Agile Project Estimation by Optimizing Various factors

A dissertation submitted in the partial fulfillment for the award of Degree of

Master of Technology

In

Software Technology

by

Swati Batra (Roll no. 2K12/SWT/13)

Under the guidance of

Prof (Dr) Daya Gupta(Former HOD)

Department of Computer Engineering, DTU

DEPARTMENT OF SOFTWARE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

BAWANA ROAD, DELHI

DECLARATION

I hereby want to declare that the thesis entitled "**Agile Project Estimation by Optimizing various Factors"** which is being submitted to the **Delhi Technological University**, in partial fulfillment of the requirements for the award of degree in **Master of Technology in Software Technology** is an authentic work carried out by me. The material contained in this thesis has not been submitted to any institution or university for the award of any degree.

_____

Swati Batra

Department of Computer Engineering

Delhi Technological University,

Delhi.

CERTIFICATE

Delhi Technological University
(Government of Delhi NCR)
Bhawana Road, New Delhi-42

This is to certify that the thesis entitled "Agile Project Estimation by Optimizing various Factors" done by Swati Batra (Roll Number: 2K12/SWT/13) for the partial fulfillment of the requirements for the award of degree of Master of Technology Degree in Software Technology in the Department of Computer Engineering, Delhi Technological University, New Delhi is an authentic work carried out by her under my guidance.

Project Guide:
Prof. Daya Gupta
Professor and Former Head of Department
Department of Computer Engineering
Delhi Technological University, Delhi

ACKNOWLEDGEMENT

I take this opportunity to express my deep sense of gratitude and respect towards my guide Prof. Daya Gupta (Former Head of Department) Department of Computer Engineering.

I am very much indebted to her for her generosity, expertise and guidance i have received from her while working on this project. Without her support and timely guidance the completion of the project would have seemed a far –fetched dream. In this respect I find myself lucky to have my guide. She have guided not only with the subject matter, but also taught the proper style and techniques of documentation and presentation. I would also like to take this opportunity to present my sincere regards to Ms. Shruti Jaiswal, Research Scholar, DTU for extending her support and valuable Guidance.

Besides my guides, I would like to thank entire teaching and non-teaching staff in the Department of Computer Engineering, DTU for all their help during my tenure at DTU. Kudos to all my friends at DTU for thought provoking discussion and making stay very pleasant.

Swati Batra
M.Tech Software Technology
2K12/SWT/13

ABSTRACT

Software project estimation is the process of predicting the most realistic use of Cost or effort required to develop or maintain software based on incomplete, uncertain and/or noisy input. Poor estimation may be the cause of significant challenges in project management and in the software quality. Therefore, it is important to make the use of estimation models and appropriate techniques to avoid losses caused by poor estimation.

Software Project estimation in Agile has been an important and difficult task since the evolution of the software. Many formal and informal methods have been proposed for software estimation. It is important for estimation methods to generate realistic software estimates to build the trust of customers as well as team members. Unrealistic estimates are major factors for either software project failure or decreasing the quality of the software.

Agile software processes try to minimize the impact of insufficient estimation accuracy by ensuring that the most important functionality is developed first. This is achieved through a flexible development process with short iterations. However, there is still a need for accurate estimates, as these are the basis for staffing, planning, prioritization and contract negotiations.

To address the Software cost estimation related issues in case of agile software development, I will be proposing a new way to estimate SW required to develop software which is going to be built by agile methodologies. This new way incorporates Optimizing various factors through Genetic Algorithms and then applying Artificial Neural Network via Matlab on project Data sets which can be helpful to increase the accuracy of Software estimation in Agile Projects.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

1.1 General Concepts

What Is Agile?

Agile methodology is a substitute to traditional project management, typically used in software development. It helps a team to respond for unpredictability through incremental, iterative work, known as sprints. Agile methodologies are an alternative to waterfall, or traditional sequential development [1].

Why Agile?

Agile development methodology provides a way to assess the direction of a project throughout the development lifecycle. This is achieved through regular iteration of work, known as sprints, at the end of which teams must present a potentially shippable product increment. By focusing on the repetition of abbreviated work cycles as well as the functional product they yield, agile methodology is described as “iterative” and “incremental” [1].

In waterfall model, development teams only have one chance to get each aspect of a project right. In an agile modeling, every aspect of development — like requirements, design, etc. — is continually revisited throughout the lifecycle. When a team stops and re-evaluates the direction of a project every two weeks, there’s always time to steer it in another direction.

Agile Model:

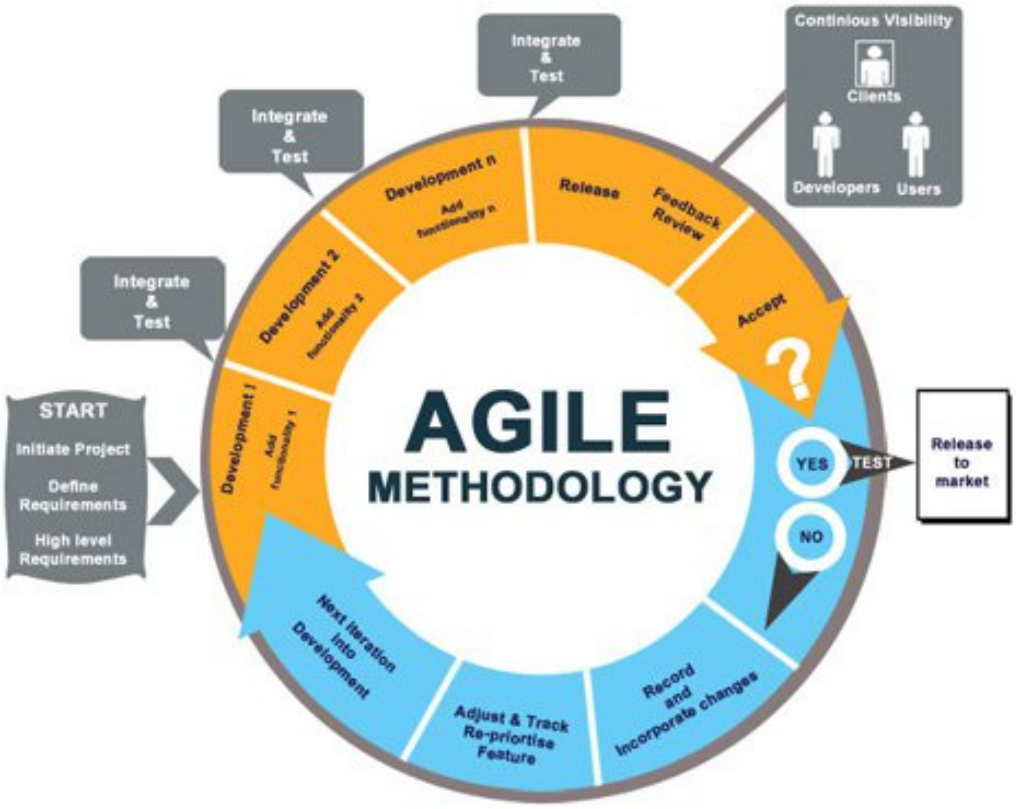

Figure 1: Agile Model [1]

Budgeting

- It defines how much we have to spend based on the scope of the work
- Also, it tends to ignore the cone of uncertainty

Estimation

- Presents an approximation of Cost,effort and duration based on size and project nature
- Focused by the cone of uncertainty (a range based on knowledge)

Planning

- Defines tasks and allocates resources
- Focused on the narrow part of the cone of uncertainty (a much smaller range)

1.2 Motivation

At early stages of software development, effort must be estimated to come up with a planned schedule and budget. Software processes constantly evolve as new and different technologies and applications are developed and used. Especially, in current software industry, where changes are arbitrary, an evolving system is required for the problem of cost estimation especially in agile environment.

The latest Release of ISBSG (International Software Benchmarking Standard Group) volume 11 [16] has reported the fact that only 6.4 % of the total projects (the one which are included in survey) in the world are developed through agile methodologies in spite of the several advantages of the methodology like quick delivery, high customer involvement, iterative and incremental model, always welcomes requirement changes etc. The graph below depicts the use of different methodologies to develop software's.

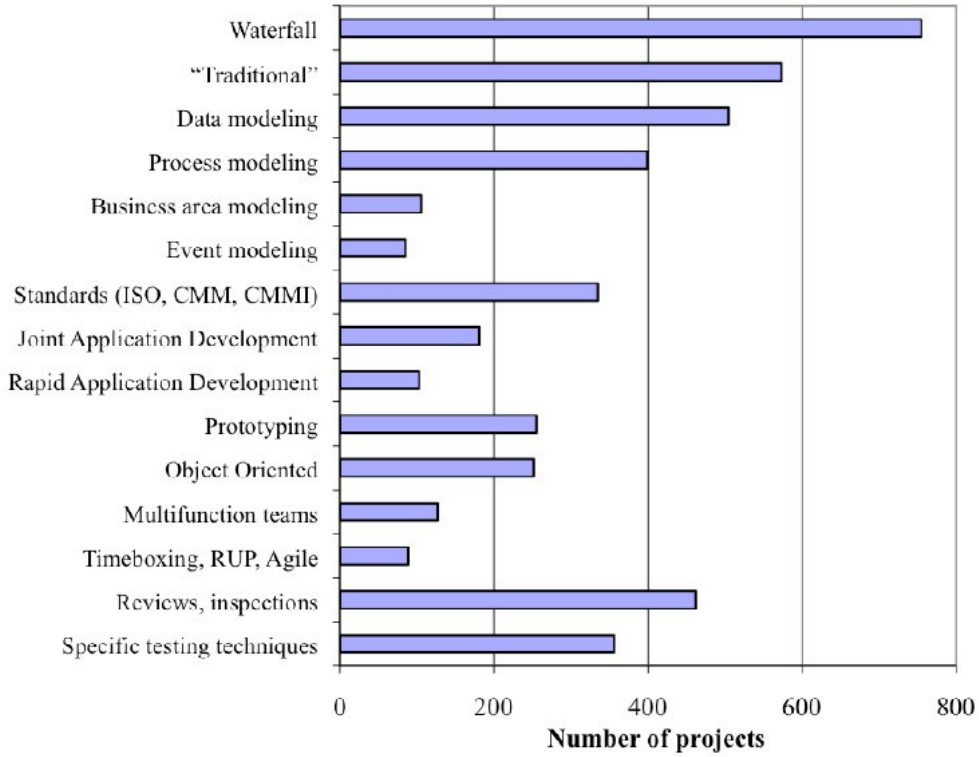

Figure 2: Comparision of Modelling techniques

Development techniques	Projects	Percent
Waterfall	745	
"Traditional"	713	
Specific techniques described	1384	
Data modeling	505	36.5 %
Process modeling	399	28.8 %
Business area modeling	106	7.7 %
Event modeling	85	6.1 %
Standards (ISO 9000; CMM, CMMI)	335	24.2 %
Joint Application Development	181	13.1 %
Rapid Application Development	103	7.4 %
Prototyping	255	18.4 %
Object Oriented Analysis/Design, UML	252	18.2 %
Multifunction teams	127	9.2 %
Timeboxing, RUP, Agile	89	6.4 %
Reviews, inspections, walkthroughs	462	33.4 %
Specific testing techniques	356	25.7 %

Table 1: Percentage usage of Different Development technique [16]

Estimation this day's has been a lightning rod for the discussion in all methods (agile, waterfall, iterative or water fountain) with the issues of predictability and standardization. Because of the controversy this is an area where a wide range of hybridization has always occurred. Organizations adjust techniques which can be fitted over governance structures, culture and risk profiles. There is no one size fits for all solution.

When surveyed with the recent research [5, 12, 22, 26] that is ongoing in agile methodology it has been found that one of the reason for not using agile methodologies is lack of efficient estimation methodology which can accurately estimate the effort and time required to develop a project. So in order to support or to enhance the use of agile methodologies need for some good estimation techniques arises which can provide accurate results.

1.3 Related work

From the Software Cost, Effort and Time Measurement point-of-view the metrics and methods from conventional lifecycle models cannot be conveniently used without changes.
J.M. Desharnais et all in [26] developed on the COSMIC (Common Software Measurement Consortium) method guidelines an estimation approach which works on COCOMO approach incorporating quality of documentation for functional analysis. It uses Cumulative Function Points (CFP) for estimation wherein each of the User Stories are defined by a single Function Point (FP) called User Story Point or USP. COSMIC has introduced the first official guideline in Agile software development methods for software sizing measurement based on function points.[27]

However the mapping between CFP and USP is subjective. It requires expert judgment and involves some degree of guesswork.

In another model given by Abrahamsson et all [16] predictors are extracted from current user stories and then used for next stories. This incorporates elements of analogy based estimates but works only in case the user stories are clearly written and well-structured.
In another research Zia et all [28] propose a SWOT analysis based method which uses influence of internal vs external factors and quantifies them. Factors such as team composition, process used, team dynamics, clarity of requirements etc are considered. In order to understand the factors that influence agility dimensions in a project, Lee and Xia[30] suggest a model which uses a trade-off relationship between response extensiveness and response efficiency of the team[30].
Asnawi et all in [20] used the Factor Analysis technique to identify 15 factors, by evaluating the responses they received in the survey. They explained contributions in several IT areas such as process/governance, quality assurance, iterative and incremental development and team communication. However these were not fully related to the aspects of projects which impact project performance such as cost, quality, deadlines and scope.
In the literature survey it was observed that recent researches are using PCA based models for software cost estimation for traditional software development.

Tosun et al. in [32] proposed feature ordering in terms of PCA based factor-importance and provided heuristics for it. They ordered the features according to absolute values of the elements of the eigenvector of the first principal component only.
and the features were ordered .
In [34] the researchers J Weng, Shixian Li, Linyang Tang demonstrated how PCA based models can provide significant improvement in reliability and accuracy of effort prediction over Traditional analogy based models. They compared the performance of PCA-based feature extraction with analogy based methods on three public datasets namely COCOMO, NASA and Desharnais. It was found that their WPCAA model outperforms the traditional analogy based models in terms of MMRE and PRED(25).

Drawing inspiration from this we explored the use of Genetic Algorithm in the Agile Software Development environment instead of PCA based extraction technique for Optimizing the various agile factors using dimensionality reduction technique and then using Artificial Neural networks to train the data sets having those optimized factors which were extracted using GA from various agile projects.

1.4 Research Problem

In the recent past researchers have proposed different methodologies for cost estimation for software projects which use Agile Development methodology. These methods use large number of project characteristics such as story points, story size, factors related to team dynamics, process model used, factors related to management, communication skills in the team, quality and clarity of requirements etc.. to estimate development cost.
In this work we apply Genetic Algorithm so that we can significantly reduce the dimensions of the attributes required; and identify the key attributes which have maximum correlation to the development cost. This might improve the cost estimation process. Also the extracted factors are trained via Artificial Neural Networks .
It was also concluded that software project estimation must be handled using an evolving system like artificial neural network rather than a static one. This study supported our notion of using machine learning methods for estimation. . This technique will help one to achieve lower Mean relative error (MRE) value and will help to reduce the losses due to inaccurate estimation.
The Research Problem can be thus be stated as follows:
To propose a robust cost estimation model for agile software projects, which extracts key factors using Genetic Algorithm and validating intense project characteristic based on artificial neural network which help in estimating the development cost using these extracted factors .

1.5 Scope of the work

This thesis reports on the results obtained by exploratory factor analysis carried out on Agile development data of various industry projects. The proposed cost estimation model uses Genetic Algorithm to derive the correlation between various project attributes and the cost of development. A Genetic Algorithm approach is used to estimate the development cost with the use of an artificial neural network based estimation method for agile software projects, which extracts key factors using Genetic Algorithm and validating intense project characteristic based on artificial neural network which help in estimating the development cost using these extracted factors.

The scope of the thesis is presented as follows:

(i) We do exploratory factor analysis of the agile development data. Using Genetic Algorithms, we extract factors called as Major Factors and their fitness values . These Major Factors are the factors which have most affect on the development cost and they account for most variation in the agile development data.

(ii) Now with the help of a trained artificial neural network, whose inputs will be these extracted Major factors for the project, will get a value of Cost as the output of the ANN.

(iii)The work is been validated with the help of the famous data set which contains about 250 projects with their actual time taken and actual cost taken to develop an software through agile methodologies. The dataset is shown in Appendix-1.

1.6 Thesis organization

Chapter 1 Begins with General introduction and related work. It addresses the topics like Motivation, Problem Statement, Scope, Related work and thesis organization.

Chapter 2 Provides a detailed description about agile methodologies, explains different types of agile methodologies, Cost estimation and Benefits of Accurate estimation.

Chapter 3 Discusses the software cost estimation models as used in traditional software development, such as LOC model, Regression models, COCOMO, CAEA and ANN etc.

Chapter 4 Presents the proposed research methodology which checks the flexibility of using Genetic Algorithm and ANN in Agile for cost estimation and presents several estimation approaches in research.

Chapter 5 Shows the implementation of the proposed methodology also the tools used in it.

Chapter 6 Presents the results and analysis part of the proposed methodology.

Chapter 7 Concludes the thesis.

Chapter 2 Agile Software Development

2.1 Introduction

Agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen tight interactions throughout the development cycle.
Agile methodology is a substitute to traditional project management, typically used in software development. It helps a team to respond for unpredictability through incremental, iterative work, known as sprints. Agile methodologies are an alternative to waterfall, or traditional sequential development.

2.2 Agile development Environment

Agile development methods promote development, teamwork, collaboration, and process adaptability throughout the life-cycle of the project.
 Iterative, incremental and evolutionary

Agile methods break tasks into small increments with minimal planning and do not directly involve long-term planning. Iterations are short time frames (time-boxes) that typically last from one to four weeks. Each iteration involves a cross-functional team working in all functions: planning, requirements analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is demonstrated to stakeholders. This minimizes overall risk and allows the project to adapt to changes quickly. An iteration might not add enough functionality to warrant a market release, but the goal is to have an available release (with minimal bugs) at the end of each iteration. Multiple iterations might be required to release a product or new features. Demonstration will minimizes overall risk in the development of project and hence will allows the project for adapting to changes rapidly. Each iteration will not increase enough functionality to the software that it can be released to the market or client-release. However but the goal was to make available a small version of the product having some or all the functionality but with minimal bugs at the end of each iteration. Many such iterations are to be required to release a product or new features.

- Extensive communication among team members

In agile teams the stakeholder or the customer appoints a representative like a Product Owner for SCRUM. He will be acting on behalf of the customer/client and has the decision making power. He is the person who will approve changes done by the team. He will also foresee the team's work so as to a make sure that they are on the correct track of development. If the team has any doubts or queries in the ,middle of the iteration he is the person responsible for correct answers. His interpretation of client requirements are considered of utmost importance.
In 2002 Alistair Cockburn coined the term "information radiator". It is used to inform the whole team and other stakeholder about the status of the project and the direction in which the product is headed. It is normally in the form of a big physical display located prominently in an office, where every member can see it.

