Frequent Change Request From User to Handle Cost on Project in Agile Model

Asia Pacific Journal of Multidisciplinary Research
Vol. 5 No.2, 26-42
May 2017
P-ISSN 2350-7756
E-ISSN 2350-8442
www.apjmr.com

Shariq Aziz Butt[1], Tauseef Jamal[2]
[1]Computer science department Superior University Lahore, Pakistan;
[2]PIEAS University, Pakistan
[1]Shariq2315@gmail.com, [2]jamal@pieas.edu.pk

Date Received: January 25, 2016; Date Revised: April 11, 2017

Abstract - *Agile has invented to improve and overcome Draw backs of software development. Now agile model is using in software development very vastly. It is facilitating the developer and client both very resourcefully. It is getting popularity than the other Software Development Life Cycle models because of its characteristics and most owing to allow change request at any level of the project. The client in the software development is the main part and asset of the company. The software house always focuses on its client because client is an asset. Thus developer has major concern with the client's requirement and change request. Agile is getting popularity because of allow change request at any stage of project, on the other hand it is also a drawback in agile model because when project starts the project's completion time and cost is decided. But due to frequent change of requests come from client the cost and completion time both increase eventually which is not good for software house's business and reputation. So there is need a cost and time estimation technique to solve change request issue in agile model.*

Keywords: *Software Development Life Cycle Models; Agile Limitations; Agile Project Cost and Time Estimation; cost and time estimation techniques.*

INTRODUCTION

Software engineering principles, techniques and tools are used to develop the software applications [1]. In the software engineering field to develop a software application there is need a pattern. These patterns are called Software development life cycle model. Many types of Software Development Life Cycle models exist in the software engineering domain to develop the software applications such as V-model, Agile model, RAD model and Big Bang model. All these models have some characteristics as well as some limitations. Agile model is one of these Software Development Life Cycle models. It is a very new and most in use model in the software houses. It has excellence on all other Software Development Life Cycle models. The reason of its excellence is that it supports the change request at any level of the project. It facilitates the client to give change request at any stage of project [5, 6]. In the software house there is a Project manager to communicate with the client and with developer to get update about application's success and change handling [2]. Instead of so much features and popularity the agile also has some limitation. This limitation is the frequent change request from user and due to this project's completion time and cost become increase. When the time and cost become increase then it impacts a bad reputation of the software house on client. The Company can lose its client and client is like an asset of the software house. To solve this issue, the project manager needs a cost and time estimation technique so that whenever the change request comes from the client then the project's cost and time become not increase [8, 9, and 14].

There are many Software Development Life Cycle models in the software engineering field such as Big Bang, RAD, V-Model and Agile [4, 5, 6, 18]. All these Software Development Life Cycle models are used for the small scale projects. All these models have some drawbacks due to which the agile model was needed to introduce and got popularity. The Analysis of these models with the Agile is shown in the Table1. The table is explaining the reasons of less use of these Software Development Life Cycle models in the software house and drawbacks of these models.

Agile Software Development Life Cycle model

The Jim High smith and Bob martin were working on agile concept. They arranged a workshop and in this workshop they exchanged ideas with each other

and with the other people, who were also involved in agile concept then the agile manifesto for software development came up in 2001 in the result of this workshop. The main reason of the popularity and adoption of the agile model was that agile focused on the customer satisfaction and allow customer to give change request at any stage of the project [5, 21, 28, and 36].

Application of agile model:

Agile model is applicable for the following situations [21, 31and 36]:

- When the frequent change requests come from the client.
- It is suitable for fixed requirements.
- It is suitable for the small scale projects.
- Where the face to face communication made between the client and developer.

Table.1 Analysis of Software Development Life Cycle Models

Model features	Big Bang	V-model	RAD	Agile
Small scale projects	Appropriate	Appropriate	Appropriate	Appropriate
Allow change request	Inappropriate	Inappropriate	Inappropriate	Appropriate
Client Satisfaction	Low	Low	Low	High
Fixed Requirement	Yes	Yes	Yes	Yes
Change request can scrap project	Yes	Yes	Yes	No
Customer involvement	High	High	High	High
Easy to manage	No	No	No	Yes
Work in Sprints/Sprints	No	No	No	Yes
Cost estimation technique	No	No	No	No

Agile Software Development:

The Agile software development became popular since the late 1990's. It became popular because at that time software was failed due to long time of requirements finalizing to first test of the develop software.

The Agile software development works in sprints. In the Figure.1 some sprints have no change request but some sprints have change request from client. The bottom Arrow in the diagram is showing the completion time of the project. The Arrow is also showing that when the numbers of sprints increase then the project's entire completion time become increase. When one sprint in the project takes too long time to complete then it ultimately affects the completion time of the other sprints in the project. As shown in the Figure .2 sprint 3 and sprint 5 have a change request from the client. Once the change request comes at any sprint then first developer complete it.

After that the developer sends sprint with change completion in sprint back to client and wait for the client's feedback. When the client gives feedback on sprint's change then developer analyze the feedback that the client is satisfied with the change or not, if not then the developer merge this change request again with the next sprint otherwise plan the next sprint. The completion time of the project also depends on the number of sprints in a project as shown in the Figure .1 [5, 17 and 21].

Issues with Agile Software Development

Frequent Change Request Issue

The change request impacts on the project development. The developer follows the client's request and completes it. The main objective of the developer and company is to satisfy the client. Because the client is the asset of the company and he is giving his time and money [14].

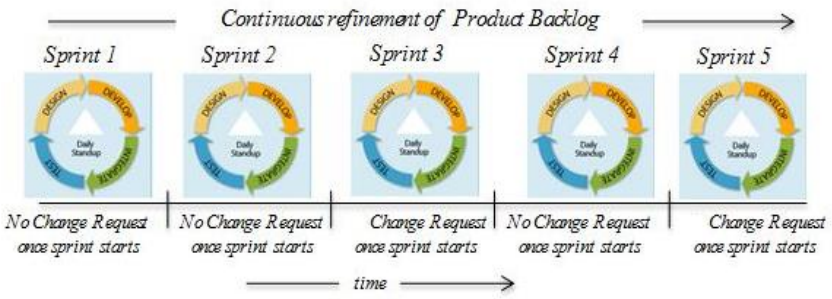


Figure.1 [48]

In the ASD (Agile Software Development) the project manager and the developer welcome the change request at any level of the project [5, 6]. Agile model splits the project into small sprints and then follows the sprint until project complete. The change request impacts on the project in many ways because the change request has many risks associated with it in terms of cost, time, effort and installation of the project. The change on one sprint affects the other sprints in the project [39, 40, 41]. The software is developed in sprints and all sprints are attached with each other. The frequent change of request on one two or more sprint impacts the whole project badly. As shown in the Figure.1 that the total no of sprints are 5 and each sprint needs some time to complete, effort and cost associate with it. Two sprints out of five, sprint 3 and 5 have change request from client. These two sprints took more time and effort to complete. The cost of these two sprints became increase. So the main characteristic of the agile software development is the main issue of it and that issue is the frequent change request come from the client. The frequently changes come from the client can disrupt the whole project. In the agile software development the client prioritize the change request and developer then follows that requirements and completes them first. This prioritization of the requirements also increases the project's complexity. Because might be possible that when the client is prioritizing the requirements then client keep the most complex change requirement at the top of the requirement list which increase the complexity of the software [39, 40, 41]. Hence the main characteristic of the ASD (Agile Software Development) allow the change request at any level of the project is the main issue for agile adoption in the software industry.

