ratio and user-story. The User-story is chosen from top to down as the highest bar has highest priority. The result shows that higher the I/E, the more is priority of the user-story as shown in Figure 5.1 which shows that the priority order of user-stories is user-story 9,1,6,2,5,4,8,7,10,3.

Now if it is assumed that there are total of three sprints in the project then the result of user-story based prioritization algorithm are shown in Table 5.3 which shows that which sprint will cover a particular user-story. The result of user-Story based prioritization algorithm shows that the sprint first consist of user-stories US-01, US-04, US-09. Sprint second consist of user-stories US-02, US-05, US-06, US-07 and sprint third has user-stories US-03, US-08, US-10.

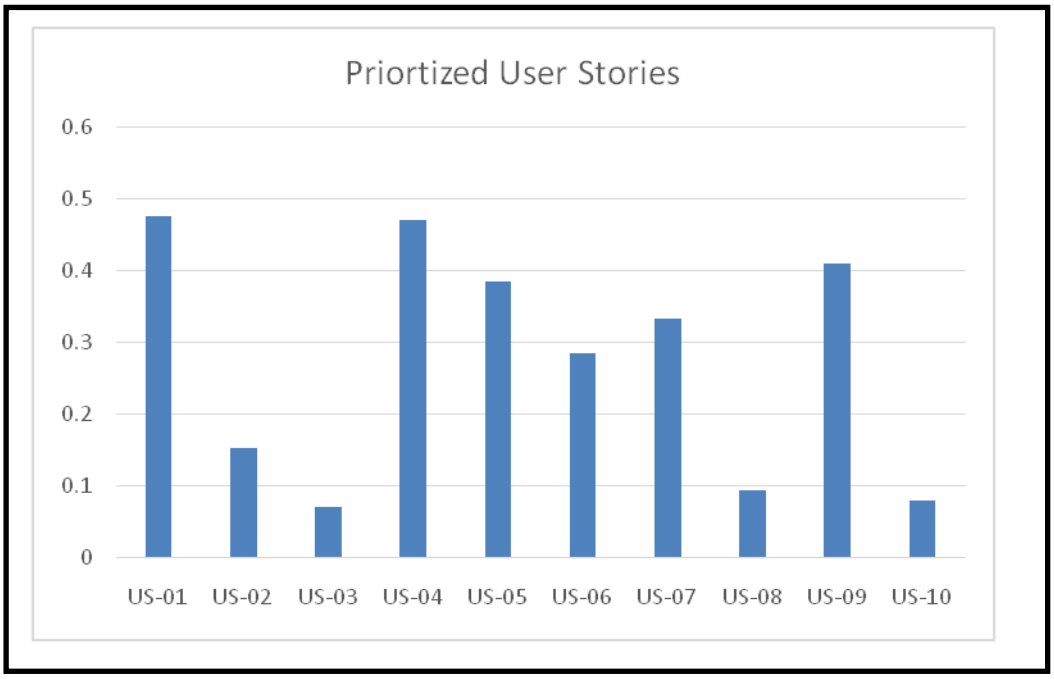


Figure 5.1: Prioritization of User-stories

Table 5.3: User-story and the Sprint Covering User-story

S.No	User-Story	Sprint
US-01	As a manager, I want to show all the existing quiz questions	1
US-02	As a manager, I should be sure that I'm subscribed to all the related topics of my skills	2
US-03	As a manager, I can add more questions and topics to quizzes.	3
US-04	As a manager, I can create a customized questionnaire	1
US-05	As a manager, I can share quiz with my staff	2
US-06	As a manager, I can create a list of students for all the related topics	2
US-07	As a manager, the quiz can be made online to students	2
US-08	As a manager, I can invite foreign university and student	3
US-09	As a manager, I can see the quiz to check students	1
US-10	As a manager, I can start a skill improvement program	3

5.4 MANAGING UNCERTAINITY IN STORY-POINTS

The proposed technique reduces uncertainty by reducing the size of the user-story to be estimated. According to the proposed approach of managing uncertainty in story-points,

user-stories are divided into small sub-stories as shown in Table 5.4. After that the fastest, practicable and fatalistic story-points for each sub-stories is calculated according to proposed formula as in Table 5.5. The estimated story-points for each user-story are calculated as shown in Table 5.6.

Table 5.4: Divided User-stories into Sub-stories

S.No	User-story	Sub-stories
US-01.	As a manager, I want to show all the existing quiz questions	1.1 Create question bank with categories 1.2 Show question bank organized categories
US-02.	As a manager, I should be sure that I'm subscribed to all the related topics of my skills.	2.1 Subscription Form 2.2 Show recommended Topics
US-03.	As a manager, I can add more questions and topics to my quizzes.	3.1 Update bank question 3.2 Update topics and related question
US-04.	As a manager, I can create a customized questionnaire	4.1 Special question bank 4.2 Create and update question
US-05.	As a manager, I can share quiz with my staff	5.1 Share option (optional)
US-06.	As a manager, I can create a list of students for all the related topics	6.1 Create Student record 6.2 Add/ Remove Student 6.3 View Student record
US-07.	As a manager, the quiz can be made online to students	7.1 Web module interaction 7.2 Store result and show them
US-08.	As a manager, I can invite foreign university and student	8.1 Special guest accounts 8.2 User addition and organized addition
US-09.	As a manager, I can see the quiz to check students	9.1 Monitor result and analysis report 9.2 Show graphs , highest marks
US-10.	As a manager, I can start a skill improvement program	10.1 Special program of difference quizzes 10.2 Let any user be added

Table 5.5: Sub-stories, Fastest, Practicable, Fatalistic, Estimated Story-points

S.No	Sub-stories	FSP	PSP	MaxFSP	ESP
US-01	1.1 Create question bank with Categories	10	15	20	15
	1.2 Show question bank Organized Categories	3	5	10	6
	2.1 Subscription Form	10	15	26	16

US-02	2.2 Show recommended Topics	20	27	42	28
US-03	3.1 Update bank question	8	19	33	20
	3.2 Update topics and related question	6	10	20	11
US-04	4.1 Special question bank	10	16	30	17
	4.2 Create and update question	14	32	45	31
US-05	5.1 Share option (optional)	17	30	50	31
US-06	6.1 Create Student record	20	38	54	38
	6.2 Add/ Remove Student	8	15	32	17
	6.3 View Student record	20	26	34	26
US-07	7.1Web module interaction	7	17	28	17
	7.2 Store result and show them	5	22	30	21
US-08	8.1 Special guest accounts	11	22	40	23
	8.2 User addition and organized addition	13	16	23	17
US-09	9.1 Monitor result and analysis report	14	18	25	19
	9.2 Show graphs , highest marks	19	27	35	27
US-10	10.1 Special program comprising of difference quizzes	17	22	25	22
	10.2 Let any user be added	9	13	18	13

Table 5.6: User-Stories, Estimated Story-points (ESP)

S.No	User-Story	(ESP)
US-01.	As a manager, I want to show all the existing quiz questions	21
US-02	As a manager, I should be sure that I'm subscribed to all the related topics of my skills	44
US-03	As a manager, I can add more questions and topics to my quizzes.	31
US-04.	As a manager, I can create a customized questionnaire	48
US-05.	As a manager, I can share quiz with my staff	31
US-06	As a manager, I can create a list of students for all the related topics	81
US-07.	As a manager, the quiz can be made online to students	38
US-08.	As a manager, I can invite foreign university and student	40
US-09.	As a manager, I can see the quiz to check students	46
US-10.	As a manager, I can start a skill improvement program	35

This technique of managing uncertainty of story point calculates estimated story-points for each sprint. Sprint first has user-stories US-01, US-04, US-09.So total ESP for first sprint are 115. The user-stories of Sprint second has user-stories US-02, US-05, US-06,

US-07.So total ESP for sprint second are 194. Sprint third has user-stories US-03, US-08, US-10.So, total ESP for third sprint are 106.

5.5 SPRINT-POINT BASED ESTIMATION ALGORITHM

In this algorithm sprint-points are calculated from estimated story-points. For checking the feasibility of the algorithm four cases have been considered. In case 1 for estimation of the user-stories the factors are not applied and estimation is done on the basis of estimated story-points instead of sprint-points. In case 2 all the factors are taken at low level (L=1), in case 3 the proposed factors are considered at medium level (L=2) and in case 4 the factors are at the high level (L=3).

- If L=1 then UV = 1 ,
- If L=2 then UV = 3 ,
- If L=3 then UV = 5

For each user-story, firstly UVSP is calculated, after that by applying the formula sprint-point is calculated.

*Unadjusted Value of Sprint-Points (UVSP) =Estimated Story-Points (ESP)*UV*

Sprint-points (SP) =ESP+0.1(UVSP)

Total Sprint-Points in the project (TSP) =∑ SP

For each user-story UVSP for each user-story is calculated in Table 5.7 and SP is calculated in Table 5.8

Table 5.7: Calculation of UVSP

S.No	User-story	Case 1	Case 2	Case 3	Case 4
US-01	As a manager, I want to show all exiting quiz questions	21	21	63	105
US-02.	As a manager, I should be sure that I'm subscribed to all the related topics of my skills	44	44	132	220
US-03.	As a manager, I can add more questions and topics to my quizzes.	31	31	93	155

US-04.	As a manager, I can create a customized questionnaire	48	48	144	240
US-05.	As a manager, I can share quiz with my staff	31	31	93	155
US-06.	As a manager, I can create a list of students for all the related topics	81	81	243	405
US-07.	As a manager, the quiz can be made online to students	38	38	114	190
US-08.	As a manager, I can invite foreign university and student	40	40	120	200
US-09.	As a manager, I can see the quiz to check students	46	46	138	230
US-10.	As a manager, I can start a skill improvement program	35	35	105	175

Table 5.8: Calculation of Sprint-points

S.No	User-story	Case 1	Case 2	Case 3	Case 4
US-01.	As a manager, I want to show all the existing quiz questions	21	23.1	27.3	31.5
US-02.	As a manager, I should be sure that I'm subscribed to all the related topics of my skills	44	48.4	57.2	66
US-03.	As a manager, I can add more questions and topics to my quizzes.	31	34.1	40.3	46.5
US-04.	As a manager, I can create a customized questionnaire	48	52.8	62.4	72
US-05.	As a manager, I can share quiz with my staff	31	34.1	40.3	46.5
US-06.	As a manager, I can create a list of students for all the related topics	81	89.1	105.3	121.5
US-07.	As a manager, the quiz can be made online to students	38	41.8	49.4	57
US-08.	As a manager, I can invite foreign university and student	40	44	52	60
US-09.	As a manager, I can see the quiz to check students	46	50.6	59.8	69
US-10.	As a manager, I can start a skill improvement program	35	38.5	45.5	52.5
		415	**456.5**	**539.5**	**622.5**

The various Agile estimation factors, velocity factor and complexity factor is shown in the three cases at the different level is shown in Table 5.9. If the factors are taken at low level then it means that the velocity is not very much affected, but if the level of factors is high then velocity will be affected more.

Table 5.9: Agile Estimation Factors

Agile Estimation Factors							
S.No	***Factor***	***Case1(Low Level)***		***Case 2(Medium Level)***		***Case 3(High Level)***	
		VF	*CF*	*VF*	*CF*	*VF*	*CF*
1	Type of project	0.98	0.1	0.94	0.3	0.89	0.5
2	Quality Requirement	0.96	0.1	0.91	0.3	0.87	0.5
3	Hardware and software requirements	0.97	0.1	0.92	0.3	0.87	0.5
4	Ease of operation	0.96	0.1	0.92	0.3	0.86	0.5
5	Complexity	0.95	0.1	0.93	0.3	0.87	0.5
6	Data transaction	0.98	0.1	0.94	0.3	0.87	0.5
7	Technical ability	0.96	0.1	0.91	0.3	0.86	0.5
8	Tool availability	0.97	0.1	0.92	0.3	0.85	0.5
9	Multiple site	0.96	0.1	0.94	0.3	0.86	0.5
10	Communication skill	0.95	0.1	0.94	0.3	0.87	0.5
11	Familiarity in team	0.98	0.1	0.94	0.3	0.87	0.5
12	Managerial skill	0.96	0.1	0.91	0.3	0.86	0.5
13	Security	0.97	0.1	0.92	0.3	0.86	0.5
14	Working time	0.96	0.1	0.94	0.3	0.87	0.5
15	Perfect Team composition	0.95	0.1	0.92	0.3	0.87	0.5

16	Working place uncomfort	0.98	0.1	0.93	0.3	0.87	0.5
17	Drifting to Agile	0.96	0.1	0.91	0.3	0.85	0.5
18	Team dynamics	0.97	0.1	0.93	0.3	0.86	0.5
19	Expected team changes	0.96	0.1	0.94	0.3	0.85	0.5
20	Introduction to new technology	0.95	0.1	0.93	0.3	0.85	0.5
21	Usability	0.98	0.1	0.94	0.3	0.86	0.5
22	Defect in third-party tools	0.96	0.1	0.91	0.3	0.87	0.5
23	Stakeholders response	0.97	0.1	0.92	0.3	0.87	0.5
24	Lack of clarity in requirements	0.96	0.1	0.94	0.3	0.89	0.5
25	Volatility of requirements	0.95	1	0.94	3	0.87	0.5
26	Change in working environment	0.96	1	0.92	3	0.89	0.5
27	Availability of Resources	0.97	1	0.92	3	0.86	0.5

The Table 5.10 shows user stories of the case study under taken and related story-points and sprint number. Consequently Table 5.11 shows the calculation of sprint-points in the case study.

Table 5.10: User-story and Sprint Covering the User-story

S.No	**User-story**	**Case 1**	**Case 2**	**Case 3**	**Case 4**	**Sprint**
US-01	As a manager, I want to show all the existing quiz questions	21	23.1	27.3	31.5	1
US-02	As a manager, I should be sure that I'm subscribed to all the related topics of my skills	44	48.4	57.2	66	2
US-03	As a manager, I can add more questions and topics to my	31	34.1	40.3	46.5	3

	quizzes.					
US-04	As a manager, I can create a customized questionnaire	48	52.8	62.4	72	1
US-05	As a manager, I can share quiz with my staff	31	34.1	40.3	46.5	2
US-06	As a manager, I can create a list of students for all the related topics	81	89.1	105.3	121.5	2
US-07	As a manager, the quiz can be made online to students	38	41.8	49.4	57	2
US-08	As a manager, I can invite foreign university and student	40	44	52	60	3
US-09	As a manager, I can see the quiz to check students	46	50.6	59.8	69	1
US-10	As a manager, I can start a skill improvement program	35	38.5	45.5	52.5	3

Table 5.11: Calculation of Total Sprint-points for each Sprint

Sprint no	**User stories covered in that sprint**	**No of defects found after regression**	**Total Sprint-points in that sprint**			
			Case 1	**Case 2**	**Case 3**	**Case 4**
1	1,4,9	0	115	126.5	149.5	172.5
2	2,5,6,7	20	194	213.4	252.2	291
3	3,8,10	37	106	116.6	137.8	159

5.6 RESULTS OF SPRINT-POINT BASED ESTIMATION ALGORITHM

In the sprint-point based estimation algorithm various inputs are supposed like the total number of user-stories, number of working days per month, number of working hours per

day, total number of working hours etc. The various inputs to the proposed algorithm are shown in Table 5.12 and results are shown in Table 5.13, Figure 5.2 and Figure 5.3.