Very short feedback loop and adaptation cycle

A common characteristic of agile development are daily status meeting or "stand-ups", e.g. *Daily Scrum (Meeting)*. In a brief session, team members report to each other what they did the previous day, what they intend to do today, and what their roadblocks are.

Quality focus

Specific tools and techniques, such as continuous integration, automated unit testing, pair programming, test-driven development, design patterns, domain-driven design, code refactoring and other techniques are often used to improve quality and enhance project agility.

2.3 How is Agile Different from Traditional Software Development

Barry Boehm described agile methods as “an outgrowth of rapid prototyping and rapid development experience as well as the resurgence of a philosophy that programming is a craft rather than an industrial process”.

TRADITIONAL APPROACH	AGILE APPROACH
Deliberate and formal, linear ordering of steps, rule-driven	Emergent, iterative and exploratory, beyond formal rules.
Optimization is the goal	Adaption, flexibility, responsiveness is the goal
In this type the environment is taken as stable and predictable.	In this type the environment is taken as turbulent and difficult to predict
Sequential and synchronous process	Concurrent and asynchronous process
It is work centered process because people will change according to different phases	It is people centered process, as the same team is developing throughout.
Project lifecycle is guided by tasks or activities.	Project lifecycle is guided by product features.
Documentation is substantial.	Documentation is minimal.
Developers do waiting until the architecture is ready.	The whole team is working at the same time on the same iteration
Too slow to provide fixes to user	Provide quick responds to user feedback
Change requirements is difficult in later stages of the project	Can respond to customer requests and changes easier
More time is spent on design so the product will be more maintainable. The "what ifs" arise earlier	There is no time for the what ifs
No communication within the team, novices stay in their rooms and try to understand things	High level of communication and interaction, reading groups, meetings
Restricted access to architecture	The whole team influences and understands the architecture. Everybody will be able to do a design presentation
Documents and review meetings are needed to solve an issue	5 minutes discussion may solve the problem
Everything is up front, everything is big before you start	The focus is on whether customer requirements are met in the current iteration

Table 2. Differences between traditional approach and agile approach to software development[1][4][6][8][13][14]

2.4 Characteristics of Agile methodologies

Some of the characteristics of the agile processes are as follows:
· The software versions are released quickly one after another as the iteration size and the release cycles are short,

· The complex functionality is designed using an easy to understand architecture,

· The clients and developers and management team, all take collective ownership over the product

· High coding standards are followed so as to produce a very sophisticated code; Refactoring and re-use of code

· continuous testing of the code just as they are developed; comprehensive regression tests and acceptance tests by the clients

· Continuous integration so that the product can be viewed a whole

2.5 Types of Agile methodologies

Extreme Programming
Extreme Programming (XP) is the most commonly used agile software development method. It is also seldom used a reference when we talk about the general characteristics of Agile software development methodologies.
According to XP principles the software development projects need to focus more on the people involved rather than the documents, processes and tools. XP provides a set of practices, values and principles [29]:
· Values : communication, simplicity, feedback, courage

Principles : incremental changes, honest measuring

· Practices : pair programming, short version cycles
These are derived from the industry experienced best practices. It has been documented and experienced that XP helps the teams in software development projects. To fulfil some of these completely varying project characteristics, the agile software development model establishes an XP product life cycle just like traditional life cycle models such as waterfall-model, or spiral model [28].

General Way of Developing Software through XP

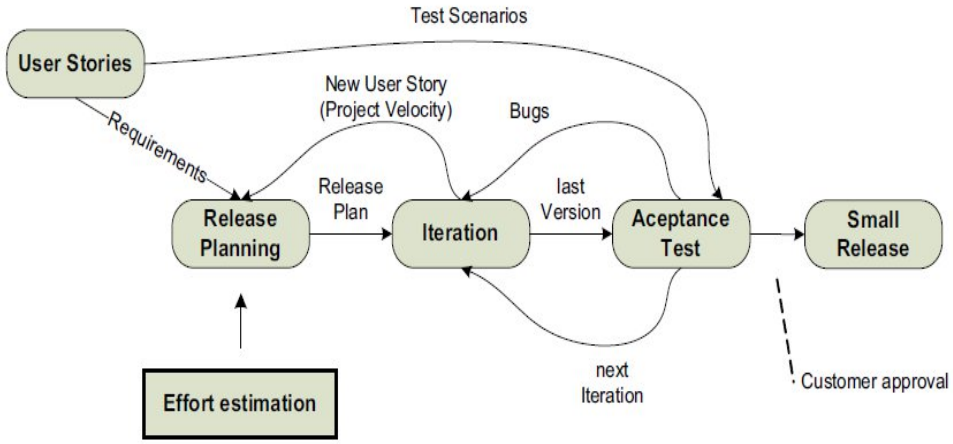

Figure 3: General Way of Developing Software through XP [29]

Agile Modeling (AM)

Modeling is an important step in software development. It enables software developers to get a blue print about complex issues before addressing them in programming. Agile Modeling (AM) was established by Scott Ambler in 2002. It is a collective set of values, principles, and practices for modeling software that can be used for software development project in an effective and easy manner [1].

The values of AM, which are considered to be an extension to the values of XP include: communication, simplicity, feedback, courage, and humility. Humility means to admit that you may not know everything; others may know things that you do not know, and thus, they may provide useful contribution to the project [1].

Again, the principles of AM are quite similar to those of XP, such as assuming simplicity, embracing changes, incremental change of the system, and rapid feedback. In addition to these Principles, AM principles include the knowledge of the purpose for modeling; having multiple effective models; the content is more important than the representation; keeping open and

honest communication between parties involved in the development process; and finally, to focus on the quality of the work [1].

The practices of AM have some commonalities with those of XP, too. An agile modeler needs to follow these practices to create a successful model for the system. AM practices highlight on active stakeholder participation; focus on group work to create the suitable models; apply the appropriate artifact as UML diagrams; verify the correctness of the model, implement it and show the resulting interface to the user; model in small increments; create several models in parallel; apply modeling standards; and other practices [1].

Agile Model Driven Development (AMDD) is the agile version of model driven development. To apply AMDD, an overall high level model for the whole system is created at the early stage of the project. During the development iterations, the modeling is performed as planned per iteration. Usually, AM is applied along with other methodologies, such as Test Driven Development (TDD), and Extreme Programming (XP), to get the best results [1].

AM basically creates a mediator between rigid methodologies and lightweight methodologies, by suggesting that developers communicate architectures through applying its practices to the modeling process [2]. In a nut, agile modeling defines a collection of values, principles, and practices which describe how to streamline the modeling and documentation efforts. It is usually applied in conjunction with agile implementation techniques for good results.

SCRUM

SCRUM methodology was initiated by Ken Swaber in 1995. It was practiced before the announcement of Agile Manifesto. Later, it was included into agile methodology since it has the same underlying concepts and rules of agile development. SCRUM has been used with the objective of simplifying project control through simple processes, easy to update documentation and higher team iteration over exhaustive documentation [4].

SCRUM shares the basic concepts and practices with the other agile methodologies, but it comprises project management as part of its practices. These practices guide the development team to find out the tasks at each development iteration. In addition to the practices defined for agility, one main mechanism recommended by SCRUM is to build a backlog. A backlog is a place where one can see all requirements pending for a project, sized based on complexity, days or some other unit of measure the team decides. Inside a product backlog, there is a simple sentence for each requirement; something that will be used by the team to start discussions and putting details of what is needed to be implemented by the team for that requirement [4].

For SCRUM, three main roles are defined as shown in Fig 4. The first role is the product owner, who mainly would be the voice of business. The second role is the SCRUM team which comprises developers, testers, and other roles. SCRUM master, the third role, is responsible for keeping the team focused on the specific tasks [4][3].

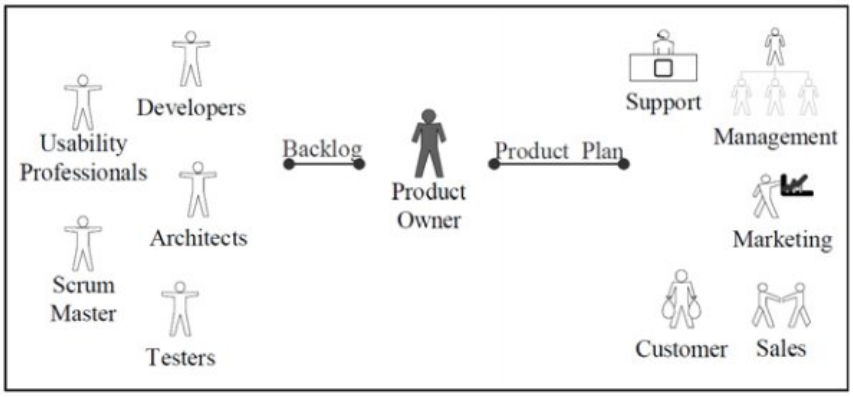

Figure 4: Key roles and interaction artifacts in SCRUM [3]

The process of development using SCRUM divides the project into phases. In each phase, one feature is fully developed, tested, and become ready to go to production. The team does not

move to a new phase until the current phase is completed. Whether what is being done adds value to the process or not, is the main concern of each phase.

Current studies on traditional SCRUM development have shown that despites its advantages, it is not best suited for products where the focus is on usability [3]. It fails to address usability needs of the user, because product owners keep their focus mainly on business issues and forget about usability. Since product owners usually come from business background, they lack the experience, skills, and motivation to design for user experiences. Moreover, traditional agile methodologies are not concerned about the user experience vision, which drives the architecture and is essential for ensuring a coherent set of user experiences.

Briefly, SCRUM is considered an iterative, incremental methodology of software development. It was proposed for software development projects, and at the same time, it can be used as a program management approach.

2.6 Why is Agile Development Necessary

In 1970, Dr. Winston Royce presented a paper entitled “Managing the Development of Large Software Systems,” which presented the short-comings of sequential development process used for software projects. He said that software should not be developed like an automobile on an assembly line, in which each piece is added in sequential phases. In this type of sequential development in phases, one phase of the project has to be done and then only the next phase can begin. Dr. Royce spoke against such phase based approach in which developers will initially be gathering all of a project’s requirements, then completely designing of all of the architecture and design, then writing all of the code, and so on. Royce’s paper specifically mentioned the drawbacks of the approach where there is not enough communication among the smaller teams that are working on a particular phase of work.

In waterfall model, development teams get a single chance only for each aspect of a project development. They should produce correct result in that chance only. In an agile modeling, every feature of development — like requirements, design, etc. — is again and again done throughout the lifecycle. At the end of each iteration if there is any mistake or changes need to be done, it is easy to do so.

By using agile development methodology we can re-evaluate the direction of a project at any point during the development lifecycle. This is done by developing the project in small iterations, known as sprints. At the end of each sprint the teams presents a potentially shippable product increment to the clients. Hence, agile methodology can also be described as "iterative" and "incremental".
.

2.7 Difficulties faced during implementation of agile methods

1) Fear of Exposure of Skill-Deficiency

In a review of 17 companies, it was found that the development team was scared that the agile process can highlight the gaps in their skills and expose their deficiencies [11]. So, they felt a pressure at all times while using agile methodologies.

To mitigate this problem, the developers need an atmosphere in which they feel the safety to project their weaknesses. They should be able to document any fears, issues or concerns due to which they didn't feel comfortable in an open forum.

2) Broader Skill Sets for Developers

Generally in software companies using agile development methodology the management requires personnel to exhibit a wide range of skill set rather than specialisation in only one area like program writing using a particular language or build deployment only [11].

To address this problem, organization goals and HR policies and expectations should be realistic. They must strive to provide their employees a well-balanced team with members becoming "masters of all" or "masters of none." The ideal situation will be when the developers have broad knowledge of the stages and features of software development however they are experts in certain areas.

3) Interpersonal Interaction among team members

Agile practices encourage collaborations, among developers and with the other stakeholders. This leads to meetings, retrospectives etc. which require the social interaction. For this the team members hone their inter-personal, communication, and presentation skills. In most of the cases management prefers constant face-to-face communication, as they could see the benefits of increased the degree of communication in agile environment. However there are people who were technically very talented but had weak communication and presentation skills[11].

This challenge can be met by providing social-skills training to the development team so that they can be more comfortable in such social work settings.

4) Understanding Agile Principles

Not all projects are suitable for application of agile values and principles. Sometimes due to irregular combination of staff personality, incorrect management style, company policy or any other factors the projects development teams were forced to implement agile methods. However it was only "on paper," and they could not achieve agility's ultimate goals.

Chapter 3: Software Estimation Techniques

There are many models for software estimation available and running in the industry. Researchers have been working on various estimation techniques since 1965. Initial work in estimation was based on regression analysis or mathematical models of other domains. Among many estimation models expert estimation, COCOMO, Function Point and derivatives of function point like Use Case Point, Object Points are most commonly used. While Lines of Code (LOC) is most commonly used as a size measure. IFPUG FPA originally invented by Allen Alrecht at IBM has been adopted by most in the industry as alternative to LOC for sizing development and enhancement of business applications. Function Point Analysis provides measure of functionality based on end user view of application software functionality. Some of the commonly used estimation techniques are as follows:

3.1 Lines of Code (LOC)

It is a formal method to measure size by counting number of lines of code. LOC is typically used to get the amount of effort that will be required to develop a program, also to estimate programming productivity or maintainability once the software is produced. Lines of Code (LOC) have two variants- Physical LOC and Logical LOC. While two measures can vary significantly.

3.2 Function Point

Software cost estimation is the process of predicting the effort for developing software. Function points are a metric since it provides a sizing gauge for products very early in the development cycle. Function Points represent the effort put into developing the desired features [28]. Once the functions of the product are identified, they are categorized into

distinct types. They are then assessed for their complexity, and function points are assigned to the features. In this paper, function points are used as base cost estimation metric at agile software projects. Common agile projects use story points for estimating the work. Desired features are identified and the total number of story points is estimated at the beginning phase of the project. The total number of story points is reduced by the completion of each user story while the project is progressing. As the story points are measured via comparisons with other stories, the total number of story point can fluctuate with small variations of the base story point.

On the other hand, function points are absolute values. The function points quantify the size and complexity of an application based on that application's inputs, outputs, inquiries, internal files, and interfaces. They are measured based on the complexity of the desired features and the interface of the user story itself. Thus, the total number of function points is more stable than any individual story point.

3.3 COCOMO 81

COCOMO 81 (Constructive Cost Model) is an empirical estimation scheme. [7] It is used for estimating effort, cost, and schedule for software projects. It was derived from the large data sets [19] from 63 software projects ranging in size from 2,000 to 100,000 lines of code. These data were analyzed to discover a set of formulae that were the best fit to the observations. These formulae link the size of the system and Effort Multipliers (EM) to give the effort required to develop a software system.

In COCOMO 81, effort is expressed as Person Months (PM) and it can be calculated as

$$PM = a * Size^{b} * \prod_{i=1}^{15} EM_i$$

Where “a” and “b” are the domain constants in the model. It contains 15 effort multipliers. This estimation scheme accounts the experience and data of the past projects, which is extremely complex to understand. Cost drives have a rating level that shows the impact of the driver on development effort. These rating can range from Extra Low to Extra High. For the purpose of analysis, each rating level of each cost driver has a weight associated with it. The weight is called Effort Multiplier(EM). The average EM assigned to a cost driver is 1.0 and the rating level associated with that weight is called Nominal [7].

3.4 COCOMO II

It is an enhanced scheme for estimating the effort for software development activities, which is called as COCOMO II. In COCOMO II, the effort requirement can be calculated as:

$$PM = a * Size^{E} * \prod_{i=1}^{17} EM_i$$

$$E = B + 0.01 * \sum_{i=1}^{5} SF_j$$

COCOMO II is associated with 31 factors; LOC measure as the estimation variable, 17 cost drives, 5 scale factors, 3 adaptation percentage of modification, 3 adaptation cost drives and requirements & volatility. Cost drives are used to capture characteristics of the software development that affect the effort to complete the project. COCOMO II used 31 parameters to predict effort and time [11] [12] and this larger number of parameters resulted in having strong co-linearity and highly variable prediction accuracy. Besides these meritorious claims, COCOMO II estimation schemes are having some disadvantages. The underlying concepts and ideas are not publicly defined and the model has been provided as a black box to the users [26]. This model uses LOC (Lines of Code) as one of the estimation variables, whereas Fenton et. al [27] explored the shortfalls of the LOC measure as an estimation variable. The COCOMO also uses FP (Function Point) as one of the estimation variables, which is highly dependent on development

the uncertainty at the input level of the COCOMO yields uncertainty at the output, which leads to gross estimation error in the effort estimation [33]. Irrespective of these drawbacks, COCOMO II models are still influencing in the effort estimation activities due to their better accuracy compared to other estimation schemes.

3.5 Planning Poker

Planning poker is the mostly used technique for estimation in agile environment. This technique is highly depended on the expert who takes part in estimation process. Participants in planning poker include all of the developers on the team [6]. The term "developers" refers to all programmers, testers, analysts etc. On an agile project, this will typically not exceed ten people. At the start of planning poker, each estimator is given a deck of cards. Each card has written on it one of the valid estimates. Each estimator may, for example, be given a deck of cards that reads number from Fibonacci Series 0, 1, 2, 3, 5, 8, 13 and 21. Fibonacci numbers have been found to be a very useful estimation sequence because the gaps in the sequence become appropriately larger as the numbers increase. These non-linear sequences work well because they reflect the greater uncertainty associated with estimates for larger units of work i.e. in case of 13 story points, it is difficult to argue whether the card is worth 13 points or 12 points [6].

For each user story to be estimated, a moderator reads the description. The moderator is usually the product owner (customer) or an analyst. The product owner answers any questions that the estimators have. The goal in planning poker is not to derive an estimate that will withstand all future scrutiny. Rather, the goal is to be somewhere well on the left of the effort line, where a valuable estimate can be arrived at cheaply. After all questions are answered, each estimator privately selects a card representing his or her estimate. Cards are not shown until each estimator has made a selection. At that time, all cards are simultaneously turned over and shown so that all participants can see each estimate.

It is very likely at this point that the estimates will differ significantly. If estimates differ, the high and low estimators explain their estimates. The moderator can take any notes he thinks will be helpful when this story is being programmed and tested. After the discussion, each estimator re-estimates by selecting a card. In many cases, the estimates will already converge by the second round. But if they do not, the process is repeated. The goal is for the estimators to converge on a single estimate that can be used for the story.