Cost increment in ASD (Agile Software Development) Issue

The project's cost is a very important part of the software development. To estimate the right cost of the project is very tough [17, 40]. The wrong cost estimation of project can be cause of the project failure [32, 40]. It can be cause of bad relationship of company with client and the client has greater value for company [14]. The main reason of the increase of the project cost is the change request come from client. Due to change request from client the cost of project exceeds from the cost decided at start of project. The ASD divides the project into multiple sprints and on these sprints the change request comes from client. Multi change requests at one sprint can come from the client multi time. Due to this multi-time change request at one sprint the cost of that particular sprint and other sprints in the project become increase. So there is need to estimate the exact cost of project. There is also need to calculate the cost of the each sprint in project. ASD needs a cost estimation technique to calculate the exact cost of the project so that whenever and no matter how much frequent change of requests come from client the decided cost of the project should not exceed [17, 32, 40].

Time Issue in ASD

The main reason of the increase of project's completion time is the change request come from the client. Due to this change request the time exceeds from the decided time. The project is not completed within the time is a big issue in the ASD. The reason is that the agile allows the client for the frequent change of request. As agile divides the project into small iterations. These iterations are the small time slots. All these iterations need some time to complete. When the frequent change of request comes from the client at a particular iteration then the time becomes increase to complete that iteration and to satisfy the client and ultimately it affects the other iteration's completion time in the project and at the end whole project delivers late. So the ASD needs a technique or model to estimate the accurate completion time of project, so that whenever and no matter how much frequent change of request comes from the client the time should not increase from the time decided at the start of the project [32, 39, 40].

Project Manager Issue in ASD

The project manager decides the cost and completion time of project with the client at the start of project. The client needs his project on the right time and within the cost but this do not happen in the agile software development because of frequent change request. Due to frequent change request the cost and completion time of project becomes increase and the client arise questions for the project manager.

Cost and Time Estimation Techniques

The Agile software development needs a time and cost estimation model so that these issues from the ASD can remove. Different cost and time estimation techniques are explained below with limitations of application of these techniques in ASD.

COCOMO I:

Dr. Barry Boehm introduced a model first time in the 1981 called COCOMO model. The COCOMO stands for Constructive Cost Model. After the publication of this model the manager and software engineers used this model to estimate the cost and time of the project.

Equations:

The equations of the COCOMO models are:

Effort=MM=a $(KDSL/KLOC)$b (1)

Time= TDEV= $2.5 \times \text{MM}^{c}$ (2)

Software Cost = MM × per person salary per month (3)

Limitations of the COCOMO Model [42, 49, 55]**:**

In the COCOMO model if a project manager wants to estimate effort and cost of the project then manager needs to know the source line of code (SLOC) at the initial phase of the project as shown in the above formula 1. The information about the source line of the code should be very accurate to find the accurate effort/person-per month of the project. The software cost is associated with the effort/person per month. Because the salaries of the software engineers are also add in the software cost formula as shown in the formula 3. So if the manager has to find the accurate cost and effort of the project then the manager should have the accurate source line of code of the project. So that when it puts all these values in the formula then the exact cost and effort of the project can be calculate. But to accurately predict the exact source line of code of any project at the initial phase of project is very hard and complex. Because no body can predict the exact source line of code of any project at the start. The source line of code varies from language to language. The size of the code becomes different when the language changes because every language has some size of code. In the Agile software development (ASD (AGILE SOFTWARE DEVELOPMENT)) the frequent change request comes from the client so the SLOC in the agile vary from sprint to sprint. To calculate and predict the exact SLOC in the ASD (AGILE SOFTWARE DEVELOPMENT) is very complex.

For every organization is very difficult to use the COCOMO model for cost, time and effort estimation because the success of the accurate estimation is truly dependent on the modification of the COCOMO model. The reason of the modification of the model is development mode and levels of the COCOMO model.

COCOMO model is also not applicable for the ASD (AGILE SOFTWARE DEVELOPMENT) because in the ASD (AGILE SOFTWARE DEVELOPMENT) the team size is very small. In the ASD (AGILE SOFTWARE DEVELOPMENT) the normal size of the team for the success of the project is not more than 9 people and COCOMO model is not resourceful for the small size team and for small size projects [31, 47].

COCOMO II:

In the mid of the 1990s the second version COCOMO II model was introduced. The main purpose of the invention of this model was to address the issues of the software engineering and introduced a new model for the estimation. It was developed in the University of the Southern California. The COCOMO II was originally published in the annals of the software engineering [45, 50, 52].

Equation:

$$PM_{NS} = AxSize^{E} x \prod_{i=1}^{n} EM_i$$

where A = 2.94 (for COCOMO II.2000) Equation 2: Person month

where $E = B + 0.01x \sum_{j=1}^{5} SF_j$

where B = 0.91 (for COCOMO II.2000)

$$TDEV_{NS} = Cx(PM_{NS})^{F}$$

where $F = D + 0.2x0.01x \sum_{j=1}^{5} SF_j$ Equation 3: Time to develop

where $F = D + 0.2 \times (E-B)$ and C = 3.67 and D = 0.28

Limitations of the COCOMO II Model

The limitation of the COCOMO II model is that it cannot estimate the project's completion time for the small scale projects. If it estimates then the estimation becomes wrong. For COCOMO II to estimate the time for the small scale project is very difficult. The ASD is applicable for the small scale project so the COCOMO II is not resourceful for ASD [42, 47].

Any type of the extension in the COCOMO II model is done during the software development. The extension is still an experiment in COCOMO II model. The extension in the COCOMO II model is not pre-calculated. If we apply the COCOMO II model in the ASD then the results of estimation becomes not accurate because the ASD welcomes the frequent change request from the client thus the COCOMO II needs extension at every change request and the cost and completion time of the project cannot accurately calculate [42].

SLOC (Source Line of Code)

Size is one of the most important attribute of the software development. It is the key indicator to tell about the cost, effort and time of the project. Size of the project is also the base unit to derive other metrics for the project type. According to the Boehm point of view about the cost estimation of the project, the size of the project is an essential part for the estimation models. So the easy way to measure the size of the project is source line of code (SLOC). SLOC is traditional, old and most popular metrics to measure the size of the software. The source line of code is not the sole contributor to estimate the cost, effort and time of the project [49, 53].

Limitations of the SLOC [40, 49]

The SLOC is like input for the cost, effort and time estimation models. It is used to measure the size of the project. It is very popular and common method to estimate the project size. The project size can be measure by the calculation of the line of code in the project and then with the help of the size of the code project's effort, time and cost can calculate. But the SLOC technique is not applicable and accurate for the ASD projects. The reason is that the ASD allows the frequent change request from the client. Due to the frequent change request the size of the line of code in the agile project can vary. There is no limit of the size of the code in the ASD project. Therefore to calculate the exact source line of the code in the project is very difficult and complex task.

Delphi Technique

It is a predictive technique. When there were need to predict the issues with software development then at that time different types of software cost estimation models were introduced to help the estimators to predict and estimate the cost and time of project. So at that time the Delphi technique was introduced by the Helmer in 1966. It is also known as expertise based technique. The Boehm also introduced the Delphi technique with the modification and with the new name called Wideband Delphi Technique. The Boehm developed this technique in the 1981 [52, 55, 56, and 57].