Table 5.12:Inputs to the Proposed Algorithm

S.No	Inputs	Value
1.	No. of User-stories	10
2.	ESP	415
3.	Initial velocity	6SP/Day
4.	No. of working days per month	22 Days
5.	No. of working hours per day	8 hrs
6.	Cost of team per day	100$
7.	Defects Fixed per hour	2
8.	No. of team members	4
9.	No. of effective working hours	5
10.	Defect Fixing Effort(DFE)= no of defects fixed per hour*no of effective working hours per day *no of persons working	40

Table 5.13: Results of Sprint-Point based Estimation Algorithm

S.No	Estimation of cost, effort and Time				
	Values	Case 1	Case 2	Case 3	Case 4
1.	ESP	407	407	407	407
2.	TSP	415	456.50	539.50	622.50
3.	VF	1	0.96	0.93	0.87
4.	DV=V*VF	6	5.78	5.56	5.20
5.	EDT=TSP/DV(Days)	69.17	78.92	96.99	119.76
6.	TEE=TSP*CF(Man-Days)	41.5	45.65	161.85	311.25
7.	TEC=TEE*cost of team per day	4150	4565	16185	31125
8.	Rework Effort(RE)= Total number of defects/DFE	3.725	3.725	3.725	3.725
9.	Total Estimated Effort using Regression testing (TEERT)= TEE+RE	45.225	49.375	165.575	314.97

S.No	Estimation of cost, effort and Time				
	Values	Case 1	Case 2	Case 3	Case 4
10.	Rework Cost(RC)= RC=RE*cost of team per day	372.5	372.5	372.5	372.5
11.	Total Estimated Cost using Regression testing(TECRT) TECRT=TEC+RC	4522.5	4937.5	16557.5	31497.5

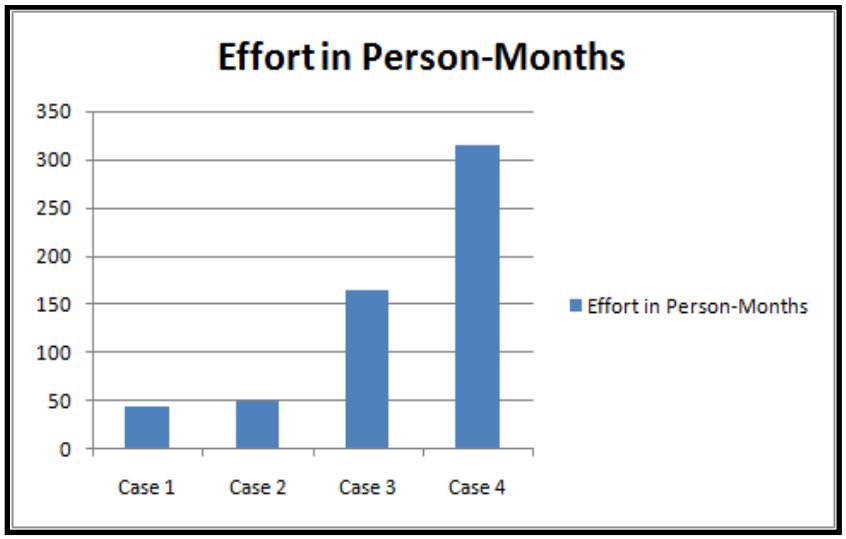


Figure 5.2: Effort in Person-Months

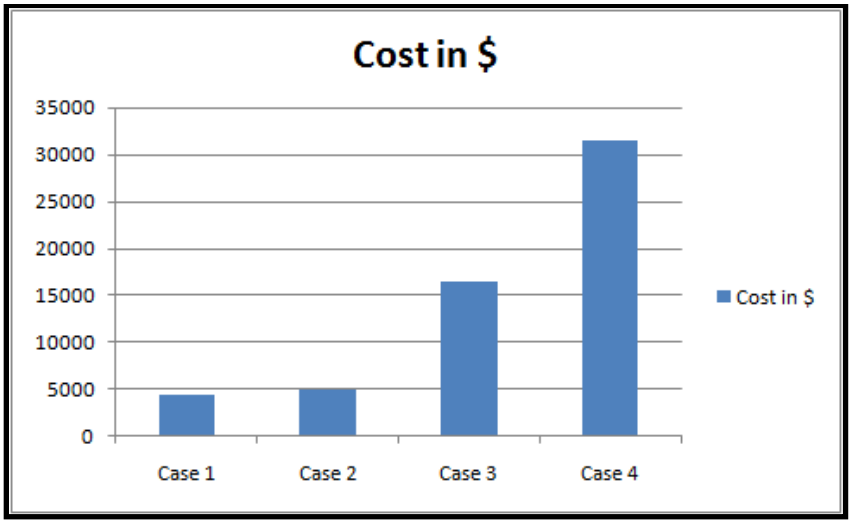


Figure 5.3:Total Estimated Cost of the Project in $

5.7 SPRINT POINT BASED ESTIMATION (SPBE) TOOL

As Agile projects are of small duration so the team has not so much amount of time to apply the mathematical algorithms**.** To resolve this issue a new Sprint–point based estimation tool(SPBE) is designed and developed in Excel to automate the sprint-point based estimation framework.The proposed SPBE tool for estimation place major emphasis on accurate estimates of effort, cost and release date by constructing detailed requirements as accurately as possible. This tool is used as a vehicle to validate the feasibility of the project.

The proposed tool is a set of individual spreadsheets with data calculated for each team separately. The estimation tool is created to provide more accuracy in velocity calculations, as well as better visibility through burn-down charts on all stages including planning, tracking and forecasting. The proposed estimation tool first decides the priority sequence of user-stories that dictates the implementation order. The priority of user-story is decided based on the importance of user-stories to the client and the effort of the scrum team. After prioritization product backlog is prepared which is the most important artifact for gathering the data. After selecting a subset of the user-stories, the sprint backlog is prepared and the period for the next iteration is decided. The tool calculates the estimated story-points for each sprint. After that the story-points are converted to sprint-points.Then sprint-point based estimation algorithm is used in the tool to estimate total cost and effort for the project.

5.7.1 Contents of SPBE Tool

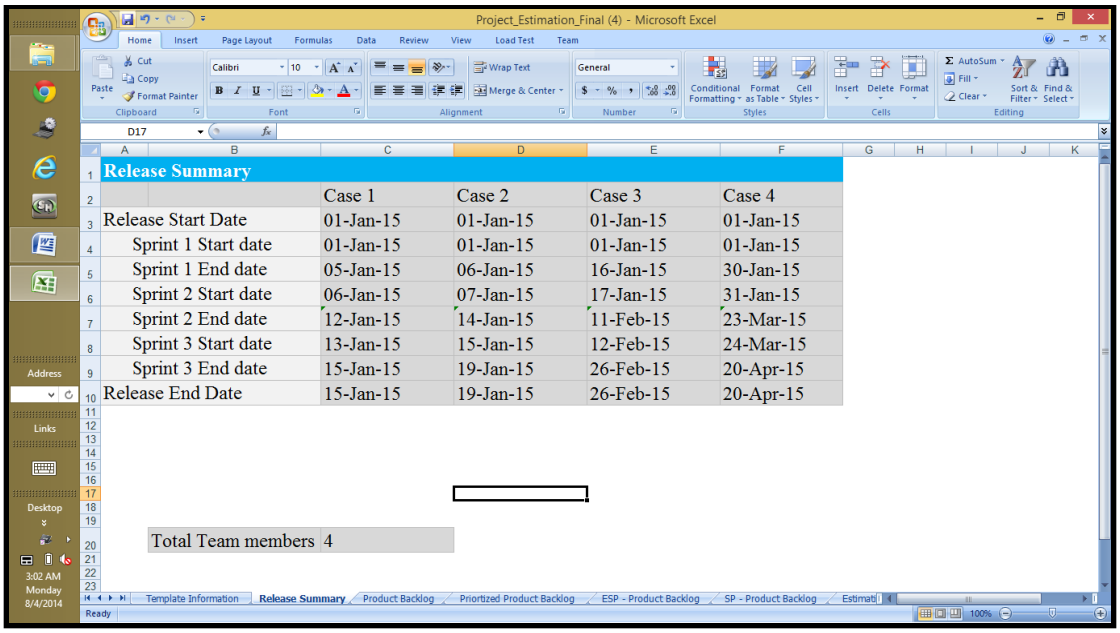
The SPBE tool contains the components. There is a separate spreadsheet for each component as in Table 5.14 like release summary, capacity management, product backlog, sprint backlog, sprint summary, defect, risk register, requirement issue log and metric analysis.

Table 5.14: Contents of SPBE Tool

S.No	Spreadsheet name	Description
1.	Release Summary	This spreadsheet contains the information about the overall planned and realized size of each release.
2.	Product Backlog	This spreadsheet lists all the user-stories in prioritized order.
3.	ESP-Product Backlog	This spreadsheet lists all the user-stories and the story-points in each user-story.
4.	TSP-Product Backlog	This spreadsheet lists all the user-stories and the sprint-points in each user-story.
5.	Estimation Summary	This spreadsheet calculates the total estimated effort, cost and time for the release.
6.	Sprint Backlog	This spreadsheet is a list of tasks identified by the Scrum team to be completed during the particular sprint
7.	Sprint Summary	This spreadsheet contains information like start date, end date of sprint.
8.	Defect	This spreadsheet describes the summary of defects, bug status, bug assignee and bug reporter and also the date of bug creation
9.	Requirement Issue log	This spreadsheet involves the various issues related to requirements.
10.	Risk Register	It shows the various risks associated with the project.
11.	Metric Analysis	This spreadsheet shows the various metrics like rework effort,defect density ratio,effort variance etc.

5.7.1.1 Release Summary

The release summary spreadsheet contains the information about the overall planned and realized size of each release. Also this spreadsheet provides information regarding sprints for each milestone. The release summary also provides the team a view of start of release, end of release and all the sprints and the start date and end date of each of the sprint (see Figure 5.4).



Release Summary

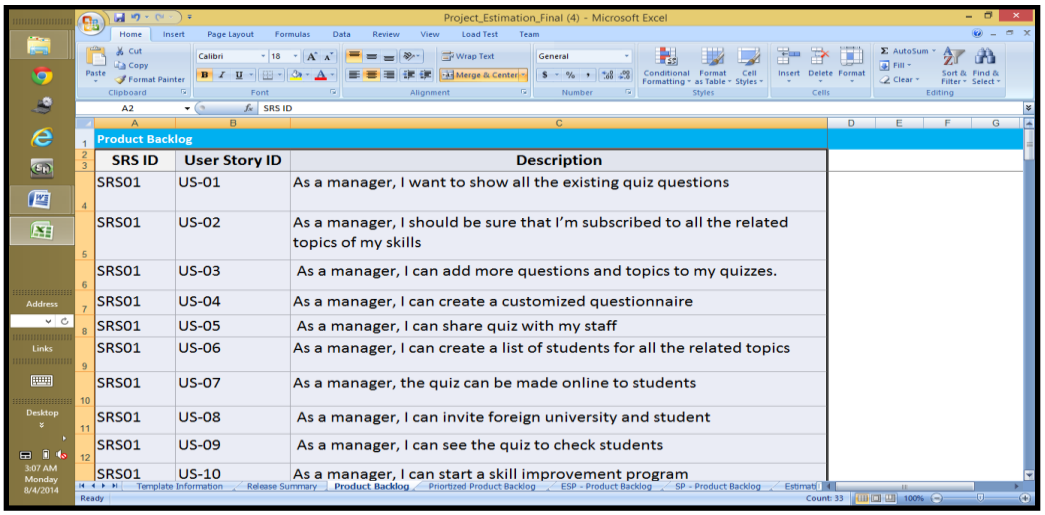
	Case 1	Case 2	Case 3	Case 4
Release Start Date	01-Jan-15	01-Jan-15	01-Jan-15	01-Jan-15
Sprint 1 Start date	01-Jan-15	01-Jan-15	01-Jan-15	01-Jan-15
Sprint 1 End date	05-Jan-15	06-Jan-15	16-Jan-15	30-Jan-15
Sprint 2 Start date	06-Jan-15	07-Jan-15	17-Jan-15	31-Jan-15
Sprint 2 End date	12-Jan-15	14-Jan-15	11-Feb-15	23-Mar-15
Sprint 3 Start date	13-Jan-15	15-Jan-15	12-Feb-15	24-Mar-15
Sprint 3 End date	15-Jan-15	19-Jan-15	26-Feb-15	20-Apr-15
Release End Date	15-Jan-15	19-Jan-15	26-Feb-15	20-Apr-15

Total Team members 4

Figure 5.4: Release Summary

5.7.1.2 Product Backlog

This spreadsheet lists all the user-stories in prioritized order. This spreadsheet works as a starting point for starting a new project. Product backlog contains also the short description of characteristic, but it doesn't contain any detailed rules. The product backlog is further divided into sprint backlog (see Figure 5.5).