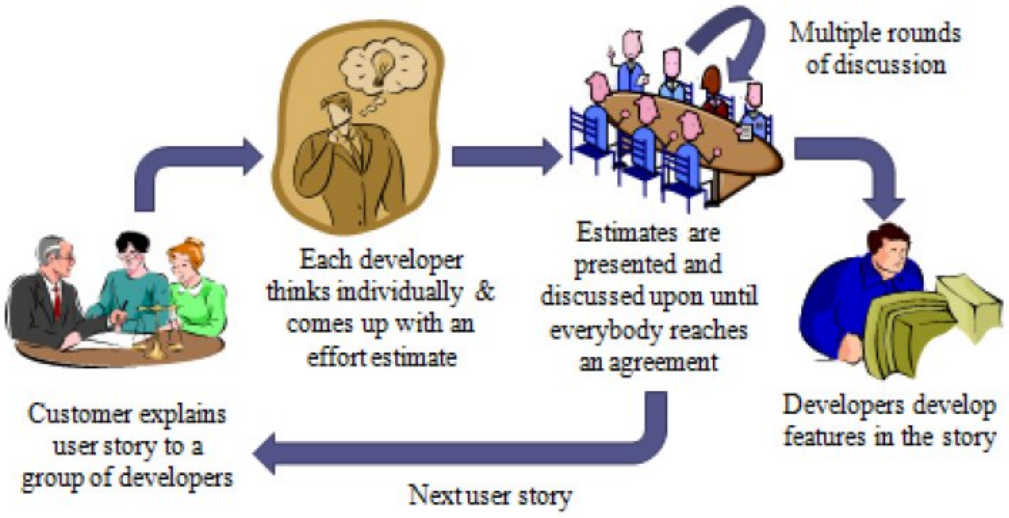

Figure 5: Planning Poker [6]

Shortcomings of Planning Poker

Based on a research [6], it has been concluded that the following areas need to be addressed:

- High Magnitude of Relative Error (MRE) in estimation

  Magnitude of Relative Error is a widely used measure for evaluating the estimation accuracy of different models. For a single estimate, it is defined as:

$$MRE = \frac{|\text{Actual Effort} - \text{Estimated Effort}|}{|\text{Actual Effort}|}$$

Mean MRE is used to quantify the accuracy for the complete model. Based on the estimation data collected from the enterprise for planning poker [6], this value comes out to be 1.0681 or 106.81% which is very high and can be reduced.

- Strong over-confidence in accuracy of estimates

  Software development projects frequently have over-optimistic effort estimates and over-confident assessments of estimation accuracy. It has been observed that [6] there is large percentage of projects which are either over or under estimated using the Planning poker:

Accurate Estimates	30
Over-estimates	49
Under-estimates	44

Table 2: Estimates using Planning Poker [6]

- An expert-dependent method

  Even though planning poker takes every developer's estimate into consideration, the bias towards estimates from experts cannot be fully avoided. From the perspective of managers and developers at the enterprise, an expert is more likely to convince his/her opinion to the rest of the team than a novice developer in the team. And it has also been observed that in absence of an expert in the team, the accuracy of estimates decreases substantially.

3.6. Constructive Agile Estimation Algorithm (CAEA)

In this section, we first propose constructive agile software estimation process followed by the CSD algorithm for the same agile software. We divide constructive agile estimation process into two phases namely; Early Estimation (EE) and Iterative Estimation (IE) as shown in Fig. 6. An estimator can update the estimates whenever the uncertainty in requirements reduces. The purpose of EE is to identify the scope of the project by envisaging the upfront with just enough requirements. This provides just enough understanding to put together an initial budget and schedule without paying the price of excessive documentation. EE also provides the platform for fix budget agile project and generates a satisfaction amongst the stakeholders by providing range of

estimate to reflect temporal risk.

On the other hand, IE is an iterative activity that starts at the beginning of iteration to incorporate new requirements/ changes. These requirements/changes may arise from the customers with detail information of system.
Constructive agile estimation process considers only clear specified requirements and other factors that affect the estimation to derive CSD of the project as shown in Fig. 6. Here, estimation process starts after the prioritization of requirements and EE is just to understand the CSD involved in project. After finalization of project, iteration planning starts and continues till all the requirements/ changes required by customers are exhausted. EE and IE use story points for estimating CSD. Existing AEMs use expert opinion and historical data for evaluating story points. In this section, we propose CAEA to evaluate story points after the inclusion of various CSD affecting factors in AEM. We also discussed the computation of variables in various projects to establish the fact that inclusion of vital factors in agile estimation generates realistic and more precise CSD estimation.
An algorithm CAEA is based on above constructive agile estimation and computes the story points for CSD estimation. It incorporates the vital factors such as project domain, performance, configuration etc. We have graded the intensityof these factors on the scale of low, medium and high based upon the complexity of the project. It is preferred to map the intensity levels with mathematical series such as square series (1, 4, 9) or Fibonacci series (2, 3, 5). Square series has been proved to be the most preferred series in agile estimation since it provides realistic level of accuracy for complex and ill-defined project [Fedrick, 2007].

Formal description of proposed CAEA is described as follows:

Algorithm: *CAEA*
// This algorithm computes the of story points on the basis of the input as the grades of vital factors of a project //

STEP 1: Vital factors of project are identified on the grade of low, medium and high using square series or Fibonacci series.
STEP 2: Compute sum of all grades of various factors for a project denoted as Unadjusted Value (UV).
STEP 3: Decompose the project in small tasks or stories.
STEP 4: Assign Story Point (SP) to each story based upon the size.
STEP 5: Compute New Story Point (NSP) by using equation (1).
STEP 6: Compute SOP by using equation (2).
STEP 7: Compute DOP through equation (3).

$$NSP = SP + 0.1 * UV \quad \text{----------(1)}$$

$$\text{Size of Project(SOP)} = \sum_{i=1..n} NSP_i \quad \text{----------(2)}$$

$$\text{Duration of project (DOP)} = SOP / \text{Velocity} \quad \text{----------(3)}$$

where SP is story point of a story,
UV is unadjusted value,
NSPi is NSP of ith the story and n is total number of stories of project,
SOP is size of project,
velocity is number of NSP developed in a iteration.

Cost of Project (COP) = NSP * Cost per story-point

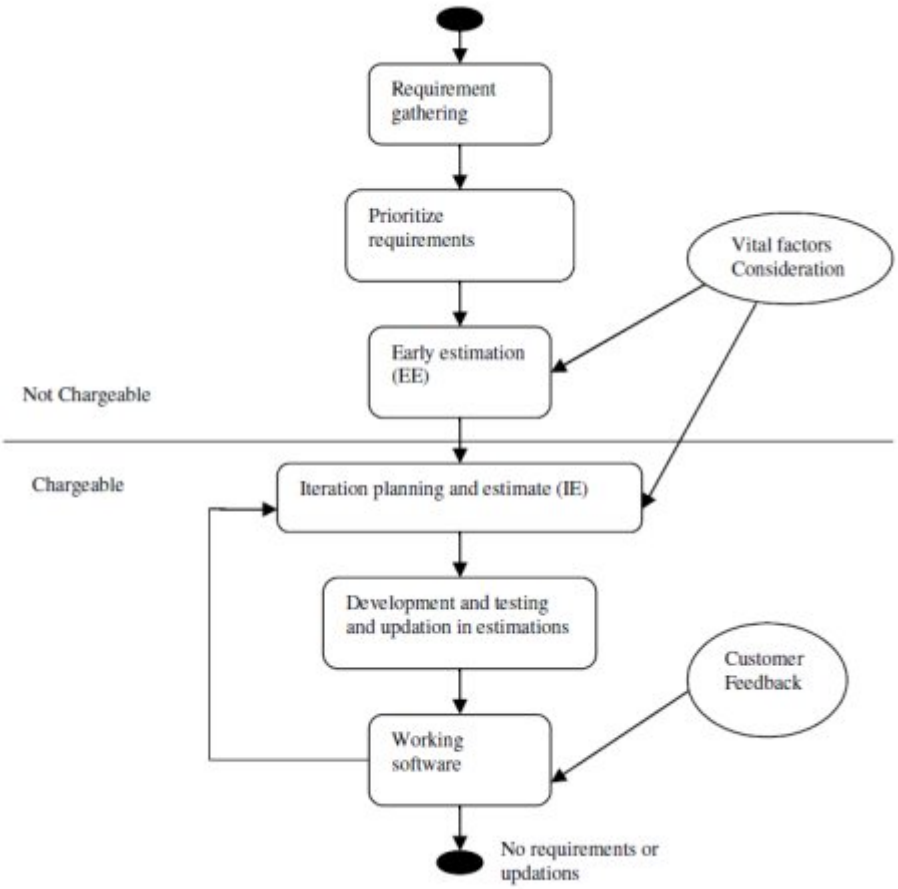

Fig 6.Estimation Activity Diagram

3.7 Estimation using artificial neural network

Artificial Neural Network (ANN) is a network composed of artificial neurons or nodes which emulate the biological neurons [11]. ANN can be trained to be used to approximate a non-linear function, to map an input to an output or to classify outputs. Now a day's ANN is highly used for estimating the cost required for developing software's by traditional methodologies [15] [17].

The most prominent topology of ANN is the feed-forward networks. Feed forward ANN layers are usually represented as input, hidden and output layer. If the hidden layer does not exist, then this is called perceptron. The perceptron can map an input to an output if the relationship between is linear. If the relationship between the input and output is non-linear, one or more hidden layers will exist between the input and output to accommodate the non-linear properties.

Several types of feed-forward NN with hidden layers exist. These include Multilayer Perceptron (MLP), Radial Basis Function neural Network (RBFNN) and General Regression (GRNN). A MLP contains at least one hidden layer and each input vector is represented by a neuron. The number of hidden neurons varies and can be determined by the trial and error so that the error is minimal. In this thesis, MLP type is used to predict software cost based on Software factors extracted via Genetic Algorithm which affect the cost in agile environment [17].

3.8 Recent Software Cost Estimation Approaches for Agile Development

1. SWOT based estimation model given by- Ziauddin, Shahid Kamal Tipu, Shahrukh Zia, "An Effort Estimation Model for Agile Software Development", *Advances in Computer Science and its Applications (ACSA)*, 2012

The authors proposed a multidimensional view to produce accurate and effective estimates using a *SWOT* according to Internal vs. External influences. They used data collected from past projects combined with mathematical formulae to develop a model to estimate the effort, project duration and

cost. The model predicts Completion time and cost for agile software project. They identified the key differences between team organisation in agile and traditional development approach as follows:

1. Agile teams are "Whole" :

It is an XP practise which implies that the team members have the requisite skills between themselves only. Within the members of the team they have the required testing skills, interfacing skills, UI designing skills, database skills, translation skills etc. and do not need any external teams dependency to complete the project.

2. Agile teams are formed of generalizing specialists:

A generalizing specialist is someone who has one or more technical specialties e.g. Java programming, project management, database administration.

3. Agile teams are stable:

This means that any significant change in the team structure or organisation can affects on the project performance.

The authors found that the scrum practitioners used a comparative scale for estimating the development effort. That is the development cost is estimated comparative to previous projects; where the scale is often a Fibonacci scale [1,2,3,5,8].This means that the story ranked 3 is approximately thrice as costly to develop than the one ranked 1.

The authors argued that this method of prediction by relative estimation does not take into account the underlying elements that affect effort and uncertainty.

Effort estimation is based on user story size and its complexity. Using these two vectors, effort of a particular User Story is determined using the following simple formula:

ES= Complexity x Size

For project cost estimation the concept of agile velocity was used. *Velocity* is calculated as

$$Velocity\ Vi = \frac{units\ of\ work\ completed}{sprint\ time}$$

This is the observed velocity or initial velocity Vi.

The authors proposed a mechanism to optimise the velocity value by two factors:

i. The Friction or consistent forces that are a constant drag on productivity and reduce Project Velocity.

ii. The Variable or Dynamic Forces that decelerate the project or team members and cause the Project Velocity to be irregular.

Friction Factors are the external factors such as environment factors, process factor, team dynamics etc which negatively impact the productivity hence increase the cost. The aim is to reduce or minimise the friction factors.
The friction value FR is given as a product of all individual friction factors values.
Variable Factors are internal factors which are unpredictable and unexpected. They are for a brief time and introduce a little irregularity in the velocity hence affect the cost. These forces need to be made consistent and predictable as much as possible so that their effect on the agile velocity can be predicted at the time of estimation.
Factors such as re-organisation of team, change in management, new tools or process, unclear requirements, personal issues of development team etc can be considered under variable factors.

Delhi Technological University, 2014 Page 49
The dynamic force DF is given as a product of all individual variable factor values.
Now the authors gave the following formulae for optimised velocity
Where V= optimized velocity, D = deceleration.
The following formula is given for estimating development cost:
Here ES refers to story point values.
Using empirical data the authors calibrated their model and found the net ratio to be 1.68.
The authors have also measured the estimation accuracy by performing experimental analysis on data of previously developed projects from different software houses. They found that MMRE for estimation of cost comes out be 61.90%.
This model also provides for uncertainty in the measurement by introducing a *"Span of Uncertainty"*. The estimator can never be 100% sure of their estimates, hence they have a *confidence level CL*. It indicates how much confidence they have in their estimates, how sure they are of their understanding of the magnitude of project factors. CL is input in terms of % value. Typically it ranges between 80% to 95%. Using the CL confidence level indicator, the model helps find the variation range in the predicted cost. The lower bound of this range is *Optimistic Point* and the upper bound is *Pessimistic Point*.

2. Chandrasekaran, R. Lavanya S., and V. Kanchana. "Multi-criteria approach for agile software cost estimation model", *Proceedings of APA*, 2007

The authors have modelled a software cost estimation process with a number of constraints imposed by stakeholders and environmental characteristics, thereby satisfying multitudinous criteria using concurrent constraint programming. The proposed work is to focus on the various factors affecting the people-oriented environment. The authors have argues taht in agile environment, the development cost of a software project is dependent to a great extend on the people and management issues. The cost factor for agile software is based on a multiple-criteria approach.
The conceptual model describes the idea of arriving at the criteria set required to emulate the agile environment. The major quality-attributes relating to the each of the agile manifestoes and affecting the agile software are identified. Numerical weights are attached to them that represent the effect of the attribute on the product quality and time of completion. the authors have called them *Quality Weights (QW)* and *Time Weights (TW)* respectively.
These attributes are the cost drivers and by combining the agile manifestoes with the various quality and time attributes, a particular set of the attribute levels are derived that forms the criteria for estimation.

The below tables show the attributes and their time and quality weights as given by the authors.

High Priority Attributes	Quality weight *QW*	Time Weight *TW*
Communication skills	5	-4
Proximity of team	3	-4
Feedback	1	5
Courage	1	-3
Managerial skills	5	-5
Consistent working	2	-2
Technical ability	3	-2
Debugging capability	2	-2
Reliability	3	-2
Function points	3	4
Ease of use	4	3
Early deliver	5	-2

Table :High priority attributes and their weights as given by the model.

High Priority Attributes	Quality weight *QW*	Time Weight *TW*
Process maturity	4	-3
Toolavailabilty	4	-2

Tool familiarity	3	5
Conservativeness	1	3
Training	4	4
Planning	4	-3
Pages of documentation	2	3
Project complexity	1	4
Other expenditure	2	3
Documentation resources	2	3
Documentation period	4	3

Table : Low priority attributes and their weights s given by model.

3. Asnawi, Ani Liza, Andrew M. Gravell, and Gary B. Wills. "Factor analysis: Investigating important aspects for agile adoption" ***AGILE India (AGILE INDIA)*****, 2012. IEEE**

The authors used the Factor Analysis technique to identify/propose 15 factors. They conducted a survey, and evaluated the practices having significant contributions in several IT areas such as process/governance, quality assurance, iterative and incremental development and team communication but not directed at project performance aspects such as cost, quality, deadlines and scope.

Factor analysis is conducted to identify the clusters of the variables (or items) and how they are inter-related to produce factors. The clusters of variables resulting from this analysis can serve as a reference to the practitioners planning to adopt the methodology.

To see how Agile adoption variables in our study are inter-related, questions regarding Agile adoption were asked to software practitioners. The suitability and appropriateness to conduct factor analysis with the data need to be checked. One of the statistical measures used to identify this is called Kaiser-Meyer-Olkin (KMO). It is a measure of sampling adequacy which ranges from 0 to 1.

Values between 0.5 and 0.7 are mediocre, values between 0.7 and 0.8 are good, values between 0.8 and 0.9 are great and lastly values above 0.9 are superb. A KMO with 0.6 is suggested as the minimum value for a good factor analysis. If the value yields more than 0.7, then the correlation on the whole are sufficient to make factor analysis suitable.

The authors on measurement found a KMO value of 0.755 was obtained from their data.

From an initial 27 factors they have extracted 8 factors. The Eigen values ranged from 0.093 to 7.852. The eight extracted factors and their related variables are described as follows:

VARIABLE	LOADING
FACTOR-1 Level of Involvement of the developer and impact of their opinion	
Responsibility of the developers towards organisation's Agile mission	0.816
Involvement of developers in setting goals for Agile activities	0.805

Importance of identifying project scope and suitability of agile in the project	0.674
Transparency and encouragement to developers	0.497
Allowing of interpersonal interactions among the developers	0.564
FACTOR-2 Organisational Culture and People Related Aspects	
Presence of people of different ethnic and racial backgrounds	0.845
Use of English as communication language	0.810
change of mindset when using Agile practices	0.434
FACTOR-3 Customer Involvement when Practicing Agile methods	
Involvement of customers in setting goals for Agile activities	0.680
Requirement of special skills for agile practises	0.656
Resposibility of the customers towards fulfilling the organisation's Agile goals	0.615
Customers knowledge of Agile Practises	0.556
FACTOR-4 Benefits/Impact of using Agile methods	
Focus on customers' satisfaction when using Agile methods	0.881
Efficiency in the software development process achieved by collaboration between both parties i.e. customers and developers.	0.867
Agile Developer's Morale	0.585
Quickness of delivery of results by Agile Methods	0.495
FACTOR-5 Importance of training and learning	
Training for Agile methods	-0.879
continuous learning helping in knowledge transfer when using Agile methods	-0.811
FACTOR-6 Importance of Technical and Technological Aspects when using Agile	
Suitability of Agile methods for that specific project and technology	-0.943
Importance of tools for supporting the usage of Agile method	-0.507
emphasises on achievement and goal accomplishment	-0.414
FACTOR-7 Importance of Sharing, Knowledge. etc	
Personal interaction between team-mates	0.614
Limitations regarding Knowledge about Agile Practices	-0.530
FACTOR-8 Team Commitment and Clarity of Purpose	
Clarity in division of knowing roles and responsibilities of each member	0.694
attitude such as team spirit and team commitment required from everyone	0.515
Early Delivery Time requirement	0.493

Table : Variables and their loadings

The authors found organisational and software developers' involvement as the top factor when using Agile methods. This result also shows that language is one of the important aspects when adopting Agile methods. In terms of the impact that Agile can deliver, high loadings (greater than 0.8) were found to be in customer satisfaction and the ease of software development as a result of collaboration between developers and customers.