Limitations of the Delphi Technique [40, 55, 56, and 57]

The Delphi technique works with the expert's predictions and opinions about the project. This technique is used to predict the future events and processes of the project. When the expert predicts the events and processes about the project then according to that predictions the project cost and effort estimates. The expert's prediction is considered accurate during the whole project development until it becomes wrong. When the prediction becomes wrong then possible that a lot of project's development time has passed. In the ASD to predict the events and issues is very tough and complex task because the ASD allows the frequent change request from client.

The limitation of the Delphi technique is the selection of expert. In other words who will be the best expert to predict the events about the project? To select the best expert is not an easy task because the future events and cost of the project is associated with the expert's opinions and predictions. So the wrong selection of expert can be cause of project failure.

Function Point Analysis

It was introduced in the October, 1979 by the Albrecht. The Albrecht introduced FP analysis in a meeting arranged by the IBM in the Monterey, California. In this meeting the Albrecht gave a presentation about the FP analysis. After the presentation the IBM announced the basic function point metrics. The FP analysis is the software estimation method. It measures the size of the project with the functionality and usability of the project [55, 58, 60].

Limitations of the FP analysis [47, 55, 59]

The big limitation in the FP analysis is that the FP analysis uses the manual approach to estimate the project. In the FP analysis the estimator needs to do the work manually which is time taking. In the manual estimation a lot of time is consumed. As in the ASD (AGILE SOFTWARE DEVELOPMENT) the projects are small scale and short duration of period. So the FP analysis is not suitable technique for the ASD (AGILE SOFTWARE DEVELOPMENT).

Price to Win Technique

The price to win technique is the technique to win the project within the price, mean that project is tried to complete within the price decide at the start of the project. The price to win is the non-algorithmic estimation technique. The non-algorithmic techniques work with the detailed information about the project. The non-algorithmic technique also uses the historical data of the previous developed project for the current project [55, 61, 63].

Limitations of the Price to Win [55, 61, 63 and 64]

The limitation of the price to win technique is the delivery of project. In the price to win technique the project is always deliver late. As discussed above in the time issue with the ASD that the completion time of the project is very important. Client always needs his project on right time and with the full functionality. When the project delivers late then the client becomes agitate. The software house can lose its client and client is the asset of software house. In the ASD the delivery of the project becomes late due to the frequent change request come from the client. To follow the request and satisfy the client the project completion within the due date becomes impossible. So the price to win technique is not suitable technique for the ASD because of its late delivery of the project.

Parkinson's Law Technique

It was introduced in the 1955 by the UK historian and author Northcote Parkinson. He gave his name Parkinson's Law to this technique. It is also known as Parkinson's Principle. This technique is based on the software estimation. The Parkinson also wrote a book on the behavior of the humans related to estimate the software. In his book he explained that how the humans estimate the project. The estimation of the software is done with the given resources for the estimation [61. 63, 65, 66].

Limitations of the Parkinson's Law[61. 63, 65, 66]

The major drawback of this technique is that it gives accurate estimation not all the time. Sometime the Parkinson's Law gives wrong estimation. Because it base on expert judgment technique. It also follows the expert opinion as like Delphi technique.

It is the un-realistic technique of estimation of project. The accuracy of the estimation is very low in the Parkinson's Law. If the manager is estimating the cost, effort and time of the project then there is no surety that the estimation is accurate.

The Parkinson's Law only measures the effort of the software. It does not measure the completion time of the project. It does not focus on the change of request come from the client. It does not focal point on the cost increment due to change request.

For to apply the Parkinson's Law technique on the project the manager should be familiar with the technique mean to say that the manager should already has some experience and practice to use the Parkinson's Law otherwise the result become not good.

Putnam's model and SLIM:

The Putnam model is based on the Nodern/Rayleigh man power distribution. The Putnam's model is an automated macro estimation model. The SLIM uses the linear programming, statistical simulation, program evaluation and review techniques to calculate the cost of the project [55, 61, 65, and 66].

The equation to estimate is:

$$S = E \times (\text{Effort})^{1/3} t_d^{4/3}$$

$$\text{Effort} = D_0 * t_d^3$$

Limitations of the Putnam's model [55, 61, 65, and 66]

The first limitation of the Putnam's model is that it is not applicable for the small scale of project. If estimator uses the model to estimate the cost, effort and time of the project then the estimation becomes wrong. Thus the Putnam's model is not suitable for the ASD (AGILE SOFTWARE DEVELOPMENT) estimation as the agile is suitable to develop the short time scale of project.

The Putnam's model does not focus on the other aspects of the software development life cycle such as deign, requirements and most important the change request come from the client. By ignoring these aspects of the software development the project can fail.

Estimation Based on Analogy

The Myers in the 1989 gave a detailed description about the Estimation based on analogy technique. In this technique the previous similar project's cost and completion time is used to get idea about the current project's cost and completion time [55, 61, 63, and 66].

Limitations of the Estimation Based on Analogy [55, 61, 63, and 66]

As the estimation based on analogy use the previous and historical project data to estimate the project cost so this is biggest drawback in the analogy technique because some time the current project is different from the previously developed project. The size and functionality of the project can be different from each other. Sometime the historical data is not available then at that time how the estimation will do? If the historical data is available then there is a chance that the data is not accurate. In the ASD due to frequent change requests the size of the project

become change at every sprint. Thus the estimator cannot calculate the cost of the project and cannot use the historical data of any project to estimate the cost and time of the project in the ASD (AGILE SOFTWARE DEVELOPMENT). If the estimator gets the similar project data then it cannot be useful for the current project due to frequent change request and project size in the ASD.

Top-Down Technique

In the top-down technique the total cost of the project can calculate from the global properties by using the either algorithmic or non-algorithmic technique. In this technique the cost of the project is divided in the various components of the system. It is more useful for the early stage of software development; the estimator can calculate the cost at the early phases of the software [55, 61, 63 & 66].

Limitations of the Top down [55, 61, 63 & 66]

The accuracy of the top-down technique is less than the others estimation techniques. The major issue with this technique is that it does not consider the low level problems and these problems can create huge difficulty in the accurate estimation of the project. These low level problems can also increase the cost of the project. The top-down technique also gives less detail and justification about the estimation.

Due to these limitations in the all these estimation techniques, there is need of more accurate model to facilitate the manager to estimate the exact cost and completion time of the project. The ASD needs a model or technique to estimate the project's accurate cost and time with the consideration of the frequent change requests come from the client. So the model can remove all these issues from the agile software development that are mentioned above. The adoption of ASD among the different software industries can increase. The manager can calculate the exact cost and time of the project. The manager can deliver the project within the decided cost and time. The business and reputation of the company can grow. Thus Author is proposing and contributed in the software engineering domain for the cost and time estimation model to remove all these issues from the ASD and from efficient software development.

Proposed Model

After the study and evaluation the above cost and time estimation techniques and issues with the ASD, Author is contributing a solution in the form of a model to remove all these issues. Author propose a model to facilitate the manager or estimator to estimate the exact cost and completion time of the project with the considerations of the frequent change requests come from the client side. Author divides the model in the five screens. Every screen is explaining the particular phase of the software development and is solving the cost and time estimation issue.