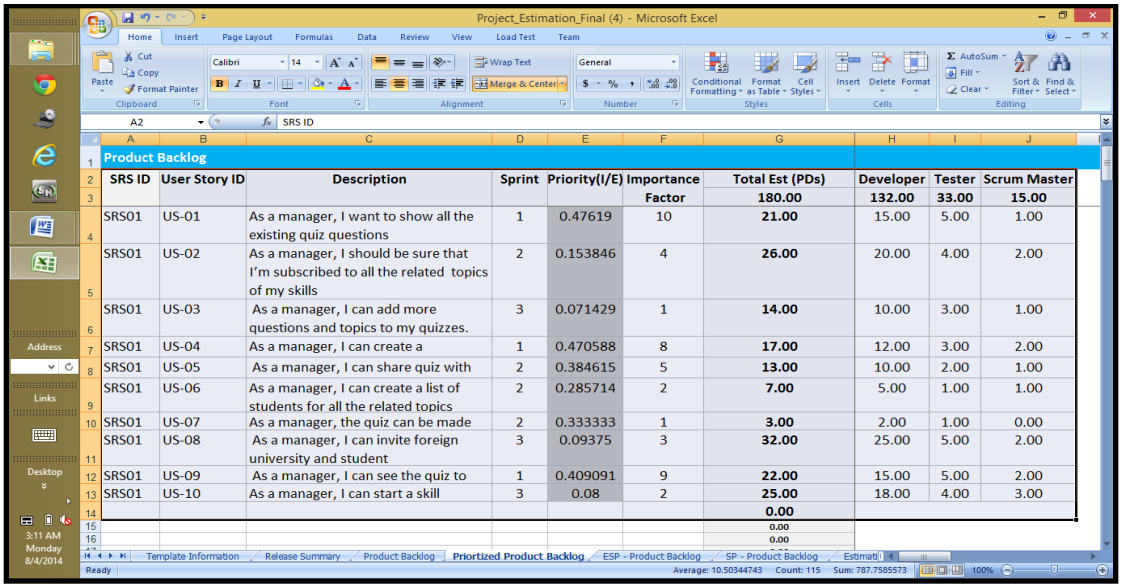
Product Backlog

SRS ID	User Story ID	Description
SRS01	US-01	As a manager, I want to show all the existing quiz questions
SRS01	US-02	As a manager, I should be sure that I'm subscribed to all the related topics of my skills
SRS01	US-03	As a manager, I can add more questions and topics to my quizzes.
SRS01	US-04	As a manager, I can create a customized questionnaire
SRS01	US-05	As a manager, I can share quiz with my staff
SRS01	US-06	As a manager, I can create a list of students for all the related topics
SRS01	US-07	As a manager, the quiz can be made online to students
SRS01	US-08	As a manager, I can invite foreign university and student
SRS01	US-09	As a manager, I can see the quiz to check students
SRS01	US-10	As a manager, I can start a skill improvement program

Figure 5.5:Product Backlog

5.7.1.3 Prioritized Product Backlog

The importance and effort of user-story has been calculated and then importance to effort (I/E) ratio for each user-story is used for prioritization. The developer, tester and scrum master provide the effort of each user-story. Then total effort for each user-story is estimated. Client provides the importance factor for each user-story. Now if it is assumed that there are total of three sprints in the project then the prioritized product backlog (see Figure 5.6) shows that which sprint will cover a particular user-story. The results show that the sprint first consist of user-stories US-01, US-04, US-09. Sprint second consist of user-stories US-02, US-05, US-06, US-07 and sprint third has user-stories US-03, US-08, US-10.



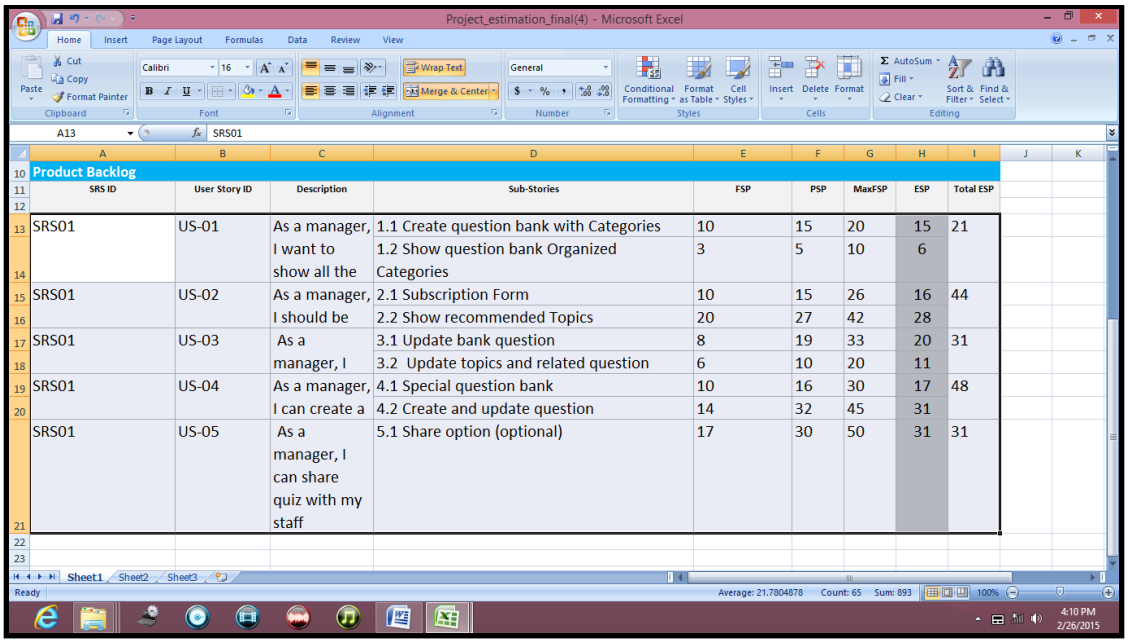
Product Backlog

SRS ID	User Story ID	Description	Sprint	Priority(I/E)	Importance Factor	Total Est (PDs)	Developer	Tester	Scrum Master
						180.00	132.00	33.00	15.00
SRS01	US-01	As a manager, I want to show all the existing quiz questions	1	0.47619	10	21.00	15.00	5.00	1.00
SRS01	US-02	As a manager, I should be sure that I'm subscribed to all the related topics of my skills	2	0.153846	4	26.00	20.00	4.00	2.00
SRS01	US-03	As a manager, I can add more questions and topics to my quizzes.	3	0.071429	1	14.00	10.00	3.00	1.00
SRS01	US-04	As a manager, I can create a	1	0.470588	8	17.00	12.00	3.00	2.00
SRS01	US-05	As a manager, I can share quiz with	2	0.384615	5	13.00	10.00	2.00	1.00
SRS01	US-06	As a manager, I can create a list of students for all the related topics	2	0.285714	2	7.00	5.00	1.00	1.00
SRS01	US-07	As a manager, the quiz can be made	2	0.333333	1	3.00	2.00	1.00	0.00
SRS01	US-08	As a manager, I can invite foreign university and student	3	0.09375	3	32.00	25.00	5.00	2.00
SRS01	US-09	As a manager, I can see the quiz to	1	0.409091	9	22.00	15.00	5.00	2.00
SRS01	US-10	As a manager, I can start a skill	3	0.08	2	25.00	18.00	4.00	3.00
						0.00			
						0.00			
						0.00			

Figure 5.6: Prioritized Product Backlog

5.7.1.4 ESP-Product Backlog

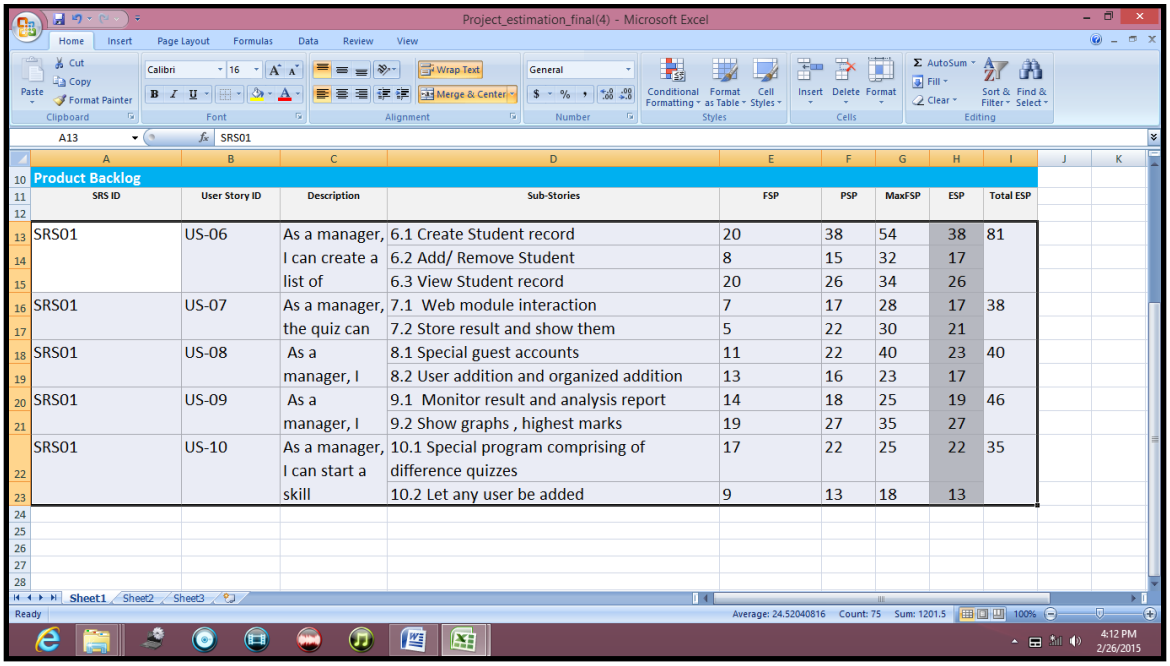
This spreadsheet lists all the user-stories and the estimated story-points in each user-story as shown in Figure 5.7 and Figure 5.8.



Product Backlog

SRS ID	User Story ID	Description	Sub-Stories	FSP	PSP	MaxFSP	ESP	Total ESP
SRS01	US-01	As a manager, I want to show all the	1.1 Create question bank with Categories	10	15	20	15	21
			1.2 Show question bank Organized Categories	3	5	10	6	
SRS01	US-02	As a manager, I should be	2.1 Subscription Form	10	15	26	16	44
			2.2 Show recommended Topics	20	27	42	28	
SRS01	US-03	As a manager, I	3.1 Update bank question	8	19	33	20	31
			3.2 Update topics and related question	6	10	20	11	
SRS01	US-04	As a manager, I can create a	4.1 Special question bank	10	16	30	17	48
			4.2 Create and update question	14	32	45	31	
SRS01	US-05	As a manager, I can share quiz with my staff	5.1 Share option (optional)	17	30	50	31	31

Figure 5.7: Estimated Product Backlog (1)



Product Backlog

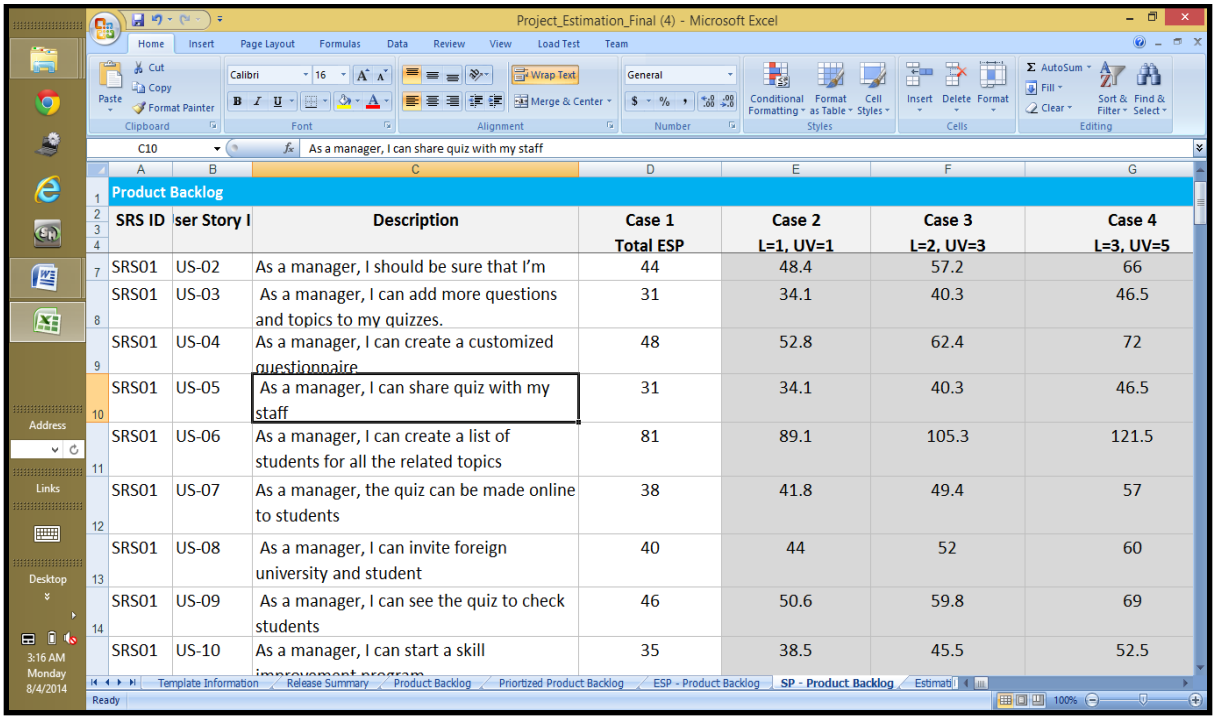
SRS ID	User Story ID	Description	Sub-Stories	FSP	PSP	MaxFSP	ESP	Total ESP
SRS01	US-06	As a manager, I can create a list of	6.1 Create Student record	20	38	54	38	81
			6.2 Add/ Remove Student	8	15	32	17	
			6.3 View Student record	20	26	34	26	
SRS01	US-07	As a manager, the quiz can	7.1 Web module interaction	7	17	28	17	38
			7.2 Store result and show them	5	22	30	21	
SRS01	US-08	As a manager, I	8.1 Special guest accounts	11	22	40	23	40
			8.2 User addition and organized addition	13	16	23	17	
SRS01	US-09	As a manager, I	9.1 Monitor result and analysis report	14	18	25	19	46
			9.2 Show graphs , highest marks	19	27	35	27	
SRS01	US-10	As a manager, I can start a skill	10.1 Special program comprising of difference quizzes	17	22	25	22	35
			10.2 Let any user be added	9	13	18	13	

Figure 5.8: Estimated Product Backlog (2)

5.7.1.5 SP-Product Backlog

Sprint point is an evaluation unit of the user-story instead of story-point. By using sprint-points, more accurate estimates can be achieved. For checking the feasibility of the

algorithm four cases have been considered. In case 1 for estimation of the user-stories the factors are not applied and estimation is done on the basis of estimated story-points instead of sprint-points. In case 2 all the factors are taken at low level (L=1), in case 3 the proposed factors are considered at medium level (L=2) and in case 4 the factors are at the high level (L=3).This spreadsheet shows the calculation of sprint-points in each user-story in four different cases (see Figure 5.9).