These extracted factors explain the practitioners' negative perceptions about the importance of agile development technical aspects in adoption process, valuing more human aspects such as customer satisfaction and collaboration among developers and customers.

The top factors identified in the study are shown in terms of (i) developer involvement and organisation-related aspects, (ii) cultural and people related aspects and (iii) customer collaboration and the need for professional skills when using Agile methods. In addition, factor analysis discovered that practitioners disagreed about the importance of the technical aspects of Agile. Moreover, the study also can help other adopters from to understand and see the suitability of Agile methods for their organisations.

4) PCA based cost estimation model for Agile Software Development Projects:

Using inspiration from recent researches involving SWOT analysis based model and use of PCA-based models for estimation in traditional development methods; we propose a cost estimation model as depicted in the figure1. The first step is to identify the factors affecting development costs by analysis of the sample data by using the Exploratory Factor Analysis with factor extraction using Principal Component Analysis (PCA). This results in generation of the coefficient matrix for each of the identified principle components. The second step is to use constraint programming for satisfying the criteria imposed by agile development environment through the agile manifesto. The development cost is determined by using the factors and coefficients generated by factor analysis while satisfying the agile manifesto conditions by the constraint programming. This estimation process is expected to enhance the level of visibility of cost estimation in the planning stages.

A. Data pre-processing

The qualitative data in the data set was converted into a set of quantitative values through summarizing on an N-point scale (where N is a prime number usually 3 or 5 or 7). The next step is smoothening of data which involves

treatment for missing data values. The missing data which was mostly empty was ignored; in case the missing data was less it was filled with the mean value. Next Step is removal of statistical noise and deletion of exceptional/extreme points in data.

B. Exploratory Factor Analysis using Principle Component Analysis

Following the pre-processing step the agile development data will be analyzed using the multivariate statistical technique Exploratory Factor analysis(EFA).In theory, EFA is a technique for exploratory data analysis consisting in data reduction or a structure simplification, to describe, if possible, the ratio of the covariance among the many variables in random and unobserved quantities named factors. Determine nature and number of latent variables that account for observed variation and co-variation among set of observed variables. These steps are illustrated in Figure2.

The objective of the use of factor analysis is to explore the sample data to generate future hypotheses about development costs involved in software development using agile practices. Factor extraction is done using statistical analysis method of Principal Component Analysis (PCA).

This method is selected because it explains the total data variance represented in the variables in data reduction to factors

The central idea of PCA is to reduce the dimensionality of a data set consisting of a large number of inter-related

variables, while retaining as much as possible of the variation present in the data set. This is achieved by transformation to a new set of variables, the PCs which are uncorrelated, and which are ordered so that the first few retain more variation than the rest of the components. Using the PCA method will reduce multiple related factors into a few comprehensive and linearly un-correlated variables which can include the most information of the agile development data.

In particular, the EFA model is represented by following set of equations

$X_1 = \ell_{11}F_1 + \ell_{12}F_2 + \ldots + \ell_{1m}F_m + \varepsilon_1$

$X_2 = \ell_{21}F_1 + \ell_{22}F_2 + \ldots + \ell_{2m}F_m + \varepsilon_2$

:

$Xp = \ell_{p1}F_1 + \ell_{p2}F_2 + \ldots + \ell_{pm}F_m + \varepsilon_p$

the coefficient ℓij is named the (factor) loading of the *i*th variable in the *j*th factor, where the letters *i* and *j* are integer index 1, 2, 3 ... , and $L_{(p \times m)}$ is the factor matrix with loadings. In this context, the factor analysis model assumes that these variables show a linear relationship with the new variables F_n, where n = 1, 2...m.The vector $\varepsilon_{(px1)}$ represents the random errors associated with measurements.

C. Constraint Programming for satisfying Agile Manifesto Criteria

On obtaining the set of principal components/attributes and the score coefficients matrix we have to determine the criteria set required to emulate the agile development environment. The agile manifestoes impose certain constraints on the factors that are to be concurrently solved to obtain these criteria. The objective of this step is to formulate a set of criteria for an agile environment from this huge domain. Constraint Solving follows the process model as shown in figure 3.

In order that the estimation best fits the agile environment, the four manifestoes of the agile are given prime importance. The PCA outputs key factors extracted and their corresponding coefficient values. The extracted factors are classified under the agile manifestos as applicable. Each extracted factor corresponds to one or more manifesto.

Let there be N extracted factors denoted as $f_1, f_2, f_3 \ldots f_N$ *with corresponding coefficient values as* $C_1, C_2, C_3 \ldots C_N$

Now we classify these extracted factors according to each of the agile manifesto and calculate the cumulative factor value for each manifesto.

Let factors f_1, f_4, f_6, f_9 be applicable to the first manifesto. These are then denoted as $f_{11}, f_{12}, f_{13}, f_{14}$; and their respective coefficient values as $C_{11}, C_{12}, C_{13}, C_{14}$.

Similarly f_{21}, f_{22}, f_{23}.. denote the factors which are applicable to the second manifesto and C_{11}, C_{12}, C_{13} are their corresponding factor coefficients; f_{31},f_{32},f_{33}.. denote the factors which are applicable to the third manifesto and C_{21}, C_{22}, C_{23} are their corresponding factor coefficients and f_{41},f_{42},f_{43}.. denote the factors which are applicable to the fourth manifesto and C_{41}, C_{42}, C_{43} are their corresponding factor coefficients.

Next we calculate the cumulative factor value for each manifesto as a summation of individual products of the value of the corresponding attribute and the coefficient value.
Cumulative Factor Value for manifesto-1 F_1 is given as:

$$F_1 = (f_{11} \times C_{11}) + (f_{12} \times C_{12}) + (f_{13} \times C_{13}) + (f_{14} \times C_{14})$$

It can be also expressed as –

$$F_1 = \sum_{i=1}^{N_1} (f_{1i} \times C_{1i})$$

Where N_1 denotes the number of factors which correspond to manifesto-1.
Similarly Cumulative Factor Value for manifesto-2 F_2 is expressed as:

$$F_2 = \sum_{i=1}^{N_2} (f_{2i} \times C_{2i})$$

Where N_2 denotes the number of factors which correspond to manifesto-2.
Cumulative Factor Value for manifesto-3 F_3 is expressed as:

$$F_3 = \sum_{i=1}^{N_3} (f_{3i} \times C_{3i})$$

Where N_3 denotes the number of factors which correspond to manifesto-3
Cumulative Factor Value for manifesto-4 F_4 is expressed as:

$$F_4 = \sum_{i=1}^{N_4} (f_{4i} \times C_{4i})$$

Where N_4 denotes the number of factors which correspond to manifesto-4.
The development cost is estimated as a product of the cumulative factor value of each of the manifesto as –

$$\boldsymbol{COST = F_1 \times F_2 \times F_3 \times F_4}$$

Chapter 4: Proposed Methodology

In this chapter we will describe the underline framework for our methodology. There are two basic objectives for performance of this methodology. First is to identify all project Factors along with their fitness value via Genetic Algorithm approach as they play a vital role in project estimation. Second is to use machine learning algorithm so that there is no ambiguity in project estimation. We propose to use ANN as they have been successfully applied in estimation of traditional methodology.

4.1 Overview of the Cost Estimation Model

Using inspiration from recent researches involving SWOT analysis based model and use of PCA-based models for estimation in traditional development methods, we propose a Cost estimation model as depicted in the figure12. The first step is to identify the factors affecting development costs by analysis of the sample data by using the Exploratory Factor Analysis with factor extraction using Genetic Algorithm. This results in generation of the fitness value for each of the identified Major Factors. The second step is to use Artificial Neural Network on identified Major Factors for Project Cost Estimation in agile. This estimation process is expected to enhance the level of visibility of cost estimation in the planning stages.

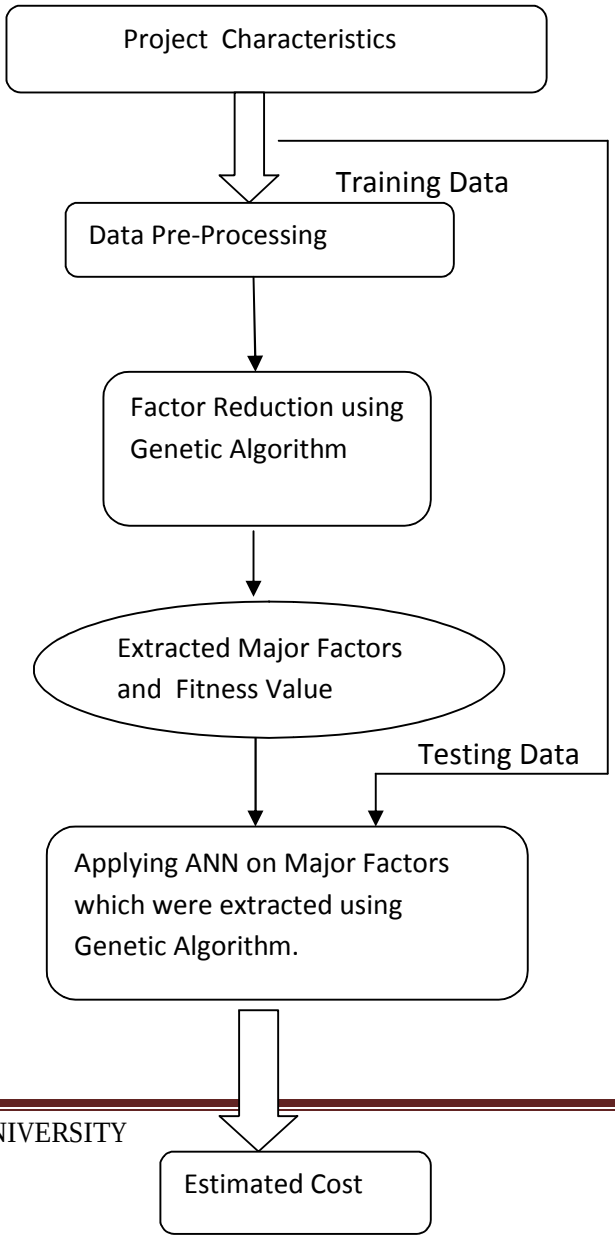
Project Characteristics
Training Data
Data Pre-Processing
Factor Reduction using Genetic Algorithm
Extracted Major Factors and Fitness Value
Testing Data
Applying ANN on Major Factors which were extracted using Genetic Algorithm.
Estimated Cost

Fig 7. Process model for cost estimation

4.2 Data pre-processing

Agile paradigm considers the people factor to be more critical than the process factor. For processes to be predictable, components in it need to behave in a predictable way. However people are not predictable components, hence there exists a difficulty in predicting and quantifying software for cost estimation. Also, uncertainty, risk factors, emerging requirements and complexity issues are presented in agile as in any other traditional software development process.

Hence the qualitative data in the data set was converted into a set of quantitative values through summarizing on an N-point scale (where N is a prime number usually 3 or 5 or 7). The next step is smoothening of data which involves treatment for missing data values. The missing data which was mostly empty was ignored; in case the missing data was less it was filled with the mean value. Next Step is removal of statistical noise and deletion of exceptional/extreme points in data.
The complete data set is divided into two parts - training data set and testing data set.

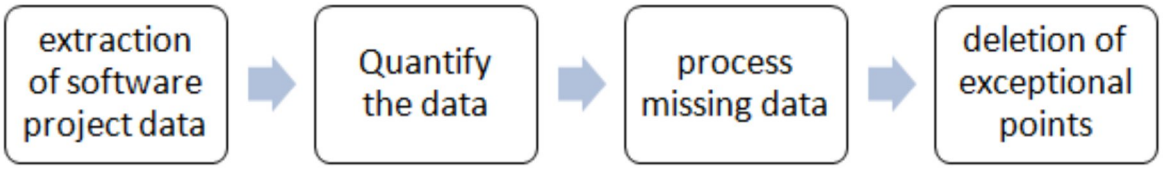

Fig 8. Steps in data pre-processing

4.3 Factors Reduction using Genetic Algorithm

Following the pre-processing step the agile development data will be analyzed using the multi-variate statistical technique Genetic Algorithm.

The idea of GA was first introduced by John Holland in 1975. GA is an algorithm which makes it easy to search a large search space. In GA process, reproduction, crossover and mutation are the three main operators that were used to find near-optimization of the problem. The fittest individual will survive more frequently and have high chances for reproduction.

In theory, genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. Genetic Algorithm solver for mixed-integer or continuous-variable optimization, constrained or unconstrained.

Steps of Genetic Algorithm :

1. Randomly generate a initial value population $X(0):=(x_1,x_2,.....,x_N)$;
2. Compute the fitness $F(x_i)$ of each of the chromosome xi in the current population X(t);
3. Create new chromosomes $X_r(t)$ by mating current chromosomes, playing mutation and recombination as the parent chromosome mate;
4. Delete numbers of the population to make room for new chromosomes;
5. Compute the fitness of $X_r(t)$ and insert these into population;
6. t :=t+1, if not (end-test) go to step 3, or else stop and return the best chromosome.

GA uses three basic genetic operators: reproduction, crossover, and mutation.

1) Reproduction operator uses the fitness of an individual solution and decides whether a string should be select for the new population, also called selection operator. There are a number of 0.83methods used to select an individual in the new population, Roulette wheel is one of the famous among them.
2) Crossover requires a mating of two randomly selected strings. A part of the string is exchanged between two. The main characteristics of the parents are transferred to their child used for the new generation.
3) Mutation operator works on the bits of the individuals. It adds information in a random way that introduces diversity in the population. A number of bits are randomly changed when the string is copied into the new population. Basic GA operations are illustrated in below figure:

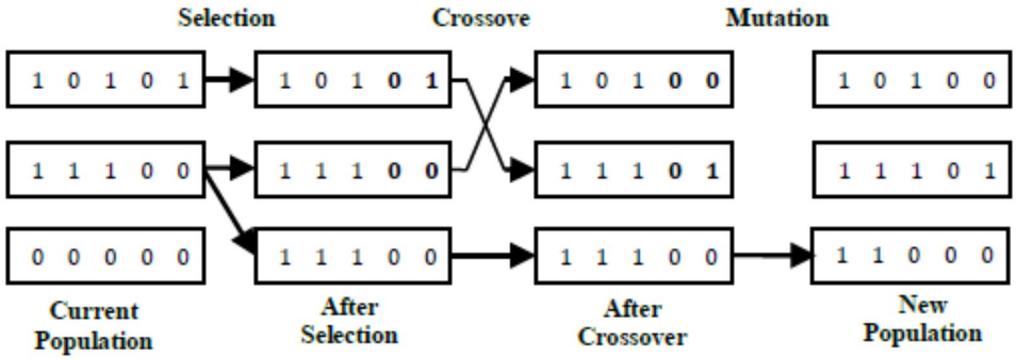

Fig 9 : Basic GA Operations

Here we use the Genetic Algorithm toolbox to optimize the various Agile factors that are driven from different project datasets for Cost Estimation. The process of GA is shown in below figure

Step 1:[Start] Generate Random population of chromosomes that is suitable solutions for the problem. We have population size as 100 and chromosomes /Individual are the total factors i.e 40 for project cost Estimation.

Step 2[Fitness] Evaluate the Fitness of each factor in the population and also evaluate predicted development cost for each new model parameter for j number of projects in the dataset.

Evaluate individual s fitness for the project t ,the equation used is :

$$? ??? ???_{??} = ? ? ?_{?}^{?} - ? ? ?_{?}^{??} / ? ? ?_{?}^{?}$$

Where s=individual number and t=project number, COP is cost of project as discussed in Ch-3.

Step 3: Individual fitness is calculated as the average value of all Project specific fitness values of an Individual achieved during steps 2 . The fitness value depends on the difference between real development cost and predicted development cost.

$$? ??? ???_{?} = \frac{?}{?} \cdot \sum_{?? ?}^{?} ? ??? ???_{??}$$

Step 4: Selection process is performed in which some chromosomes are chosen from the population to be parents. The selection operator is implemented in a number of ways. In our thesis, we use Roulette-wheel selection method. The selection method, roulette wheel selection is used to form a set of individuals.

Step 5: Crossover: we use the single point crossover. In single point crossover two individuals are selected randomly. Then a single crossover point is set between 0 and the length of the individuals. The part of the individuals is exchanged behind the crossover point, and the two new individuals for the new population are generated. crossover size is 0.5

Step 6: Mutation operator is performed in order to prevent the algorithm to be trapped in local minimum. Probability of mutation determines the number of mutations occurred in the population. we kept the mutation rate at 0.1.

For example, if the **Population size** is `20`, the **Elite count** is `2`, and the **Crossover fraction** is `0.5`, the numbers of each type of children in the next generation are as follows:

- There are two elite children.
- There are 18 individuals other than elite children, so the algorithm rounds 0.5*18 = 9 to get the number of crossover children.
- The remaining 9 individuals, other than elite children, are mutation children.

Step 7: The stopping condition describes when the algorithm must be finished. We consider maximum generation as the termination criteria.