Screen1:

Author give his model name is Shariq Screens (SS) method. The estimation starts in the model through the review session. In the review session the program manager discusses the whole project with the software engineers and find out the module's size, cost, time period and effort. The basic purpose of arranging review session is to share the experience of development of software engineers and on the base of that experience they can communicate with each other to find out what are the easy and complex modules of the software? After finding out the type of the module the project manager decides the cost, effort and completion time of the software. The review session team members also look that how much modules has developed earlier related to current software application and how much modules has not been developed before this time. How much no of modules are new for team? The Author divides the model in different categories and every category is explaining the specific type of project.

Module's Categories:

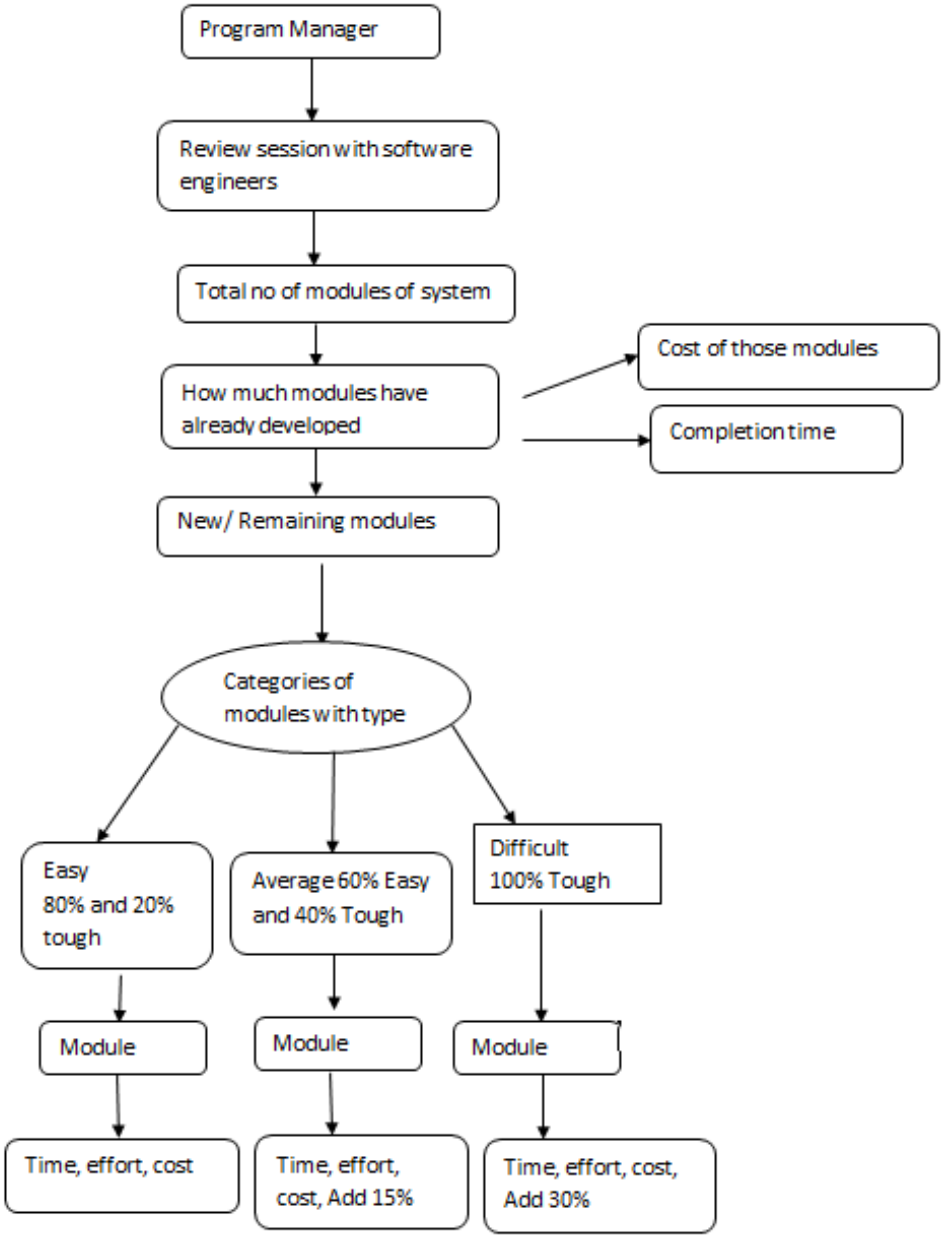
1. **Easy:** The first category of the module is the easy module means that the majority of modules have been already developed in another application. The software engineers just have to replace these modules with the current application's modules or they need some modifications in previously developed modules to meet the current application's requirements. As shown in the screen 1 Figure below. The 80% easy and 20% tough is the easy module category.

2. **Average:** The second category is the Average module means that minimum numbers of modules have been already developed in another application the software engineers just have to replace these modules with the current application modules or they need some modifications in previously developed modules to meet the current application requirement. In the average category

module the number of previously developed modules is less than the easy module category. . As shown in the screen 1 Figure below. The 60% easy and 40% is tough.

3. **Difficult:** The third category in the Screen 1 is the difficult module means that the whole application is new the software engineers have never developed this type of application before this time. They have no development experience related to current application. As shown in the Screen 1 Figure that module/Project is 100% difficult. The developers have to spend a lot of effort and time to complete application and they need more time to handle the frequent change request in application. The time, effort and cost of the application decide in the review session. The difficult category is the very tough and complex.

Screen 1. Figure



Screen 2:
The total cost of the project can calculate by:

Total cost = Average Cost + Expected Cost (1)

Average cost:

Here the average cost is representing the cost of previous develop modules. It adds in the total cost of the current project.

Expected cost:

The expected cost is the cost decided in the review session by the participants in the session.
The total time of the project can calculate by

Total time = Average Time + Expected Time (2)

Average time:

The average time is the completion time of the previously developed modules. It adds in the total time of the current project.
For Average Category:
Total cost = Average Cost + Expected Cost + 15% (3)
Total time = Average Time + Expected Time +15% (4)

For Difficult Category:
Total cost = Expected Cost + 30% (5)
Total time = Expected Time +30% (6)

Screen 3:
If the change request comes after some modules have been developed then the manager can estimate:
Total modules of software =?
No of completed module=?
SLOC of the completed module=?
Remaining module=?
Total time of completion of completed module=?
Effort required for the completed module=?
Per day SLOC=?
Per person SLOC=?

Average of size of the remaining modules:

$\sqrt[Completed\ no\ of\ module]{SLOC\ of\ completed\ module}$ =
Average size of the remaining module.

Screen 4:

The screen 4 is dealing with the very important part of the Shariq screens method that is, when the project development starts the team develop the first module and sends to the client for the feedback. If at the first module the change request comes from the client side then for the developers and manager it is very risky and tough to estimate the size of the change request, affect on completion time and cost of the remaining modules of project. So to handle this part of development the manager or estimator can use this screen 4 to estimate the size of the change request, completion time of project, cost and affect of change request on whole project.

Total no of modules=?
Total time of the modules=?
Divide the time on per module=?
Divide Effort per module=?

Screen 5:

The screen five is explaining that who is best person in the team to handle the change request and to handle the toughest module of the project?