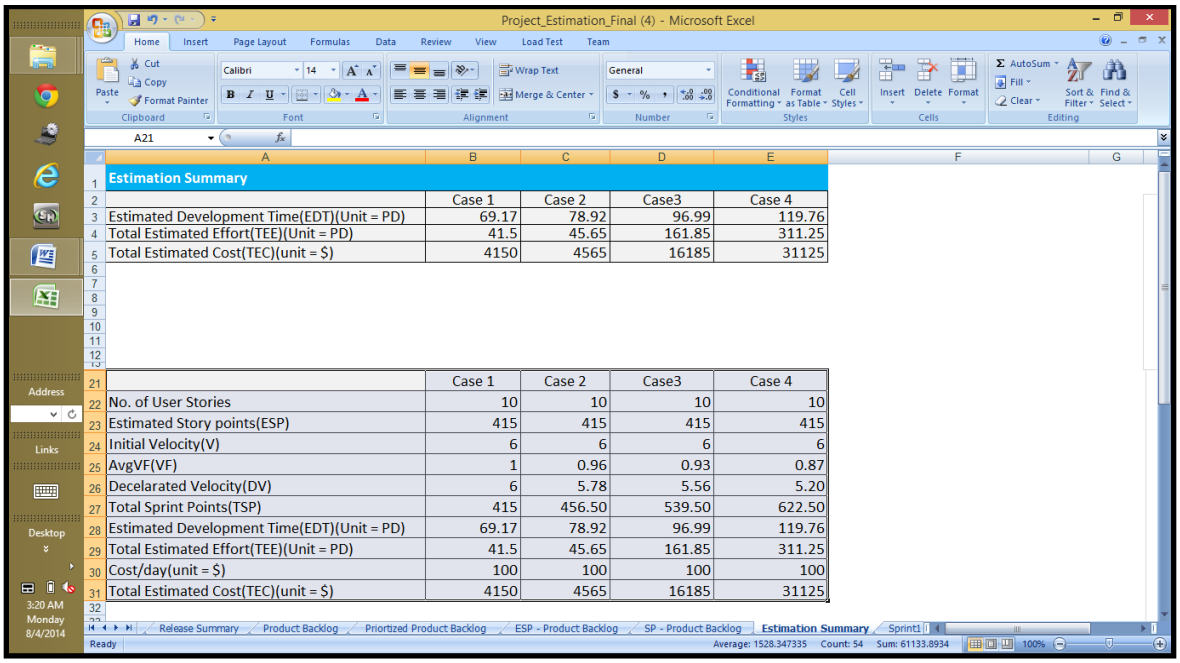
Product Backlog

SRS ID	ser Story I	Description	Case 1 Total ESP	Case 2 L=1, UV=1	Case 3 L=2, UV=3	Case 4 L=3, UV=5
SRS01	US-02	As a manager, I should be sure that I'm	44	48.4	57.2	66
SRS01	US-03	As a manager, I can add more questions and topics to my quizzes.	31	34.1	40.3	46.5
SRS01	US-04	As a manager, I can create a customized questionnaire	48	52.8	62.4	72
SRS01	US-05	As a manager, I can share quiz with my staff	31	34.1	40.3	46.5
SRS01	US-06	As a manager, I can create a list of students for all the related topics	81	89.1	105.3	121.5
SRS01	US-07	As a manager, the quiz can be made online to students	38	41.8	49.4	57
SRS01	US-08	As a manager, I can invite foreign university and student	40	44	52	60
SRS01	US-09	As a manager, I can see the quiz to check students	46	50.6	59.8	69
SRS01	US-10	As a manager, I can start a skill improvement program	35	38.5	45.5	52.5

Figure 5.9: Sprint-Points in each User-story

5.7.1.6 Estimation Summary

The results of sprint-point based estimation algorithm are shown in this spreadsheet. The results show that total estimated story-points are changed if the level of the factors is changed which affects the total effort to complete the project and thus affects total cost and time of the project. The spreadsheet shown in Figure 5.10 details about the initial velocity, the decelerated velocity, total estimated story-points (TSP), sprint-points, total estimated cost (TEC), total estimated effort (TEE) and estimated development time (EDT).Regression testing efforts in a project are calculated in Figure 5.11.

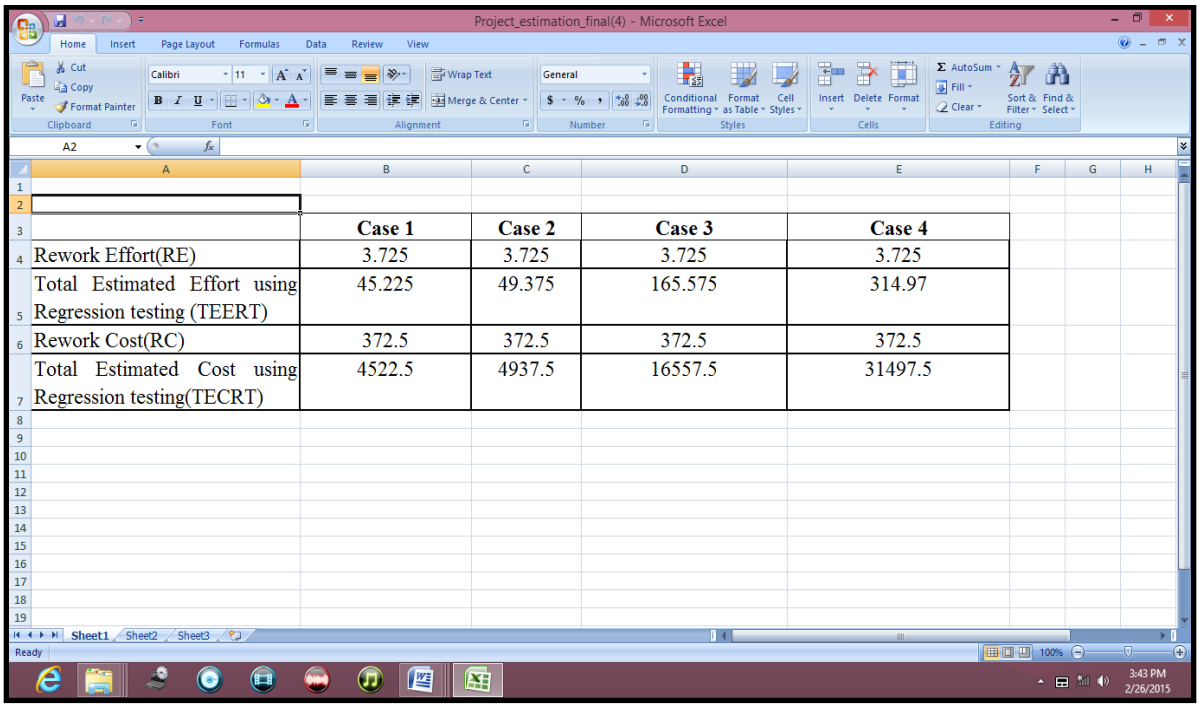


Estimation Summary

	Case 1	Case 2	Case3	Case 4
Estimated Development Time(EDT)(Unit = PD)	69.17	78.92	96.99	119.76
Total Estimated Effort(TEE)(Unit = PD)	41.5	45.65	161.85	311.25
Total Estimated Cost(TEC)(unit = $)	4150	4565	16185	31125

	Case 1	Case 2	Case3	Case 4
No. of User Stories	10	10	10	10
Estimated Story points(ESP)	415	415	415	415
Initial Velocity(V)	6	6	6	6
AvgVF(VF)	1	0.96	0.93	0.87
Decelarated Velocity(DV)	6	5.78	5.56	5.20
Total Sprint Points(TSP)	415	456.50	539.50	622.50
Estimated Development Time(EDT)(Unit = PD)	69.17	78.92	96.99	119.76
Total Estimated Effort(TEE)(Unit = PD)	41.5	45.65	161.85	311.25
Cost/day(unit = $)	100	100	100	100
Total Estimated Cost(TEC)(unit = $)	4150	4565	16185	31125

Figure 5.10: Estimation Summary



	Case 1	Case 2	Case 3	Case 4
Rework Effort(RE)	3.725	3.725	3.725	3.725
Total Estimated Effort using Regression testing (TEERT)	45.225	49.375	165.575	314.97
Rework Cost(RC)	372.5	372.5	372.5	372.5
Total Estimated Cost using Regression testing(TECRT)	4522.5	4937.5	16557.5	31497.5

Figure 5.11: Estimation using Regression Testing Effort

5.7.1.7 Sprint Summary

A sprint also known as iteration is a short (ideally two to four week) period in which the development team implements and delivers a discrete product increment, e.g. a working milestone version.

The sprint summary sheet for each sprint describes the start date, end date of the sprint 1, and also the number of user-stories to be covered in that sprint (see Figure 5.12, Figure 5.13, Figure 5.14).The sprint summary spreadsheet contains the information about the overall planned sprint including start date and end date of each sprint. Also this spreadsheet provides information regarding the number of sprint-points to be completed in that particular sprint.

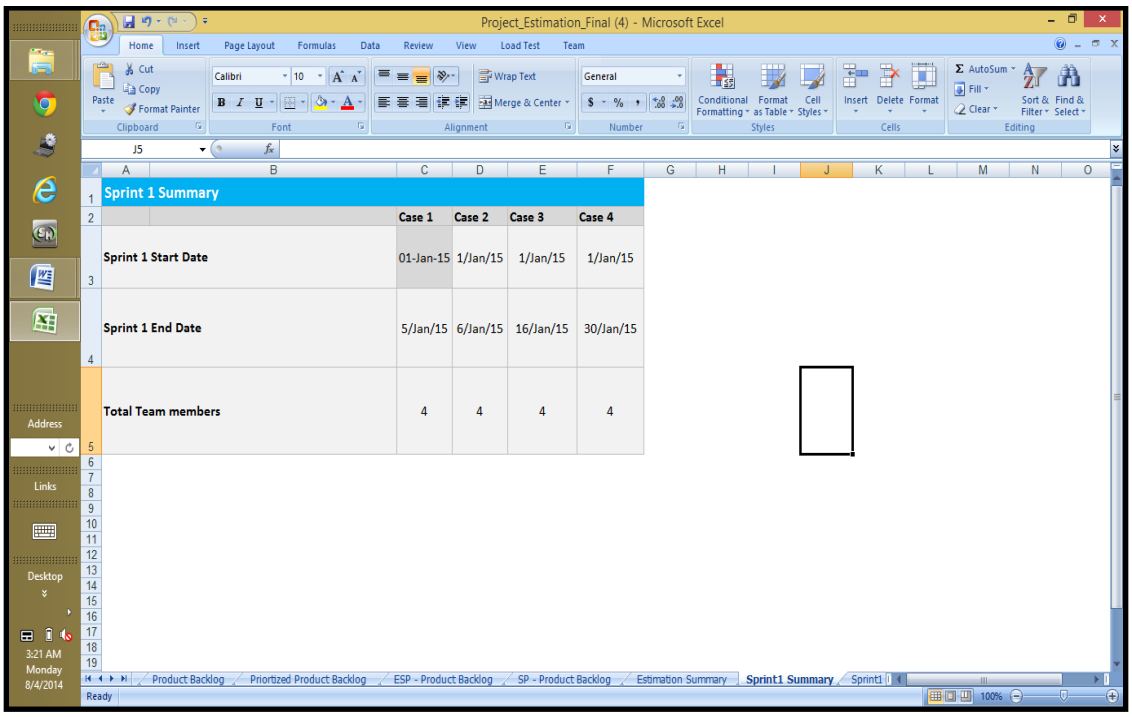


Figure 5.12: Sprint1 Summary

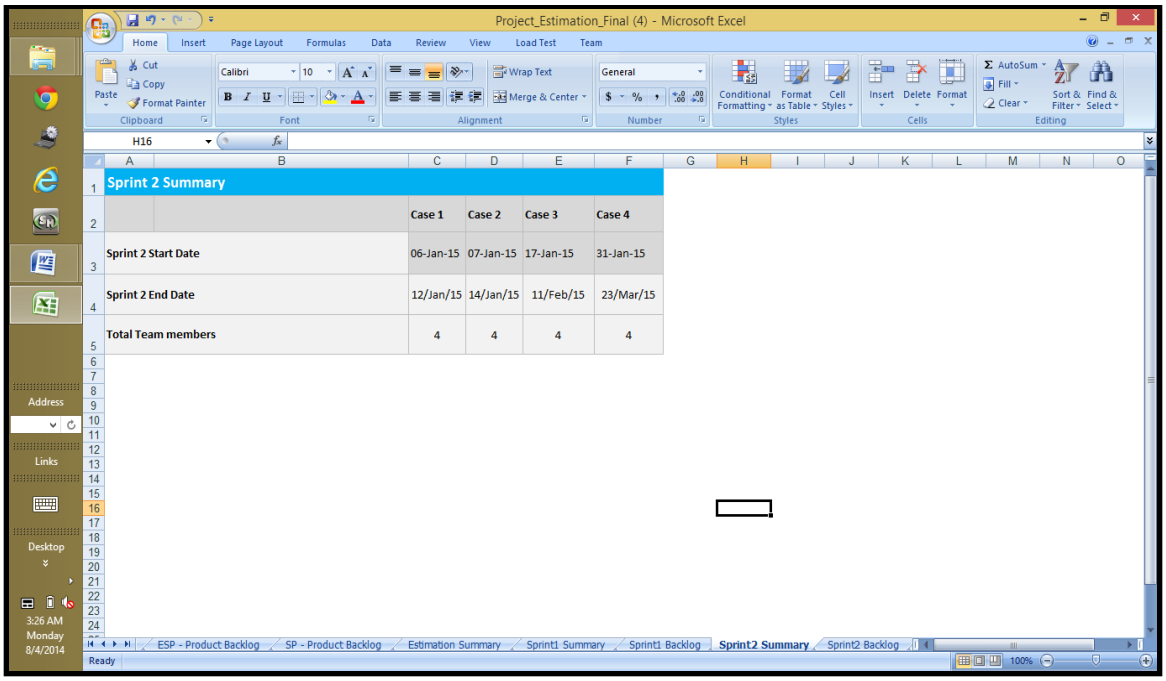


Figure 5.13: Sprint 2 Summary

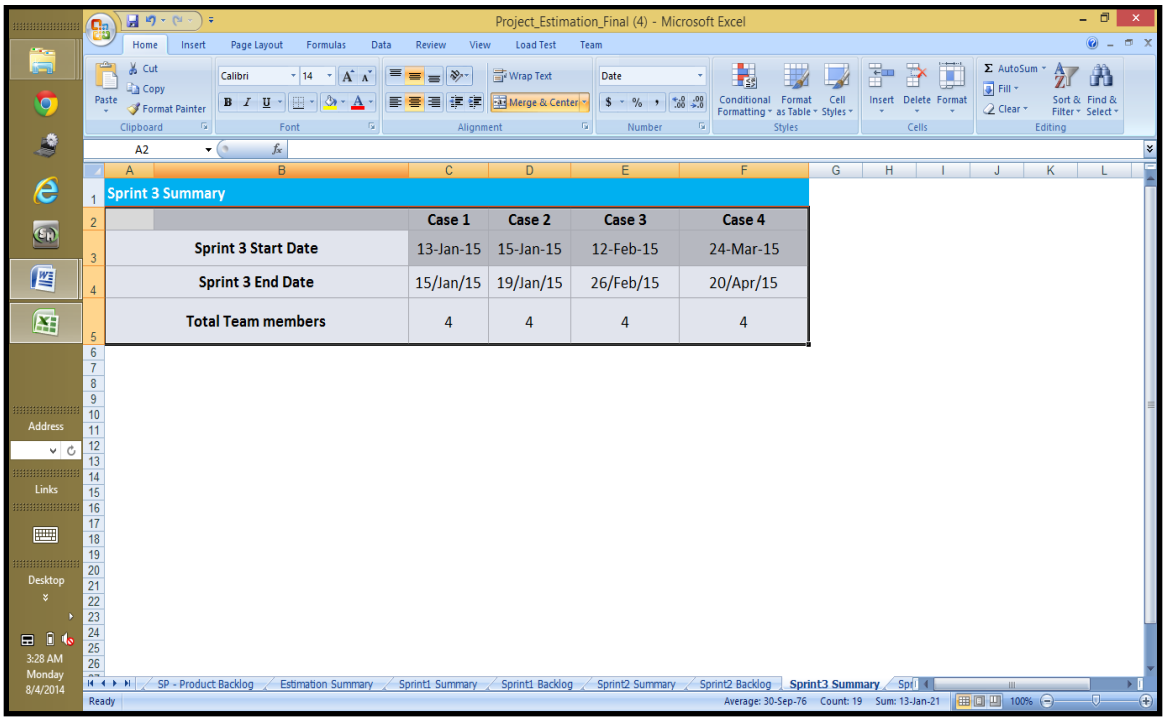


Figure 5.14: Sprint 3 Summary

5.7.1.8 Sprint Backlog

This spreadsheet is a list of tasks identified by the scrum team to be completed during the particular sprint. New tasks cannot be added into sprint backlog when the sprint has started. Only way to add new tasks are if the scrum team wants to add something from product backlog, most teams also estimate how many hours each task will take someone on the team to complete. A sprint backlog contains the list of tasks that need to be completed to implement the features planned for a particular sprint. Ideally, each task in a sprint is relatively short and can be picked up by a team member rather than being assigned. The Figures below shows the sprint backlog for sprint 1, sprint 2 and sprint 3(see Figure 5.15 and 5.16 and Figure 5.17).

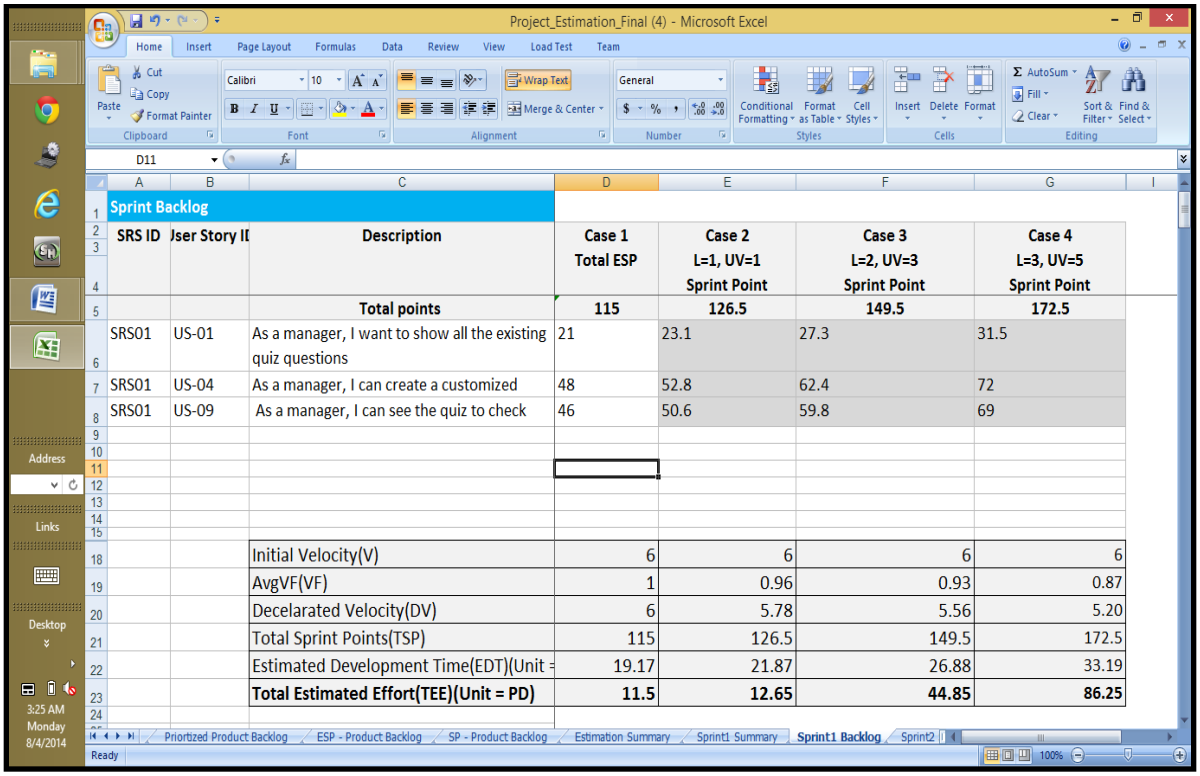
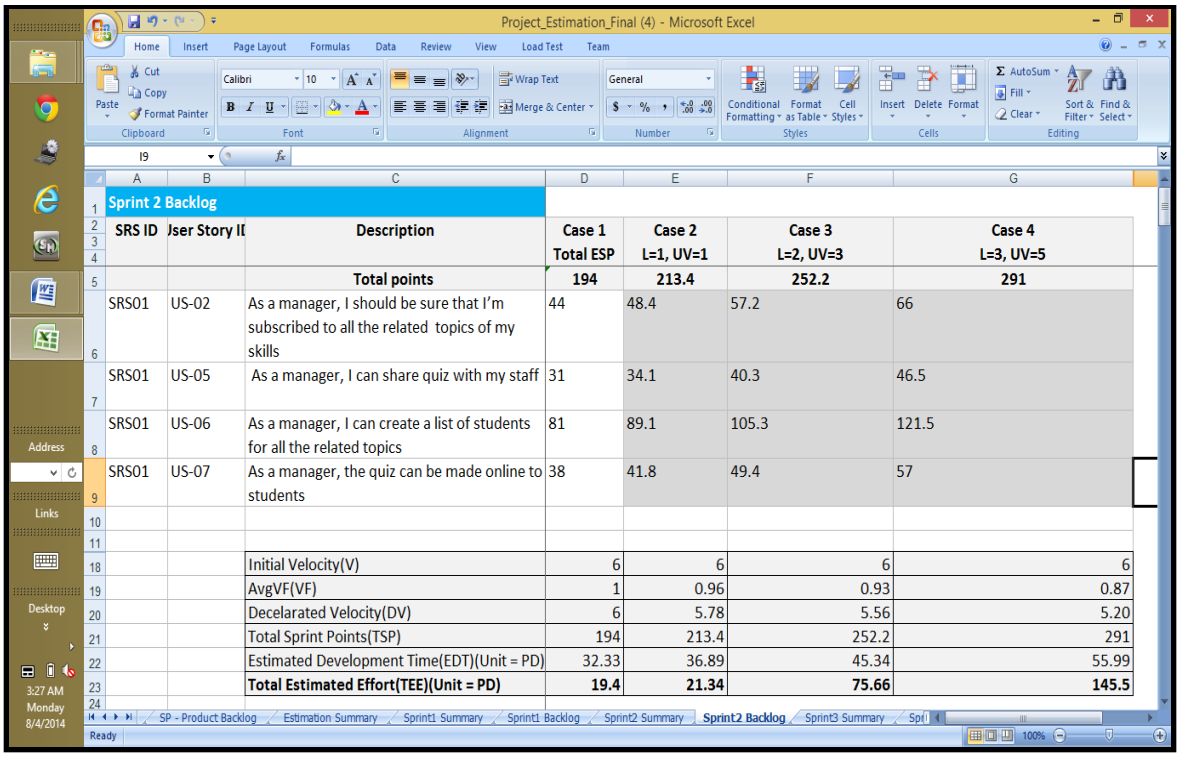


Figure 5.15: Sprint 1 Backlog

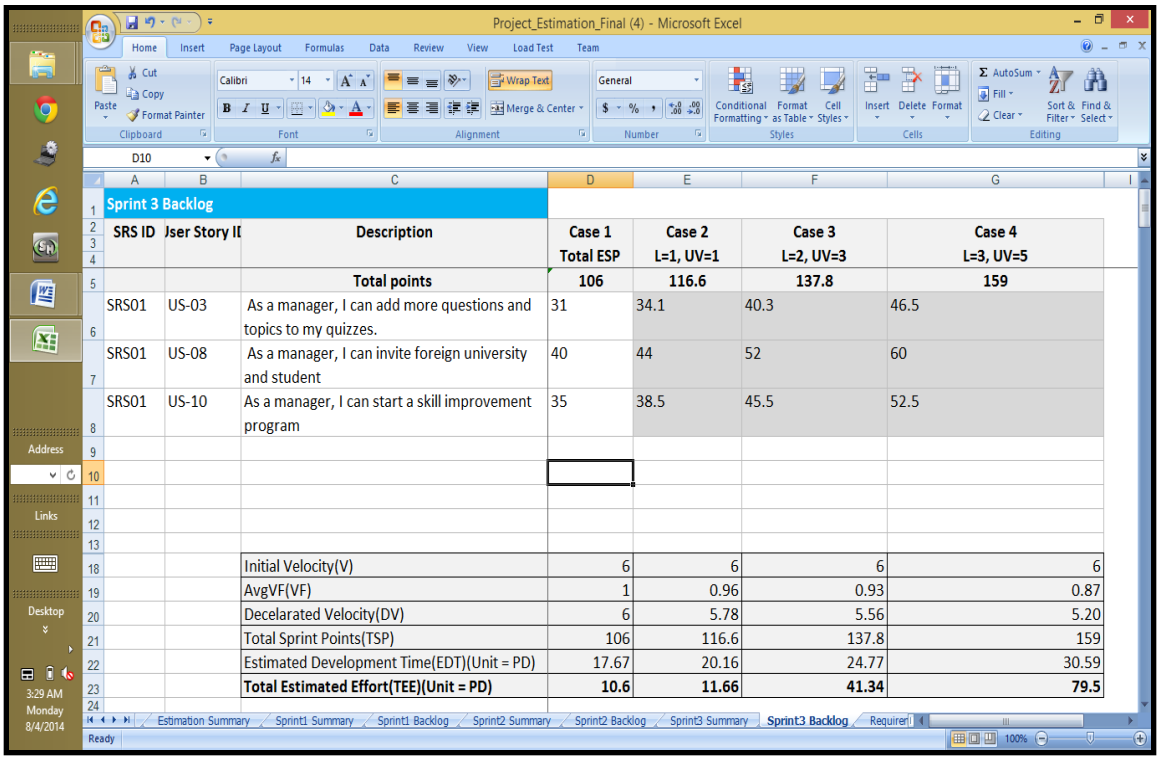


Sprint 2 Backlog

SRS ID	User Story ID	Description	Case 1 Total ESP	Case 2 L=1, UV=1	Case 3 L=2, UV=3	Case 4 L=3, UV=5
		Total points	**194**	**213.4**	**252.2**	**291**
SRS01	US-02	As a manager, I should be sure that I'm subscribed to all the related topics of my skills	44	48.4	57.2	66
SRS01	US-05	As a manager, I can share quiz with my staff	31	34.1	40.3	46.5
SRS01	US-06	As a manager, I can create a list of students for all the related topics	81	89.1	105.3	121.5
SRS01	US-07	As a manager, the quiz can be made online to students	38	41.8	49.4	57

Initial Velocity(V)	6	6	6	6
AvgVF(VF)	1	0.96	0.93	0.87
Decelarated Velocity(DV)	6	5.78	5.56	5.20
Total Sprint Points(TSP)	194	213.4	252.2	291
Estimated Development Time(EDT)(Unit = PD)	32.33	36.89	45.34	55.99
Total Estimated Effort(TEE)(Unit = PD)	**19.4**	**21.34**	**75.66**	**145.5**

Figure 5.16: Sprint 2 Backlog



Sprint 3 Backlog

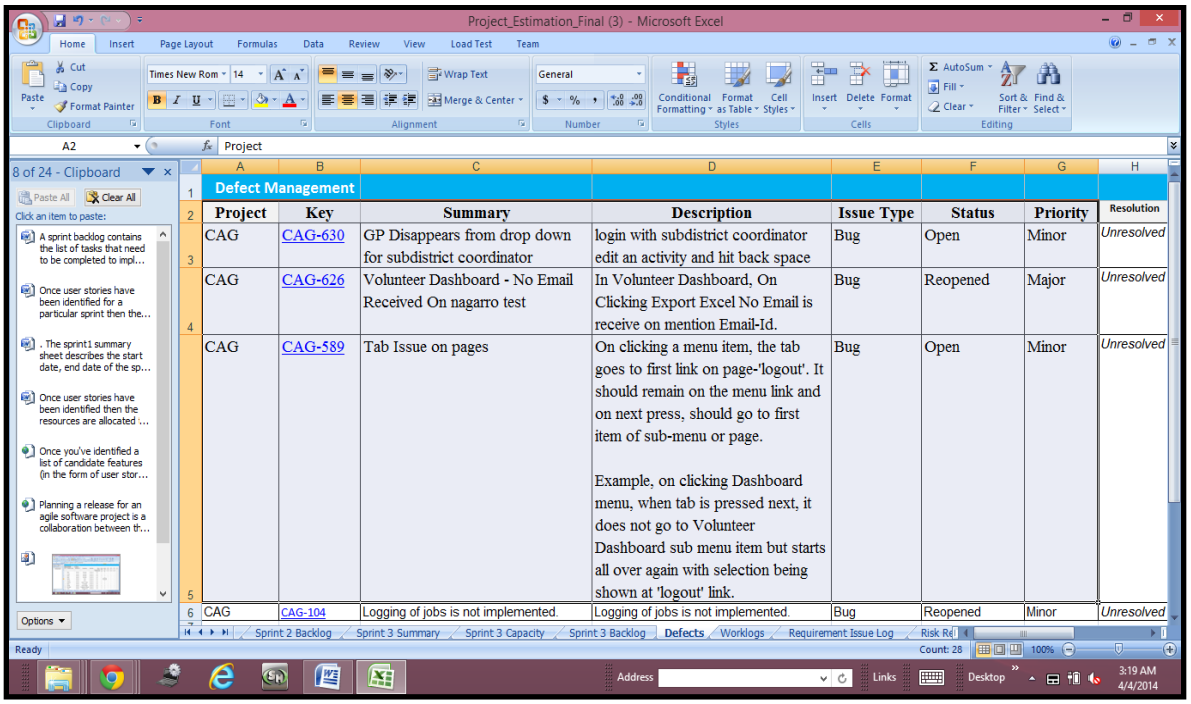
SRS ID	User Story ID	Description	Case 1 Total ESP	Case 2 L=1, UV=1	Case 3 L=2, UV=3	Case 4 L=3, UV=5
		Total points	**106**	**116.6**	**137.8**	**159**
SRS01	US-03	As a manager, I can add more questions and topics to my quizzes.	31	34.1	40.3	46.5
SRS01	US-08	As a manager, I can invite foreign university and student	40	44	52	60
SRS01	US-10	As a manager, I can start a skill improvement program	35	38.5	45.5	52.5

Initial Velocity(V)	6	6	6	6
AvgVF(VF)	1	0.96	0.93	0.87
Decelarated Velocity(DV)	6	5.78	5.56	5.20
Total Sprint Points(TSP)	106	116.6	137.8	159
Estimated Development Time(EDT)(Unit = PD)	17.67	20.16	24.77	30.59
Total Estimated Effort(TEE)(Unit = PD)	**10.6**	**11.66**	**41.34**	**79.5**

Figure 5.17: Sprint 3 Backlog

5.7.1.9 Defect

This spreadsheet describes the summary of defects, bug status, bug assignee and bug reporter and also the date of bug creation as in Figure 5.18.



Defect Management

Project	Key	Summary	Description	Issue Type	Status	Priority	Resolution
CAG	CAG-630	GP Disappears from drop down for subdistrict coordinator	login with subdistrict coordinator edit an activity and hit back space	Bug	Open	Minor	Unresolved
CAG	CAG-626	Volunteer Dashboard - No Email Received On nagarro test	In Volunteer Dashboard, On Clicking Export Excel No Email is receive on mention Email-Id.	Bug	Reopened	Major	Unresolved
CAG	CAG-589	Tab Issue on pages	On clicking a menu item, the tab goes to first link on page-'logout'. It should remain on the menu link and on next press, should go to first item of sub-menu or page. Example, on clicking Dashboard menu, when tab is pressed next, it does not go to Volunteer Dashboard sub menu item but starts all over again with selection being shown at 'logout' link.	Bug	Open	Minor	Unresolved
CAG	CAG-104	Logging of jobs is not implemented.	Logging of jobs is not implemented.	Bug	Reopened	Minor	Unresolved

Figure 5.18: Defect Sheet

5.7.1.10 Requirement Issue log

This spreadsheet involves the various requirements and various issues related to these requirements. It basically describes which issue is raised by which team member, and which team member will work to resolve the issue. It also describes that what is the priority of the issue(see Figure 5.19).

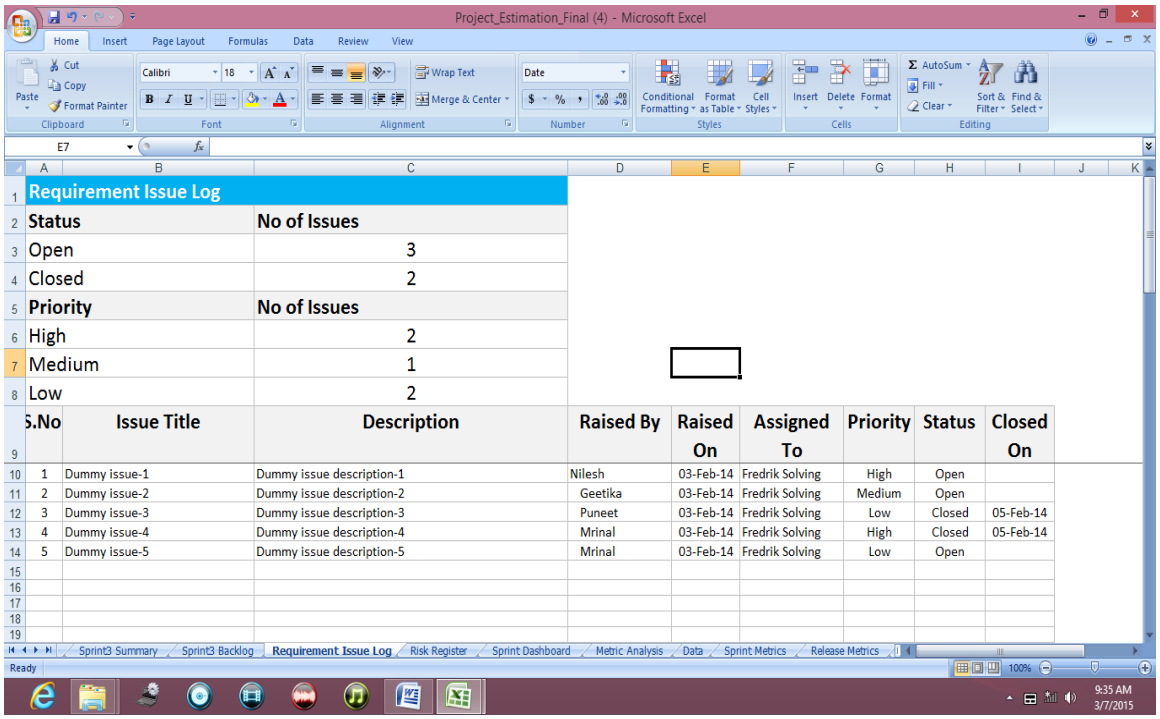


Figure 5.19 Requirement Issue Log

5.7.1.11 Metric Analysis

This spreadsheet shows the various metrics like rework effort, defect density ratio, effort variance etc.

5.8 CONCLUSION

All the proposed approaches have been numerically analyzed on a case study named as enable quiz. As Agile projects are of small duration so the team has not so much amount of time to apply the mathematical algorithms for estimation of cost, effort and time. For resolving this problem a new Sprint–point based estimation tool(SPBE) has been designed and developed to automate the sprint-point based estimation framework.The proposed SPBE tool for estimation places major emphasis on accurate estimates of effort, cost and release date by constructing detailed requirements as accurately as possible. This tool may be used as a vehicle to validate the feasibility of the project.

Chapter VI

CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSIONS

This chapter presents the major achievements of the research work and lists the scope of future work in this area. The outcome of this research contributed an Agile model, Sprint-point based estimation framework and Sprint-point based estimation tool. This research will help the Agile community in supporting the growing demand from organizations that want to adopt Agile practices. The benefits of proposed design are summarized in the following section.

6.2 BENEFITS OF PROPOSED DESIGN

The significant achievements of the proposed design are listed below:

- **A Model for Transitioning from Traditional software development methods to Agile.**

  An Agile model has been proposed for adopting Agile processes in the software industry. A mapping function has also been presented for transformation from traditional software development model to new Agile model. It is the base to implement Agile.

- **Prioritization of user-stories**

  The proposed design of Sprint-point based estimation framework prioritizes the user-stories on the basis of importance of the client and effort by the developers which is a step towards understanding how Agile projects produce value to the

clients or to the product owners through the requirements prioritization activity. The proposed user-story based prioritization algorithm suggests a method of prioritization that helps in choosing the optimal order of user-stories.

- **Managing Uncertainty in Story-points**

  In Agile uncertainty cannot be eliminated completely but when estimating work, some steps can be taken to reduce it. The proposed technique reduces uncertainty by reducing the size of the user-story to be estimated. If the size of the user-story is small the results of story-point estimation will be more reliable. The proposed strategy of decreasing the "granularity" (size) of items to be estimated improves accuracy and reduces uncertainty.

- **Agile Estimation Factors**

  The proposed framework presents three types of factors i.e. project-related, people-related, resistance factors that can increase or decelerate the velocity of project and affect on productivity, thereby increasing or delaying the duration of the project.

- **Sprint-Point Based Estimation Algorithm**

  A Sprint-point based estimation framework in Agile has been proposed based on various Agile estimation factors and regression testing efforts. This algorithm helps to estimate the accurate cost, time and effort.

- **Sprint-Point Based Estimation Tool**

  To automate the sprint-point based estimation framework,a Sprint-point based estimation tool has been designed. It is a set of individual spreadsheets with data calculated for each team separately. The estimation tool is created to provide

more accuracy in velocity calculations, as well as better visibility through burn-down charts on all stages including planning, tracking and forecasting. This tool is used as a vehicle to validate the feasibility of the project

6.3 FUTURE SCOPE

The work contained in this thesis can be extended with the following list of possible future research issues of ASD.

- **Additional Factors in ASD for Estimation**

  The proposed framework has included three types of factors i.e. people related, project related and resistance factors that affect the cost, effort and time of the software project. In future certain other factors that may affect the estimation can be added so that estimation is more accurate and efficient.

- **Integration and Performance Testing Effort in Estimation**

  In the proposed Sprint-point based estimation framework, regression testing efforts has been considered. But in future, certain other testing efforts may also be considered such as integration testing effort, performance testing effort etc. as per the need of the project.

- **Scaling of Agile Project**

  The present research work is being applicable on small sized and medium-sized software projects. Scaling up an ASD to large projects and large teams is an interesting and challenging topic that may affect the current and proposed methods of estimation. This may necessitate to develop some new models for estimation which is an open research area in fast growth of software industry.

REFERENCES

[1] A. Ananda Rao and Kiran Kumar J,"An Approach to Cost Effective Regression Testing in Black-Box Testing Environment", *International Journal of Computer Science Issue* ,ISSN (Online): 1694-0814,Vol. 8, Issue 3, No. 1, May 2011,pp. 198-207.

[2] A. Qumer, B. Henderson-Sellers, "A Framework to Support the Evaluation, Adoption and Improvement of Agile methods in Practice", *The Journal of Systems and Software*, 2008, pp. 1899-1919.

[3] A.Abran, J.W Moore, P.Bourque,R. Dupuis,"Guide to the Software Engineering Body of Knowledge", *Los Alamitos, CA: IEEE Computer Society*,2004.

[4] A.M.Awad,"A Comparison between Agile and Traditional Software Development Methodologies" *Unpublished doctoral dissertation, The University of Western Australia*, Australia,2005.

[5] A.Maglyas, U. Nikula,K. Smolander, "Comparison of Two Models of Success Prediction in Software Development Projects", *6th Software Engineering Conference (CEE-SECR)* ,13-15 October 2010, pp. 43-49.

[6] Agile Manifesto,"Manifesto for Agile Software Development",http://www.agile manifesto.org.,2009.

[7] Alistair Cockburn, *Agile Software Development,* Pearson Education, 2002.

[8] B. Boehm, "Get Ready for Agile methods with Care", *IEEE Computer Society*,2002, pp. 64–69.

[9] Balasubramaniam Ramesh, Lan Cao,Richard Baskerville,"Agile Requirements Engineering Practices and Challenges: An Empirical Study", *Information Systems Journal*, Volume 20, Issue 5, September 2010, pp. 449–480.

[10] Barry Boehm, *Spiral Development: Experience, Principles, and Refinements,* Wilfred J.Hansen, 2000.

[11] Buglione, "Improving Estimations in Agile Projects: Issues and Avenues", *4th Software Measurement European Forum*, Rome, Italy, 2007.

[12] Cockburn, *Agile Software Development*, Pearson Education, Asia Low Price Edition, 2007.

[13] D. Turk, R. France, B. Rumpe," Assumptions underlying Agile Software Development Processes", *Journal of Database Management*, 2005, pp. 62–87.

[14] D.F Rico," What is The ROI of Agile vs. Traditional Methods? An Analysis of Extreme Programming, Test Driven Development, Pair Programming and Scrum" *Tick IT International Journal*, 2008, pp. 9-18.

[15] D.Leffingwell,*Scaling Software Agility: Best Practices for Large Enterprises*,Upper Saddle River, NJ: Addison-Wesley,2007.

[16] D.Preston, "Using Collaborative Learning Research to Enhance Pair Programming Pedagogy", *ACM SIGITE Newsletter*, Vol.3, No.1, January 2006, pp.16-21.

[17] D.S Janzen, H. Saiedian," Test Driven Learning: Intrinsic Integration of Testing into CS/SE Curriculum", *37th ACM Technical Symposium on Computer Science Education (SIGCSE 2006*), Houston, Texas, USA, 2006,pp. 254-258.

[18] D.S Janzen, H. Saiedian," Test Driven Learning in Early Programming Courses", *39th ACM Technical Symposium on Computer Science Education (SIGCSE 2008),* Portland, Oregon, USA,2008,pp 532-536.

[19] Dagnino**,**Tore Dyba,"Empirical Studies of Agile Software Development: A Systematic Review", *Information and Software Technology, Science Direct*,2008,pp.833-859.

[20] Daniel D. Galorath, "The 10 Step Software Estimation Process for Successful Software Planning, Measurement and Control",Copyright Galorath Incorporated,2006,pp.1-13.

[21] Esther Derby, Diana Larsen, Ken Schwaber, *Agile Retrospectives: Making Good Teams Great,* 4 Aug 2006.

[22] Frank Maurer,Thomas Chau," Knowledge Sharing:Agile Methods Vs. Tayloristic Methods",*International Conference on Enabling Technology: Infrastructure for collaborative Enterprises*, IEEE Computer Society,2003.

[23] F. J. Heemstra, "Software Cost Estimation", Information and Software Technology, vol. 34, no.10,1992, pp. 627-639

[24] F. Maurer and S. Martel, "Extreme Programming: Rapid Development for Web-Based Applications", *IEEE Internet Computing*, 6(1), Feb 2002, pp. 86-91.

[25] F.Chan, J.Thong," Acceptance of Agile Methodologies: A Critical Review and Conceptual Framework" *Decision Support Systems*,2009, pp 803-814.

[26] Feature Driven Development, "http:// feature driven development.com"

[27] G.Hedin, L.Bendix, B.Magnusson,"Introducing Software Engineering by Means of Extreme Programming", *25th International Conference on Software Engineering (ICSE)*, Portland, Oregon,2003, pp. 586-593.

[28] H. Merisalo-Rantanen, T. Tuure, R. Matti," Is Extreme Programming Just Old Wine in New Bottles: A Comparison of Two Cases", *Journal of Database Management*, 2005,pp. 41–61.

[29] Ian Sommerville, *Software Engineering*, Addison Wesley, 7th edition, 2004.

[30] J. Patel, R. Lee and Kim Haeng-Kon, "Architectural View in Software Development Life-Cycle Practices", *6th IEEE/ACIS International Conference on Computer and Information Science*, 2007, pp. 194-199.

[31] J. Stapleton, *DSDM: The Method in Practice*, Addison Wesley Longman, 2003.

[32] J.AHighsmith,*Agile Software Development Ecosystems*, Boston, MA: Addison Wesley,2002.

[33] J.J Kuhl, "Project Lifecycle Models: How They Differ and When to Use Them", www.businessesolutions.com,2002.

[34] James Shore, Shane Warden, *The Art of Agile Development*, November 2,2007.

[35] Jennifier 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,Göteborg, Sweden*, November 2011.

[36] K. Beck, *Extreme Programming Explained: Embrace Change*, Second edition, Addison-Wesley, 2005.

[37] K. McDaid, D. Greer, F. Keenan, P. Prior, P. Taylor, G. Coleman, "Managing Uncertainty in Agile Release Planning", *18th International Conference on Software Engineering and Knowledge Engineering (SEKE'06)*, 2006,pp. 138-143.

[38] K. Petersen,C.Wohlin,"A Comparison of Issues and Advantages in Agile and Incremental Development between State of The Art and an Industrial Case", *Journal of Systems and Software*, 2009,pp.1479-1490.

[39] K. Wiegers, "First Things First: Prioritizing Requirements in Software Development", *IEEE Software* ,vol. 7, no. 9, 1999.

[40] K.Beck,*Extreme programming: Embrace change.Upper Saddle River*,Addison-Wesley,2001

[41] K.Logue,K.McDaid," Agile Release Planning: Dealing with Uncertainty in Development Time and Business Value Engineering of Computer Based Systems", *15th International Conference and Workshop*,IEEE,2008 ,pp. 437-442.

[42] K.McDaid, D.Greer,F.Keenan, P.Prior, P.Taylor,G.Coleman," Managing Uncertainty in Agile Release Planning", *18th International Conference on Software Engineering and Knowledge Engineering* ,2006,pp. 138-143.

[43] K.Schwaber, M.Beedle, *Agile Software Development with Scrum, Upper Saddle River,* NJ, USA: Prentice Hall PTR,2001.

[44] Karlstro¨m and Runeson," Combining Agile Methods With Stage-gate Project Management",*IEEE Journal*,Vol 22,Issue 3,May 2005.

[45] Ken Schwaber, Mike Beedle, *Agile Software Development***,** Prentice Hall, 2001, pp. 100-101.

[46] Ken Schwaber**,***Agile Project Management with Scrum*, Microsoft Press, 2004.

[47] Kent Beck, *Test Driven Development*, Addison Wesley, 2002.

[48] Kiamars Fathi Hafshajani, Mohammad Mehdi Movahhedi, Mohammad Hosein Aboee Mehrizi,"Analysis of Organizational Agility in Auto Industry and identifying Improvement Strategies Using Quality Function Deployment", *International Journal of Economics and Management Sciences*, Vol. 1, No. 7, 2012, pp. 08-18.

[49] Kieran Conboy, "Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development", *Information Systems Research* Vol. 20, No. 3, ISSN 1047-7047,September 2009, pp. 329–354.

[50] Koch, *Agile Software Development - Evaluating the Methods for Your Organization*, Artech House Incorporated, ISBN 1-58053-842-8 Norwood, Massachusetts, 2005.

[51] L. Layman, L. Williams, and L. Cunningham," Exploring Extreme Programming in Context: An Industrial Case Study,"*Agile Development Conference*, 2004,pp. 32-41.

[52] L.Layman, L.Williams,L.Cunningham," Motivations and Measurements in an Agile Case stud*y", Journal of Systems Architecture* ,2006,pp. 654-667.

[53] Laura C. Rodriguez Martinez, Manuel Mora , Francisco, J. Alvarez, "A Descriptive/Comparative Study of the Evolution of Process Models of Software Development Life Cycles", *International Conference on Computer Science IEEE Computer Society*, Washington, DC, USA, 2009,pp.298-303.

[54] Mary Lynn Manns, Linda Rising, *Fearness Change: Patterns for Introducing New Ideas*, Addison Wesley.

[55] M. Ceschi, A. Sillitti, G. Succi, S. De Panfilis, "Project Management in Plan Based and Agile Companies",*IEEE Software* ,2005,pp. 21–27.

[56] M.Cohn, *User stories Applied: For Agile Software Development*, Boston, MA: Addison-Wesley,2004

[57] M.Jorgensen, "Top-Down and Bottom-Up Expert Estimation of Software Development Effort", *Information and Software Technology*,46**,** 2003,pp. 3-16.

[58] M.Jorgensen,U. Indahil, D.Sjoberg, "Software Effort Estimation by Analogy and Regression Toward the Mean". *Journal of Traditional Estimation Methods and Software,* 2003, pp. 253-262.

[59] M.Pikkarainen,J.Haikara, O.Salo, P.Abrahamsson,J. Still," The Impact of Agile Practices on Communication in Software Development", *Empire Software Engineerin*g, 2008,pp. 303–337.

[60] Malik Hneif, Siew Hockow, "Review of Agile Methodologies in Software Development", *International Journal of Research and Reviews in Applied Sciences,* ISSN: 2076-734X, EISSN: 2076-7366, Volume 1, Issue 1, October 2009,pp. 1-9.

[61] Mike Beedle, Martine Devos, Yonat Sharon, Ken Schwaber, Jeff Sutherland, "SCRUM: An Extension Pattern Language for Hyper Productive Software Development", http://citeseerx.ist.psu.edu/viewdoc/summary,2000.

[62] Mike Cohn, *Agile Estimating and Planning*, Addison-Wesley,2005.

[63] Miller, Steve**,** "Agile Scrum-An Overview", *Pragmatic Software Newsletter*, *Pragmatic Software*, 17 Jun 2009.

[64] Mitch Lacey," Scrum as a Cost Saving Measure", *Microsoft Corporation*, Lisbon Portugal,2009.

[65] N.Jacobson, S.K Schaefer," Pair programming in CS1: Overcoming objections to its adoption", *SIGCSE Bulletin*, 40(2), 2008, pp 93-96.

[66] Nabil Mohammed Ali Munassar and A. Govardhan, "A Comparison Between Five Models of Software Engineering", *International Journal of Computer Science Issues(IJCSI),* Vol. 7, Issue 5,September 2010,pp 95-101.

[67] Neno Loje," Talk and slides: Kontinuier liches Feedback, www.teamsystempro.ch and www.scrum.org", *Excellence in Software Engineering Conference (ESE)* Zürich, 2012.

[68] Nupur Garg, "Prioritization of Requirements based upon Quality Approach and Interactive Genetic algorithm focused on Agile Methodologies". *International Journal of Computer Science & Communication Networks*, 2012, pp. 111-113.

[69] O.Benediktsson,D.Dalcher, and H.Thorbergsson,"Comparison of Software Development Life Cycles:A Multiproject Experiment," *IEEE Proceedings–Software*,vol 153, 2006, pp.87-101.

[70] O.Salo,P.Abrahamsson,"Agile Methods in European Embedded Software Development Organisations: A Survey On The Actual Use and Usefulness of Extreme Programming and Scrum", *IET Software* , 2008,pp. 58-64.

[71] Orit Hazzan, Yael Dubinsky, *Agile Software Engineering*, Springer International Edition,2011

[72] P. Abrahamsson,J. Koskela, "Extreme Programming: A Survey of Empirical Data from a Controlled Case Study", *International Symposium on Empirical Software Engineering*,2004, pp. 73-82.

[73] P. Meso, R. Jain, "Agile Software Development: Adaptive Systems Principles and Best Practices", *Information Systems Management* 23 (3),2006,pp 19–30.

[74] Pekka Abrahmson, Outi Salo, Jussi Ronkainen, Juhani Warsta, *Agile Software Development Methods*, VTT Publications 478, ESPOO 2002.

[75] R.Duque,C.Bravo,"Analyzing Work Productivity and Program Quality in Collaborative Programming", *3rd International Conference on Software Engineering Advances*, 2008, pp.270-276.

[76] R.N Charette," Why Software Fail" IEEE Spectrum: http://www.spectrum.ieee. org/sep05/1685,2009.

[77] 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.

[78] Rajlich," Agile Software Development-Software Change, Agile 2014".

[79] Rashmi Popli,Naresh Chauhan, "Mapping of Traditional Software Development Methods to Agile Methodology" *Third International Conference on Computer Science, Engineering & Applications*, Delhi, IEEE, May 2013,pp.117-123.

[80] Rashmi Popli,Naresh Chauhan ,"Sprint-Point Based Estimation in Scrum "*International Conference on Information Systems and Computer networks(ISCON)*, IEEE, GLA University Mathura, March 2013,pp.98-103.

[81] Rashmi Popli,Naresh Chauhan," Impact of Key Factors on Agile Estimation", *International Conference On Research And Development Prospects On Engineering and Technology (ICRDPET)*, IEEE ,Tamilnadu,2013.

[82] Rashmi Popli,Naresh Chauhan," An Agile Software Estimation Technique based on Regression Testing Efforts" *13th Annual International Software Testing Conference*, IEEE, Bangalore, India, 04 – 05 December 2013,pp. 1-9.

[83] Rashmi Popli,Naresh Chauhan,"Prioritizing User Stories in Agile Environment" *International Conference on Issues and Challenges in Intelligent Computing Techniques*, IEEE ,Ghaziabad,7-8 Feb,2014,pp. 515-519.

[84] Rashmi Popli,Naresh Chauhan,"Agile Estimation using People and Project Related factors", *International Conference on "Computing for Sustainable Global*

Development", ,Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), IEEE, New Delhi , 5th– 7th March, 2014,pp. 564-569.

[85] Rashmi Popli, Naresh Chauhan," Cost and Effort Estimation in Agile Software Development", *International Conference on Reliability, Optimization and Information Technology (ICROIT),* IEEE,Manav Rachna International University, Faridabad, , February 6-8, 2014,pp. 57-61.

[86] Rashmi Popli, Naresh Chauhan," Estimation in Agile Environment using Resistance Factors", *International Conference on Information Systems and Computer Networks(ISCON),* IEEE ,GLA University Mathura, March 2014,pp 60-65.

[87] Rashmi Popli,Priyanka,Naresh Chauhan," Managing Uncertainity of Time In Agile Environment", Dhinaharan Nagamalai et al. (Eds) : *ACITY,* WiMoN, CSIA, AIAA, DPPR, NECO, CS & IT-CSCP 2014 DOI : 10.5121/csit.2014.4506,2014, pp. 47–56.

[88] Rashmi Popli,Hemant Sharma,Naresh chauhan," Agile Release Planning by Reducing Uncertainity", *13th Annual International Software Testing Conference in India, Bangalore,* India,04 – 05 December 2013.

[89] Rashmi Popli,Naresh Chauhan," Managing Uncertainity of Story-points in Agile Software",*International Conference on Emerging Trends in Ad-Hoc Networks, Internet Technologies and Software Testing*", Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), IEEE, New Delhi (INDIA), 11– 13 March, 2014.

[90] Rashmi Popli,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.

[91] Rashmi Popli, Priyanka, Naresh Chauhan," Management of Time Uncertainty in Agile Environment",*International Journal of Software Engineering and Applications(IJSEA)* Vol 4,No. 4,July 2014,pp. 122-133.

[92] Rashmi Popli,Naresh Chauhan ,"Research Challenges of Agile Estimation in "*International Journal of Information Technology and Knowledge management*" Volume 7, Number 1 , ISSN 0973-4414, December 2013, pp. 108-111.

[93] Rashmi Popli,Naresh Chauhan,"Agile Software Development", *International Journal of Computer Science and Communication* ,Volume 4,Number 2 ISSN-0973-7391, September 2013,pp.153-156.

[94] Rashmi Popli, Priyanka, Naresh Chauhan," Estimating Regression Effort in Agile Environment", *International Journal of Computer Science and Communication* ,Volume 5 , ISSN-0973-7391 Number 1,Sep 2014, pp.23-28.

[95] Rashmi Popli, Naresh Chauhan, Rajesh Kumar," Estimation In Scrum Using Project Delay-Related-Factors", *International Journal of Research,* YMCAUST.

[96] Rick Botta, A. Terry Bahill, "A Prioritization Process"; *Engineering Management Journal* Vol. 19 No. 4, December 2007,pp. 779-783.

[97] Rlewallen, "Software Development Life Cycle Models ",http://codebeter.com, 2005.

[98] S. Bhalerao and Maya Ingle, "Incorporating Vital Factors in Agile Estimation through Alogorithmic Method" *International Journal of Computer Science and Applications, Technomathematics Research Foundation* ,Vol. 6, No. 1,2009, pp. 85 – 97.

[99] S. Nerur, R. Mahapatra, G. Mangalaraj, "Challenges of Migrating to Agile Methodologies", *Communications of the ACM* , May 2005,pp. 72–78.

[100] 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, ISSN 2304-0777,2012.

[101] S.Kollanus, V.Isomottonen," Test driven development in education: Experiences with critical viewpoints",*13th Annual Conference on Innovation and Technology in Computer Science Education (ITICSE),* Madrid, Spain,2008,pp. 124-127.

[102] S.Nerur, V. Balijepally, "Theoretical Reflections on Agile Development methodologies", *Communications of the ACM* , 50 (3), 2007,pp.79–83.

[103] 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,pp. 285-289.

[104] Steve Easterbrook, "Requirement Engineering: A Roadmap", *International Conference on Future of Software Engineering*", ACM Digital library, 2001,pp. 35-46.

[105] T.Briggs, & C.D. Girard, "Tools and Techniques for Test driven learning in CS1", *Journal of Computing Sciences in Colleges*, 22(3), 2007,pp. 37-43.

[106] T.Stober,U.Hansmann,*Agile Software Development Best Practices for Large Software Development Projects*,Springer Publishing,NewYork 2009.

[107] Timeboxing,"DSDM Consortium,DSDM Agile project Framework", www.dsdm.org September 2013.

[108] Todd L. Graves, Mary Jean Harroldy, Jung-Min Kimz, Adam Porterx, Gregg Rothermel "An Empirical Studies of Regression Test Selection Techniques",*20th International Conference on Software Engineering*,1998,pp. 188-197.

[109] Victor Szalvay**,** "An Introduction to Agile Software Development", Copyright Danube Technologies, November 2004,pp. 1-11.

[110] Z. Racheva., M. Daneva, K. Sikkel, "Value Creation by Agile Projects: Methodology or Mystery?" *10th International Conference on Product-Focused Software Process Improvement*, Oulu (Finland), 2009, pp. 141-155.

[111] Zhaohao Sun,"A Waterfall Model for Knowledge management and Experience Management", *4th International Conference on Hybrid Management System*,University of Wollongong ,zsun@uow.edu.au,2004,pp.472-475.

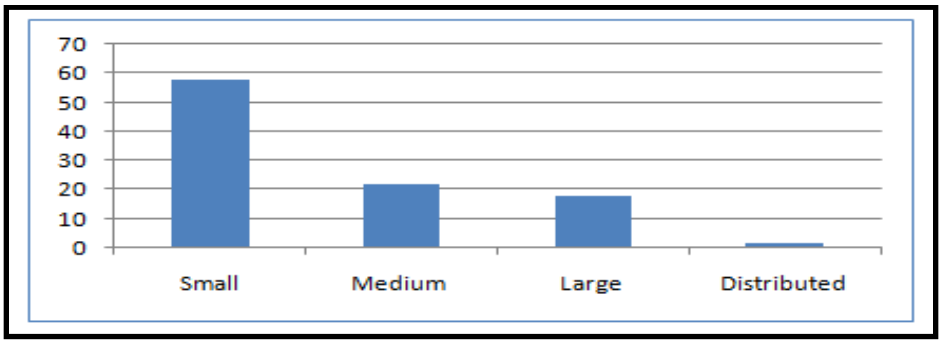
[112] Zornita Racheva,"A Conceptual Model and Process for Client Driven Agile Requirement Prioritization", *4th International Conference on Research Challenges in Information Science*, IEEE,Nice, France, May 2010,pp 5-7.

Appendix: Survey Regarding Problems in ASD

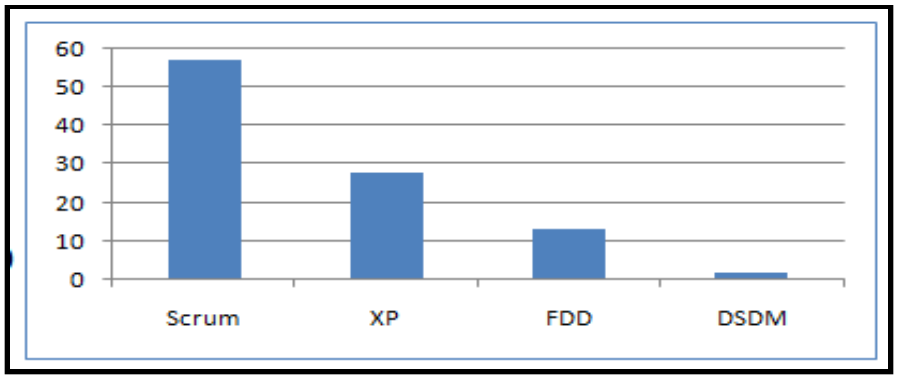
QUESTIONNAIRE

In this work, a survey is being conducted based on the questions asked in this questionnaire. We received 98 responses from software professionals working with Agile experience. The respondents were mostly project managers, scrum masters, team leads followed by software development team members. This survey was done to find out the problems the Agile Team working to develop software projects face. The various questions of the survey and the responses are as below:

1. Q. Which is your Agile Team Type?
    a. Small
    b. Medium
    c. Large Collocated
    d. Distributed



2. Q. Which Agile Method you use?
    a. Scrum
    b. XP
    c. FDD (Feature Driven Development)
    d. DSDM (Dynamic Systems Development



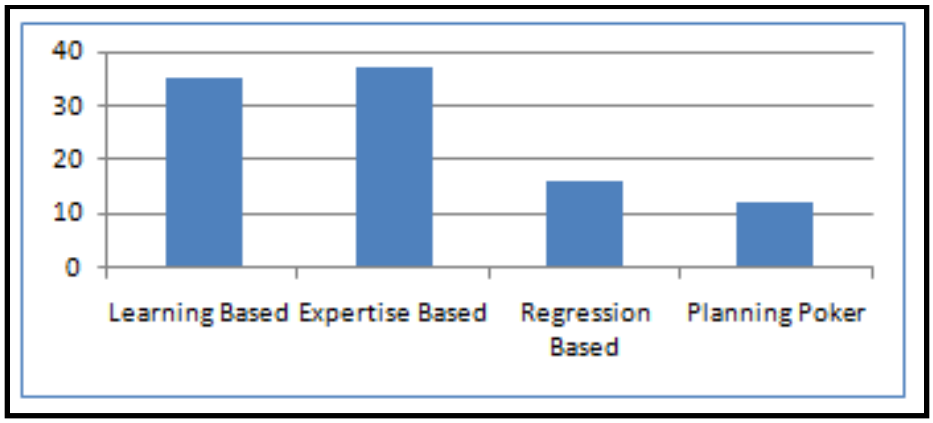
3. Q. How Effort Estimation is performed

a. Learning-Based Approach

b. Expertise based Approach

c. Regression Based Approach

d. Planning Poker



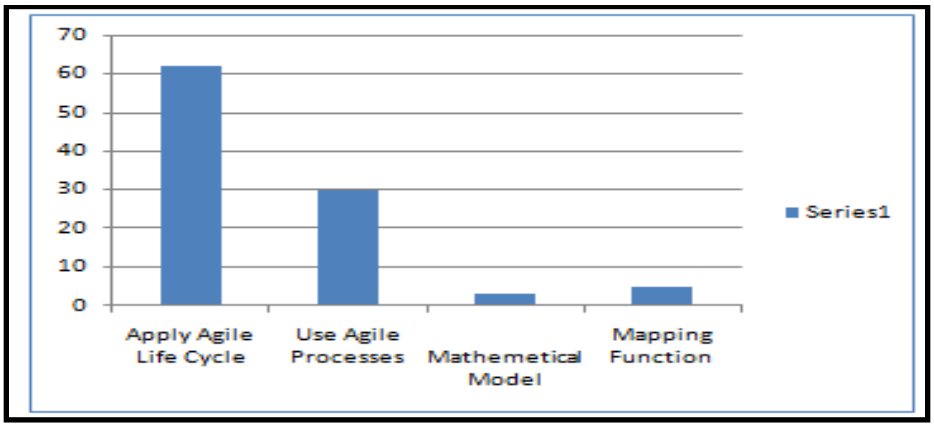
4. Q. How Transitioning is performed if management wants to switch from some traditional method to Agile

a. Directly Apply Agile Life Cycle

b. Use Agile Processes initially

c. Some mathematical model

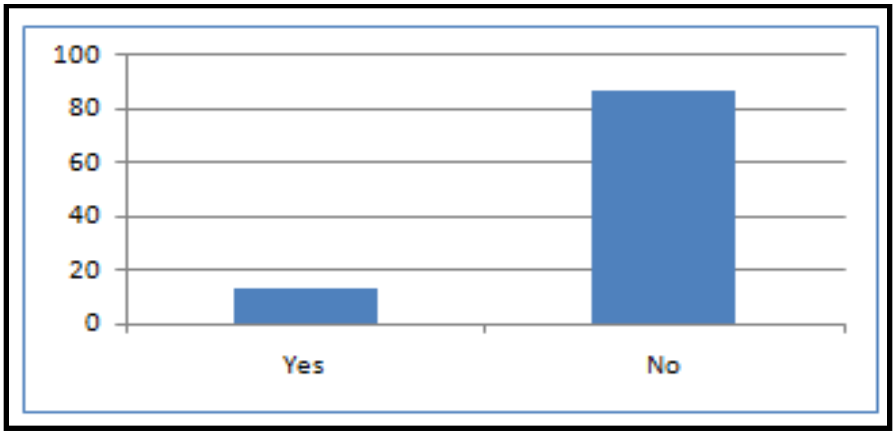
d. Mapping Function



5. Q. Have you ever tried any method of prioritization of user-stories.

a. Yes

b. No



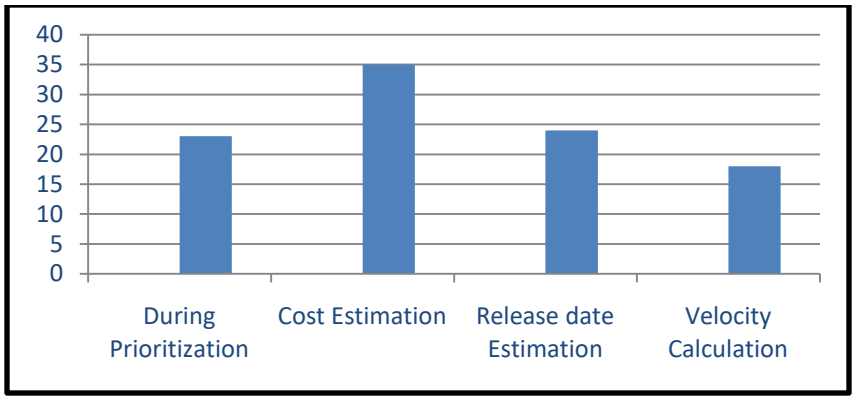
6. Q. Currently At what stage, you face problem while working with Agile

a. During prioritization of user-stories

b. Cost Estimation

c. Release Date Estimation

d. Velocity Calculation



7. Q. How Have Agile Approaches Affected the Cost of System Development?

8. Q. Specify problems which you face frequently in your company while working with agile.

9. Q. Specify any estimation tools which is used frequently in your company.

10. Q. How uncertainty of user-stories is removed?

A majority of the respondents (58%) indicated that they use small teams while working with Agile projects and they are using scrum method of Agile. Scrum is the most popular Agile methodology.

The respondents currently use different estimations approaches like learning-based, expertise based regression based and planning poker. More than 85% of the respondents said that there is no estimation tool while working with agile project. The estimation is done in ad-hoc basis. The respondents currently face problems while prioritization of user-stories, while estimating the project. Ninety percent of respondents said that transitioning form a traditional method to Agile is not an easy and one step process. Rather a mapping function must be there for transitioning process. The respondents to a certain extent attribute ASD acceptance problems to organizational resistance and administrative disinterest. Upper management support and lack of training are also identified as challenges compounding the view that lack of administrative actions are probably the biggest roadblocks to the adoption and diffusion of Agile practices.

BRIEF PROFILE OF THE RESEARCH SCHOLAR

Rashmi Popli is pursuing her Ph.D in Computer Engineering from YMCA University of Science and Technology, Faridabad. She is M.Tech (CE) from M.D University in year 2008, B.Tech (IT) from M.D University in the year 2004.She has 11 years of experience in teaching. Presently she is working as Assistant Professor in Department of Computer Engineering in YMCA University of Science & Technology, Faridabad. Her interests include Agile Software Development, Software Engineering, Software Testing and Software Quality. She had 23 research papers published in various International journals and International Conferences.

LIST OF PUBLICATIONS

List of Published Papers in International Journal

S.No	Title of the paper along with volume, Issue No, year of publication	Publisher	Impact Factor	Referred or Non-Referred	Whether you paid any money or not for publication	Remarks
1.	A Mapping Model for transforming Traditional Software Development Methods to Agile Methodology, International Journal of Software Engineering and Applications(IJSEA)Volume 4,No. 4,July 2013.	AIRCC[Academy & Industry Research Collaboration Center] Publishing Co-operation. ISSN-0975-9018		Referred	No	
2.	Management of time uncertainty in Agile Environment, International Journal of Software Engineering and Applications (IJSEA) Volume 4, No. 4, July 2013.	AIRCC[[Academy & Industry Research Collaboration Center] Publishing Co-operation,ISSN-0975-9018		Referred	No	
3.	Research Challenges of Agile Estimation, in International Journal of Information Technology and Knowledge management, IJITKM Volume 7 ,Number 1 ,December 2013 pp. 108-111 (ISSN 0973-4414)	Serial Publications ISSN 0973-4414	1.84	Referred	No	
4.	Agile Software Development, International Journal of Computer Science and Communication IJCSC Volume 4, Number 2	Serial Publications ISSN-0973-7391	1.9	Referred	Yes	

	September 2013 pp.153-156 ISSN-0973-7391.					
5.	Estimating Regression Effort in Agile Environment, International Journal of Computer Science and Communication IJCSC Volume 5, Number 1 March-Sep 2014,pp.23-28 ISSN-0973-7391	Serial Publications ISSN-0973-7391	1.9	Referred	No	
6.	Estimation In SCRUM Using Project Delay-Related-Factors, International Journal of Research, YMCAUST. Volume 2, issue 1. ISSN:2319-9377, Jan 2014	YMCAUST		Referred	No	

List of Communicated papers in International Journal

S.No	Title of the paper along with volume, Issue No, year of publication	Publisher	Impact Factor	Refereed or Non-Refereed	Whether you paid any money or not for publication	Remarks
1.	A Sprint-point based Estimation Technique in Agile Environment, Inderscience publishers.	Inderscience publisher		Referred	No	

LIST OF RESEARCH PAPERS

List of Published Papers

INTERNATIONAL CONFERENCES

1. Rashmi Popli,Naresh Chauhan "Mapping of Traditional Software Development Methods to Agile Methodology" Third International Conference on Computer Science, Engineering & Applications(ICCSEA-2013),24th-26th May 2013, Delhi, India.
2. Rashmi Popli,Naresh Chauhan "Sprint-Point Based Estimation in Scrum " proceedings of IEEE International Conference on Information Systems and Computer Networks, conference held on 9th-10th march 2013 at GLA University, Mathura.
3. Rashmi Popli,Naresh Chauhan," Impact of Key Factors on Agile Estimation", International Conference On Research And Development Prospects On Engineering and Technology (ICRDPET-2013),IEEE, Tamilnadu, India.
4. Rashmi Popli,Naresh Chauhan," An Agile Software Estimation Technique based on Regression Testing Efforts" 13th Annual International Software Testing Conference in India,04 – 05 December 2013, Bangalore, India.
5. Rashmi Popli,Naresh Chauhan," Prioritizing User Stories In Agile Environment" International Conference on Issues and Challenges in Intelligent Computing Techniques,IEEE, Ghaziabad,India 7-8 Feb,2014.
6. Rashmi Popli,Naresh Chauhan,"Agile Estimation using People and Project Related factors", International Conference on "Computing for Sustainable Global Development",IEEE, Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), New Delhi (INDIA), 5th– 7th March, 2014
7. Rashmi Popli, Naresh Chauhan," Cost and Effort Estimation in Agile Software Development",International Conference on Reliability, Optimization and Information Technology (ICROIT'14), IEEE,Manav Rachna International University, Faridabad, February 6-8, 2014.

8. Rashmi Popli, Naresh Chauhan," Estimation in Agile Environment using Resistance Factors",International Conference on Information Systems and Computer networks,IEEE,GLA University Mathura, March 2014.
9. Rashmi Popli,Priyanka,Naresh Chauhan," Managing Uncertainity of Time In Agile Environment", Dhinaharan Nagamalai et al. (Eds) : ACITY, WiMoN, CSIA, AIAA, DPPR, NECO, InWeS – 2014,pp. 47–56, 2014. © CS & IT-CSCP 2014 DOI : 10.5121/csit.2014.4506
10. Rashmi Popli,Hemant Sharma,Naresh chauhan," Agile Release Planning by Reducing Uncertainity", 13th Annual International Software Testing Conference in India , Bangalore, India, 04 – 05 December 2013
11. Rashmi Popli,Naresh Chauhan," Managing Uncertainity of Story-points in Agile Software" International Conference on "Computing for Sustainable Global Development",IEEE,Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), New Delhi (INDIA). 11th– 13th March 2015.

INTERNATIONAL JOURNALS

1. Rashmi Popli,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.
2. Rashmi Popli, Priyanka, Naresh Chauhan," Management of time uncertainty in Agile Environment", International Journal of Software Engineering and Applications(IJSEA) Vol 4,No. 4,July 2014.
3. Rashmi Popli,Naresh Chauhan ,"Research Challenges of Agile Estimation in "International Journal of Information Technology and Knowledge management" IJITKM Vol 7 Number 1, ISSN 0973-4414, December 2013 pp. 108-111
4. Rashmi Popli,Naresh Chauhan,"Agile Software Development", International Journal of Computer Science and Communication IJCSC "Vol 4, Number 2 ISSN-0973-7391,September 2013 pp.153-156.

5. Rashmi Popli, Priyanka, Naresh Chauhan," Estimating Regression Effort in Agile Environment", IJCSC International Journal of Computer Science and Communication Vol 5 , Number 1 March-Sep 2014 pp.23-28 ISSN-0973-7391.
6. Rashmi Popli, Naresh Chauhan, Rajesh Kumar," Estimation In SCRUM Using Project Delay-Related-Factors", International Journal of Research, Jan 2014,Volume 2:Issue 1,YMCAUST,ISSN:2319-9377

List of Communicated Papers

1 Rashmi Popli,Naresh Chauhan," A Sprint-point based Estimation Technique in Agile Environment, Inderscience publishers.