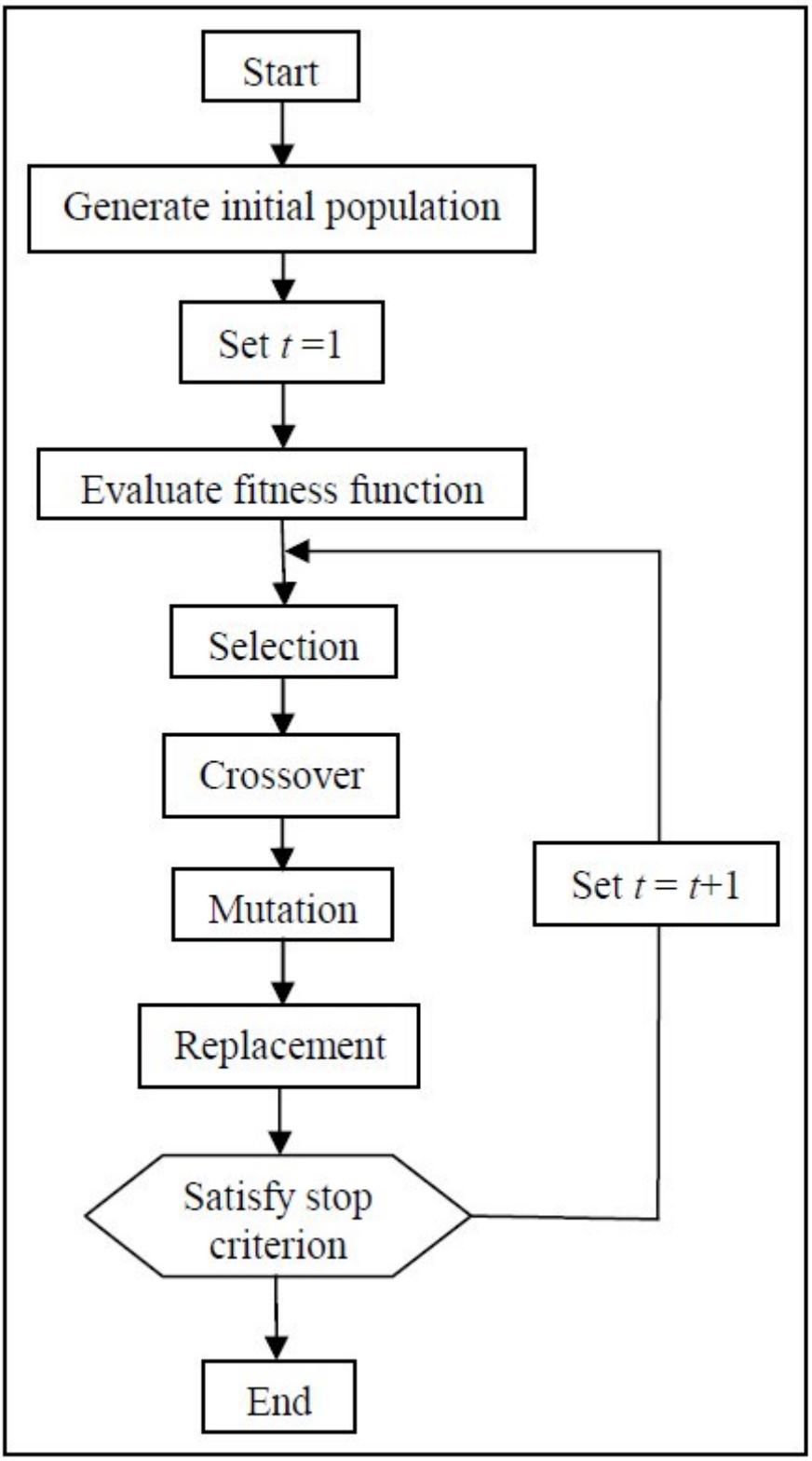
Start
Generate initial population
Set t =1
Evaluate fitness function
Selection
Crossover
Mutation
Replacement
Satisfy stop criterion
Set t = t+1
End

Fig 10. Proposed GA Flowchart

Genetic Parameters	Magnitude
Number of Generations	150
Population size	100
Crossover size	0.5
Type of crossover	Single point
Mutation rate	0.1
Selection type	Roulette Wheel

Table 3 :GA Parameters and Values

GA in Matlab : We can optimize the factors using Genetic Algorithm ,to use the genetic algorithm at the command line, call the genetic algorithm function ga with the syntax

[x fval] = ga(@fitnessfun, nvars, options)

where

- @fitnessfun is a handle to the fitness function.
- nvars is the number of independent variables for the fitness function.
- options is a structure containing options for the genetic algorithm. If you do not pass in this argument, ga uses its default options.

The results are given by

- x — Point at which the final value is attained
- fval — Final value of the fitness function

The selection of this technique is motivated by the argument that variables can be grouped based on the fitness value . The agile development data contains data from multiple projects. This data contains values of recorded attributes for each of the software project. The recorded attributes are:

Table 4. Attributes for cost estimation in data-set

Language Type	CMMI
Data Base System	Architecture
Organization Type	ISO
Hardware	Type of Server
Following process model	Package Customization
Training	Project complexity
Technical ability	Function points
Planning	Reliability
Debugging capability	Pages of documents
Client/Server	Risk taking
Communication skills	Managerial skills
Process maturity	Ease of use
Proximity of team	Documentation resources
Tool availability	Early delivery
Feedback	Documentation period
Tool familiarity	Other Expenditure
Software size	Application Type
IDE	Programming Language
Productivity	Operating System
Development Platform	Team Size

4.4 Training the ANN

Artificial Neural Network as discussed in chapter 3 is used for modeling complex relationships between inputs and outputs or to find patterns in data. Here to model the relationships between inputs and outputs a raw data which consist of the various factors of 2500 old projects is passed to ANN, which will train the neural network to establish a relationship between the given set of inputs and output. The data set is collected from different sources and Major factors are extracted via Genetic Algorithm. The data set is shown in Appendix 1. To create and train the ANN different tools (nftool, nprtool) from Matlab are used which are discussed in chapter 3.

ANN Model used for this research:

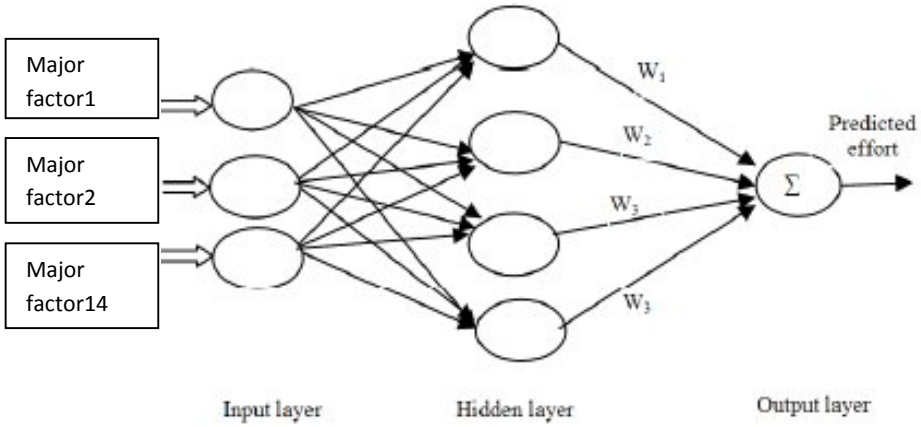

Fig 11: ANN Model

The neural network process starts by developing the structure of the network and establishing the technique used to train the network using an existingdata set. Neural network architectures are divided into two groups:
1. Feed forward networks where no loops in the network path occur.
2. Feedback networks that have recursive loops.

The most common architecture of neural networks which is used in software cost estimation is the Back-Propagation trained Feed Forward networks . The training algorithm of back propagation involves four stages:

1. Initialization of weights
2. Feed forward
3 . Back Propagation of errors
4. Updation of the weights and biases

The model proposed uses the identity function at the input layer which is defined by The hidden and the output layer uses unipolar sigmoid function defined by

$$f(x) = \frac{1}{1+e^{-x}}.$$

We will be using CAEA algorithm for estimating cost using ANN and Major Factors extracted via Genetic Algorithm.

As discussed above CAEA:

Cost of Project (COP) = NSP * Cost per story-point

NSP=SP+0.1*UV

UV=$\sum_{??\ ?}^{?}$? ?

Where n is no of factors ,here n=14 the factors which were extracted using Genetic algorithm will be used for cost estimation. X is the value /grade assigned to the particular factor.

We can take Cost per story point as constant A and SP as constant B

Now COP= (B+0.1 *($\sum_{??}^{?}$? ?))* A

The above equation in ANN becomes as:

COP= (B+0.1*(b2 + ?₂*?₂ + ?₂ * ?₂ +…….+?₂₂?₂₂)* (?₂ A +b1)

Here b1 and b2 are the biases and the coefficients Yi and Zi are the additional terms used in the model which act as the weights from the input layer to the hidden layer

The weights associated to the input nodes are denoted by Zi for $1 \leq i \leq 14$ for each input and b1 .On the other hand weights associated to the hidden layer for cost per story point aye yi and b2. These weights are initialized as Z_i=1and y_i=0. The weights from the hidden layer to the output layer are denoted by p and q and initialized as p=q=1.

Training Algorithm:

The feed forward back propagation procedure is used to train the network by iteratively processing a set of training samples and comparing the network's prediction with the actual value. For each training sample, the weights are modified so as to minimize the error between the networks predicted value and the actual value. The following algorithm is used for training the proposed network and for calculating the new set of weights:

Step 1: Initialize the weights and learning rate α

$(0 < \alpha \leq 1)$.

Step 2: Perform steps 3-10 when stopping condition is false.

Step 3: Perform steps 4-9 for each training pair.

Step 4: Each input unit receives input signal and sends it to the hidden unit.

Step 5: Each hidden unit C1and C_2 sums its weighted input signals to calculate net input given by:

C1=b2+∑? ? * Xi for i=1 to 14
C2=b1+ Yi *A for i=1 to 14

Apply sigmoidal activation function over C_1 and C_2 and send the output signal from the hidden unit

to the input of output layer units.

Step 6: The output unit C_{PM}, calculates the net input given by:
$C_{PM} = C_1*p + C_2*q$

Apply sigmoidal activation function over C_{PM} to compute the output signal C_{est}.

Step 7: Calculate the error correction term as:

$\delta = E_{act} - E_{est}$,

where C_{act} is the actual cost from the dataset and C_{est} is the estimated cost from step 6.

Step 8: Update the weights between hidden and the output layer as:

$p(new) = p(old) + \alpha * \delta * C_1$
$q(new) = q(old) + \alpha * \delta * C_2$

Step 9: Update the weights and bias between input and hidden layers as:

$x_i(new) = x_i(old) + \alpha * \delta_1 * z_i$ for i=1 to 14
$y_i(new) = y_i(old) + \alpha * \delta_{2F} * z_i$ for i=1 to 14
$b_1(new) = b_1(old) + \alpha * \delta_1$
$b_2(new) = b_2(old) + \alpha * \delta_2$

The error is calculated as
$\Delta_1 = \delta * p$; $\delta_2 = \delta * q$;

Step 10: Check for the stopping condition. The stopping condition may be certain number of epochs reached or if the error is smaller than a specific tolerance.

Using this approach, we iterate forward and backward until the terminating condition is satisfied. The variable α used in the above formula is the learning rate,a constant, typically having a value between 0 and 1.
The learning rate can be increased or decreased by theexpert judgment indicating their opinion of the input effect. In other words the error should have more effect on the expert's indication that a certain input had more contribution to the error propagation or vice versa. For each project, the expert estimator can identify the importance of the input value to the error in the estimation. If none selected by the expert, the changes in the weights are as specified by the learning algorithm.

4.5 Cost Prediction:

In this final step we will find out the cost required by passing different inputs to the well trained ANN. The inputs will be the 14 factors which were extracted using Genetic Algorithm based on the fitness function which is going to be work on the project. All these inputs are identified in the above steps. The cost will be predicated on the basis of the values of inputs are been passed to ANN. While training the ANN, it establishes a relation between the inputs (14 extracted Major factors via Genetic Algorithm) and the output (Cost). This relation will help one to predict the cost by just providing the values of inputs. It has been found that the predicated cost poses less inaccuracy which indeed will help us to develop software's with high quality.

Chapter 5: Implementation

5.1 Introduction

In this chapter we will implement the proposed estimation model on the sample data set and demonstrate the steps of the cost estimation process.

5.2 Data-Set Description

The proposed two step model for cost estimation in agile software projects was used on the data-set which is used in research in the domain of agile development methodologies and agile studies at the 'Centre For Systems And Software Engineering' at the 'School of Engg, University of South California'. The dataset contains knowledge about software projects that are *'standardised, verified, recent and representative of current technologies'*.

It is a repository of the software development project data from about 250 development projects using agile technologies. In other words it is a record of values of 40 attributes for 250 projects. The data is collected from reputed software development firms in various countries- Switzerland, USA Australia, Netherlands, Spain, China, Finland, France, Germany, Italy, and Japan.

The data is collected primarily from middle-level to big level teams with a team size ranging from 25 to 70 persons. The projects considered are also of varying size, complexity and costs. The suitability and appropriateness of the data to conduct factor analysis should to be checked. This validation of data used for analysis has been tested by the Kaiser-Meyer-Olkin (KMO) method.

The KMO method is used to measure sampling adequacy and it ranges from 0 to 1. Values between 0.5 and 0.7 are mediocre, values between 0.7 and 0.8 are good, values between 0.8 and 0.9 are great and lastly values above 0.9 are superb. A KMO with 0.6 is suggested as the minimum value for a good factor analysis. If the value yields more than 0.7, then the correlation on the whole are sufficient to make factor analysis suitable [35].

The KMO value founded was 0.816, which according to research, corresponds to a data-set which is of good quality and suitable for analysis.

The complete data-set is provided in Appendix-1
This data can be used for estimation, benchmarking, project management, infrastructure planning, bid planning, outsources management, standards compliance and budget support. The data set is essentially a 250*40 matrix i.e. there is a record of values of 40 attributes for 250 projects. 70% of the data i.e. 175 records are considered as the training data set and the remaining 75 records are used for Testing Data-set.

5.3 Tools used

This section states about the tool which will be going to be used for implementing Genetic Algorithm and Artificial Neural Network:

MATLAB (matrix laboratory) is a numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran [24].

1)Optimization Toolbox

Genetic Algorithm Solver :This tool corresponds to the ga function.

Fitness function (required) is the objective function you want to minimize. You can specify the function as a function handle of the form @objfun, where objfun.m is a function file that returns a scalar.

Number of variables (required) is the number of independent variables for the fitness function.

Options : Specify options for the Genetic Algorithm solver.

Population

Population options specifies options for the population of the genetic algorithm.

Population type specifies the type of the input to the fitness function. Types and their restrictions:

Double vector — Required when there are integer constraints.

Bit string — For Creation function and Mutation function, use Uniform or Custom. For Crossover function, use Scattered, Single point, Two point, or Custom. You cannot use a Hybrid function or Nonlinear constraint function.

Custom — For Crossover function and Mutation function, use Custom. For Creation function, either use Custom, or provide an Initial population. You cannot use a Hybrid function or Nonlinear constraint function.

Population size specifies how many individuals there are in each generation. If you set Population size to be a vector of length greater than 1, the algorithm creates multiple subpopulations. Each entry of the vector specifies the size of a subpopulation.

Creation function specifies the function that creates the initial population (ignored with integer constraints):

Constraint dependent chooses:

Uniform if there are no linear constraints, or if there are integer constraints

Feasible population if there are linear constraints and no integer constraints

Uniform creates a random initial population with a uniform distribution within any bounds. Find more details under Initial range.

Feasible population creates a random initial population that satisfies the bounds and linear constraints. Feasible population ignores any setting of Initial range.

Custom enables you to provide your own creation function, which must generate data of the type that you specify in Population type. Enter a function handle of the form @CreationFcn, where CreationFcn.m is a function file with syntax described here.

Initial population enables you to specify an initial population for the genetic algorithm. If you do not specify an initial population, the algorithm creates one using the Creation function. You can specify fewer than Population size individuals; if you do, the Creation function creates the rest.

Initial scores enables you to specify scores for the initial population. If you do not specify Initial scores, the algorithm computes the scores using the fitness function. Ignored when there are integer constraints.

Initial range specifies lower and upper bounds for the entries of the vectors in the initial population for the Uniform Creation function. You can specify Initial range as a matrix with 2 rows and Initial length columns. The first row contains lower bounds for the entries of the vectors in the initial population, while the second row contains upper bounds. If you specify Initial range as a 2-by-1 matrix, the two scalars expand to constant vectors of length Initial length. Each integer-constrainted component has an implicit lower bound of -9,999 and an implicit upper bound of 10,001. Other components have implicit lower and upper bounds of -10 and 10 respectively. Any explicit Initial range you give override these implicit bounds. Also, ga internally modifies the Initial range to satisfy the bounds in Constraints: Bounds.

Use the Optimization App

To open the Optimization app, enter

```
optimtool('ga')
```

at the command line, or enter optimtool and then choose ga from the Solver menu.

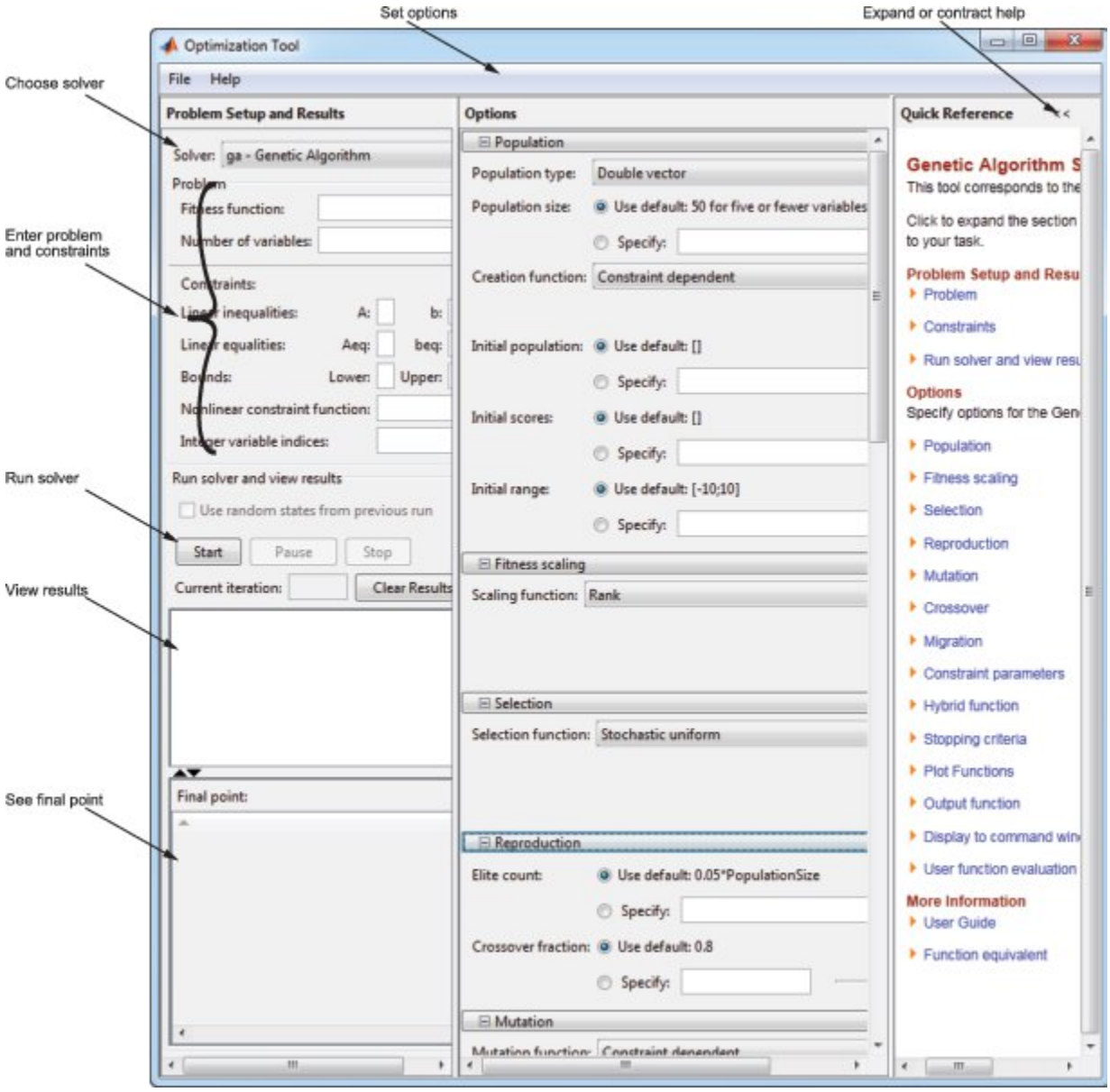

Fig 12 :Optimization Toolbox

You can also start the tool from the MATLAB **Apps** tab.