1. When the project starts then in the review session the project's tough and complex modules are decided. After the declaration of these modules now the task is to decide the developer to work on these modules. Give all these modules to most highly experienced developer in the team. Because due to the high experience the style of writing code become different from the developer who has low experience. Then in the last the low experience developer remains with easy module to develop.
2. When the change request comes from the client side then the most experienced developer of the team handle it. Because due to his experience he writes the code in few line as compare to other less experienced developer. He can meet the change request more quickly and can save the time.

Analysis with the Techniques:

In this section Analysis of the above mentioned techniques with the Shariq Screens method is explained.

COCOMO Vs SS Method

The COCOMO model is compared with the SS method as shown in the Table. The COCOMO model estimates the project with the help of the SLOC. It uses the SLOC (source line of code) size for the project cost and time estimation. The main issue with COCOMO model is the SLOC because due to the frequent change of requests come from the client the estimator cannot measure the exact SLOC for the

accurate cost and time estimation but in the SS method the estimator does not need to depend on the SLOC for the estimation. Using the SS method estimator can calculate the exact cost; effort and completion time of the project with the consideration of frequent change of request come from the client. The SS method does not need any type of medication in the model during the estimation while the COCOMO model needs modification when the change of request comes from the client. The SS method is applicable for the all types of projects medium, large and small scale. The SS method is applicable for the all team size.

Analysis and Benefits:

1. The SS method does not base on the source line of code for time, effort and cost estimation as like COCOMO model.
2. During the use of the SS method in the estimation, estimator does not need to modify the method for accurate values, but the COCOMO is required modification during estimation.
3. Modification in the COCOMO model is not pre-calculated but in the SS method is pre-calculated.
4. The SS method is accurately useful for the frequent change of request but the COCOMO model is not suitable.
5. COCOMO model is applicable only for where the team size is large but the SS method is applicable for all team and project size.
6. Accuracy rate of estimation of SS method is more than the COCOMO model.

Techniques	Modification	Allow frequent change request	Size of the team	SLOC for estimation
COCOMO	Experimental and not pre-calculated	NA	Only for large team	Based on estimation
Shariq Screen	Not experimental and pre-calculated	A	For all team	Not based for estimation

NA- Not Appropriate ; A – Appropriate

COCOMO II Vs Shariq Screens Method:

The COCOMO II is the second version of the COCOMO model. The main drawbacks of the model are shown in the Table. The COCOMO II model is not suitable model for the small scale projects but the Agile is used for the small scale projects. The SS method is not only applicable for the Agile but also applicable for all SOFTWARE DEVELOPMENT LIFE CYCLE models use for the all type of projects. The COCOMO II model does not work with the iterations but the SS method can works with the iterations and calculates the accurate completion time, effort and cost of the project. It is more applicable for the frequent change of request comes from the client rather than COCOMO II model.

Analysis and Benefits:

1. The COCOMO II model is not suitable for the iteration software development but the SS method is suitable.
2. The COCOMO II model is not ingenious for the small scale project; on the other hand the SS method is ingenious for all projects.
3. Less accuracy of COCOMO II model for the frequent change request than the SS method.
4. SS method is less risky than the COCOMO II model.

Tech-niques	Iteration/ Sprints	Small scale projects	Frequent change of request	Extension
COCOMO II	Not work with iterations/sprints	NA	NA	Experimental and not pre-calculated
Shariq Screen	work with iterations/sprints	A	A	Not experimental and pre-calculated

NA- Not Appropriate ; A – Appropriate

1. SLOC VS SS method:

The SLOC is also an estimation technique use the exact SLOC for estimation at the start of the project. The main issue with the SLOC is that when the frequent change of request comes from the client side then at that time the size of the line of code vary from one sprint to another sprint so on the behalf of that source line of code to calculate the cost, effort and completion time of the project is very complex and risky. The estimator by using the SLOC can calculate the cost and completion time of project when he/she knows the exact SLOC at the start of the project but in the SS method the estimation is not dependent on the SLOC. The SS method also use the SLOC for estimation as mentioned in the Screen 3 and 4, but when the estimator has some exact SLOC of some completed modules and not calculate as like the

SLOC at the start of the project. In the SS method there is no language barrier for the estimator.

Analysis and Benefits:

1. The SLOC is based on the exact pre-calculated source line of code but the SS method does not base on the source line of code.
2. SLCO does not allow the frequent change of request but the SS method allows the change request.
3. Due to programming languages the accuracy of the SLOC affects but the SS method estimation does not affect.

Techniques	SLOC	Developer's code writing experience	Frequent change of request	Programming Language barrier
SLOC	Exact size of SLOC required	Issue	NA	Yes
Shariq Screen	Without the exact size of SLOC	No issue	A	No

NA- Not Appropriate ; A – Appropriate

2. Delphi VS SS method:

The Delphi is based on the expert opinions about the issues in the software application. The technique uses the expert opinions and suggestions about the future issues of the software application and on that basis the estimator calculates the project cost, completion time and effort. The issue with the Delphi technique is that who is the expert to give suggestions and opinions about the issues? In other words the selection of the Expert in the Delphi is an issue which the SS method is solving through the concept of review session between the software developers and project manager. In the review session they decide the project's completion time, cost and effort according to their experience. The SS method is also base on the expert opinions but SS method is modifying the expert opinion. In the SS method there is no chance that the expert can be optimistic as like in the Delphi technique. According to the SS method the experts are only within the development team. The experts are software developers of the team.

Analysis and Benefits:

1. There is no proper methodology in the Delphi technique for the estimation but the SS method has proper methodology for estimation.
2. In the SS method the experts are not biased as compare to Delphi technique.
3. In the SS method the selection of the members for the review session is not complex as compare to Delphi.

Tech-niques	Methodology for estimation	Expert can biased, optimistic	Rely on experts	Expert selection
Delphi	No	Yes	Yes	Tough/Issue
Shariq Screen	Yes	No	Modified	Easy

3. FP Analysis VS SS method:

The difference between the FP Analysis and the SS method is that the FP analysis is the time taking estimation process but the SS method is not time taking method. For the use of SS method there is no experience required anyone can use the method for estimation. It is very easy to use.

Analysis and Benefits:

1. The FP analysis is the time taking analysis as compare to SS method.
2. There is no experience required for the use of the SS method but in the FP analysis pre-use experience is required.
3. The SS method is more easy to use as compare to FP analysis.
4. The SS method is less risky than the FP analysis.

Techniques	Time taking estimation	For use the Technique
FP analysis	Yes	Experience required
Shariq Screen	No	No Experience required

4. Price to Win VS SS method:

The price to win technique always delivers project late. The late delivery is already an issue for the ASD (AGILE SOFTWARE DEVELOPMENT). So the SS method is removing this late delivery issue from the agile. In the SS method the project completion time is decided at the start of the project and delivers within that decided time. In the price to win the expert estimation can be wrong and the whole project can scrap but in the SS method the expert opinions never becomes wrong. There is no pressure on the developers to complete the work in the SS method.

Analysis and Benefits:

1. In the Price to win always project delivers late but this does not happen in the SS method.

2. There is no pressure on developers and manager in the SS method as compare to price to win.
3. The rate of accurate Cost estimation is more than the price to win.
4. In the SS method there is no chance of wrong estimation.