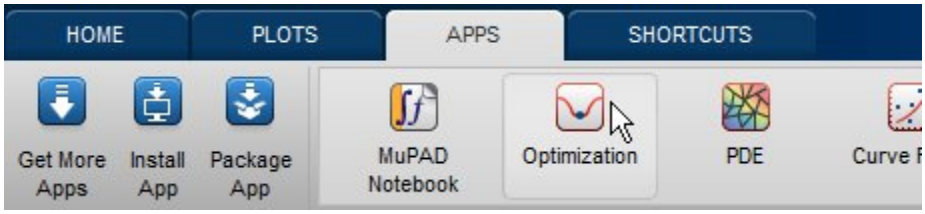

Fig 13 :MATLAB APPS

To use the Optimization app, you must first enter the following information:

- **Fitness function** — The objective function you want to minimize. Enter the fitness function in the form @fitnessfun, where fitnessfun.m is a file that computes the fitness function. Compute Objective Functions explains how write this file. The @ sign creates a function handle to fitnessfun.
- **Number of variables** — The length of the input vector to the fitness function. For the function my_fun described in Compute Objective Functions, you would enter 2.

You can enter constraints or a nonlinear constraint function for the problem in the Constraints pane. If the problem is unconstrained, leave these fields blank.

To run the genetic algorithm, click the Start button. The tool displays the results of the optimization in the Run solver and view results pane.

You can change the options for the genetic algorithm in the Options pane. To view the options in one of the categories listed in the pane, click the + sign next to it.

2. Neural Network Toolbox

It contains a set of MatLab functions which implement architectures and learning algorithms for several types of neural networks. The user can specify the architecture, the activation functions, the connectivity, the learning algorithm. To get a first idea on the facilities offered by NN toolbox just use help nnet (it lists all functions related with neural networks).

There are several demos which can be activated by using nnd. The toolbox contains both general functions (as sim to simulate a network or adapt and train to train a network) and

particular functions for given architectures (as newlin, newff, newrbf etc.). Any neural network implemented in NN Toolbox is a structured object having as components arrays, functions for simulation and training and control parameters.

There are some applications with GUIs:

- Pattern recognition: nprtool
- Pattern fitting: nftool
- Clustering: nctool

NFTOOL:

nftool (neural network fitting tool) provides a graphical user interface for designing and training a feedforward neural network for solving approximation (fitting) problems. The networks created by nftool are characterized by:

- One hidden layer (the number of hidden units can be changed by the user; the default value is 20)
- The hidden units have a sigmoidal activation function (tansig or logsig) while the output units have a linear activation function
- The training algorithm is Backpropagation based on a Levenberg-Marquardt minimization method.

The learning process is controlled by a cross-validation technique based on a random division of the initial set of data in 3 subsets: for training (weights adjustment), for learning process control (validation) and for evaluation of the quality of approximation (testing). The quality of the approximation can be evaluated by:

- Mean Squared Error (MSE): it expresses the difference between to correct outputs and those provided by the network the approximation is better if MSE is smaller (closer to 0)

- Pearson's Correlation Coefficient (R): it measures the correlation between the correct outputs and those provided by the network; as R is closer to 1 as the approximation is better.

5.4 Steps of Cost Estimation Process :

The Major Factors are identified using the MATLAB software through Optimization toolbox using Genetic Algorithm.
From the processed training data set we obtain the various Factors which affects the cost estimation in agile projects
The factors extracted contains the Fitness value for each factor which explains the co-relation between the corresponding factor and the cost of development.
The following table shows these fitnesss value for each of the 40 factors.

Major Factors	Fval
Software size	0.72
Architecture	0.68
Risk Resolution	0.52
Process Maturity	0.77
Team Size	0.89
App Type	0.67
Required SW reliability	0.79
Product/project Complexity	0.59
Development Platform	0.7
Prog Language and Toolset experience	0.66
OS	0.84
CMMI	0.71
Function Points	0.68
ISO	0.82
Managerial skills	0.45
Platform experience/Tech Ability	0.5
Hardware	0.41
Pages of Documents	0.34
Developed for Reusability	0.29
Development Flexibility	0.49
DB Size/SW size	0.43
IDE	0.21
Productivity	0.35
Org Type	0.44

Process Model	0.49
Training	0.34
Planning	0.41
Debugging Capability	0.49
Client/Server	0.34
Tool avlb	0.29
Feedback	0.21
Tool Familiarity	0.41
Team Prox	0.19
Typer of server	0.16
Package Customization	0.014
Ease of Use	0.33
Documentation Resources	0.13
Early Delivery	0.37
Documentation Period	0.21
Other Expenditure	0.18

Table 5 :Agile factors and fitness values

From the above Table we extract 14 factors which are having high fitness value which can be used in extimating the cost for various agile projects .After doing crossover and Mutation for multiple iterations 14 factors are extracted so we can use them for extimating the cost in Agile projects instead of taking all the 40 factors which are of less or minimal use in cost estimatuion as there will be very less variation while considering such factors and will not influence the cost estimation.

Hence the extracted Factors using Genetic Algorithm in Matlab are:

Table 6 : Extracted Agile Factors

Function Points	Product Complexity
Architecture	Development Platform
Risk Resolution	Prog Language and Toolset experience
Process Maturity	OS
Team Size	CMMI
App Type	Software Size
Required SW reliability	ISO