Techniques	Project deliver late	Pressure with S.E	Cost fixed	Expert estimation
Price to win	Always	Yes	Yes	Wrong
Shariq Screen	No	No	Yes	Right

5. Parkinson's Law:

The Parkinson's Law is the used for the project estimation but the estimation becomes wrong. The estimation accuracy rate in the Parkinson's Law is not good as compare to SS method. The SS method estimation always becomes right and exact. The Parkinson's Law only measures the effort of the project not the completion time of the project but the SS method measures the time, effort and cost of the project. For the use of the SS method there is no experience required but for the Parkinson's Law use some experience is required.

Analysis and Benefits:

1. The SS method is less risky than the Parkinson's Law.
2. Rate of accurate estimation is better than the Parkinson's Law.
3. In the SS method the estimation is realistic but in the Parkinson's Law the estimation is not realistic.
4. There is no experience required for the use of the SS method but in the Parkinson's Law pre-use experience is required.
5. The Parkinson's Law only measure the effort but the SS method measures the effort, completion time and cost of the project.

Techniques	Estimation accuracy	Realistic estimation	Measure only effort	For use technique
Parkinson's Law	No	No	Yes	Experience required
Shariq Screen	Always	Yes	Measures Effort, cost and time.	No Experience required

6. Putnam's model VS SS method:

The Putnam's method is not used for small scale projects. It is only useful for the large scale projects. The ASD is used for the small scale projects so the Putnam's model is not useful for the ASD but the SS method is useful for the all types of projects medium size, large size and small size projects.

Analysis and Benefits:

1. The Putnam's model is only applicable for the small scale projects but the SS method is applicable for the all types of projects.
2. The SS method is more accurate than the Putnam's model.

Techniques	Useful for small scale projects
Putnam's model	No
Shariq Screen	Yes

7. Estimation based VS SS method:

The major difference between the Estimation based analogy and SS method is that the, Estimation based analogy use the historical data for the estimation. It is based on the historical data but the SS method is not based on the historical data. The SS method works for the data availability and not availability both but the Estimation based analogy only works for the data availability. If the data is not available then the Estimation based analogy not estimates the project.

Analysis and Benefits:

1. The SS method's calculation not depends on the historical data as like Estimation Based Analogy.
2. The estimation in the Estimation Based Analogy become wrong due to use of historical data but this not happens in the SS method.
3. Accuracy of estimation in the Estimation Based Analogy is less than the SS method because of depends on historical data.

Techniques	Based on historical data	Historical data need
Estimation based analogy	Yes	Yes
Shariq Screen	No	No

8. Top- Down VS SS method:

The major difference between the SS method and Top-Down technique is that the SS method focuses on the all issues in the software development and estimation but the Top-Down does not focus on the small issues with the software development and estimation.

Analysis and Benefits:

1. Less accuracy due to ignorance of low level problems in the Top-Down technique.
2. SS method is more helpful in estimation than the Top-Down technique.

Techniques	Low level problems handling
Top down	No
Shariq Screen	Yes

Application of the Method

The SS method is not only applicable for the small scale projects and for agile software development but it is also applicable for the large scale projects. It is useful for all Software Development Life Cycle models which are used for the small scale projects. It is also helpful estimation method for the frequent change of requests. It is practical for the all Software Development Life Cycle models which are not allowing the frequent change of requests come from the client such as the models are mentioned above in the Software Development Life Cycle models section Big bang, V-model, RAD model. Whenever the change request come from the client then these models become fail to manage the project's cost, completion time and effort because these models have no estimation techniques. The project cost, completion time and effort become increase. Thus the SS method is solution to remove these issues from all these Software Development Life Cycle models and is making them more efficient for the software development with exact and accurate cost and time estimation. The SS method is applicable and useful for the all Software Development Life Cycle models, for all large and small scale projects. This SS method is more applicable than the other estimation techniques. The SS method can give the more accurate exact time, cot and effort estimation.

Implementation of SS model

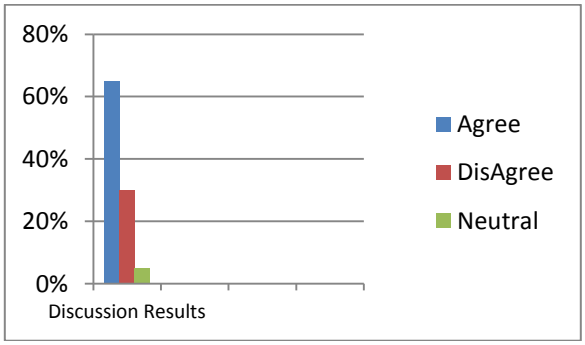
Company	Implementation
Arbi soft	Questionnaire
Soft solution	Project
Systems	Questionnaire
Sigmatech	
Xavor	Questionnaire
Shaukat khanum	Questionnaire
Urban unit	Questionnaire
I2C	Questionnaire
InfoTech	Questionnaire
Consensus	Project

For the results and implementation we use multiple software industries to get results of proposed model and discuss with Several Developers, Project Managers, Business Analysts and teams. We arrange sessions in some companies to get results and feedback.

In the mentioned industries, we arrange sessions with the software developers, project managers, team leads and discusses with them about our SS method and then asked some questions and we get results and in some software houses we use our method on real time projects and get results.

Questionnaire:

RQ1. Is SS method helpful to manage the cost and time?
RQ2. Is Review Session an efficient way to manage the project?
RQ3. Can project cost and time accurately estimate in the review session?
RQ4. Can the SS method's all screens control the cost and time from increase?
RQ5. Can the SS method remove the cost and time issue from the agile model?
RQ6. Do you think that SS method can control the cost and time with the frequent change of requests?
All the developers give different answers and opinions about the SS method the results are shown in the give below:

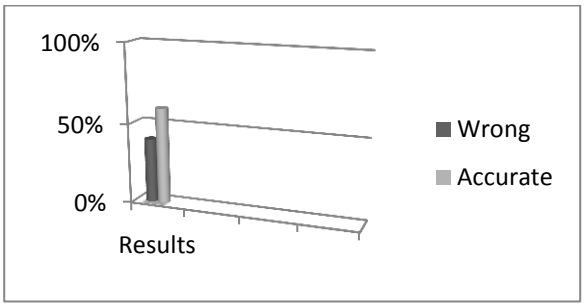


The chart is showing the results for the SS method the 65% is in the favor of the SS method, 35% is against the SS method and 5% is neutral.

No.	Modules	Category	Completion-Time	Pre-develop module	Company
1	8	Average	12 month	3	Soft Solution
2	10	Complex	10 month	0	Consensus

We use the SS method in real time projects in the 2 companies for real results. The projects description is explained in the table below:

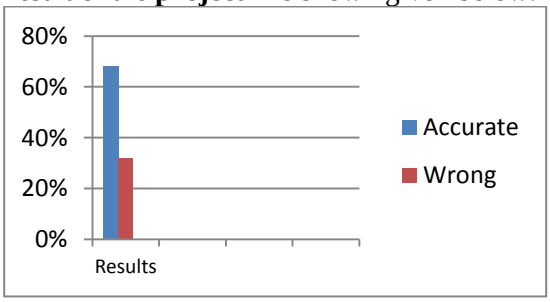
The **project 1** is a **pension management system** that has 8 modules with 12 Months, from which 3 modules has been developed before this time in any other application. Due to category of SS method the **project 1** falls in the Average category. The **Screen 3** of the SS method is used for the time and cost estimation. Project Manager applied all formulas and techniques to estimate the project according to SS method. The module 5 and 7 has large frequent change of requests from the client side. Thus these two modules impacts of other modules in the project. Results are given below:



The ratio of the accuracy of the SS method is 60%-40% Percent in the **project 1**.

The SS method is used in another software house Consensus, project type is **online education system,** has 10 modules with 10 months. According to the SS method the **Project 2** is falling in the difficult category. The reason is developers do not work on this type of application before this time. The project manager use **Screen 4 and 5** to Estimate the project. This project is tough and complex for the developers and more change of requests comes from the client side.

The result of the **project 2** is shown given below:



The ratio of the accuracy of the SS method is 68%-32% Percent in the project 2.

CONCLUSION

The Agile model allows the client to give change request at any stage of the project. The main characteristic of the Agile model is the main issue for the agile adoption in software industry because due to the frequent change request come from the client the cost and completion time of project becomes increase. There are many cost and time estimation techniques but all have some drawbacks due to which these estimation technique are not applicable and helpful for the project manager to estimate the exact and accurate completion time and cost of project in the Agile Software Development. The SS method facilitates the project manager in Agile Software development to estimate the accurate time and cost of the project with the considerations of the frequent change of request come from client. The SS method is solving the main drawback from the Agile Software Development.

REFERENCES

[1] Douglas T. Ross, John B. Good enough, C.A. Irvine, "Software Engineering Rocess, Principles, And Goals", SoftTech Info. Available At: https://goo.gl/nA7h3p.

[2] Sriramasundararajan Rajagopalan ,"*Mapping of the Project Manager Role to the Scrum Master Role",* 10 March 2014. Available At: https://goo.gl/tXWQhF.

[3] Nabil Mohammed Ali Munassar and A. Govardhan, "*A Analysis Between Five Models Of Software Engineering*", IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 5, September 2010.

[4] Apoorva Mishra and Deepty Dubey, "*A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios*", International Journal of Advance Research in Computer Science and Management Studies, 1(5). , October 2013.

[5] S.Balaji and Dr.M.Sundararajan Murugaiyan, "*Wateerfallvs V-MODEL Vs AGILE: A Comparative Study On Software Development Life Cycle "* International Journal of Information Technology and Business Management, 29th June 2012. Vol.2 No. 1.

[6] Nayan B. Ruparelia, "*Software Development Lifecycle Models"* ACM SIGSOFT Software Engineering, May 2010 Volume 35 Number 3.

[7] S.Thulasee Krishna ,Dr. S.Sreekanth ,K.Perumal and K.Rajesh Kumar Reddy, "*Explore 10 Different Types of Software Development Process Models*", IJCSIT International Journal of Computer Science and Information Technologies Vol.3 (4).

[8] Sérgio Sequeira, Eurico Lopes, *"Simple Method Proposal for Cost Estimation from Work Breakdown Structure",* Elsevier Procedia Computer Science 64 (2015) 537 – 544.

[9] Firas Glaiel , "*Agile Project Dynamics: A Strategic Project Management Approach to the study of Large-*

Scale Software Development Using System Dynamics", Composite Information Systems Laboratory (CISL) June 2012.

[10] Adel Alshamrani and Abdullah Bahattab, *"A Analysis Between Three SOFTWARE DEVELOPMENT LIFE CYCLE Models Waterfall Model, Spiral Model, and Incremental/Iterative Model*", International Journal of Computer Science Issues Volume 12, Issue 1, No1, January 2015.

[11] Jim Highsmith,Alistair Cockburn, *"Agile Software Development: The Business of Innovation"*, IEEE Computer Society/ACM joint task force on Computing Curricula 2001.

[12] Handbook of the Secure Agile Software Development Life Cycle, ISBN number: 978-952-62-0341-6, Juvenes Print oulu, University of oulu 2014.

[13] M. A. Awad, *"A Analysis between Agile and Traditional Software Development Methodologies", University of Western Australia, 2005.* Available: https://goo.gl/CyPBRH

[14] *A decade of agile methodologies: Towards explaining agile software development*, Elsevier The Journal of Systems and Software 85 (2012) 1213–1221.

[15] Professor L. Bolliet, Dr . H. J. Helms, "*Software Engineering", Nato Science Committee Garmisch, Germany, 7th to 11th October 1968*.

[16] Barry Boehm, "*A View of 20th and 21st Century Software Engineering*", ICSE'06, May 20–28, 2006, Shanghai, China Copyright 2006 ACM. Available At: https://goo.gl/ZIEslh.

[17] Muhammad Amir, Khalid Khan, Adnan Khan and M.N.A. Khan, "*An Appraisal of Agile Software Development Process",* International Journal of Advanced Science and Technology Vol.58, (2013).

[18] Yu Beng Leau , Wooi Khong Loo, Wai Yip Tham and Soo Fun Tan, "*Software Development Life Cycle AGILE vs Traditional Approaches",* International Conference on Information and Network Technology (ICINT 2012), IPCSIT vol. 37 (2012) Singapore.

[19] Tore Dyba and Torgeir Dingsøyr, "*Empirical studies of agile software development: A systematic review",* Elsevier Information and Software Technology 50 (2008) 833–859.

[20] Marian STOICA, Marinela MIRCEA, Bogdan GHILIC-MICU, "*Software Development: Agile vs. Traditional",* Informatica Economică vol. 17, no. 4/2013.

[21] Baban, N. S. *Processing Models of Software Development Life Cycle ".* Available At: http://www.csimumbai.org/ .

[22] Project Communication Handbook, Second Edition 2007. Available At: https://goo.gl/nSBqS5.

[23] Sergio F. Ochoa, José A. Pino, Luis A. Guerrero, César A. Collazos, *"SSP: A Simple Software Process for Small-Size Software Development Projects",* Springer US 2006 Series Volume 219.

[24] Software Development Life Cycle tutorials 2016. Available At: https://goo.gl/e4EeKj.

[25] Bhuvaneswari, T. & Prabaharan, *S. A Survey on Software Development Life Cycle Models*, IJCSMC, Vol. 2, Issue. 5, May 2013.

[26] Nancy Bindal and Aanchal Mehta, *"Survey On Software Developmentprocessing Models",* IJEETE Vol. 02, Issue 03, MAY-JUN, 2015.

[27] Apoorva Mishra and Deepty Duba, *"Suitability Analysis of Various Software Development Life Cycle Models",* International Journal of Electronics Communication and Computer Engineering Volume 4, Issue (6) NCRTCST-2013.

[28] Hassan Hajjdiab and Al Shaima Taleb, *"Adopting Agile Software Development: Issues and Challenges*", International Journal of Managing Value and Supply Chains (IJMVSC) Vol. 2, No. 3, September 2011.

[29] Software Development Life Cycle 8 March 2016. Available At: https://goo.gl/yLJWHh.

[30] Dan Turk, Robert France Bernhard Rumpe, *"Limitations of Agile Software Processes",* Third International Conference on Extreme Programming and Flexible 22 Sep 2014. Available At: http://arxiv.org/abs/1409.6600 .

[31] Andrea De Lucia and Abdallah Qusef , *"Requirements Engineering in Agile Software Development*", Journal Of Emerging Technologies In Web Intelligence, 2(3), 2010.

[32] Rashmi Popli and Naresh Chauhan, *"Cost and Effort Estimation in Agile Sofware Development",* International Conference on Reliability, Optimization and Information Technology 2014.

[33] Sakshi Garg, Daya Gupta, *"PCA based cost estimation model for agile software development projects*", Proceedings of the 20151nternational Conference on Industrial Engineering and Operations Management Dubai, United Arab Emirates (UAE), March 3 - 5, 2015.

[34] Gilbert Fridgen, Julia Klier, Martina Beer And Thomas Wolf, *"Improving Business Value Assurance in Large-Scale IT Projects A Quantitative Method Based on Founded Requirements Assessment",* ACM Transactions on Management Information System, Vol. 5, No. 3, Article 12.

[35] Muhammad Usman Emilia Mendes and Jürgen Börstler, *"Effort Estimation in Agile Software Development: A Survey on the State of the Practice",* ACM 978-1-4503-3350-4/15/04 2015.

[36] Leo R. Vijayasarathy And Dan Turk, "*Agile Software Development: A Survey Of Early Adopters",* Journal of Information Technology Management Volume XIX, Number 2, 2008.

[37] R. Max Wideman, *"Managing the Project Environment".* Available At: https://goo.gl/3cpqyf.

[38] Peter Forbrig and Michael Herczeg, "*Managing the Agile Process of Human-Centered Design and*

Software Development", INTERACT 2015, Bamberg, 14-18 Sept. 2015.

[39] A B M Moniruzzaman and Dr Syed Akhter Hossain, *"Comparative Study on Agile software development methodologies",* Global Journal of Computer Science and Technology Volume 13 Issue 7 Version I.

[40] Aymen S. Elhady and Hisham M. Abushama, "*RACI Scrum Model For Controlling of Change User Requirement In Software Projects",* IJAIEM Volume 4, Issue 1, January 2015.

[41] Tor Stålhane, "*Change Impact Analysis in Agile Development",* 2014. Available: https://goo.gl/vwQUee.

[42] Nancy Merlo – Schett, "*COCOMO (Constructive Cost Model)",* Seminar on Software Cost Estimation WS 2002 / 2003 University of Zurich, Switzerland.

[43] Sommerville , *" Software Engineering Edition 8th",* 2007.

[44] M. Steven Palmquis,Mary Ann Lapham ,Suzanne Miller ,Timothy Chick and Ipek Ozkaya , "*Parallel Worlds: Agile and Waterfall Differences and Similarities",* October 2013. Available At: https://goo.gl/5xa0og.

[45] Dr. Barry Boehm, Ricardo Valerdi, Jo Ann Lane, and A. Winsor Brown, *"COCOMO Suite Methodology and Evolution,"* CROSSTALK The Journal of Defense Software Engineering April 2005.

[46] Allan Caine, "*Constructive Cost Model COCOMO".* Available At: https://goo.gl/l2Wl5g.

[47] Vikash Lalsing, Somveer Kishnah and Sameerchand Pudaruth, *"People Factors In Agile Software Development And Project Management* ", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.

[48] Chand Warrier, "*A Sprint is Not a Mini Waterfall* ", 19 January 2012. Available At: https://goo.gl/tjd3jU.

[49] Kaushal Bhatt, Vinit Tarey and Pushpraj Patel, "Analysis Of Source Lines Of Code (SLOC) Metric," International Journal of Emerging Technology and Advanced Engineering Volume 2, Issue 5, May 2012.\

[50] R. Dillibabu and K. Krishnaiah, "Cost estimation of a software product using COCOMO II.2000 model – a case study" , Elsevier International Journal of Project Management 23 (2005) 297–307.

[51] Ali Javed, Mirza Ahsan Ullah and Aziz-ur-Rehman, "*Factors Affecting Software Cost Estimation in Developing Countries",* I.J. Information Technology and Computer Science, 2013, 05, 54-59.

[52] Barry Boehma, Chris Abts and Sunita Chulani, "*Software development cost estimation approaches A survey ,"* Springer November 2000, Volume 10, Issue 1, pp 177–205.

[53] Vu Nguyen, Sophia Deeds-Rubin*, Thomas Tan and Barry Boehm, "*A SLOC Counting Standard",* COCOMO Forum 2007. Available At: https://goo.gl/4JFZc5.

[54] Robert E. Park, "*Software Size Measurement: A Framework for Counting Source Statements",* 1992. Available At: https://goo.gl/ptmdMN.

[55] Shivangi Shekhar and Umesh Kumar, "*Review of Various Software Cost Estimation Techniques*", International Journal of Computer Applications (0975 – 8887) Volume 141 – No.11, May 2016.

[56] Ricardo Valerdi, *"Convergence of Expert Opinion via the Wideband Delphi Method: An Application in Cost Estimation Models",* 2011. Available At: https://goo.gl/1T7cwe.

[57] Murali Chemuturi Hand Book, *Software Estimation Best Practices, Tools & Techniques A Complete Guide for Software Project Estimators*, 2009. Available At: https://goo.gl/ZWjnxs.

[58] Heejun Park and Seung Baek, *"An empirical validation of a neural network model for software effort estimation",* Elsevier Expert Systems with Applications 35 (2008) 929–937.

[59] T.S.Shiny Angel, Dr. Paul Rodrigues , John T. Mesiah Dhas and S. SelvaKumara Samy, "*Limitations of Function Point Analysis in E-Learning System Estimation,"* International Journal of Computational Engineering Research (IJCER) ISSN: 2250-3005 National Conference on Architecture, Software system and Green computing.

[60] Yinhuan Zheng , Beizhan Wang, Yilong Zheng and Liang Shi, "*Estimation of software projects effort based on function point" ,* IEEE Proceedings of 2009 4th internation conference on computer science and education.

[61] Hareton Leung and Zhang Fan, "*Software Cost Estimation*" , CiteSeer X 2006.

[62] Torgeir Dingsøyr and Nils Brede Moe, Research Challenges in Large-Scale Agile Software Development,

[63] Ian Sommerville, "*Software Engineering 7th edition* ", 2004.\

[64] Nils Brede Moe, Torgeir Dingsøyr and Tore Dybå, "*A teamwork model for understanding an agile team: A case study of a Scrum project",* Elsevier Information and Software Technology 2009.

[65] Syeda Binish Zahra and Mohsin Nazir, "*A Review of Analysis among Software Estimation Techniques",* Bahria University Journal of Information & Communication Technology Vol. 5, Issue 1 December 2012.

[66] Mehwish Nasir, *"A Survey of Software Estimation Techniques and Project Planning Practices",* Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing IEEE 2006.

[67] Outi Salo, *"Enabling Software Process Improvement in Agile Software Development Teams and*

Organisations", VTT Technical Research Centre of Finland 2006.

[68] Jörg Pechau and Petra Becker-Pechau, "*Challenges: Agile Values Meet Different Value Systems",* ACM 978-1-60558-220-7/08/10.

[69] Oualid Ktata and Ghislain Lévesque, "*Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects*", ACM 978-1-60558-401-0/09/05.

[70] Juyun Cho, *"Issues And Challenges Of Agile Software Development With Scrum ",* Issues in Information Systems VOL IX, No. 2, 2008.