Training the ANN

Step 1: use command nftool in the command line of Matlab

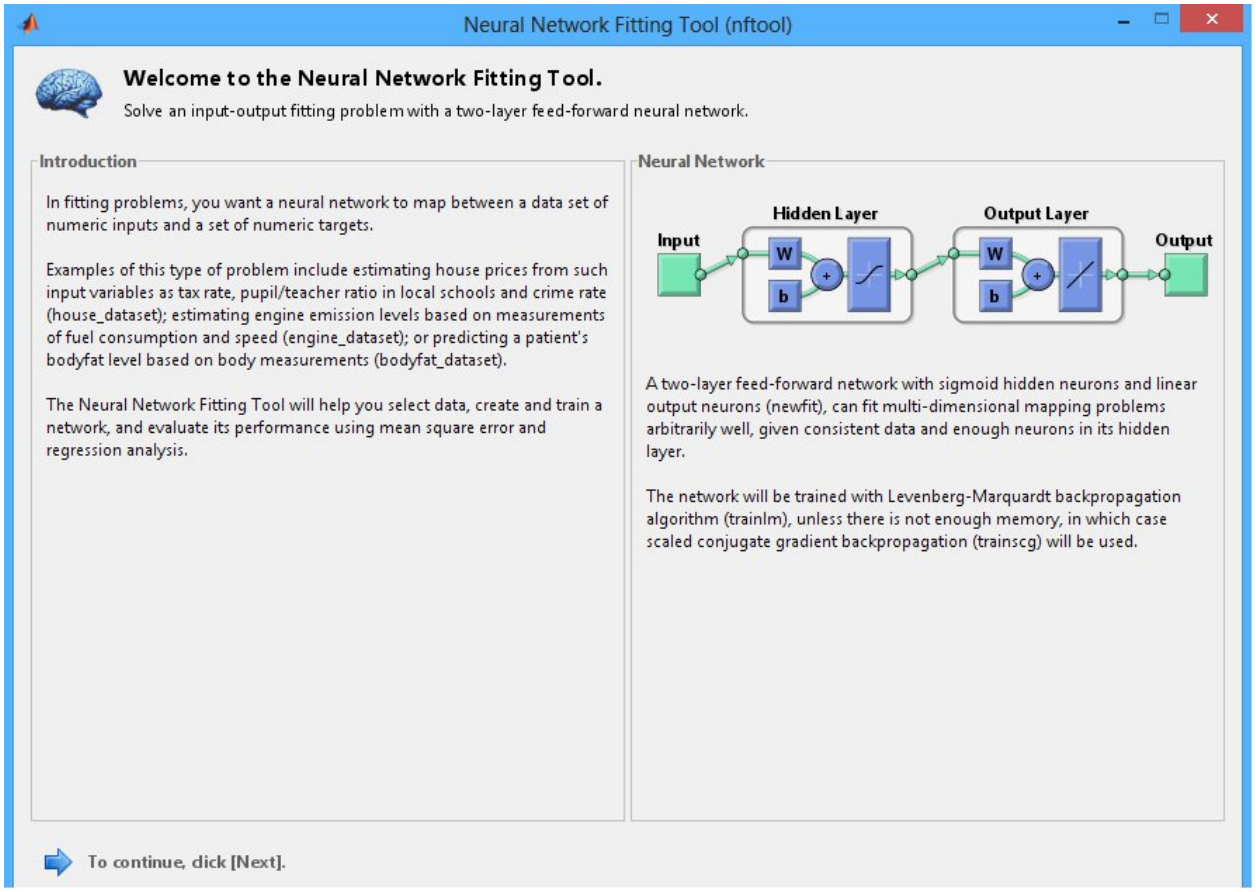

Fig 14:Starting nftool

Step 2: Selecting input and output parameters

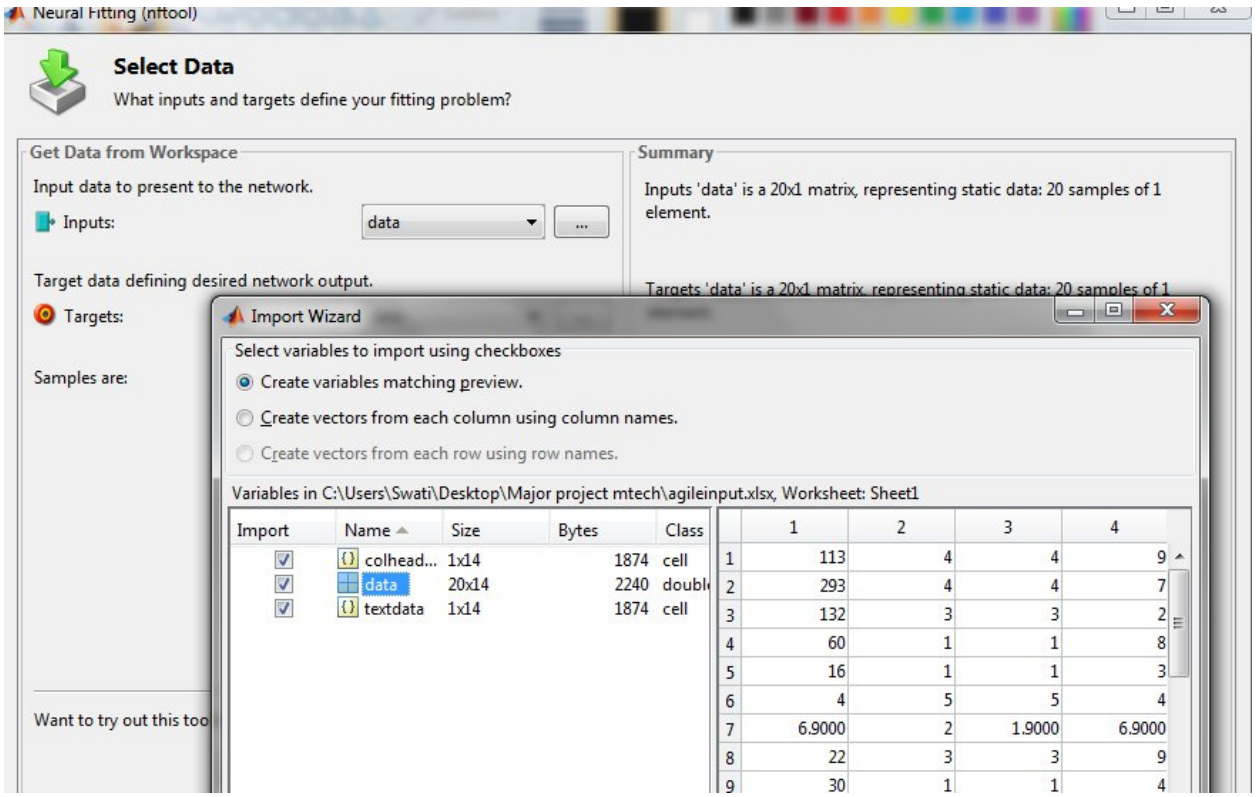

Fig 15: Input and Output of ANN

Step 3: Assigning percentage of data for validation and testing

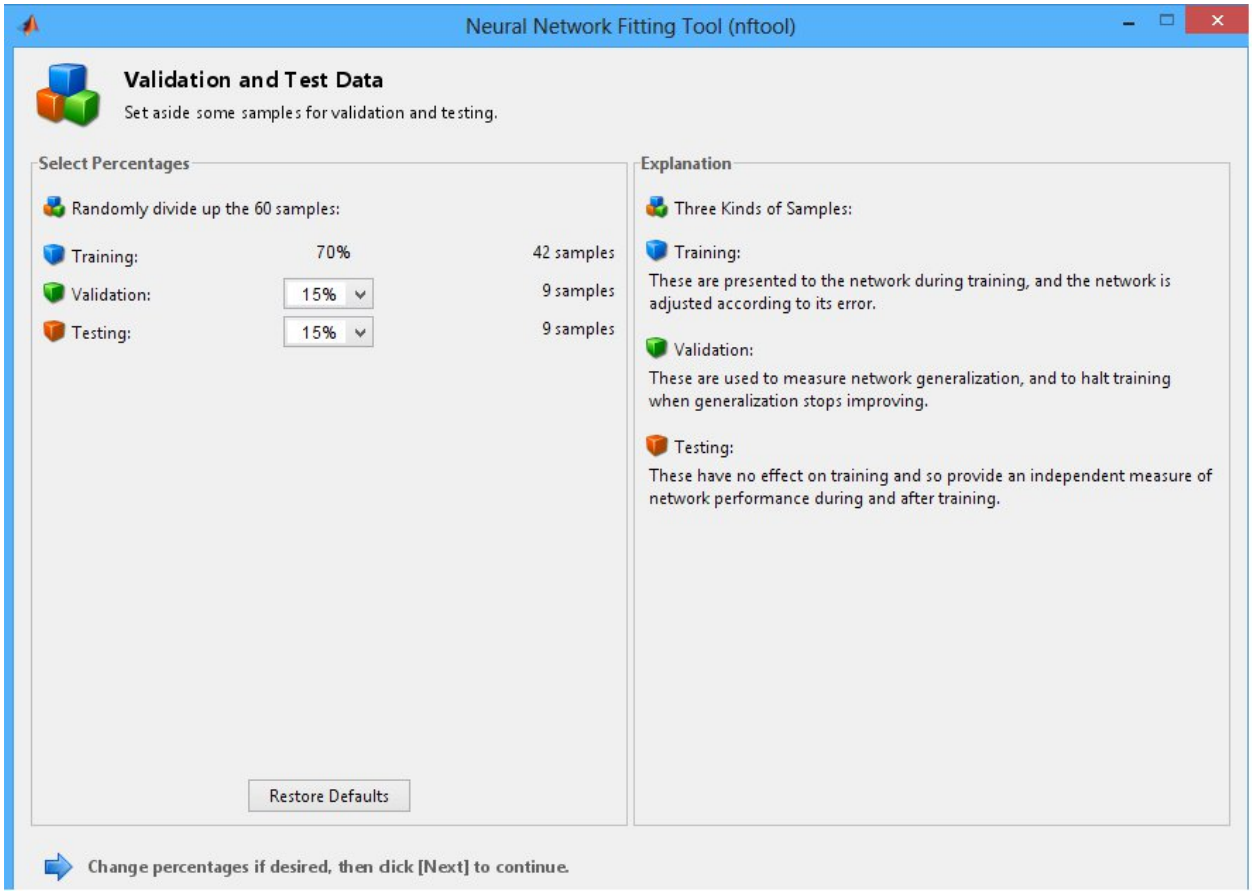

Fig 16 :Sample Classification

Step 4: Selecting number of neurons for the hidden layers

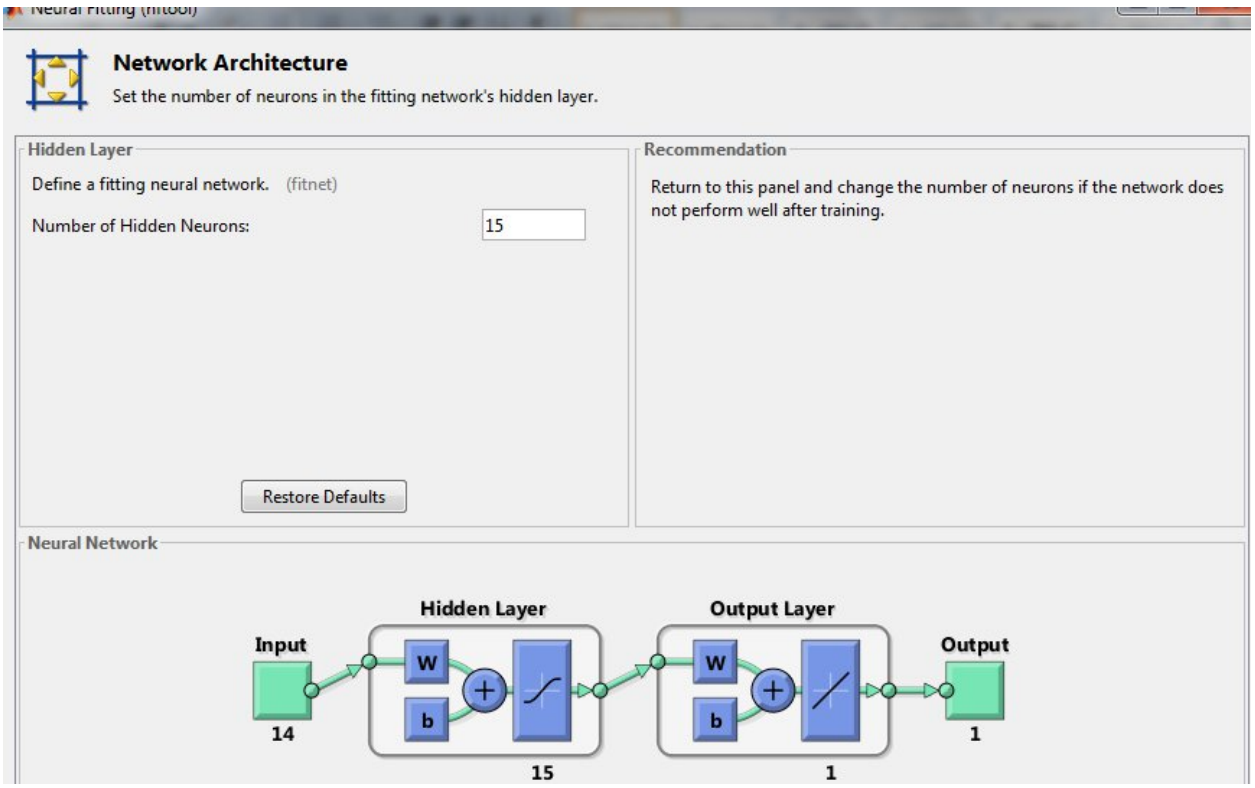

Fig 17: Selecting Number of Neurons

Step 5: ANN is ready for training. Just need to press train button

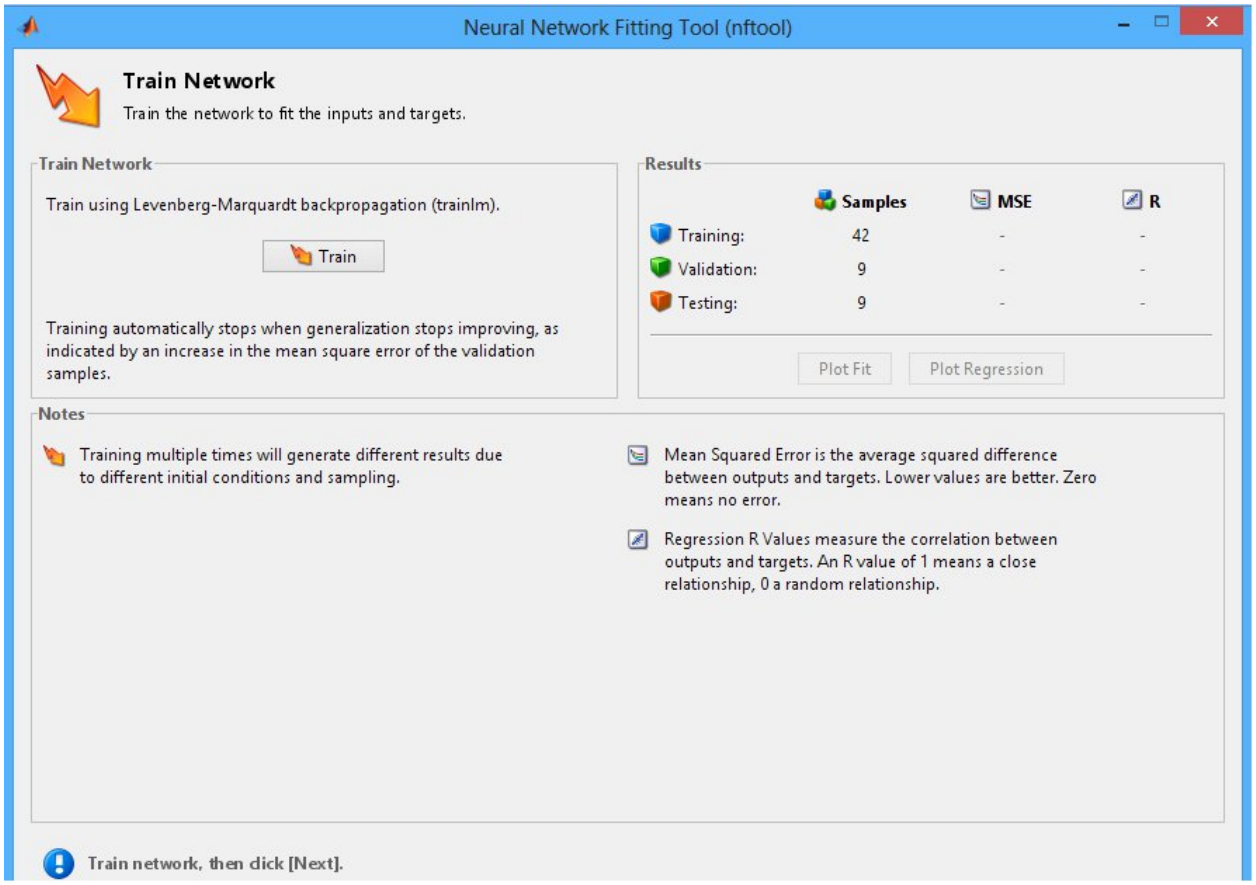

Figure 18: Training the Network

Step 6: Output of training the ANN

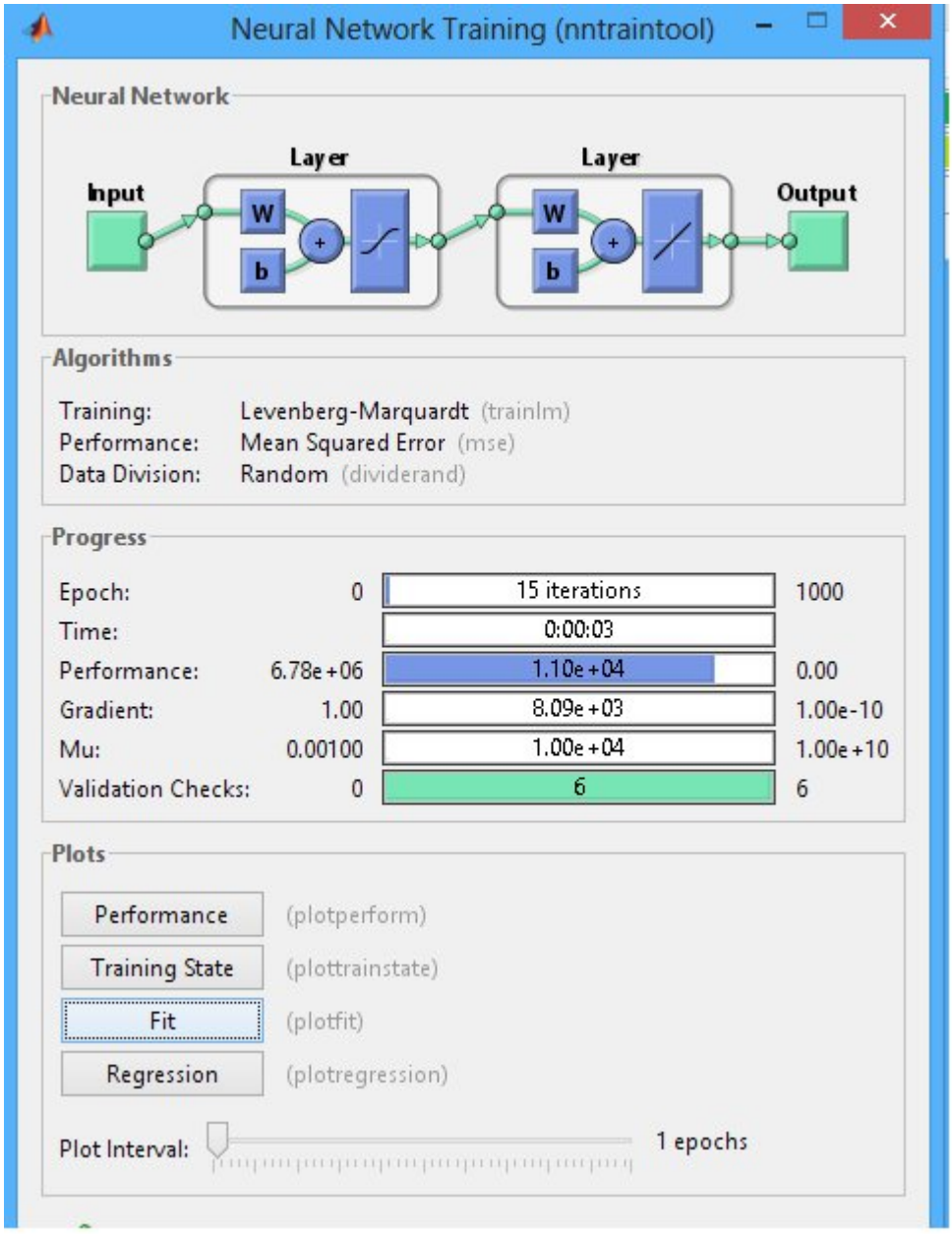

Figure 19: Results of ANN

Step 7: Saving the results and generating M file for the ANN.

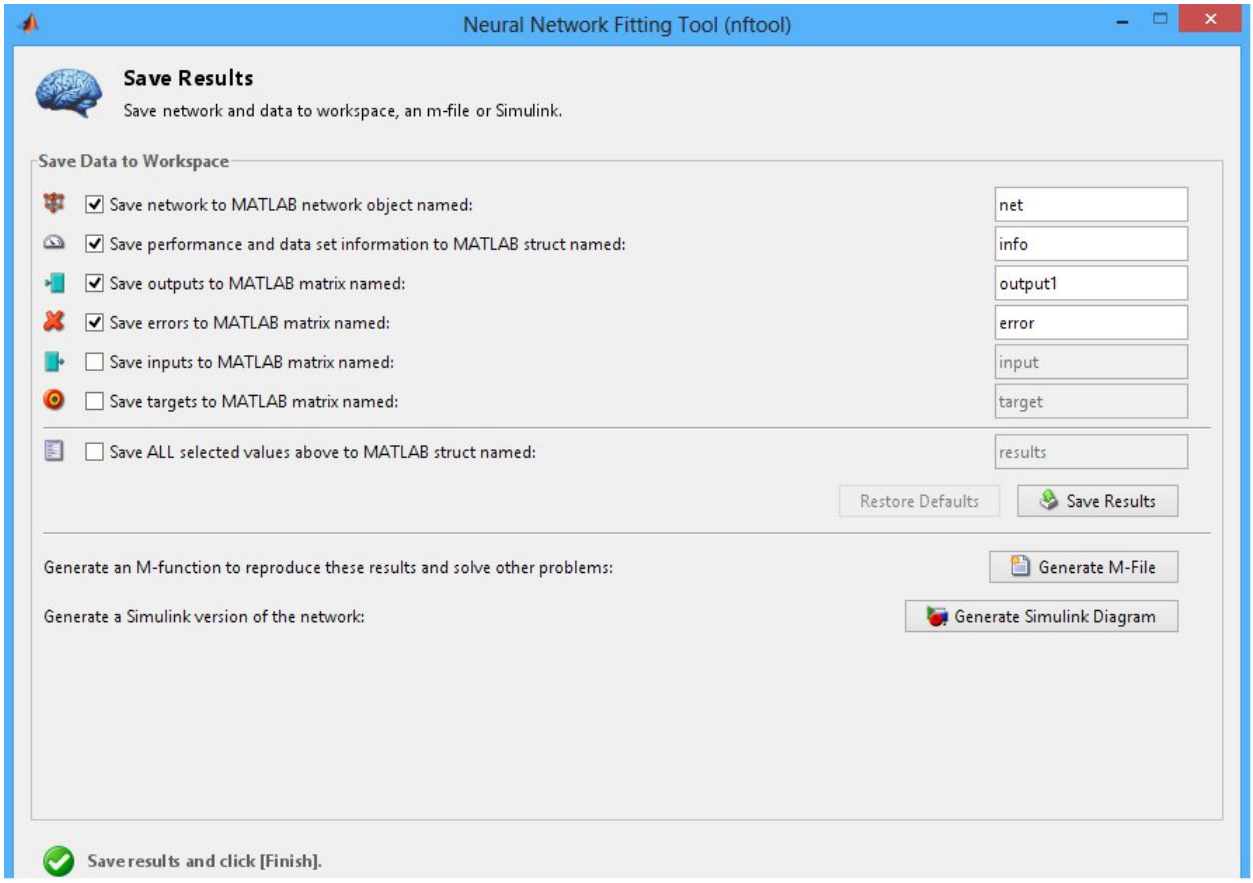

Figure 20: Saving results

Chapter 6: Results and Analysis

This section shows the results of above proposed methodology and the analysis of the result to check whether it can serve our purpose or not.

6.1 Over Estimation and Under Estimation

Problems with over-estimation

Managers and other project stakeholders sometimes fear that, if a project is overestimated, Parkinson's Law will kick in—the idea that work will expand to fill available time. For example, if you give a developer 5 days to deliver a task that could be completed in 4 days; the developer will find something to do with the extra day. As a result, some managers consciously squeeze the estimates to try to avoid Parkinson's Law [11].

Problems with under-estimation

Reduced effectiveness of project plans

Low estimates undermine effective planning by feeding bad assumptions into plans for specific activities. If the estimation errors caused the plans to be off by only 5% or 10%, those errors wouldn't cause any significant problems. But numerous studies [11] have found that software estimates are often inaccurate by 100% or more. When the planning assumptions are wrong by this magnitude, the average project's plans are based on assumptions that are so far off that the plans are virtually useless.

Poor technical foundation leads to worse-than-nominal results

A low estimate can cause you to spend too little time on understanding requirements. If you don't put enough focus on understanding them, you'll have to revisit them later in the project at greater cost than if you had done that well in the first place. This ultimately makes your project take longer than it would have taken with an accurate estimate [17].

Destructive late-project dynamics make the project worse than nominal

Once a project gets into "late" status, project teams engage in numerous activities that they don't need to engage in during an "on-time" project. For example, more status meetings, frequent re-estimation, defects arising from quick and dirty workarounds etc.

Over-estimation vs. Under-estimation

As Figure shows, the best project results come from the most accurate estimates. If the estimate is too low, planning inefficiencies will drive up the actual cost and schedule of the project. If the estimate is too high, Parkinson's Law kicks in.

Figure 21: Penalties for underestimation vs. Penalties for overestimation [14]

Work does expand to fill available time. But deliberately underestimating a project because of Parkinson's Law makes sense only if the penalty for overestimation is worse than the penalty for underestimation. In software, the penalty for overestimation is linear and bounded - work will expand to fill available time, but it will not expand any further. But the penalty for underestimation is nonlinear and unbounded - planning errors, shortchanging understanding of

requirements, and the creation of more defects cause more damage than overestimation does, and with little ability to predict the extent of the damage ahead of time [2][11][12].

6.2 Analysis using a Data set

The data set which has been used to train the ANN is shown in Appendix 1. The data set consist of various attributes like function point, Team Size, Architecture, Risk resolution etc for around 250 projects. Also from the attribute of data set one can easily find the project characteristics and thereby can have a good idea about the project.

6.2.1 Selection of inputs

The inputs to the ANN are selected on the basis of their quality to accurately identify the cost required. The input must be in direct correspondence with the cost. All 14 factors generated via Genetic Algorithm are taken as Input to Artificial Neural Network.

6.2.2 Evaluation Criteria

The evaluation criteria for the proposed methodology is regression values and MMRE (Mean Magnitude relative error), The estimated values were then compared against the Actual cost values. This was done using MMRE (Mean Magnitude of Relative Error) . Different error measurements have been used by various researchers. I have choose the mean relative Error (MRE) and regression as the major measurement tool:

MMRE: MRE is calculated as follows:

$$MRE = \frac{|? ??? ? ? ? ??? ? ???? \ ? ??? \ ? ???|}{? ??? ? ? \ ? ???}$$

MMRE = (100/N)*(MRE1 + MRE2 + ... + MREN)

.

Table 7 : Actual and Estimated Cost

Project No	Actual Development Cost	Estimated Development Cost
1	2040	1234
2	1600	1345.34
3	243	177.32
4	240	312

5	33	24.5
6	43	21.42
7	80	51.21
8	1075	765.42
9	423	234.45
10	321	189.45
11	218	170.62
12	201	121.12
13	79	43.45
14	60	34.32
15	61	45.21
16	40	51.23
17	9	14.23
18	11400	789.34
19	6600	4523.21
20	6400	8723.45
21	2455	921.21
22	724	543.21
23	539	260.37
24	453	276.54
25	523	175.87
26	387	181.23
27	88	56.21
28	98	110.21
29	7	4.5
30	5	2.34
31	1063	456.23
32	702	957.45
33	605	359.23
34	230	241.23
35	82	98.23
36	55	35.076
37	47	77.67
38	12	21.09
39	8	5.23
40	8	42.51

For the above results we derive that the

MMRE = 50.95

The below plot shows the trend of MRE of the estimate cost with respect to rising project cost. We can see that

- The error value is localised within the a small range around the mean value with the exception of a few projects.
- Increased cost of development has no effect on the estimation error value.
- This means that even for very complex and costly projects the error in estimation will be the same.

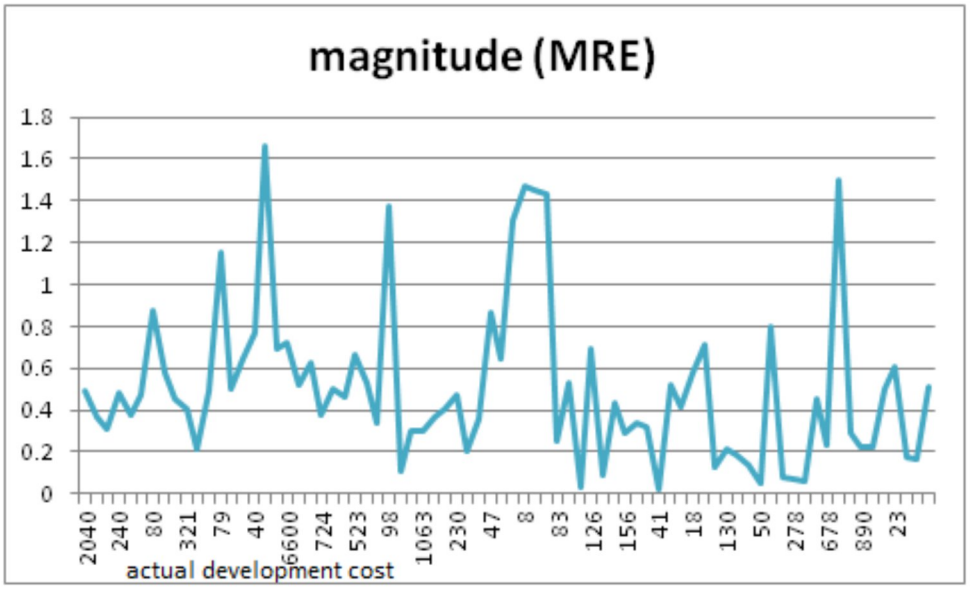

Fig 22. MRE vs actual development cost for proposed Model.

6.3 Results

The result of the well trained ANN will be the graph which shows different regression values during training, validation and Testing. It has been observed that changing the number of hidden layers in ANN or changing the number of neurons in the hidden layer of ANN may affect the regression values. So for good approximate results the training of ANN has to be done several times by changing the hidden layers. So the testing has been done by changing the hidden layer many times from 1 to 35 but the most significant values of regression is obtained on 15.

Below shown the regression plot of the data set shown in appendix-1 which is being used to train the ANN with different number of neurons in the hidden layers.

Number of neurons in hidden layer is 15 and the regression value while testing the trained ANN is around 0.99 that means around 0.01 is error value.

Training:

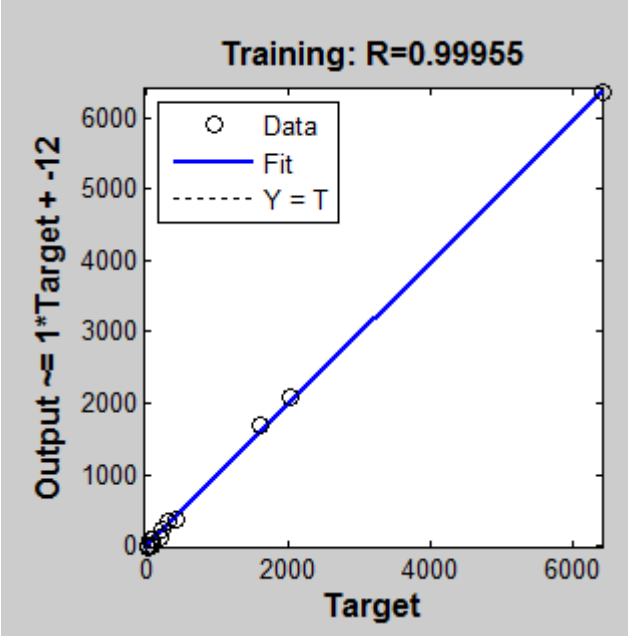

Figure 23: Regression Curve of Training

Validitaion:

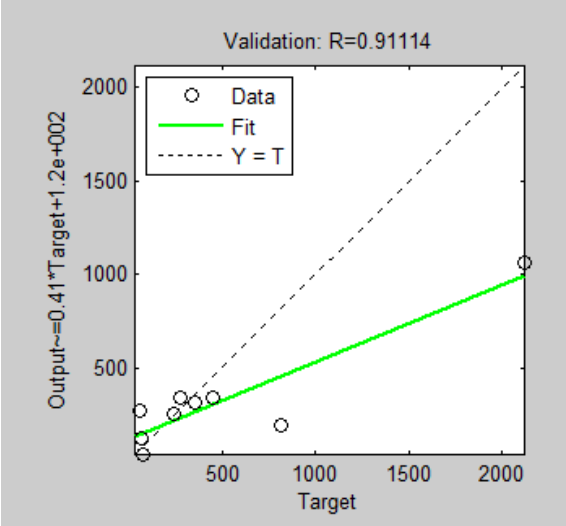

Figure 24: Regression Curve of Validation

Testing:

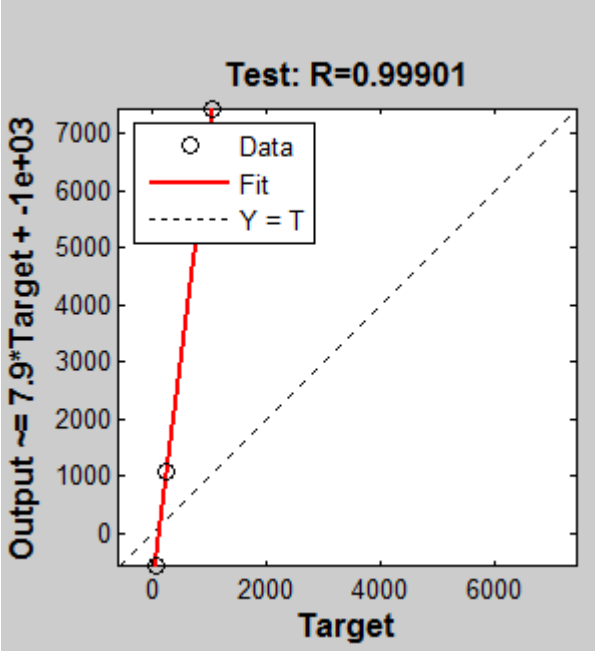

Figure 25: Regression Curve of Testing

Overall:

Training	Validation	Testing
0.99955	0.91119	0.99901

Table 8: Collective values for Regression Curve for 15 hidden neurons

Number of neurons in hidden layer is 10 the regression value here is around 0.96 which is less than the above regression value:

Training:

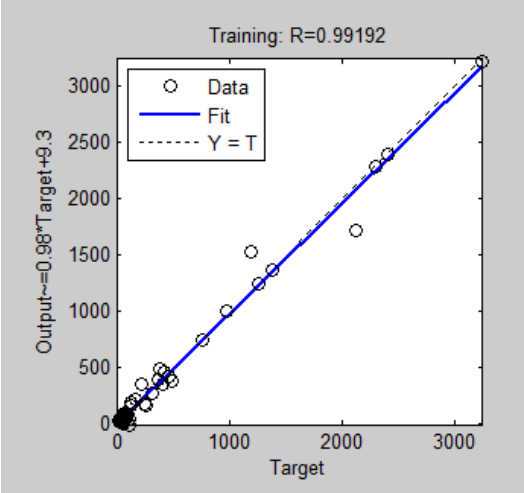

Figure 26: Regression Curve of Training

Validation:

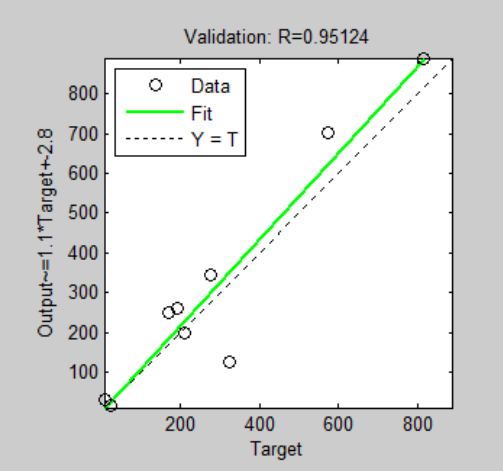

Figure 27: Regression Curve of Validation

Testing:

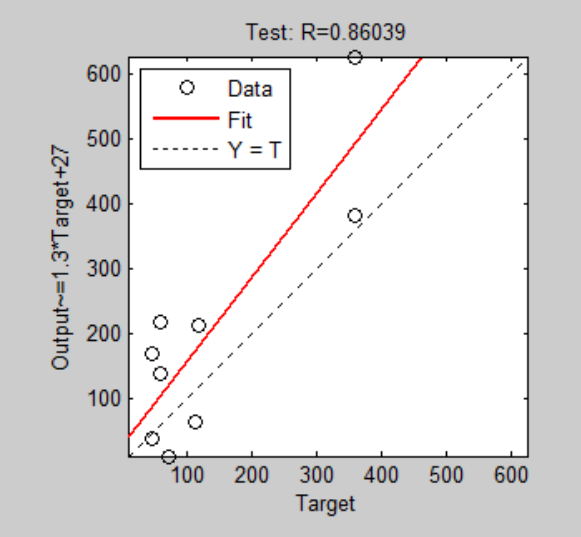

Figure 28: Regression Curve of Validation

Overall:

Training	Validation	Testing
0.99192	0.95124	0.86039

Table 9: Collective Values of Regression Curve for 10 Hidden neurons

So comparing table 6 and table 7, one can easily indentify that the better values of regression are in table 6 which is close to 0.99 that means the Relative error comes out to be 0.01 and this relative error is the minimum one till date with respect to all the COST estimation methodologies using ANN. So here I can say that it has been shown that on the basis of 250 data sets that the ANN can also be very efficient for cost estimation of various projects which are developing through agile techniques as well.

Results can be stated as follows:

- Decreased MRE and increased the estimation accuracy

  Magnitude of relative error is a widely used measure for estimation accuracy.

  It is defined as:

$$RE_i = (estimate_i - actual_i) / (actual_i)$$

$$MRE_i = abs(RE_i)$$

$$MMRE = (100/N)*(MRE_1 + MRE_2 + ... + MRE_N)$$

  Proposed methodology was able to decrease the estimation error, which naturally increased the accuracy of estimation, in comparison with other available methods like planning poker. The result has shown that MRE values are reduced up to 0.01%.

- Reduced losses due to estimation inaccuracy

  With the application of proposed methodology and analysis, one can be able to show that in the long run, we can reduce the losses incurred due to estimation inaccuracy, i.e. under-estimation and over-estimation.

- Naturally fitting method for agile development

We have also identified the next steps to validate and adopt proposed methodology in order to improve the estimation process. The idea is to train ANN with the data sets of agile projects of previously developed projects estimate data in the first part of the project, and get more precise results in the latter half.

6.3.1 Comparison with previously existing cost estimation Model

In this section we will compare the proposed model with the CAEA model which is suggested in Ch-3.

We used the CAEA method for cost estimation using our data set.

The results of estimation process are as shown in below table;

Project No	Actual Development Cost	Estimated Development Cost
1	2040	3214.34
2	1600	2123.32
3	243	345.54
4	240	455.23
5	33	45.67
6	43	57.89
7	80	323.23
8	1075	324.34
9	423	231.43
10	321	603.23
11	218	212.23
12	201	189.56
13	79	72.12
14	60	78.9
15	61	23.45
16	40	56.67
17	9	45.56
18	11400	2134.45
19	6600	6123.23
20	6400	7213.12
21	2455	3452.21
22	724	879.23
23	539	1234.45
24	453	567.89
25	523	981.112

26	387	456.34
27	88	121.23
28	98	145.23
29	7	5.32
30	5	10.32
31	1063	923.23
32	702	566.45
33	605	450.23
34	230	74.45
35	82	87.23
36	55	77.21
37	47	41.23
38	12	10.11
39	8	5.32
40	8	5.34

Table 10. Actual and Estimated cost using CAEA

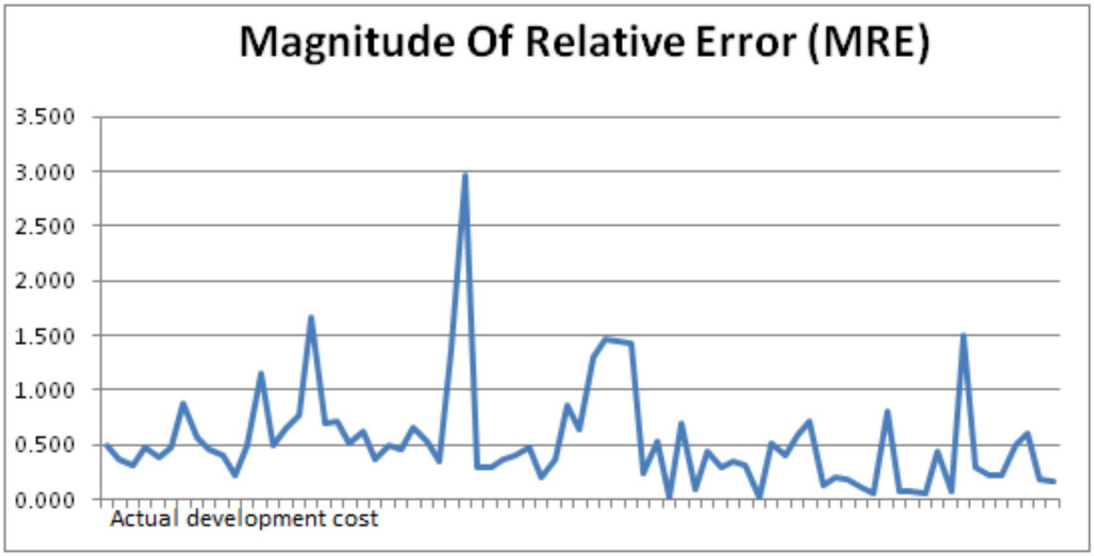

Fig 29 : MRE vs actual development cost for CAEA

From the above Results we can see that –

MMRE value for CAEA is = 53.72

This is higher than the results of our cost estimation approach. Hence our approach using GA and ANN will be more suited to real-world Agile software projects.

Chapter 7: Conclusion

This thesis proposed Genetic Algorithm and feed-forward Artificial Neural Network (ANN) model to predict software cost in agile environment based on the different project characteristics metrics. The inputs of the proposed model are various factors which can affect the cost required to develop software's .These factors were extracted using Genetic Algorithm in Matlab by doing dimension reductionality to get the major factors which will impact the cost estimation in Agile projects. Further, to evaluate the ANN model, a multiple linear regression model was developed that has the same inputs as the ANN model. The regression based ANN models was trained using 250 projects and evaluated using extracted factors. The ANN model was then evaluated against the regression and produced great results which are shown above.

Results show that the proposed ANN model for agile estimation outperforms the existing methods for agile cost estimation based on the MMRE and Regression values criteria and can be used an as alternative method to predict software cost in agile environments.

Advantages of New Technique:

- An expert independent method: The estimation model which uses ANN is independent of the experts. That means ANN has removed the role of the experts from Estimation process which can be very helpful to produce accurate results.
- Lower MMRE values in estimation: This technique has shown a decent lower MRE value so one can expect to achieve a good accuracy while estimating cost..
- Evolve with time: As it uses ANN which can evolve itself with time, as the new projects are being developed, the data of the projects can be useful to train the neural network and thereafter can be useful to predict the effort and cost of new projects.

Future work will focus on trying other models such as Radial Basis Function Neural Network and General Regression Neural Network.

References

[1] Agile Modeling Home Page. Effective Practices for Modeling and Documentation. [Online] Retrieved 17th March 2009. Available at: www.agilemodeling.com .

[2] J. Erickson, K. Lyytinen and K. Siau, Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research. In Journal of Database Management, 16(4), 2005, 88-100.

[3] M. Singh, U-SCRUM: An Agile Methodology for Promoting Usability. In Ag. AGILE '08. Conference, Toronto, 2008, 555-560.

[4] M. Cristal, D. Wildt and R. Prikladnicki, Usage of SCRUM Practices within a Global Company. Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on, 2008, 222-226.

[5] Abrahamsson, P., Fronza, I., Moser, R., Vlasenko, J., Pedrycz, W., Predicting Development Effort from User Stories, International Symposium on Empirical Software Engineering and Measurement, 2011

[6] Haugen, N., an Empirical Study of Using Planning Poker for User Story Estimation, Proceedings of AGILE 2006 Conference, 2006

[7] Zivadinovic, J., Medic, Z., Maksimovic, D., Damnjanovic, A., Vujcic, S., Methods of Effort Estimation in Software Engineering, International Symposium Engineering Management and Competitiveness 2011 (EMC2011)

[8] Cohn, M., Agile Estimating and Planning, 2002

[9] Molokken, K., Jorgensen, M., A Review of Software Surveys on Software Effort Estimation, International Symposium on Empirical Software Engineering, 2003.

[10] Goldratt, E.M., Critical Chain, Great Barrington, MA: The North River Press, 1997

[11] Caruana, R., Niculescu-Mizil, A., An Empirical Comparison of Supervised Learning Algorithms, Proceedings of the 23rd International Conference on Machine Learning, 2006

12] Cheng, B., Xuejun, Y., The Selection of Agile Development's Effort Estimation Factors based on Principal Component Analysis, International Conference on Information and Computer Applications, 2012

[13] Baskeles, B., Turhan, B., Bener, A., Software Effort Estimation Using Machine Learning Methods, 22nd International Symposium on Computer and Information Sciences, 2007

[14] C. Lopez-Martin, C. Isaza and A. Chavoya, "Software development effort prediction of industrial projects applying a general regression neural network," Empirical Software Engineering, vol. 17, pp. 1-19, 2011.

[15] C.W. Dawson."A Neural Network Approach to Software Projects Effort Estimation", TransactionS on Information and Communication Technology 1996.

[16] ISBSG – Repository Data Release 11

[17] A. B. Nassif, L. F. Capretz and D. Ho, "Software effort estimation in the early stages of the software life cycle using a cascade correlation neural network model," in 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), Kyoto, Japan, 2012, pp. 589-594.

[18] Iman Attarzadeh, Siew Hock Ow. Proposing a New Software Cost Estimation Model Based on Artificial Neural Networks. International Conference on Computer Engineering and Technology, 2010, PP:487-491.

[19] L. Pickard, B. Kitchenham, and S. linkman, "An Investigation Analysis Techniques for Software Datasets," in Proc. of Sixth IEEE International Software Metrics, Symposium, 1999, pp. 1-13.

[20] Y. Zheng, B. Wang, Y. Zheng, and L. Shi, "Estimation of Software Project effort based on functional Point," in Proc. of 4th International Conference on Computer Science and Education , pp. 941-943, 2009.

[21] M. Sadiq and S. Ahmed, "Relationship between Lines of Code and Function Point and its Application in the Computation of Effort and Duration of a Software using Software Equation," in Proc International Conference on Emerging Technologies and Applications in Engineering, Technology and Sciences Rajkot, Gujarat, 2008, India.

[22] R Moser, W Pedrycz, G Succi Incremental effort prediction models in Agile Development using Radial Basis Functions Proc. 19th International Conf. on Software Engineering & Knowledge.

[23] Aggrawal KK , Yogesh Singh, Software Engineering 2nd Edition New age Publication.

[24] Matlab - www.mathworks.com

[25] Sommerville I. Software Engineering – 8th Edition

[26] Gupta D, Diwedi R, Kumar S, Domain Specific priority based implementation of mobile service – an Agile Way , International Conference of Software Engineering and Research, Las Vegas, USA.

[27] Coelho E, Basu A, Effort Estimation in Agile software Development using Story Points, International Journal of applied Information System, New York, USA Volume 3-No. 7 August 2012.

[28] Cagley Thomas, Agile Estimation using Functional Metrics- one of current thread of thought as voiced by Tim Lister on the Software Process and Measurement Cast 51.

[29] B. Tessem, Experiences in Learning XP Practices: A Qualitative Study. In Extreme Programming and Agile Processes in Software Engineering. 2003, 131-137.

[30] Capers Jones,"Estimating Software Costs: Bringing Realism to Estimating",2007, Tata McGraw-Hill , Second Edition.

Appendix -1

This shows the data set which is used to train the ANN. Following are the fields: