A novel Estimation model for agile development: Meta heuristic approach

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

Master of Technology

In

Software Engineering

by

Pradeep Kumar (2K14SWE14)

Under the guidance of

Prof (Dr) Daya Gupta (Former HOD)

Department of Computer Engineering, DTU

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

BAWANA ROAD, DELHI

DECLARATION

I hereby want to declare that the thesis entitled "**A novel Estimation model for agile development : Meta heuristic approach"** 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 Engineering** 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.

_____

Pradeep Kumar

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 **"A novel Estimation model for agile development: Meta heuristic approach"** done by **Pradeep Kumar** (2K14SWE14) for the partial fulfillment of the requirements for the award of degree of **Master of Technology** Degree in **Software Engineering** in the **Department of Computer Science Engineering**, Delhi Technological University, New Delhi is an authentic work carried out by his 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 Science 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.

Pradeep Kumar
M. Tech Software Engineering
2K14/SWE/14

ABSTRACT

Effective software project estimation is one of the most challenging and important activities in software development. Proper project planning and control is not possible without a sound and reliable estimate. As a whole, the software industry doesn't estimate projects well and doesn't use estimates appropriately. We suffer far more than we should as a result and we need to focus some effort on improving the situation.

Like any other Software paradigm Project estimation in Agile is also a tedious but important task for the software. A large number of formal and informal methods have been proposed for software estimation. The major factor for the project failure is unrealistic estimates which also affects the quality of 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.

In agile development many methods are proposed in the literature for the cost estimations of software projects. These methods use a large number of factors to estimate the development cost. In this thesis we apply Binary Cuckoo Search so that we can significantly reduce the number of attributes required. The identified attributes have maximum correlation to the development cost of the project. It is also evident from the literature that project estimation must be handled using an evolving system, hence Binary Cuckoo Search is one of the suitable technique for it. For the cost estimation, the concept of Binary Cuckoo Search is adopted where search follows quasi-random manner. The optimal solution obtained from the Binary Cuckoo Search shows that it is far efficient than other metaheuristic techniques like Genetic Algorithm and Particle Swarm optimization

Chapter 1

Introduction

Software engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software**.** Heuristic techniques such as genetic algorithms, particles warm optimization, Binary Cuckoo Search algorithm and tabu search have found wide applications in SE [1]. Meta heuristic techniques are used for optimization of problems by iterative improvement of a candidate solution.

1.1 Agile software development

Agile software development is a group of software development methods in which solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages rapid and flexible response to change.

The agile method is based on giving high priority to customer participation, from the very beginning of the development cycle. The objective is to keep the client involved at every step so that they have a product that they are happy with at the end. This method saves the client money and time because the client tests and approves the product at each step of development. If there are defects or challenges, then changes can be made during production cycles to fix the issue [2]. Traditional models of project management would not find defects as early because they do not test as often. Typically, (in traditional methods of production) defects that are not discovered at the different stages can find their way into the final product. This can result in increased overhead prices and client dissatisfaction. Model for Agile methodology is shown in Figure 1.1.

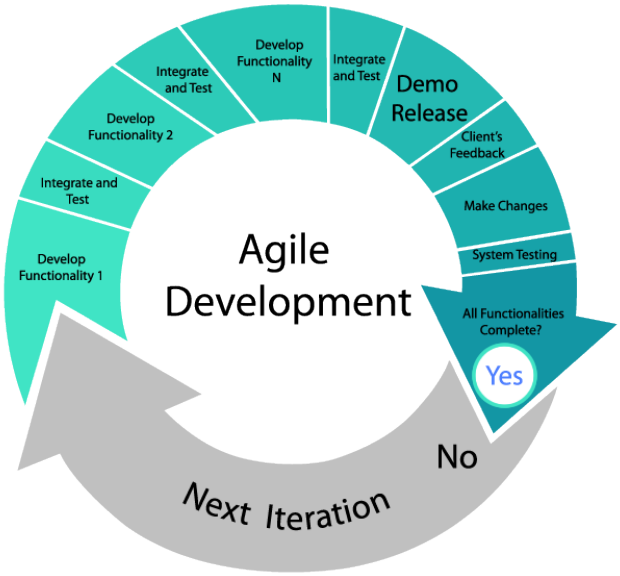

Figure 1.1 Model for Agile methodology

Businesses have proven this model of project management with their increased client satisfaction rate. The values for businesses that use this model include:

- Lower Cost
- Enables clients to be happier with the end product by making improvements and involving clients with development decisions throughout the process.
- Encourages open communication among team members, and clients.
- Providing teams with a competitive advantage by catching defects and making changes throughout the development process, instead of at the end.
- Speeds up time spent on evaluations since each evaluation is only on a small part of the whole project.
- Ensures changes can be made quicker and throughout the development process by having consistent evaluations to assess the product with the expected outcomes requested.
- It keeps each project transparent by having regular consistent meetings with the clients and systems that allow everyone involved to access the project data and progress.

1.2 Motivation

It is required to determine the development effort at the earliest stage so that team can produce a plan, a time line called the schedule and cost for the project. As the software is continuously evolving process and the changes are arbitrary, different new technologies, development environment and applications are developed and are being currently used. In current software industries, since software changes are arbitrary, requirement is an evolving system for carrying out the estimation process, especially in agile environment.

In the software development process three major point of the discussion are Estimation of the cost, size and duration. These three points are discussed in each type of the project whether it is agile, iterative, rapid application development or waterfall. Organization and development team modify and personalize the estimation and development technique so that they can be fitted over governance structures, culture and risk profiles. There is no one size that fits for all solutions.

ISBSG (International Standard Benchmarking Standard Group) whose work is the global and independent source of data and analysis for the IT industries, has latest released the volume 13. The comparison between volume 13 and 12 are shown in the Table 1.1 and Table 1.2

Table 1.1 As per reported by ISGSB vol. 13

S.NO	Development Techniques	% projects
1	Agile	14.6
2	Data modeling	32
3	Event modeling	8.1
4	Joint application development	9.9
5	Just in time RUP	7
6	Multifunction teams	9.7
7	Process modeling	29.27
8	Prototyping	16.2

9	Rapid application development	5.4
10	Reviews, walkthroughs, inspections	35.8
11	Sandwich, hybrid	35
12	Search based development	3.1
13	Specific techniques described	15
14	Traditional	71.3
15	UML, object modeling	17
16	Waterfall	25.7

Table 1.2 As per reported by ISGSB vol. 12

S.NO	Development Techniques	% projects
1	Agile	9.8
2	Data modeling	32
3	Event modeling	8.1
4	Joint application development	9.9
5	Just in time RUP	4.3
6	Multifunction teams	9.7
7	Process modeling	29.27
8	Prototyping	16.2
9	Rapid application development	5.4
10	Reviews, walkthroughs, inspections	35.8
11	Sandwich, hybrid	35
12	Search based development	3.1
13	Specific techniques described	15
14	Traditional	71.3

15	UML, object modeling	17
16	Waterfall	25.7

Comparing the above two tables we can see that there is a significant increase in the agile development methodologies for project developments. So in order to support and enhance the use of agile methodologies need for some good estimation techniques arise which can provide accurate results.

In my thesis I have investigated the domain of cost estimation practices for agile software development projects. So the knowledge of cost estimation techniques currently used in the industry and research are important to find drawback and limitation in current techniques. The following aspects are examined

- Conditions assumed by the projects that use agile process.
- Basic approach and state of the art of the estimation in projects which currently used agile execution.
- Applicability of the classic methods of estimation and academic backgrounds while using the agile procedures.

In literature survey, different cost estimation techniques for agile software developments have been studied to gather knowledge about the possibilities and borders of cost estimation task and new challenges.

1.3 Related Work

As we know the methods and metrics used in conventional development models are different from the agile developments methodologies, so they cannot be conveniently used for software cost, effort and time measurement. Software development cost overruns often that induces project managers to cut down manpower at the expense of software quality. Accurate effort estimation is beneficial to the prevention of cost overruns.

In a research by J.M. Desgarnais et al. in [11] they have developed an COSMIC (Common Software Measurement Consortium) method that guided an estimation approach which works on

COCOMO approach incorporating quality of documentation for functional analysis. It uses cumulative function point (CFP) for estimation where in each of the user stories are defined by single function point (FP) called user stories point or USP. COSMIC has introduced the first official guidance in Agile Software development methods for software sizing measurement based on functional point.

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

In another work by Zia et al. [10] propose a SWOT analysis based method which uses influence of external vs internal 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 [11] suggest a model which uses a trade-off relationship between response extensiveness and response efficiency of team.

Asnawi et al. in [12] 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 communications. However, these were not fully related to the aspects of projects which impact project performance such as cost, quality deadline and scope.

In [14] the researchers demonstrated how PCA based model can provide significant improvement in reliability and accuracy of effort predication over traditional analogy based model. They proposed a new cost estimation model for agile software development projects. In the model they apply Principal Component Analysis (PCA) to reduce the dimensions of the attributes required and identify the key attributes which have maximum correlation to the development cost; and then they use constraint solving approach to satisfy the criteria imposed by agile manifesto. They have extracted 12 key attributes out of initial 40 attributes given in the dataset. Their proposed methodology is found to be suitable for agile projects as it uses constraint programming to explicitly check for satisfaction of agile manifestos.

In another model [7] based on Artificial Neural Network (ANN) researchers has extracted 9 key project factors that have an effect on agile development effort. The main reason for using ANN for this problem is to keep the estimation process up-to-date by incorporating up-to-date project data. If the effort estimation process evolves with new projects, the estimation model is kept up-to-date. This overcomes the main problem in simple estimation models.

Another researcher in [8] found a new way that incorporates various optimizing factors through Genetic Algorithms and then applying Artificial Neural Network (ANN) via Matlab on project Data sets which can be helpful to increase the accuracy of Software estimation in Agile Projects.

D. Rodrigu in [13] proposed feature selection using binary cuckoo search algorithm and provided heuristics for it. In this paper the research were carried out in the context of theft detection in power distribution systems in two datasets obtained from a Brazilian electrical power company, and have demonstrated the robustness of the proposed technique against with several others nature-inspired optimization techniques.

In the literature survey it was observed those recent researchers are using any meta heuristic algorithm to extract the important key attribute's out of a list of available attributes.

Drawing inspiration from this we explored the Binary Cuckoo Search Algorithm and Artificial Neural Network Model for cost estimation in the agile software development environment. The dataset used in [7,8,14] will be used in the thesis for cost estimation using our proposed model. And also the MMRE value for our approach is compared with other models presented in [7,8,14].

1.4 Research Problem

Researchers have proposed a lot of methodologies for cost estimation for software projects using Agile. These methods use large number of project factors for determining the development cost

involved in the project. A few factors include team size, team quality, size of software, clarity of requirements, development environment, technologies chosen etc.

In this thesis we apply Binary Cuckoo Search so that we can significantly reduce the number of required project attributes. The identified attributes have maximum correlation to the development cost of the project. This might enhance the cost estimation process. To increase the accuracy, the constraints of agile manifesto are considered.

The research problem can be thus stated as follows:

"To propose a robust cost estimation model for Agile software project which extract key factors using Binary Cuckoo Search Algorithm and estimate the development cost and satisfies the agile manifesto constraints"

1.5 Scope of the work

In this study we report on the results obtained by exploratory factory analysis carried out on agile development of various industry projects. The proposed cost estimation model uses Binary Cuckoo Search Algorithm to determine correlation between various project attributes and the cost of development.

The scope of the thesis is presented as follows:

(i) We have done exploratory factor analysis of the agile development data. Using Binary Cuckoo Search as discussed in the above section 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) Next, 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 proposed model is validated by comparison with the existing cost estimation model in terms of mean magnitude of relative errors (MMRE) to show improved accuracy of estimations. The dataset is shared by the senior M. Tech student with us. The dataset is shown in Appendix-1.

1.6 Organization of Thesis

The rest of the thesis is organized as follows

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 and discusses the Binary Cuckoo Search etc.

Chapter 4 Presents the proposed research methodology which checks the flexibility of using Binary Cuckoo Search 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

This chapter describes the Agile software development. Agile Manifesto [14] stresses the importance of four main things (i) people and interactions over processes and tools, (ii) working software instead of detailed documentation, (iii) active customer participation and involvement rather than time and effort expended on negotiating contracts, and (iv) willingness and ability to take on changes over stead fast commitment to a static plan. Agile software development methods including eXtreme Programming (XP), Scrum, Adaptive Software Development and Feature-Driven Development are based on the principles of the Agile Manifesto and geared towards realizing its goals and objectives as shown in figure 2.1. In general, the feedback from organizations that have implemented agile development is positive. Some of the benefits attributed to agile development are:

- Increased productivity
- Expanded test coverage
- Improved quality/fewer defects
- Reduced time and costs
- Understandable, maintainable and extensible code
- Improved morale, better collaboration, and higher customer satisfaction.

The adoption of agile development has also revealed some challenges such as slow participant buy-in, opposition to pair-programming, lack of detailed cost evaluation, scope creep, reduced focus on code base's technical infrastructure and maintainability, difficulty evaluating and rewarding individual performance, and the need for significant on-site customer involvement, management support, competent managers and developers, and extensive training [15]. It is a substitute to traditional project management, used in software development. Agile methodologies are an alternative to waterfall, or traditional sequential development. Agile Methodology is shown in Figure 2.1

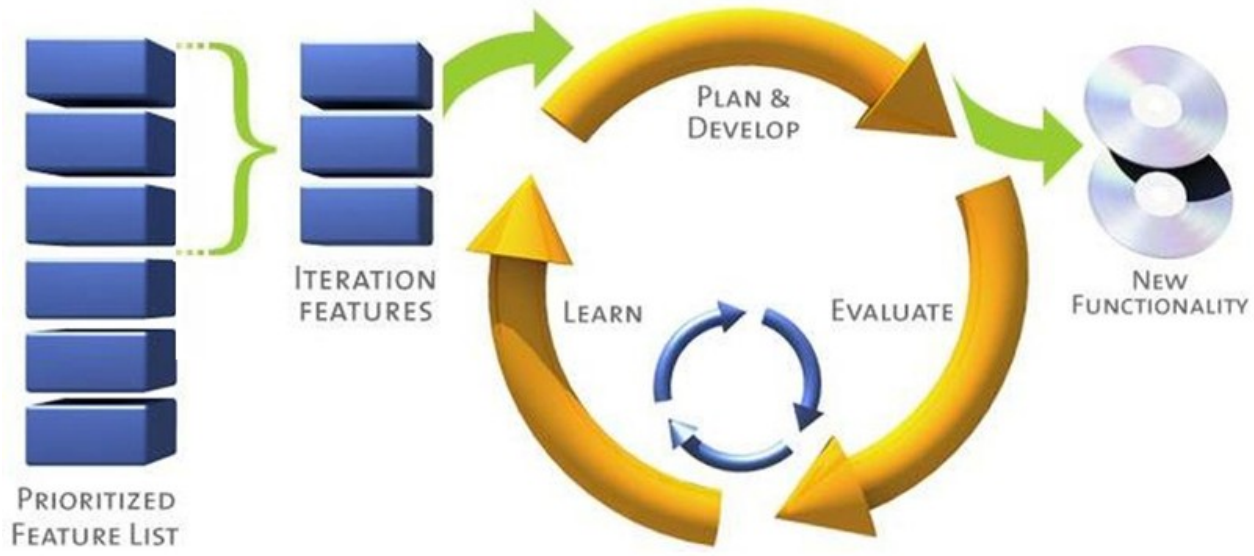

Figure 2.2 Agile Methodology

2.1 Factors affecting Agile development

This section discusses the various factors that affects the Agile process. Agile development methods promote development, teamwork, collaboration, and process adaptability throughout the life-cycle of the project. The factors that are considered are Organizational Factors, People factors, Process factors, technical factors and project factors [9]. Factor affecting the Agile Development is Shown in Figure in 2.2.

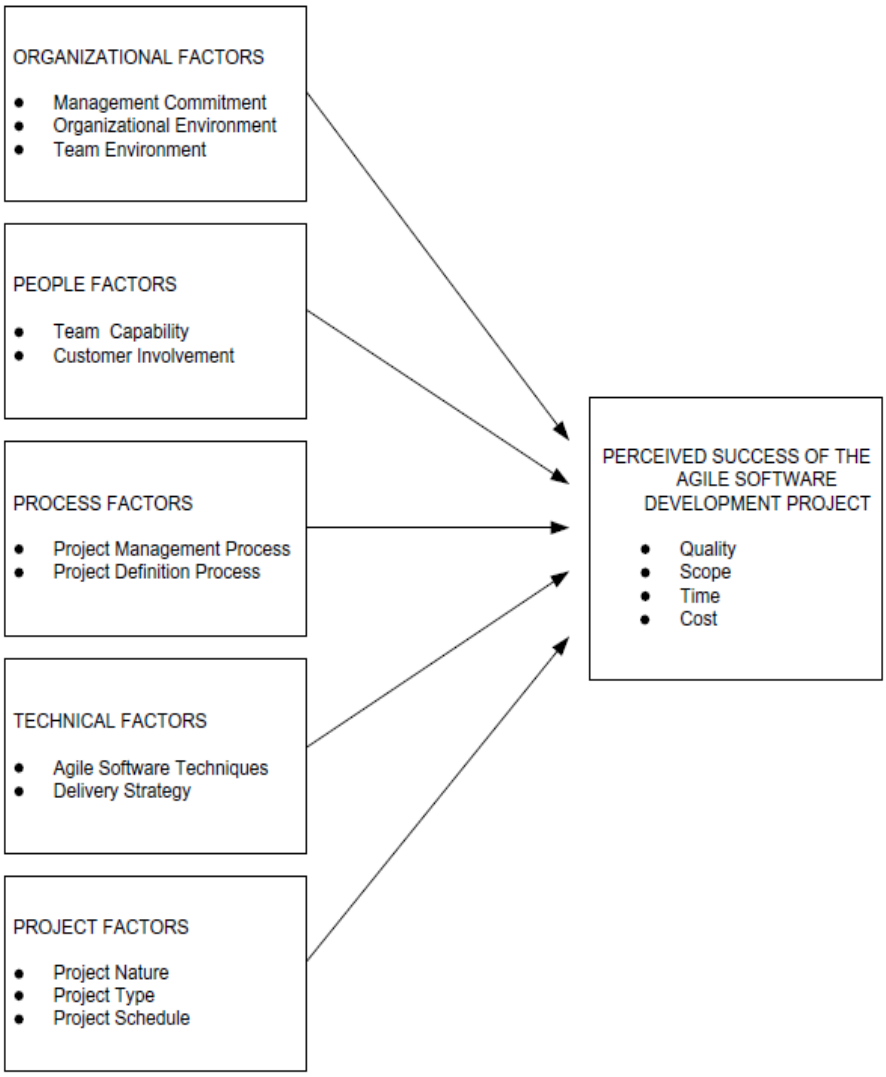

Figure 2.3 Factors affecting Agile development [15]

2.2 Project management activities in Agile

There are a number of activities involved for managing the team in Agile development a few are discussed in table 2.1.

Table 2.3 Agile development practices and description

Agile Practice	*Description*
Iteration Planning	The iteration planning session is a meeting that takes place at the start of each iteration where the team collectively defines and plans tasks that must be completed during the next iteration (Beck and Andres 2005; Schwaber and Beedle 2002)
Daily Stand-Up	The daily stand-up is a short daily status team meeting lasting a maximum of 10-15 minutes typically conducted at the same time each day with team members standing up. During the meeting, team members explain briefly what they accomplished since the previous meeting, what will be completed by the next meeting, and indicate any impediments that may prevent them from completing their current tasks (Elssamadisy 2008; Schwaber and Beedle 2002).
Iteration Retrospective	An iteration retrospective is a meeting that is held at the end of each iteration where the project team reflects on what went well in the iteration, what did not, and what could be improved for future iterations (Elssamadisy 2007; Schwaber and Beedle 2002).

2.3 Methodologies used in Agile Development

There are various methodologies that are collectively known as agile, as they promote the values of the agile manifesto and they are consistent with the above principles. The most popular ones are:

DSDM is probably the original agile development method. DSDM was around before the term 'agile' was even invented, but is absolutely based on all the principles we've come to know as agile. DSDM seems to be much less well-known outside of the UK.

Scrum is also an agile development method, which concentrates particularly on how to manage tasks within a team-based development environment. Scrum is the most popular and widely adopted agile method – I think because it is relatively simple to implement and addresses many

of the management issues that have plagued IT development teams for decades. Scrum was applied in 1990s by Ken Schwaber and Mike Beedle. It is an agile, iterative, incremental developing method which assumes that changes and chaos exist through entire life-circle of the project and attempt to solve these problems [16]. Scrum is designed to add energy, focus, clarity and transparency to project teams' development software systems. It allows team to operate in close proximity to foster rapid system evolution. Scrum methodology is shown in Figure 2.3.

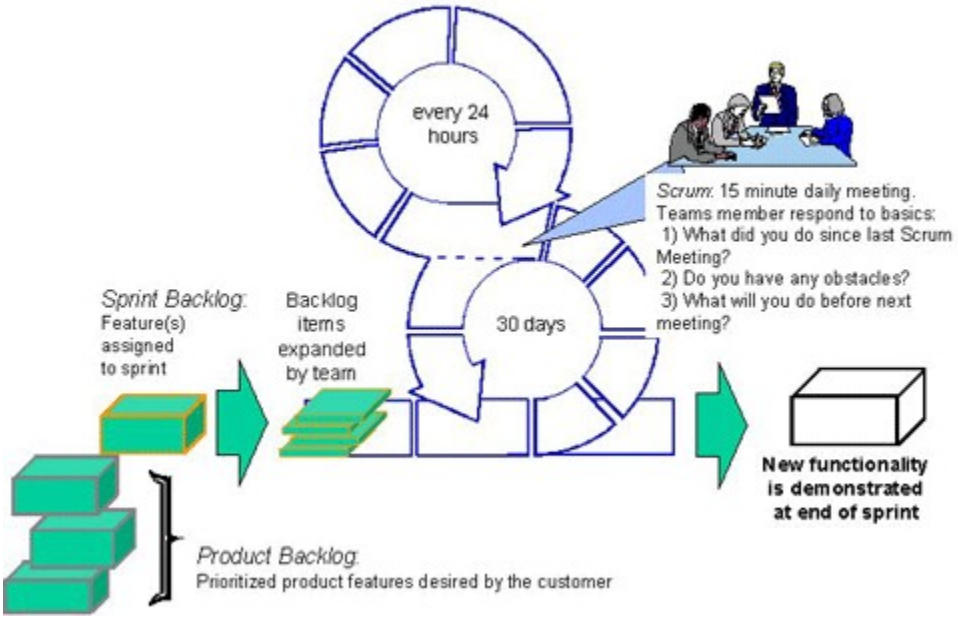

Figure 2.4 Scrum methodology in Agile

XP (Extreme Programming) is a more radical agile methodology, focusing more on the software engineering process and addressing the analysis, development and test phases with novel approaches that make a substantial difference to the quality of the end product.

Extreme Programming is one of the most widely adopted agile methodologies. The XP methodology was created by Kent Beck. The XP improves a software project in four essential ways which are communication, simplicity, feedback and courage. It is introduced twelve best practices with which XP programmers are able to courageously respond to changing requirements and technology [17]. XP methodology is shown in Figure 2.4.

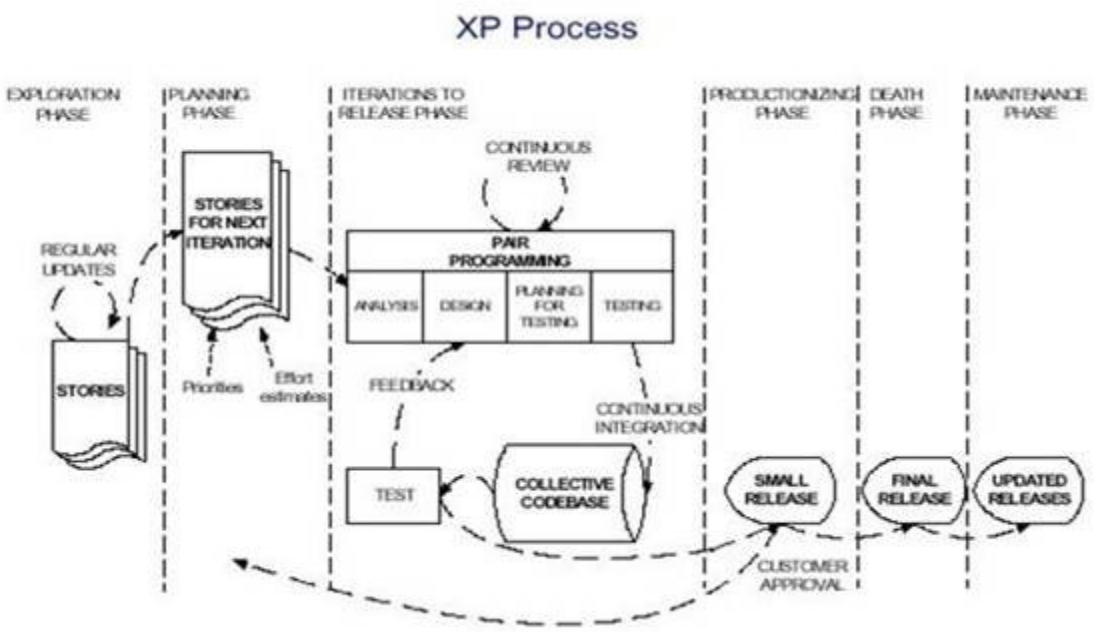

Figure 2.5 XP methodology in Agile

Dynamic System Development Method (DSDM) is probably the most complete agile methodology, whereas Scrum and XP are easier to implement and complementary because they tackle different aspects of software development projects and are both founded on very similar concepts.

DSDM, dating back to 1994, grew out of the need to provide an industry standard project delivery framework for what was referred to as Rapid Application Development (RAD) at the time. While RAD was extremely popular in the early 1990‘s, the RAD approach to software delivery evolved in a fairly unstructured manner. As a result, the DSDM Consortium was created and convened in 1994 with the goal of devising and promoting a common industry framework for rapid software delivery. Since 1994, the DSDM methodology has evolved and matured to provide a comprehensive foundation for planning, managing, executing, and scaling agile process and iterative software development projects

2.4 Advantages of Agile Development

1. The Agile methodology allows for changes to be made after the initial planning. Re-writes to the program, as the client decides to make changes, are expected.

2. Because the Agile methodology allows you to make changes, it's easier to add features that will keep you up to date with the latest developments in your industry.

3. At the end of each sprint, project priorities are evaluated. This allows clients to add their feedback so that they ultimately get the product they desire.

4. The testing at the end of each sprint ensures that the bugs are caught and taken care of in the development cycle. They won't be found at the end.

5. Because the products are tested so thoroughly with Agile, the product could be launched at the end of any cycle. As a result, it's more likely to reach its launch date.

2.5 Limitations of Agile Development

In general, we all are more convinced about the benefits of the agile development. Here we have found some limitations of agile development.

1. Limited support for Distributed Development Environments

2. Limited support for Development Involving large teams

3. Steel learning Curve

4. Limited support for sub-contracting

5. Limited support for developing safety critical software and Legacy software

6. Poor project documentation

7. Limited support for developing large and complex software

8. Lack of standards

9. Inadequate project requirements and testing

10. Limited support for building reusable artifacts

11. Longer project turnaround time

12. Inability to work in CMMI environment

13. Lack of predictive control

2.6 Difference between Agile and traditional development

Table 2.2 gives a comparative analysis of Agile and traditional development system.
Table 2.3 discuss the scenario in which Agile can be used over traditional approach.

Table 2.4 Differences between traditional approach and agile approach to software development[17][18]

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 2.5 When to use which model

Factor	Status	Agile	Waterfall
Project size and complexity	Smaller, less complex	x	
	Larger or more complex		x
Requirements	Not stable requirements	x	
	Stable requirements		x
Solution (product)	Not a clear picture of the final solution (product)	x	
	Clear picture of the final solution (product)		x
Customer availability	Available frequently throughout the project	x	
	Customer cannot commit to extensive involvement		x
Customer tolerance for scope and cost changes	Budget and schedule are flexible	x	
	Budget and/or schedule are fixed and difficult to modify		x
Integration with external systems	Low level of integration or not needed	x	
	High level of integration, complex or unknown		x
Time to market	Rapid deployment needed; can have limited feature set	x	
	Full featured application must be delivered, within a determined timeline		x
Project team	High level of skill set and ability to work independently	x	
	Lower level of skill set or require direct supervision		x
Tracking, control and reporting	Low (complexity is low, probability of variances is low)	x	
	High (large budget or timeline constraint, finite resource timing, legal)		x
Customer preference on methodology	The customer has stated methodology requirements	Generally, as requested	
Organization processes and procedures	The organization has processes and procedures that we have to apply	We use the processes and procedures	

2.7 Hurdles in using Agile methods

This section discusses some of the hurdles that needs to be overcome while using Agile methodology.

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 [20]. 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 specialization in only one area like program writing using a particular language or build deployment only [20].

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[20].

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

Chapter 3

Software Estimation Techniques

Software estimation has always been an active research area. Accurate software estimation is desirable in any software project, not only to properly schedule budget, resources, time and cost and avoid over run but also to reasonably estimate as software organizations with better estimates and planning will be able to get the projects in bidding. Pre-bid estimation is paramount in getting business for the company. Accuracy of pre-bid estimation governs the smooth running and success of a project. Estimation output forms the basis for sub sequent project plan and activities as well as client commitments. According to Porter [21] competitive advantage is achieved through innovation, cost leadership and quick response for customers. Therefore, software process change is inevitable and should ideally be based on software measurement programs [22].

As shown in figure 3.1 processes should to be facilitated to smooth the overall estimation procedure. Practice recommended for software estimation is

1. Software Size computation
2. Effort estimation in person-hour is derived from software size.
3. Cost & budget calculation.
4. Proper scheduling, resource allocation is done as a final step.

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. Process of project estimation is shown in Figure 3.1.

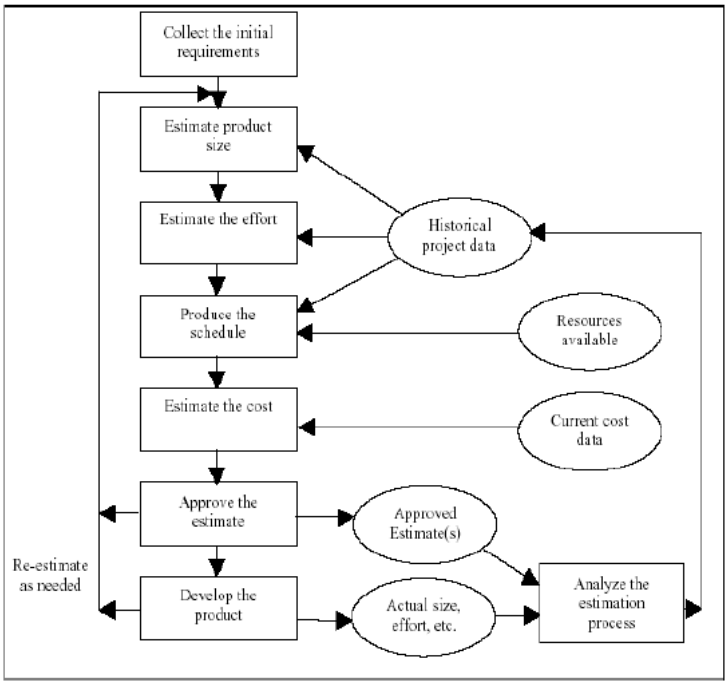

Figure 3.6 Process of Project Estimation [23]

3.1 Methodologies of Estimation

Various methodologies as discussed in [24] are discussed in this section.

3.1.1 Analogy Method

In analogy approach the project to be estimated is compared with the already complete projects of that type if exists. The historical data of previously completed projects helps in the estimation. However, it works only when previous data is available. Needs systematically maintained database.

3.1.2 Top Down Method

Top down approach requires less functional and non-functional requirements and is concerned with the overall characteristics of the system to be developed. This estimation is quite abstract at the start and accuracy improves step by step. It can under estimate the cost of solving difficult low-level technical components. However, top down approach takes into account integration, configuration management and documentation costs.

3.1.3 Bottom Up Method.

This method does estimation of each and every individual component and combines all components to give the overall, complete estimation of project. This approach can be an accurate method if the system has been designed in detail. However, bottom up method can underestimate the cost of system level activities such as integration and documentation.

3.2 Estimation techniques

Various techniques exist to cater the estimation procedure. Some of these techniques are discussed next.

3.2.1 Lines of Code (LOC)

Direct software size can be measured in terms of LOC (Lines of code), one of the oldest techniques. This measure was first proposed when programs were typed on cards with one line per card. Its disadvantage is that accuracy of LOC is highly dependent on the software completion and before that only expert judgment estimates can be given. Also LOC is language dependent. 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.

3.2.2 Function point

Albrecht of IBM developed the Function Point metrics in 1979 [25]. 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. 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.

3.2.3 Cosmic Full Function Points

The COSMIC-FFP method was first published in late 1999 after proposed by St-Pierre et al. [26] and became stable with the publication of an International Standard definition in 2003. COSMIC-FFP is now a standard (ISO/IEC 19761:2003)

Three other methods, IFPUG originating in the USA, Mark II FPA method from the UK and the NESMA method from the Netherlands have also been approved by ISO. These three first generation functional sizing methods were all designed 10 – 20 years ago to work for business application software. Compared with 1st Generation Measurement methods, the COSMIC-FFP method has the following main advantages and benefits.

1. The COSMIC-FFP method can be applied to business, real-time and infrastructure Software. Operating system software, telecom and process control) And to hybrids of all these types.

2. COSMIC-FFP underlying concepts are compatible with modern methods of determining software requirements and constructing software, the method is easier to learn and apply, and automatic sizing is possible with the right tools.

3.2.4 COCOMO

COCOMO (Constructive Cost Model) is an empirical estimation scheme. [26] 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.

COCOMO is a hierarchy of software cost estimation models, which include following three sub models

- Basic COCOMO applies the parameterized equation without much detailed consideration of project characteristics.

  Effort Applied (E) = $a_b(KLOC)^{b}_{b}$ [man months]
  Development Time (D) = $c_b(\text{Effort Applied})^{d}_{b}$ [months]

- Intermediate COCOMO : The same basic equation for the model is used, but
  Fifteen cost drivers are rated on a scale of 'very low' to 'very high' to calculate the specific effort multiplier and each of them returns an adjustment factor which multiplied yields in the total EAF (Effort Adjustment Factor). The adjustment factor is 1 for a cost driver that's judged as normal. In addition to the EAF, the model parameter "a" is slightly different in Intermediate COCOMO from the basic model. The parameter "b" remains the same in both models.

  $E=a_i(KLoC)^{(b_i)}(EAF)$

  EAF is the Effort Adjustment Factors.

- Detailed COCOMO : The Advanced COCOMO model computes effort as a function of program size and a set of cost drivers weighted according to each phase of the software lifecycle. The Advanced model applies the Intermediate model at the component level, and then a phase-based approach is used to

Consolidate the estimate [ ]

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

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

Where “a = 2.5” and “b = 0.91” are the domain constants in the model.

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

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

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

In the above equation a = 2.5 and B is 0.91 which are a constant same as COCOMO 81. 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 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. 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. 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.

COCOMO model is provided for three operational modes:

1. Organic. Applied in projects that have a small, experienced development team developing applications in a familiar environment.
2. Semi-detached. Semi-detached mode is for projects somewhere in between.
3. Embedded. Embedded mode should be applied to large projects, especially when the project is unfamiliar or there are severe time constraints.

3.2.5 Use Case Estimation

Now-a-days estimation with use case [27] is also gaining popularity as most of the functional and non-functional requirement is captured initially in the form of Use cases. However, because of many different styles and formats of use cases, this technique is a little problematic. Also just a count of use case, scenarios per use case didn't give an accurate idea of the complexity of problem to be solved. Still there are many case studies and papers that revealed the success and effectiveness of use case estimation. For parametric computations, three estimates optimistic, likely and pessimistic are computed and an average is considered.

3.2.6 Planning Poker

Planning Poker is the most widely used technique for estimation in agile environment, which is consensus, based. This technique is highly depended on the experts who are in the team of estimation process. All the developers are the part of team. The term "developers" refers to all analysts, programmer, testers etc. The max limits for developer typically not exceed ten people for the agile projects.

Each estimator is holding a deck of Planning Poker cards with values like 0, 1, 2, 3, 5, 8, 13, 21, and 34, which is the Fibonacci sequence we recommend. The values represent the number of story points, ideal days, or other units in which the team estimates. 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 [5]. The

values represent the number of story points, ideal days, or other units in which the team estimates.

The estimators discuss the feature, asking questions of the product owner as needed. When the feature has been fully discussed, each estimator privately selects one card to represent his or her estimate. All cards are then revealed at the same time. If the estimates are significantly different then the high and low estimator explains their estimates. After the discussion, each estimator re-estimates by selecting a card. This process of re estimation is repeated until the estimators converge on a single estimate that can be used for the story. Figure 3.2 shows how the planning poker works.

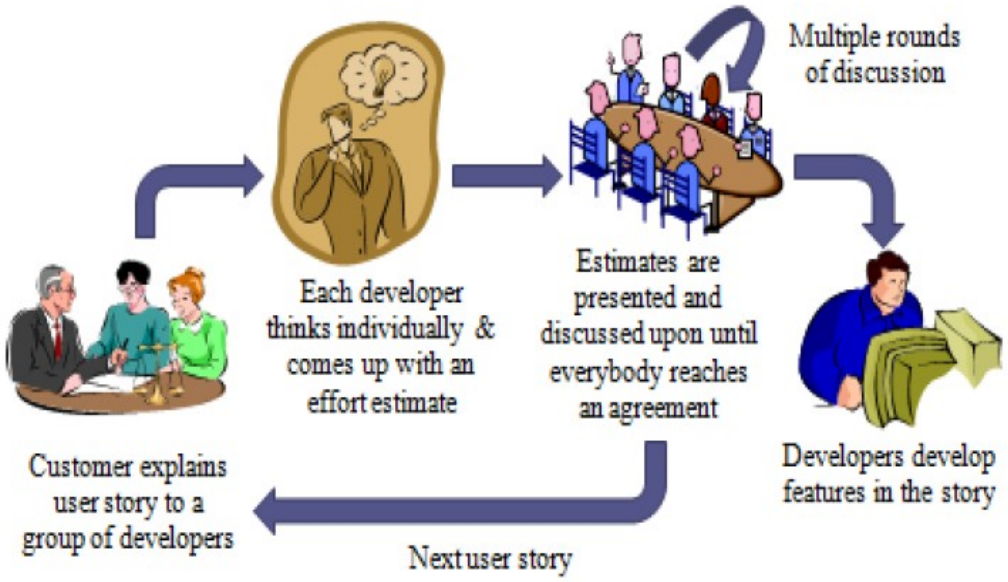

Figure 3.7 Planning poker Method

Shortcomings of Planning Poker

Based on a research [5], 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 [5], this value comes out to be 1.0681 or 106.81% which is very high and can be reduced.

- **An expert-dependent method**

  An expert is more likely to convince his/her opinion to the rest of the team than the novice developer in the team. And if the expert is absent than accuracy decreases substantially.

3.2.7 PCA Based Model

First step of this model is identifying the factor affecting development cost by analysis of the sample data using the Exploratory Factor Analysis (EFA) with factor extraction using Principal Component Analysis (PCA) [14]. 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 constraints programming. This estimation process is expected to enhance the level of visibility of cost estimation in the planning stages. Figure 3.3 Shows the stages of PCA based model.

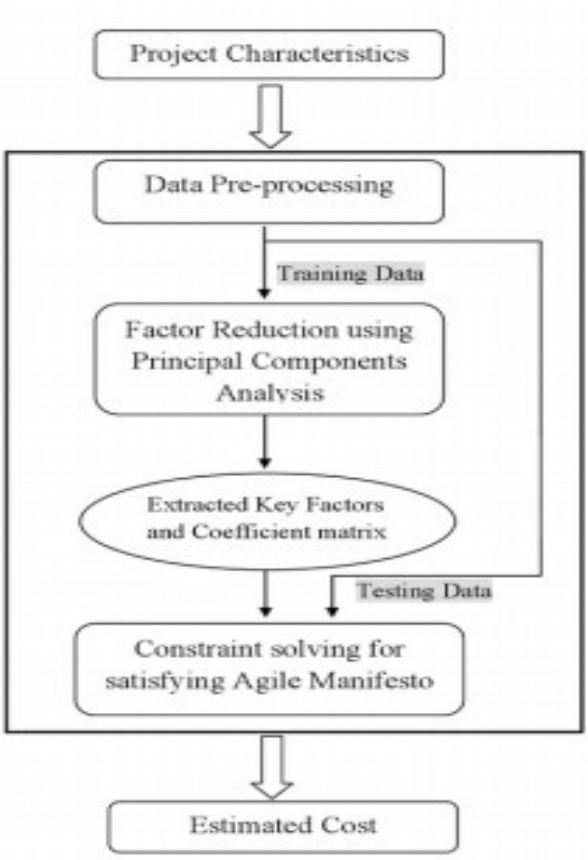

Figure 3.8 PCA model for cost estimation

3.2.8 Estimation using artificial neural network

An artificial neural network (ANN), usually called neural network (NN), is a mathematical model or computational model that is inspired by the functional aspects of biological neurons [31]. 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 effort required for developing software's by traditional methodologies [17].

One of the most widely used NN is Feed Forward Network in which information flows in one direction, which is from the input layer to the final output layer via the hidden layers. Different types of feed-forward NN with hidden layers exist. These include Radial Basis Function neural Network (RBFNN), General Regression (GRNN), and Multilayer Perception (MLP), A MLP contains at least one hidden layer and each input vector is represented by a neuron. To minimize the error the number of hidden neurons can be determined by the trial and error method. MLP

type is used to predict software effort based on Software size calculated based on the FPA method, Team Size and cumulative weight of metrics which affect the effort in agile environment. Two most recently proposed model which use Artificial Neural Network (ANN) are discussed below.

- **Feasibility of ANN in Agile Methodologies for Effort Estimation [7]**

  The two basic objectives of this methodology are first is to identify all project characteristics along with their intensity and weights 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. Flow chart in Figure shows the overall framework for this methodology:

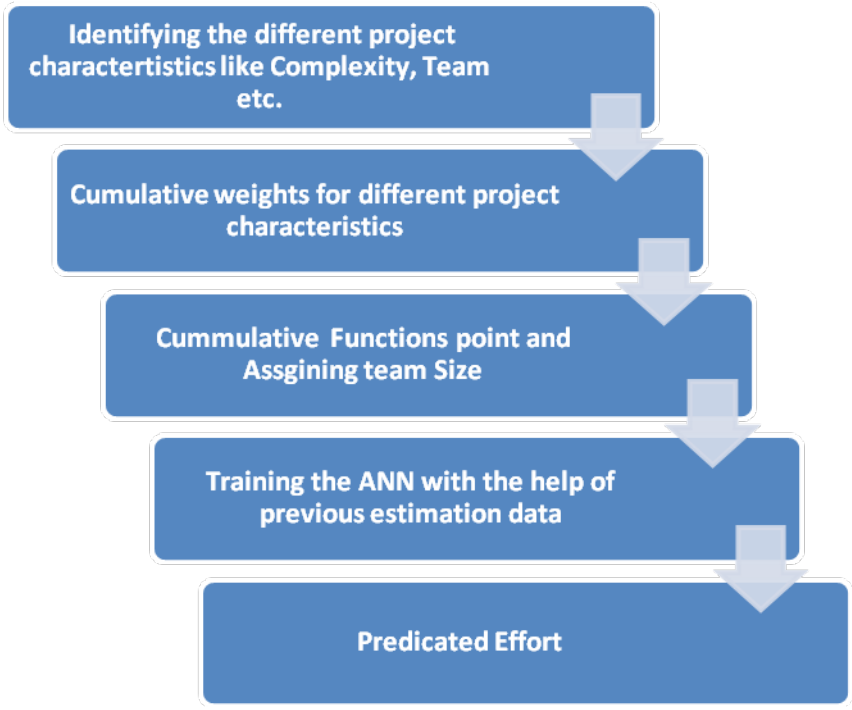

Figure 3.9 Proposed methodology

All characteristics are project driven and the project manager has to select different intensity level for each factor based on functionality of project on the scale of Very low,

low medium, high and very high. Then project manager will decide the weight age of different metrics by assigning different intensity level to the factors in the metrics. To identify the intensity level of different vectors in different metrics a tool in PHP has been developed by them which gave user a GUI to select different intensity level on the grade of very low, low, medium, high and very high. Once the intensity level of different vectors are been identified, then through some general algorithm the weight of individual metrics are found. Now to get a cumulative or overall weight of the metrics, all the individual weights of different metrics are added. This cumulative weight will be the vital input to the ANN. This weight tells about the nature of the project, the more its value the more critical is project and vice versa.

- **Agile Project Estimation by optimizing various factors [8]**

The two basic objectives for performance of this methodology are 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. Figure shows the step of agile project estimation by optimizing various factors

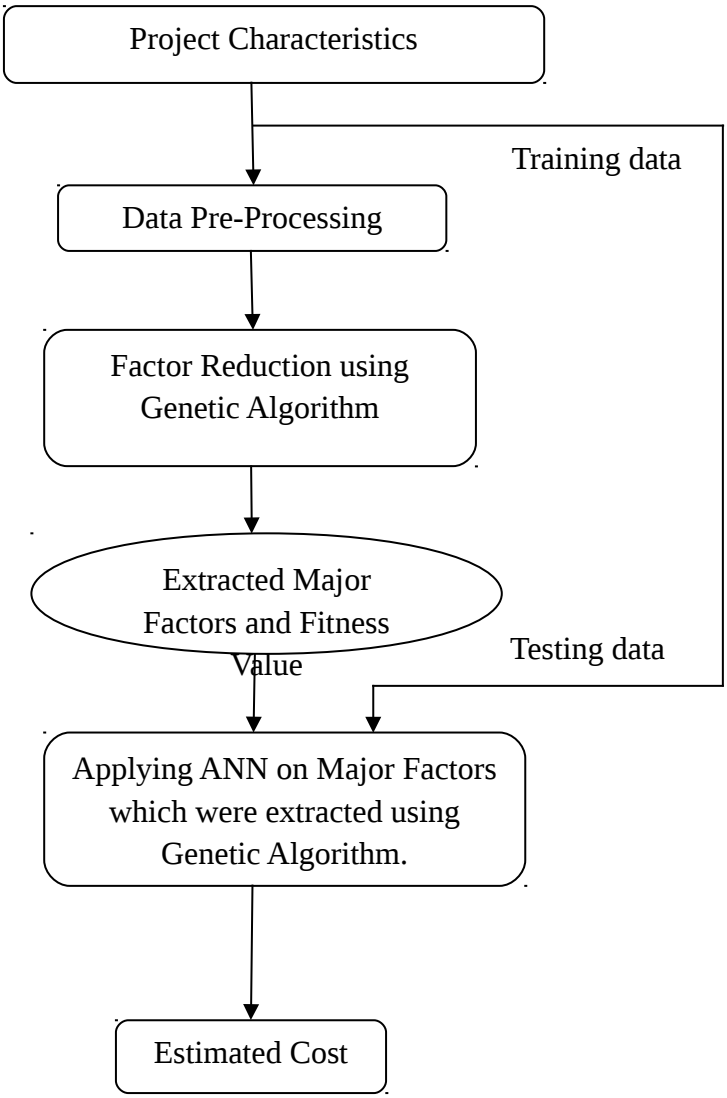

Figure 3.0.10 Process model for cost estimation model

Feasibility of ANN in Agile Methodologies for Effort Estimation [7] and Agile Project Estimation by optimizing various factors [8] concluded that software effort estimation must be handled using an evolving system like artificial neural network rather than a static one. This technique will help one to achieve lower Mean relative error (MRE) value and will help to reduce the losses due to inaccurate estimation.

Chapter 4

Proposed Methodology

This chapter describes the framework and techniques used for our work. There are two basic objectives for performance of this methodology. First is to identify all project Factors along with their fitness value via Binary Cuckoo Search approach as they play a vital role in project estimation. Second is to use the data from various companies around the globe to validate the results. We propose to use Binary Cuckoo Search in Agile for the factor extraction of project.

4.1 Overview of cost estimation model

Going throw the PCA based model, Genetic Algorithm and SWOT based model we proposed a new method for cost estimation which is based on Binary Cuckoo Search algorithm. All the above algorithm extract selected set of attribute which have high influence in project cost estimation as compared to other set of attributes.

We use binary cuckoo algorithm for the most influential 14 factors attribute extraction which play major role in cost estimation. 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. The Proposed methodology is shown in Figure 4.1.

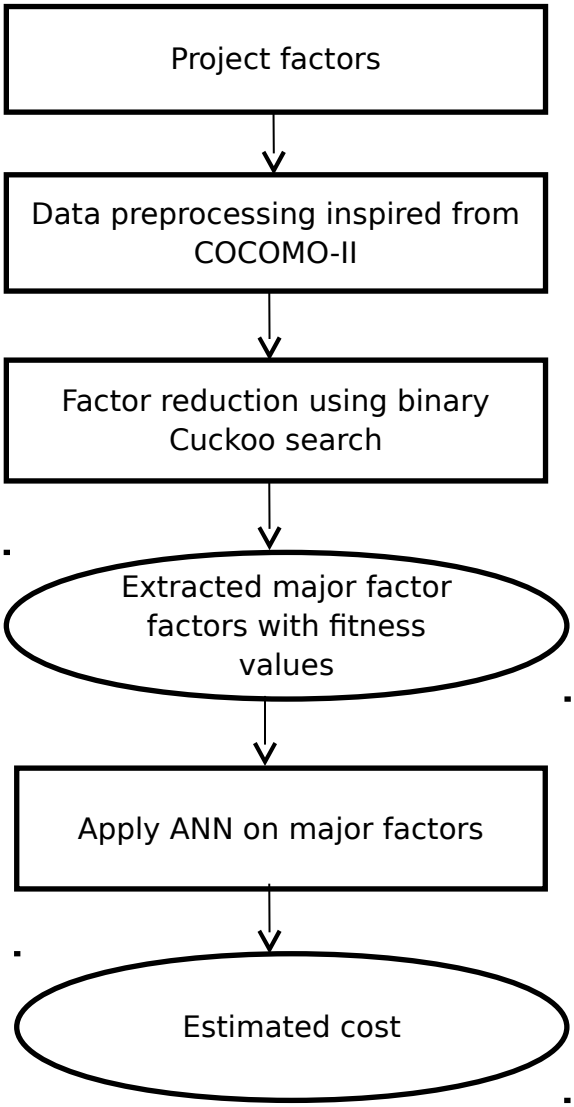

Figure 4.11 Flow chart of proposed methodology

4.2 Data Preprocessing

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 As

we know factors related to man are more important than the process in agile development we select the 9 factors related to man using COCOMO II.

4.3 Description of Cuckoo Search Algorithm

There are many nature inspired meta heuristic algorithms cuckoo is the latest one out of these. It is based on the brood parasitism of cuckoo bird. Using levy flight of these birds we can further enhance this. Recent studies show that cuckoo search is far efficient than its pear of the same group like BBO, PSO and genetic algorithms. For optimization problem, the quality or fitness of a solution can simply be proportional to the value of the objective function. In cuckoo some Forms of fitness can be defined in a similar way to the fitness function defined in genetic algorithms.

The idea of CSA is developed by the mimic behavior of cuckoo to save new generation. Cuckoo makes fascinating sound as well as aggressive and unique reproduction technique. Few cuckoo species like ani and Guira lay their eggs in communal nests i.e. nest managed by some other bird, though they may eliminate other‘s eggs to increase their own egg hatching probability [50].28 There are three types of brood parasitism: co-operative breeding, nest takeover and intra specific brood parasitism. A notable number of species engage in obligate brood parasitism by laying their own eggs in the nests of other host birds more often in some other species nest. Some host birds can engage in direct conflict with the intruding cuckoos. If a host bird found that the eggs are not their own, they will either throw away found alien eggs or simply leave its own nest and build a new nest somewhere else.

Some cuckoo species such as the New World brood-parasitic Tapera have evolved to such extent that they can mimic color and pattern of the eggs of a few chosen host species. This reduces the probability of their eggs being destroyed and thus increases their reproduction probability. Timing of laying egg of some species is also fascinating. In general, the cuckoo eggs hatch slightly earlier than their host eggs. Once the first cuckoo offspring is hatched, the first action it will take is to evict the host eggs by blindly throwing the eggs out of the nest, which increases the cuckoo offspring‘s share of food provided by its host bird. Cuckoo search idealized such breeding behavior, and thus can be applied for various optimization problems. Cuckoo Search is simulated by set of independent nests. Eggs in the nest represent an independent separate

solution and a cuckoo egg represents a possible new solution. The goal is to use the new potentially better solutions (cuckoos) to replace a bad solution in the nests. For simplicity, it can be considered that each nest contain one egg that is, one solution but in the case of optimizing multi-objective function multiple egg in a nest represents multiple solution corresponding to several objectives. The algorithm can further be extended to more complicated cases according to the need of situation and problem. In the basic CS, levy flight of cuckoo i is defined as following.

$$x_i(t+1)=x_i(t)+\alpha \oplus Levy(\lambda)$$

where $x_i(t)$ and $x_i(t+1)$ represent the position of the cuckoo before and after applying levy flights, respectively, also $\alpha>0$ is step size. Cuckoo search algorithm is defined with the help of following 4 steps.

- Neighborhood Structure: This represents a schematic solution representation of the problem.
- The step of movement: usually it is defined as distance between two solutions
- Kill some % of new eggs : as mention earlier, at each iteration of the algorithm some % of new eggs will be killed which most of them have low quality or low fitness.
- Keep the best eggs and nest: Sort the population based on their fitness and selects the best Nmax of them. If the stopping criterion is met stop, otherwise continue.

4.4 Factors used in Cuckoo search

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 shown in Table 4.1.

Table 4.6 List of the project Characteristics (attributes)

1. Language Type	2. Development Platform
3. Language Type	4. CMMI
5. Data Base System	6. Architecture
7. Organization Type	8. ISO
9. Hardware	10. Type of Server
11. Following process model	12. Package Customization
13. Training	14. Project complexity
15. Technical ability	16. Function points
17. Planning	18. Reliability
19. Debugging capability	20. Pages of documents
21. Client/Server	22. Risk taking
23. Communication skills	24. Managerial skills
25. Process maturity	26. Ease of use
27. Proximity of team	28. Documentation resources
29. Tool availability	30. Early delivery
31. Feedback	32. Documentation period
33. Tool familiarity	34. Application Type
35. Software size	36. Programming Language
37. IDE	38. Operating System
39. Productivity	40. Team Size

4.5 ANN and Its Roles

Artificial Neural Network 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 175 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 Cuckoo search. The data set is shown in Appendix 1. To create and train the ANN different tools (nftool, nprtool) from Matlab are used. ANN Model used for this research is shown in Figure 4.2

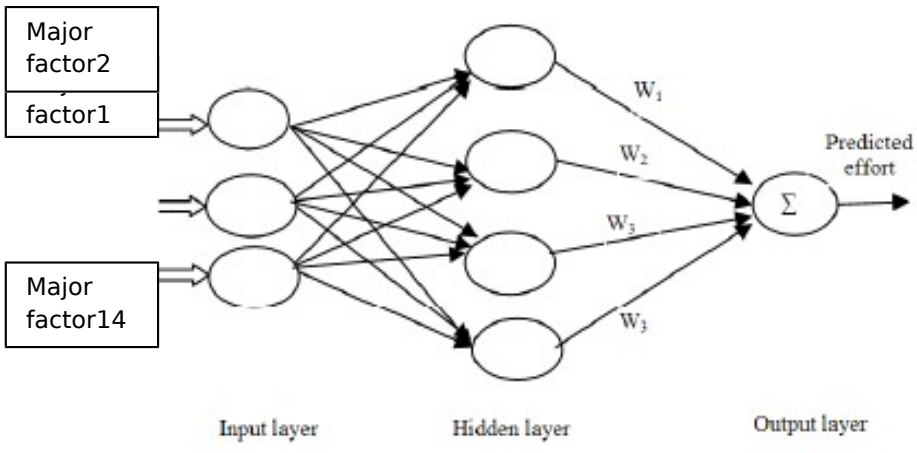

Figure 4.12 ANN model used

4.6 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 Binary Cuckoo Search 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 binary Cuckoo search) 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 details

This Chapter discusses in detail the proposed cost estimation model on the sample data set and high lights the step by step procedure used for cost estimation. It also presents a detailed description of the data set that has been used, the data set is provided in Annexure 1. And finally the tools used for the implementation are also discussed.

5.1. Feature represents

Single dimensional array is taken whose size is equal to the number of features present in the project. Therefore, the size of array will be 40 because the data set taken for implementation has 40 different features (attributes). Next each index of the array is randomly initialized with 0 or 1, where 0 represent that particular feature is not selected and 1 represent that particular feature is selected. A sample array with random 0 and 1 values is shown in Figure 5.1.

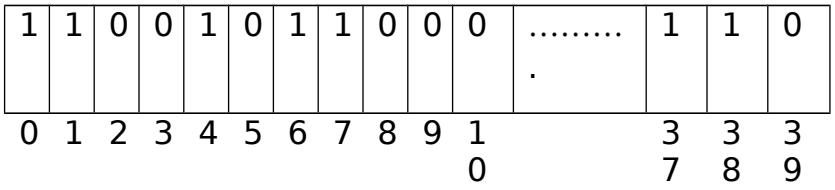

Figure 5.13 Sample array with Random values.

In the Figure 5.1, at index 0 binary value 1 is present which represent that feature 1 is selected and at index 3 presence of binary value 0 represent that feature 3 is not selected.

5.1.1 Nest

Nest is a container which holds egg in it. We take one egg in each nest for simplicity. Each egg represents which set of features is selected or not. Every egg represents different set of features.

5.1.2 Population

- Population size is fixed.
- Population represents the collection of different set of features.

5.1.3 Fitness function

$$\mathbf{PC_{initial} = A^{*}\ (size)^{B}}$$

Where $PC_{initial}$ = initial project cost

A = constant which is 2.5 (Taken from COCOMO II)

B = scale factor

B = 0.91 + 0.01 * (sum of the nine man related factors in projects which are given below)

1. Team size
2. Tech ability
3. Tool familiarity
4. Dev. Platform
5. Managerial skills
6. Architecture
7. Process maturity
8. CMMI
9. Risk taking

$$\mathbf{PC_{total} = PC_{initial} * [} \prod_{i=1}^{30} remaning\ therity\ project\ factors]$$

5.2 Flow chart for BCS

Flow chart for Binary Cuckoo Search (BCS) is shown in Figure 5.2 followed with explanation.

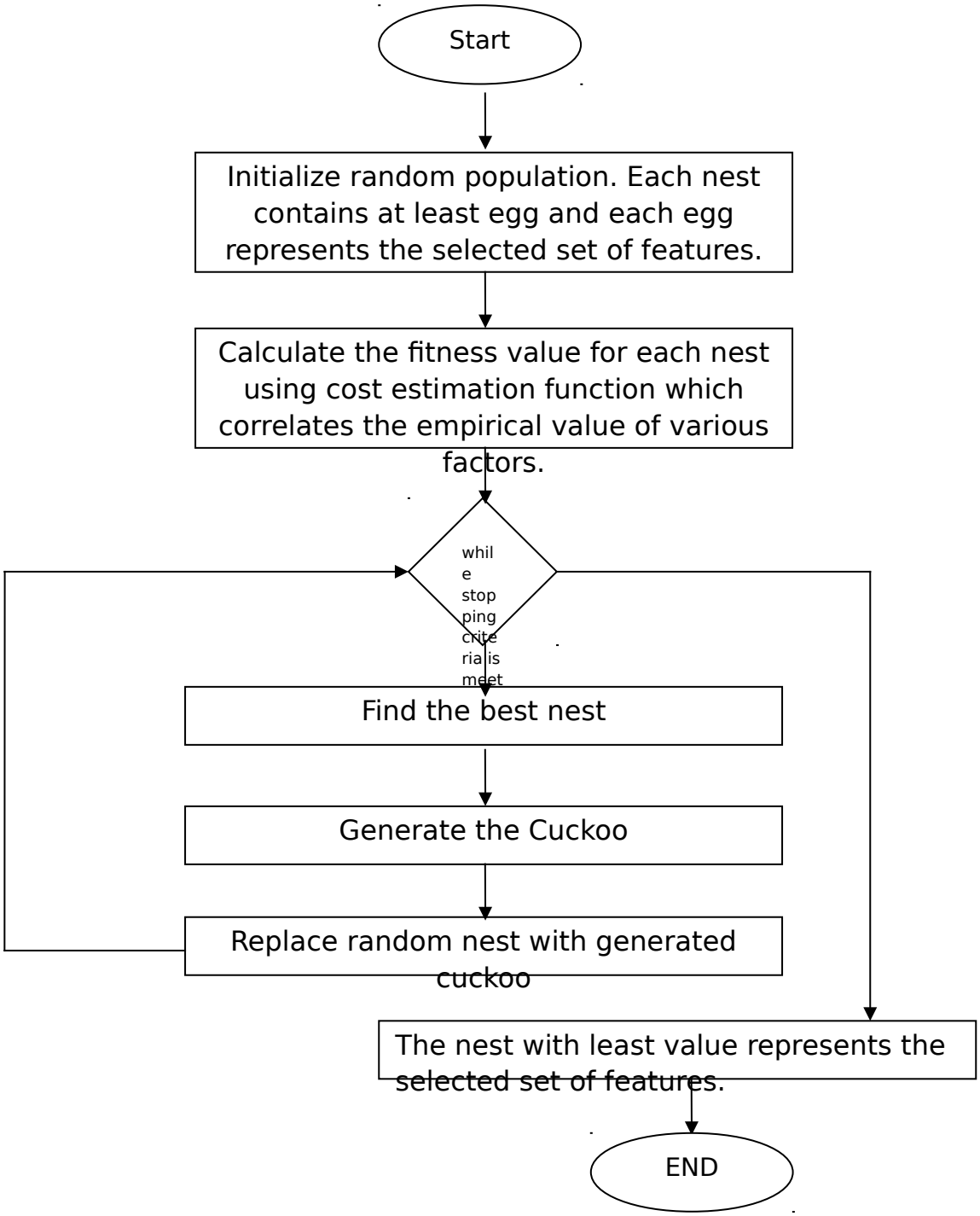

Figure 5.14 Flow chart for BCS

1. Initialize random population of nest. Each nest contains one egg and each egg represents the selected set of features.

2. Calculate the fitness value for each nest using cost estimation function that correlates the empirical value of various factors.

3. While stopping criteria (stopping criteria is the no of iterations) is met do the following:

a) Find the best nest which is based on the selected set of features using the fitness value.

b) Select a random nest to generate a new cuckoo.

c) Generate the new cuckoo using following equation which is explained in detail later.

$$xji(t) = xj\ i(t - 1) + \alpha \oplus Levy(\lambda)$$

d) Select random nest j which is not the fittest solution in the population and replaced the egg in this nest with newly generated egg.

4. The nest with least value represents the selected set of features.

5.3 The BCS Algorithm used in feature extraction

A binary version of the Cuckoo Search is called Binary Cuckoo Search (BCS). It is used for feature selection purpose. The search space for feature selection is modeled as a *d*-cube, where *d* stands for the number of attributes.

The main idea is to associate each nest to a set of binary coordinates that denote whether a attribute will belong to the final set of attribute or not, and the function to be maximized is the one given by COCOMO II , which we have explained earlier. The quality of the solution is related with the number of nests. If nest chosen are too small than the accuracy is low but if we select too large nest number then it take more time to execution. So in our case we have to choose 14 nests. So we need to evaluate each one of them by training a classifier with the selected features encoded by the egg's quality and also to classify an evaluating set.

We have a data set of 250 projects. The data set contains 40 cost attributes and the actual cost of projects. As we know a problem with *d* attributes, has 2^d possible solutions, which makes an

exhaustive search impracticable for high dimensional attribute spaces. In our sample data there are 40 project attributes, so we have 2^{40} possible solutions.

We have summarized BCS using following rules with our data set, as follows:

- Each cuckoo lays one egg at a time, one egg means an array with size 40 attributes. Each index of array is initialized with 0 or 1 randomly.
- This egg is dumps in a randomly chosen nest. The total number of nest is equal to the number of project attribute we want to extract out of 40. Here we want to extract 14 factors so our total nests are 14. One nest must contain at least one egg.
- The nests with the best of eggs (solutions) will move on to the next generations means the nest which have more factors which effect more on cost estimation process as compared to the other factor are selected.
- The nests with low cost effect factor are destroyed and new nest is build.
- Destroying the nest and building the new nest depends on the COCOMO II equation.

Algorithmically, each host nest n is defined as an agent which can contain a simple egg x (unique dimension problem) or more than one, when the problem concerns to multiple dimensions. CS starts by placing the nest population randomly in the search space. In each algorithm iteration (in our case max iteration is 100), the nests are updated using random walk via Levy flights:

$$xji(t) = xji(t - 1) + \alpha \oplus Levy(\lambda)$$

Here,

$Xji(t)$ stands for the jth egg (feature of 40 attributes) at nest i in current iteration t (in our case 14 because we extract 14 attribute out of 40 attributes), $i = 1,2, \ldots, m$ and $j = 1,2, \ldots, d$.

$xji(t - 1)$ stands for the jth egg (feature of 40 attributes) at nest i in previous iteration t-1

α is a randomly chosen value belong to 0 or 1 which decides whether large step is taken or not, if 1 then large step is taken

Large step is denoted by changing value of more than 10 features, exact no of which is chosen randomly. This large step denotes the levy flight in our feature selection problem.

Finally, the nests which have eggs with the lowest quality, are replaced to new ones according to a probability $pa \in [0,1]$.

At the end of final iteration (i.e.100^{th}) we got 14 attributes which effect project cost more as compared to others attributes.

The Cuckoo Search Algorithm used is as follows:

STEP 1: Each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest. Objective function f(x),x=($x1$, $x2$, … xu)T .Initialize image Generating initial population of 14 host nests xi Where,(i=1,2,…n).

STEP 2: The best nests with high quality of eggs will carry over to the next generation (select the high quality egg based on COCOMO II equation)

While (t<Max Generations) and (!termin.condit.) // termination condition or 100 iteration

Move a cuckoo randomly

Evaluate its fitness Fi // evaluation using levy flight equation

Randomly choose nest among n available nests

(for example j)

STEP3: The number of available hosts nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $\varrho\alpha\in$ (0,1).(collecting matching points and rearrange the points until satisfied fitness value)

If (Fi > Fj) Replace j by the new solution;

Fraction of worse nests is abandoned and new nests are being built; Keep the best solutions or nests with quality solutions; Rank the solutions and find the current best.

End while

Post process

The flow chart of the same is shown in the Figure 5.2.

5.4 Tools used

This section states about the tools that are used for implementing Binary Cuckoo Search and Artificial Neural Network:

MATLAB (**mat**rix **lab**oratory) is a numerical computing environment and fourth-generation programming language. Developed by Math Works, 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.

1. **Cuckoo optimization :** The parameters used for cuckoo optimization are as follows:

- numCuckooS = 14; % number of initial population
- minNumberOfEggs = 1; % minimum number of eggs for each cuckoo
- maxNumberOfEggs = 5; % maximum number of eggs for each cuckoo
- maxIter = 100; % maximum iterations of the Cuckoo Algorithm
- knnClusterNum = 1; % number of clusters that we want to make
- motionCoeff = 9; % Lambda variable in COA paper, default=2
- accuracy = -inf; % How much accuracy in answer is needed
- maxNumOfCuckoos = 10; % maximum number of cuckoos that can live at the same time
- radiusCoeff = 5; % Control parameter of egg laying
- cuckooPopVariance = 1e-13; % population variance that cuts the optimization

Figure 5.3 shows the data set given to the binary cuckoo search algorithm

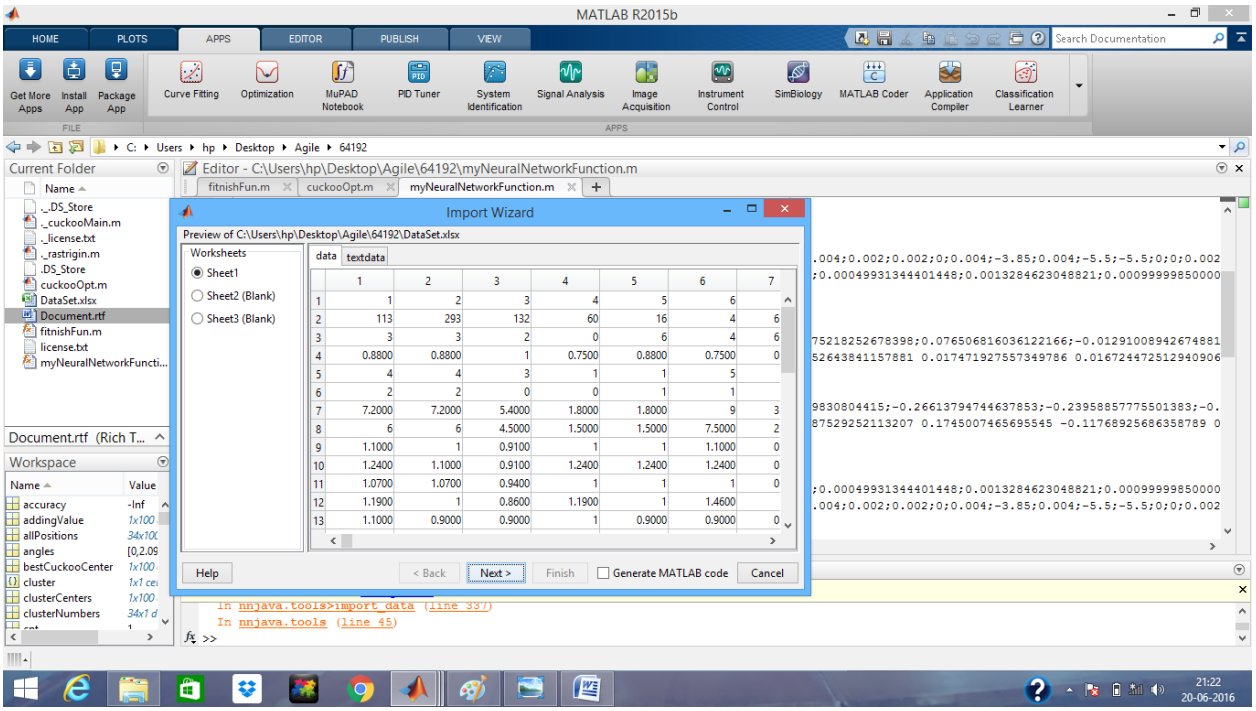

Figure 5.15 Data set to be given to cuckoo optimization

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

NF TOOL: 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 Back propagation 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.

5.5 Cost Estimation Process:

The following steps are used in cost estimation process:

Step 1: First of all, the Major Factors are identified using the MATLAB software through Optimization toolbox using binary Cuckoo optimization.

Step 2: From the processed training data set we obtain the various Factors which affects the cost estimation in agile projects.

Step 3: The factors extracted contains the Fitness value for each factor which explains the co-relation between the corresponding factor and the cost of development.

Table 5.1 shows the fitness value for each of the 40 factors.

Table 5.7 Fitness value of each factor

Major Factors	**Fitness Value**
Software size	0.71
Architecture	0.67
Risk Resolution	0.51
Process Maturity	0.72
Team Size	0.81
App Type	0.72
Required SW reliability	0.71
Product/project Complexity	0.62
Development Platform	0.7
Prog. Language and Toolset experience	0.62
OS	0.81
CMMI	0.70
Function Points	0.68
ISO	0.81
Managerial skills	0.42
Platform experience/Tech Ability	0.5
Hardware	0.40
Pages of Documents	0.37
Developed for Reusability	0.24
Development Flexibility	0.41
DB Size/SW size	0.44
IDE	0.24
Productivity	0.36
Org Type	0.48
Process Model	0.41
Training	0.35
Planning	0.47
Debugging Capability	0.44
Client/Server	0.33
Tool avlb	0.22
Feedback	0.25
Tool Familiarity	0.46
Team Prox.	0.13
Type of server	0.16

Package Customization	0.014
Ease of Use	0.39
Documentation Resources	0.16
Early Delivery	0.34
Documentation Period	0.28
Other Expenditure	0.19

From the above data we extract 14 factors which are having high fitness value which can be used in extimating the cost for various agile projects .First 14 factors have value greater than 0.5 so we can use them for estimating 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 Binary Cuckoo Search in Matlab are shown in Table 5.2.

Table 5.8 Extracted Agile Factors

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

Training the ANN

Step 1: use command nftool in the command line of Matlab is shown in Figure 5.4.

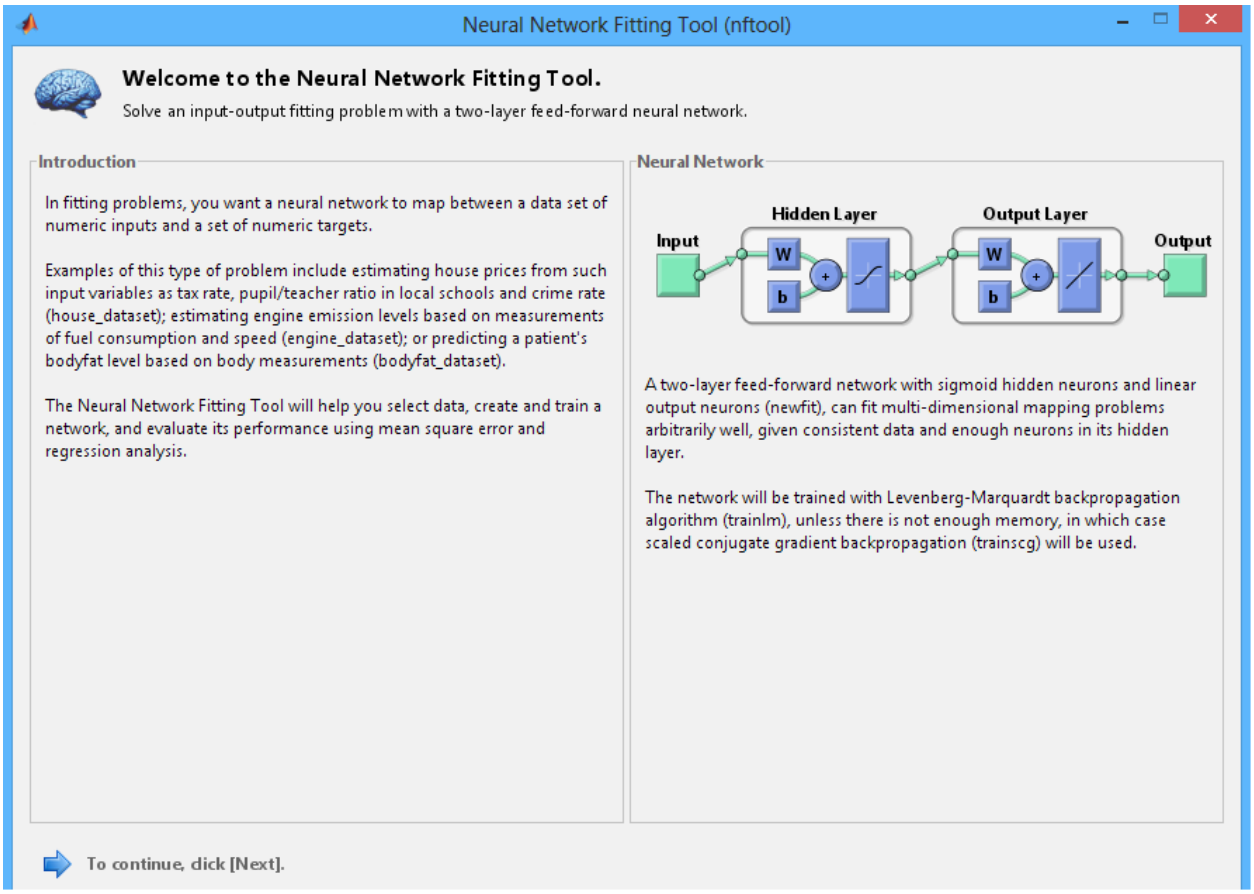

Figure 5.16 Nf tool

Step 2: Selecting input and output parameters for ANN tool is shown in Figure 5.5.

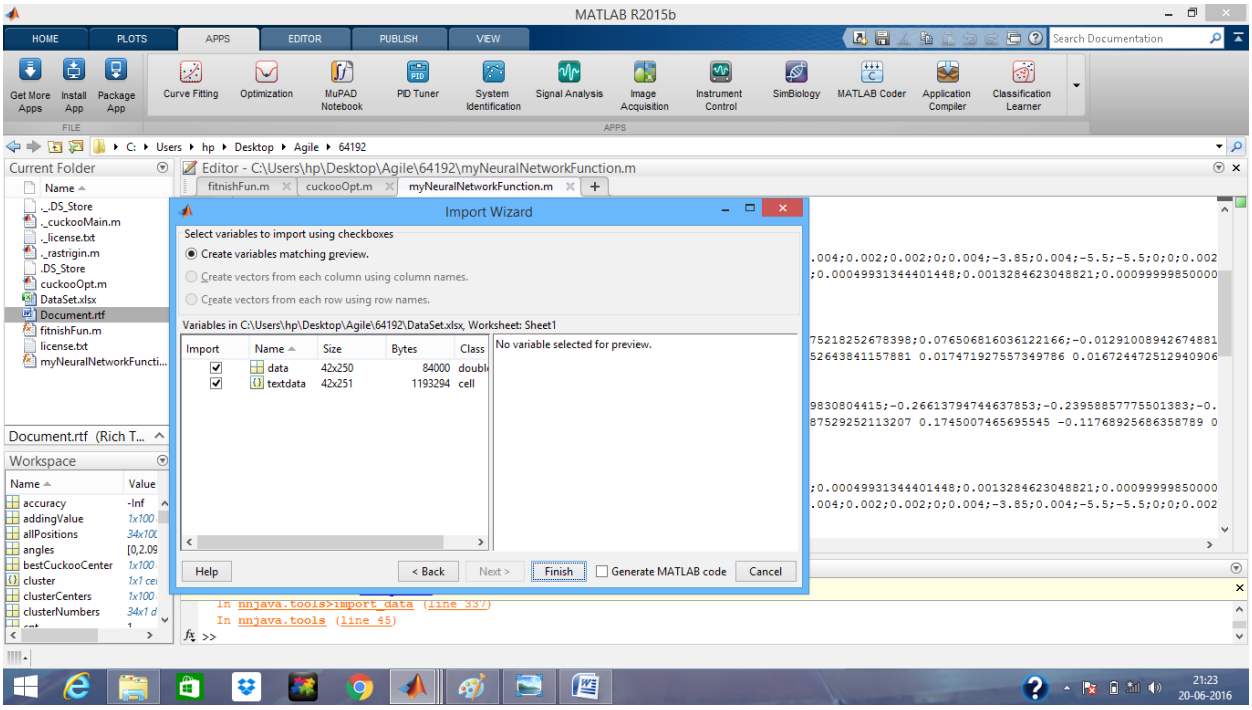

Figure 5.17 Giving Input to ANN

Step 3 Validation and testing data for ANN is shown in Figure 5.6.

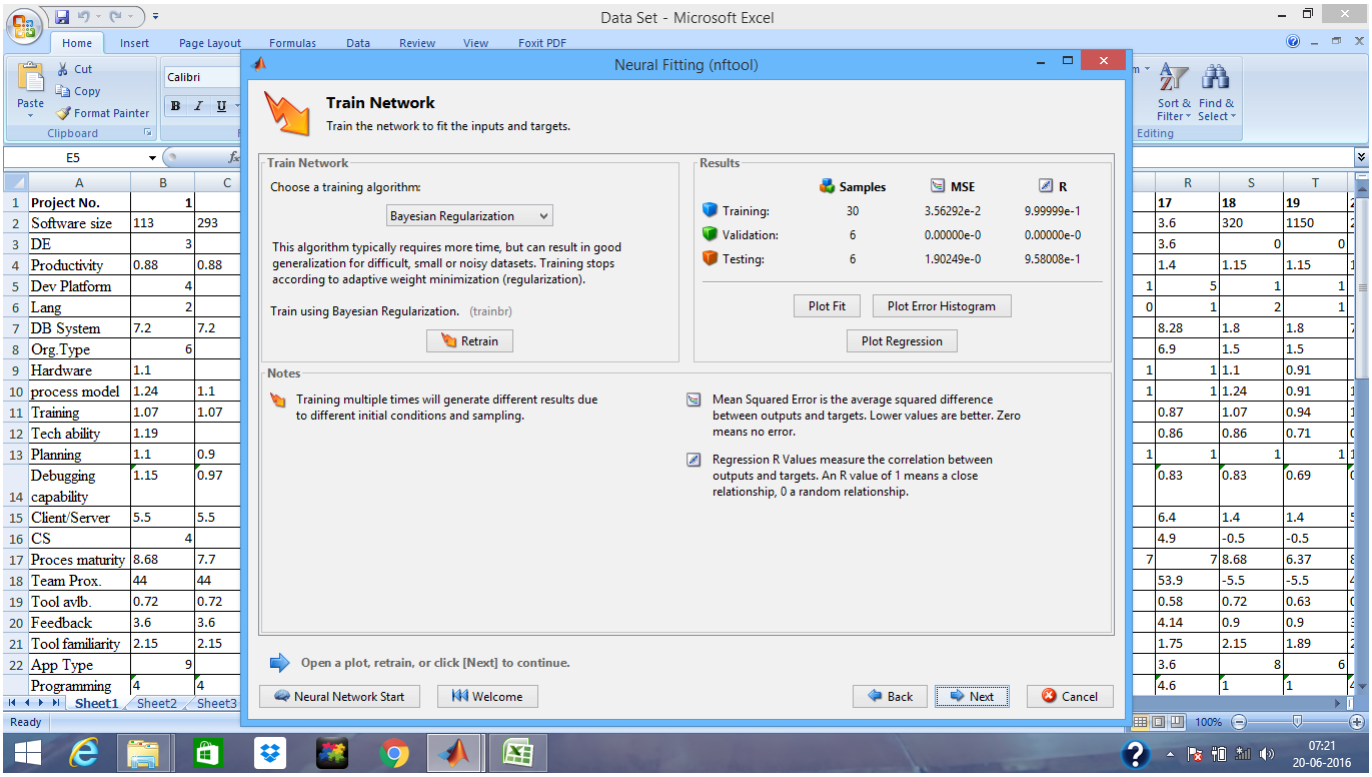

Figure 5.18 Data for training, Validation and Testing

Step 4: Layers in ANN network is shown in Figure 5.7.

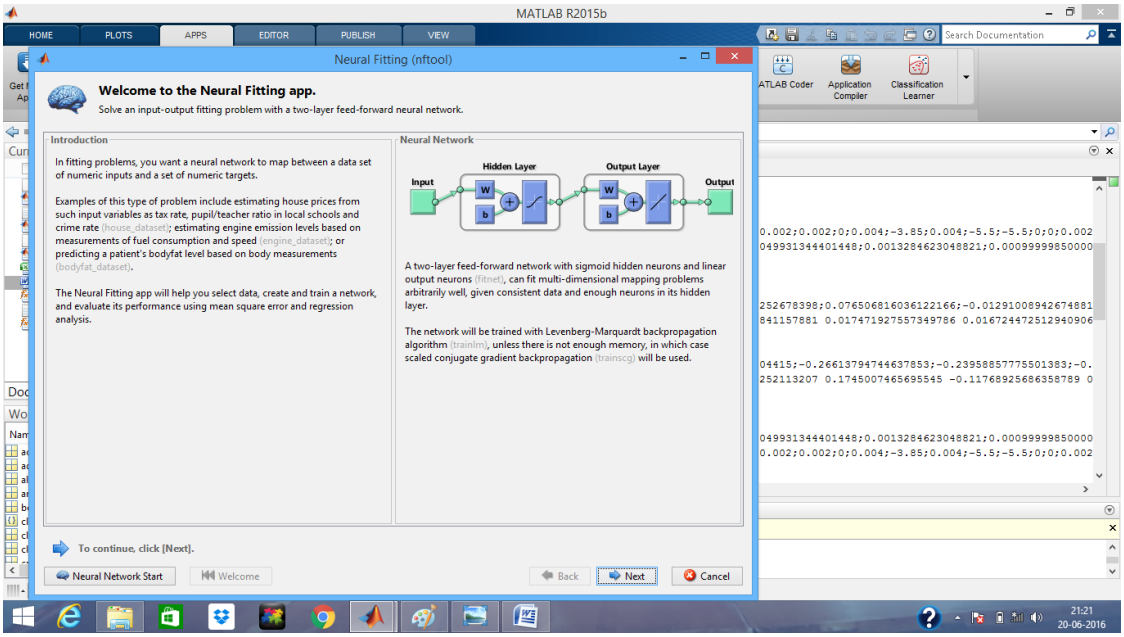

Figure 5.19 Using ANN network

Step 6: Output of training the ANN is shown in Figure 5.8.

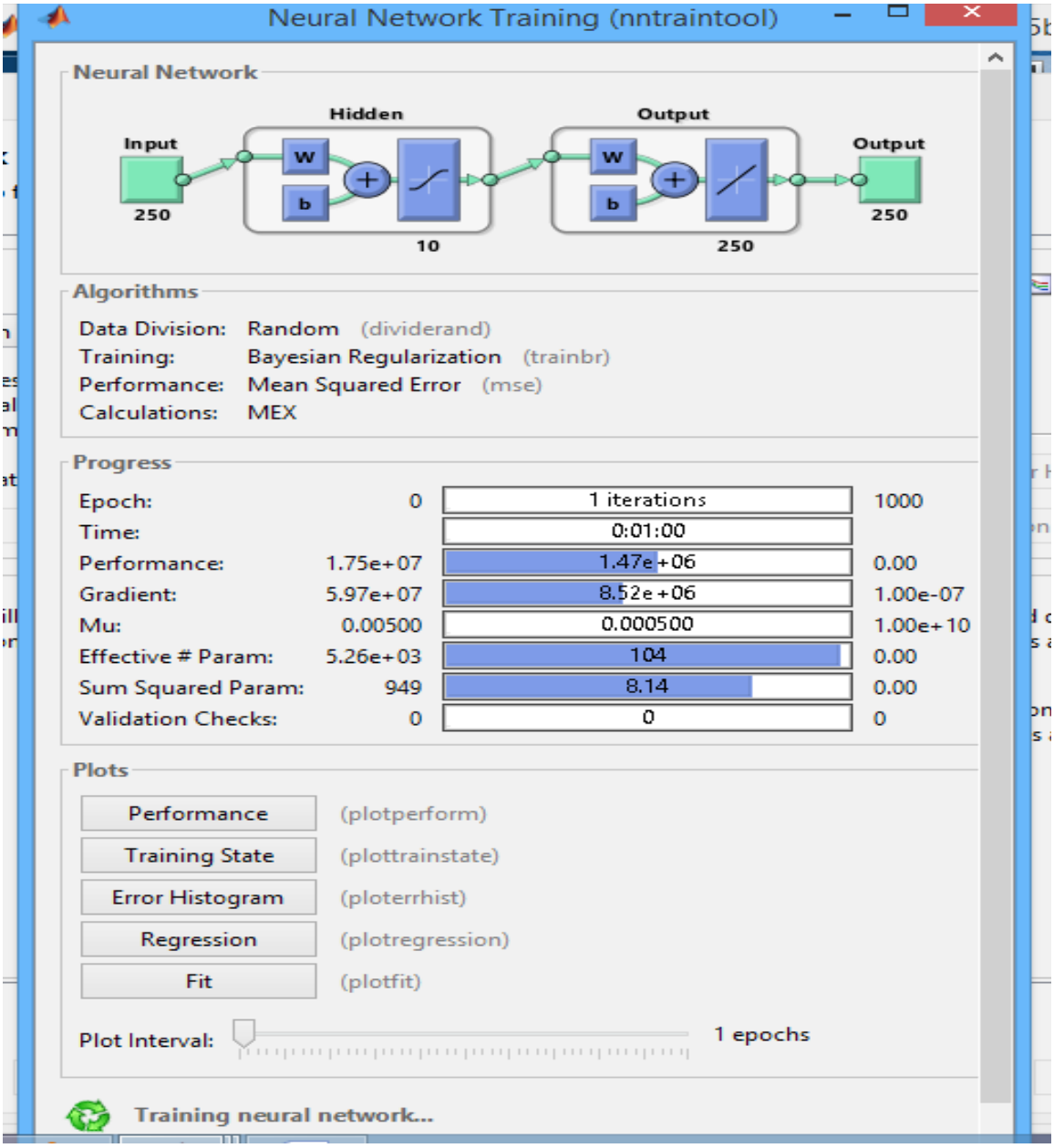

Figure 5.20 Results of ANN

Step 7: Saving the results and generating M file for the ANN is shown in Figure 5.9.

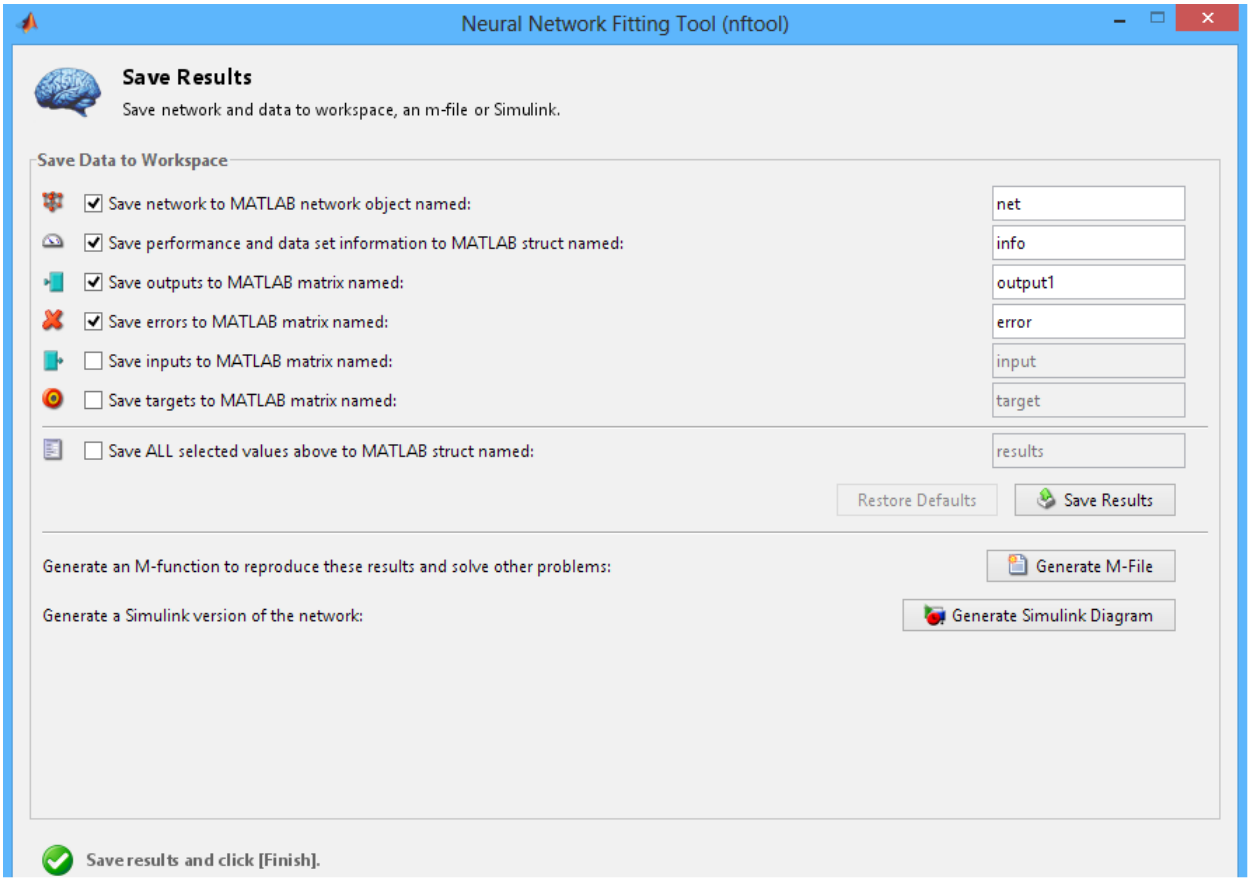

Figure 5.21 Saving results

Chapter 6

Results and Analysis

This section shows the results of above proposed methodology and the analysis of the result by comparing it with exiting PCA Based estimation model and SWOT based estimation model. The model is evaluated by finding the mean magnitude of relative error (MMRE)

6.1 Description of data used

In this thesis we have used a standard data set which is used in research in the domain of agile development methodologies. The two step proposed model for cost estimation make use of data-set of “Centre for Systems and Software Engineering” at the School of Engineering, University of South California. The dataset contains knowledge about software projects that are 'standardized, 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.

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. Data sample is shown in Table 6.1

Table 6.9 Some Project data from the data set

Project No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Software size	113	293	132	60	16	4	6.9	22

DE	3	3	2	0	6	4	6.9	2
Productivity	0.88	0.88	1	0.75	0.88	0.75	0.75	1.15
Dev Platform	4	4	3	1	1	5	2	3
Lang	2	2	0	0	1	1	1	1
DB System	7.2	7.2	5.4	1.8	1.8	9	3.42	5.4
Org.Type	6	6	4.5	1.5	1.5	7.5	2.85	4.5
Hardware	1.1	1	0.91	1	1	1.1	0.91	1.1
process model	1.24	1.1	0.91	1.24	1.24	1.24	0.91	1.1
Training	1.07	1.07	0.94	1	1	1	0.87	1
Tech ability	1.19	1	0.86	1.19	1	1.46	1	0.71
Planning	1.1	0.9	0.9	1	0.9	0.9	0.9	1.21
Debugging capability	1.15	0.97	0.83	1.15	0.97	1.42	0.97	0.69
Client/Server	5.5	5.5	4.2	1.4	1.4	6.9	2.6	4.2
CS	4	4	2.5	-0.5	-0.5	5.5	0.85	2.5
Proces maturity	8.68	7.7	6.37	8.68	8.68	8.68	6.37	7.7
Team Prox.	44	44	27.5	-5.5	-5.5	60.5	9.35	27.5
Tool avlb.	0.72	0.72	0.63	0.67	0.67	0.67	0.58	0.67
Feedback	3.6	3.6	2.7	0.9	0.9	4.5	1.71	2.7
Tool familiarity	2.15	2.15	1.89	2.01	2.01	2.01	1.75	2.01
App Type	9	7	2	8	3	4	6.9	9
Programming Language	4	4	3	1	1	5	1.9	3
OS	14.4	14.4	10.8	3.6	3.6	18	6.84	10.8
Team Size	66	66	50	17	17	83	31	50
CMMI	2	2	6	3	3	4	4	6
Architecture	3.5	1.5	2.5	2.5	3.5	4.5	1.4	3.5
SO	3.0	3.0	3.5	3.5	3.5	1.5	1.9	3.5
Type of Server	3	3	2	1	1	3	1	2
Package Customization	1.17	2.17	3.17	4.17	5.17	6.17	7.17	8.17
Project complexity	0.95	0.74	0.21	0.84	0.32	0.42	0.72	0.95
Function points	1130	2930	1320	600	160	40	69	220
Reliability	0.88	0.88	1	0.75	0.88	0.75	0.75	1.15
Pages of documents	4500	3500	1000	4000	1500	2000	3450	4500
Risk taking	-0.001	-0.001	0.000	-0.003	-0.001	-0.003	-0.003	0.002

Managerial skills	30	30	23	8	8	38	14	23
Ease of use	0.37	0.37	0.20	0.28	0.28	0.28	0.11	0.28
Documentation resources	108	84	24	96	36	48	82.8	108
Early delivery	1	1	1	1	1	1	1	1.66
Documentation period	6.4	4.9	1.4	5.6	2.1	2.8	4.9	6.4
Other Expenditure	14.83	11.63	1.77	1.74	0.24	0.31	0.58	7.82
Actual Cost	2040	1600	243	240	33	43	80	1075

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.

6.2 Problems of Over Estimation and Under Estimation

This section discusses the problems in over and under estimation in a project.

6.2.1 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.

6.2.2 Problems with under-estimation

1. **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.

2. **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.

3. **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.

6.3 Binary Cuckoo Optimization

Binary Cuckoo Search optimization is used with different iterations the output of which is shown in this section. Cuckoo iteration plot for first 8 iterations is shown in Figure 6.1 and after iteration 72 is shown in Figure 6.2.

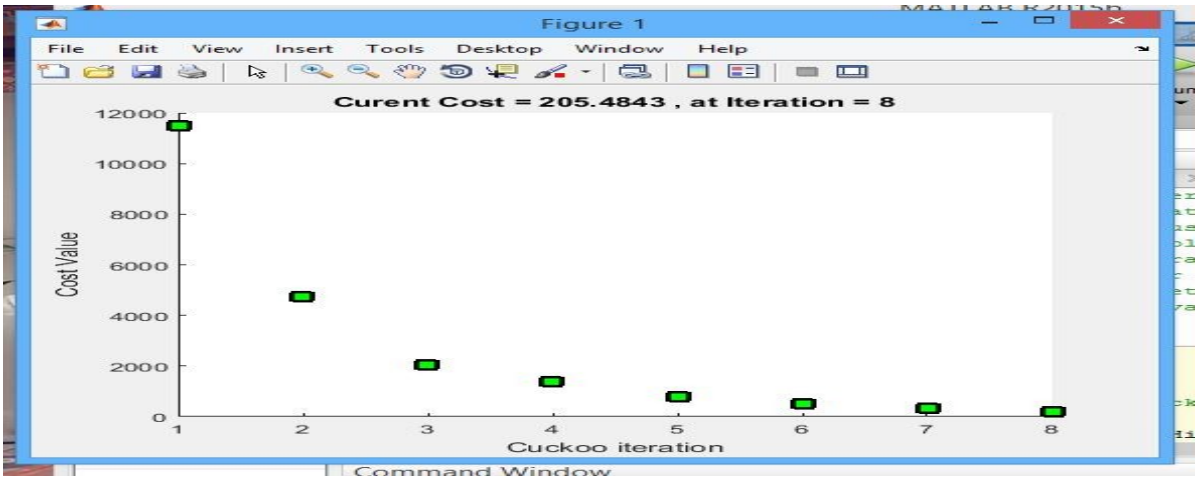

Figure 6.22 Cuckoo iteration plot

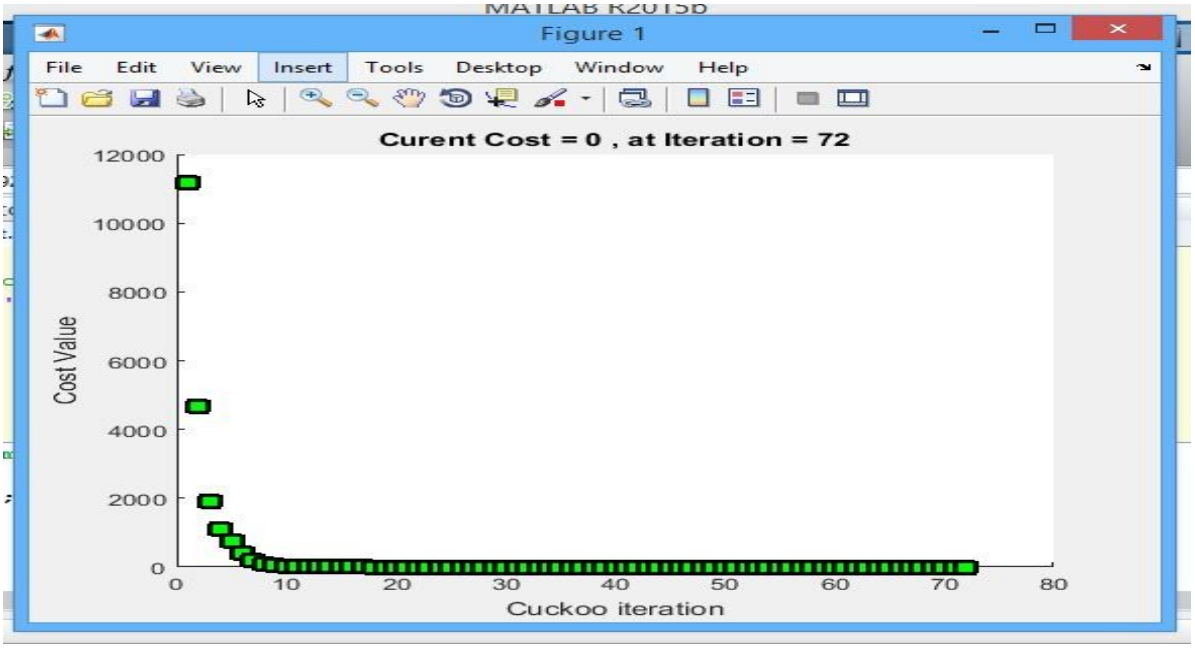

Figure 6.23 Cuckoo after 72th iteration

6.4 Over-estimation vs. Under-estimation

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.

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.

6.5 Analysis using a Data set

The data set which has been used for cuckoo optimization and 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.

6.5.1 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 with the help of Binary Cuckoo Search taken as Input to Artificial Neural Network.

6.5.2 Evaluation Criteria

The evaluation criteria for the proposed methodology is regression values and MMRE (Mean magnitude of relative error), the ideal values of the regression is close to one (more closer the value to one, more accurate results) and zero of the MMRE (more closer the value to zero, more accurate result). Different error measurements have been used by various researchers. I have chosen the mean relative Error (MRE) and regression as the major measurement tool:

MRE: MRE is calculated as follows:

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

$$MRE_i = abs(RE_i)$$

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

6.6 Results

As shown in figure 6.3, it shows the number of sample used for training, testing and validation.

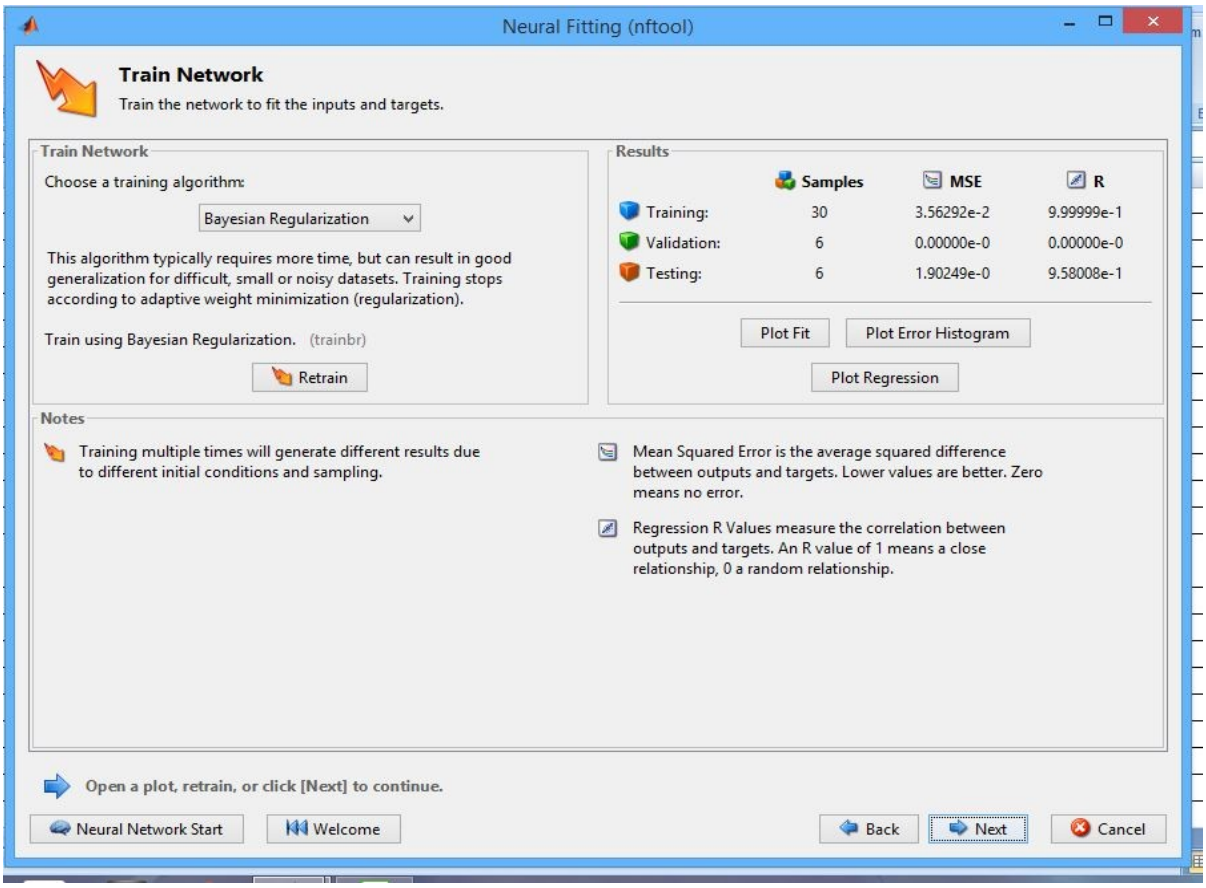

Figure 6.24 Number of samples used for training, validation and testing

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 30 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. For hidden layer 15 regression curve for training is shown in Figure 6.4.

Training:

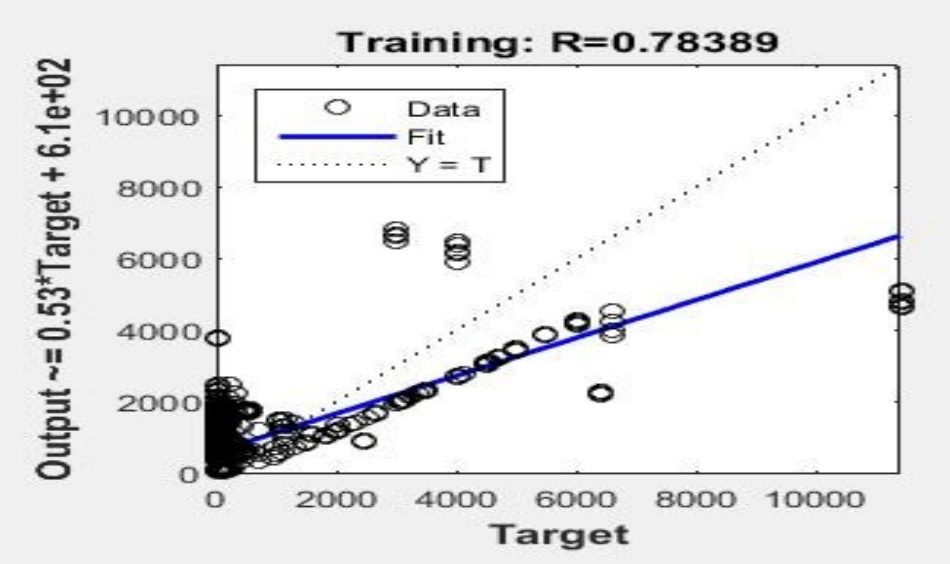

Figure 6.25 Regression curve of training

For hidden layer 15 regression curve for validation is shown in Figure 6.5

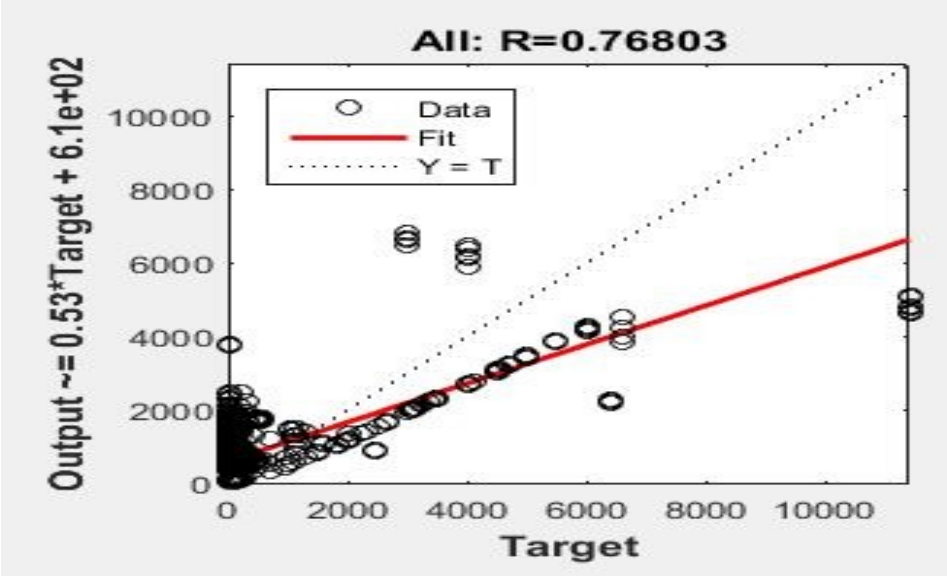

Figure 6.26 Regression Curve of Validation

For hidden layer 15 regression testing curve is shown in Figure 6.6.

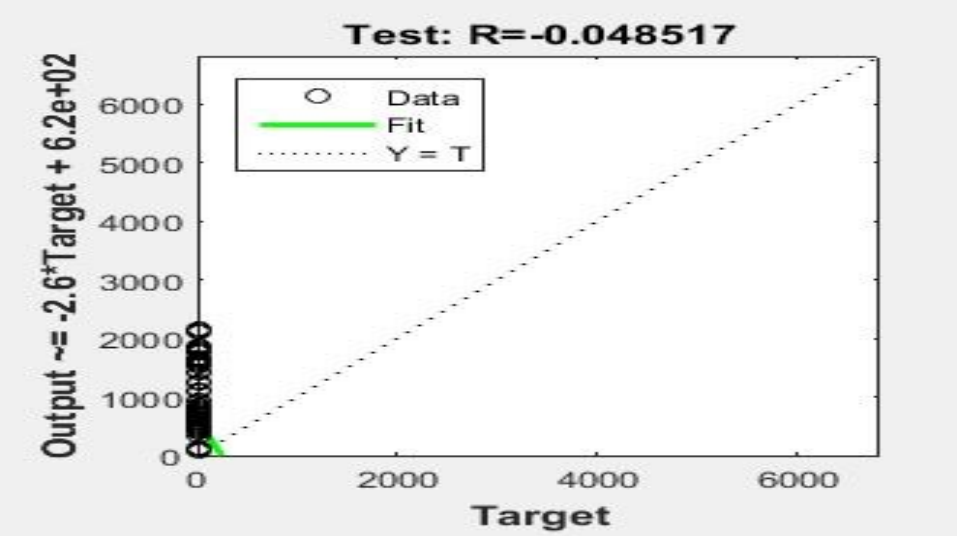

Figure 6.27 Regression Curve of Testing

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

For hidden layer 10 regression curve for training is shown in Figure 6.7.

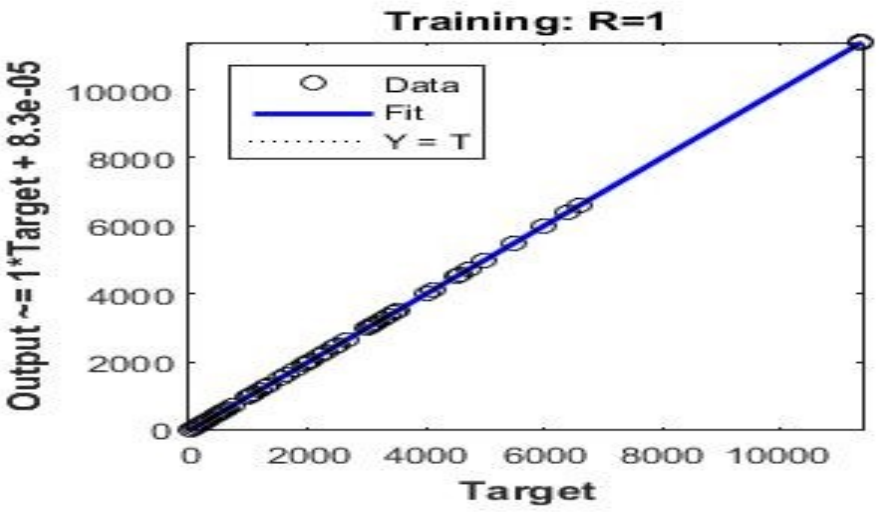

Figure 6.28 Regression curve for training

For hidden layer 10 regression curve for testing is shown in Figure 6.8

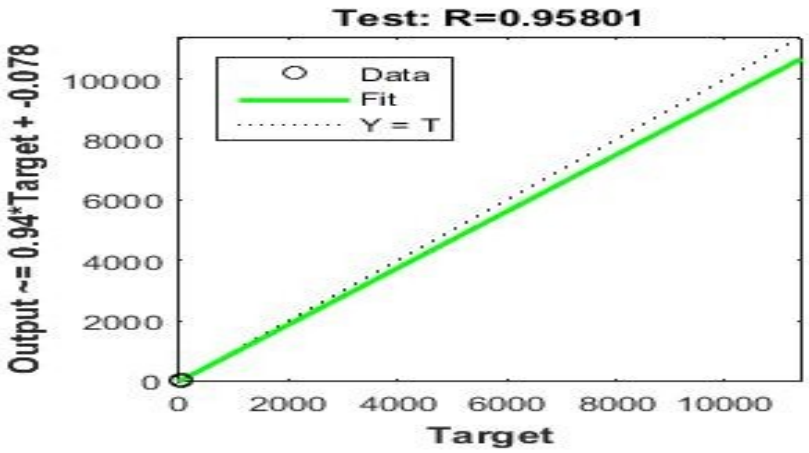

Figure 6.29 Regression Curve for testing

Combined regression curve is shown in figure 6.9.

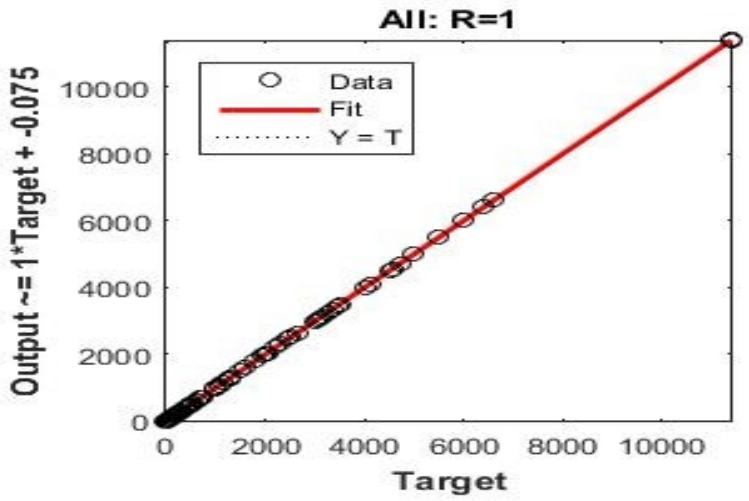

Figure 6.30 Regression curve for all

We can easily identify that the better values of regression are in figure 6.9 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 we can say that it has been shown that on the basis of 250 Projects data sets that the ANN can also be very efficient for cost estimation of various projects which are developing through agile techniques as well.

6.7 Results can be stated as follows:

- **Decreased MMRE 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 MMRE 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 on the features based on Binary Cuckoo Search optimization 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.

- **Comparison with Previous Existing Estimation models**

  We find that our model provide low MMRE value 0.01, which is marginally lower than the previous models as proposed in [8] that is having value 0.03 and [14] having value 0.05 There for we can say our model has increased level of accuracy because of lower MMRE value.

Chapter 7
Conclusion

This thesis proposed Binary Cuckoo Search optimization 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 facts to cuckoo optimization which can affect the cost required to develop software's. These factors were extracted using Binary Cuckoo Search in Matlab by doing dimension reductions 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 model was trained using 175 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 MRE 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 MRE values in estimation:** This technique has shown a decent lower MRE value so one can expect to achieve a good accuracy while estimating effort.
- **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 ANN bye extracting major factor using Binary Cuckoo Search and thereafter can be useful to predict the effort and cost of new projects.
- **Future work** we can use Binary Bat Algorithm(BBA),Binary firefly Algorithm (BFFA), binary particle Swarm Optimization (BPSO) and compare the result with Binary Cuckoo Search Algorithm.

References

[1] Humphrey, Watts S. A discipline for software engineering. Addison-Wesley Longman Publishing Co., Inc., 1995.

[2] Maruping, Likoebe M., ViswanathVenkatesh, and Ritu Agarwal. "A control theory perspective on agile methodology use and changing user requirements." *Information Systems Research* 20.3 (2009): 377-399.
[3] Boehm, B., Turner, R., 2003. Using risk to balance agile and plan-driven methods. Computer 36 (6), 57–66.
[4] Reel, J.S., 1999. Critical success factors in software projects. IEEE Software 16 (3), 18–23.
[5] Haugen, N., an Empirical Study of Using Planning Poker for User Story Estimation, Proceedings of AGILE 2006 Conference, 2006.
[6] DeMarco, T., and Lister, T., (1987). *Peopleware: Productive Projects and Teams*, Dorset House Publishing Co., NY.
[7] Arpit Sanghavi, "Feasibility of ANN in Agile Methodologies for Effort Estimation", M. Tech Thesis, Computer Science & Engineering Department, Delhi Technological University, Delhi (2013)
[8] Swati Batra, " Agile Project Estimation by Optimizing Various factors ", M. Tech Thesis, Computer Science & Engineering Department, Delhi Technological University, Delhi (2015)
[9] .Beecham, S., Baddo, N., Hall, T., Robinson, H., and Sharp, H., (2008). Motivation in Software Engineering: A Systematic LiteratureReview. *Information and Software Technology*, (50:9-10): 860-878.
[10] Whitworth, E., and Biddle, R., Motivation and Cohesion in Agile Teams, *Proceedings of the 8th International Conference, XP 2007*, Como, Italy, June 18-22, 2007, pp. 62-69.
[11] Maruping, L. M., Venkatesh, V., and Agarwal, R., (2009). A Control Theory Perspective on Agile Methodology Use and Changing UserRequirements. *Information Systems Research*, (20:3): 377-399.
[12] F. Song, Z. Guo and D. Mei, "Feature Selection Using Principal Component Analysis," International conference on System Science, Engineering Design and Manufacturing Informatization (ICSEM), Vol 1, pp. 27-30, Changchun, China, 2010
[13] D. Rodrigues, L. A. M. Pereira, T. N. S. Almeida, J. P. Papa, A. N. Souza, C. C. O. Ramos, and X. S. Yang, "A binary cuckoo search algorithm for feature selection," in Proceedings of IEEE International Symposium on Circuits and Systems, 2013, pp. 465-468.
[14] Garg, Sakshi, and Daya Gupta. "PCA based cost estimation model for agile software development projects." Industrial Engineering and Operations Management (IEOM), 2015 International Conference on. IEEE, 2015.

[15] Cohn, M., Ford, D., 2003. Introducing an agile process to an organization.Computer 36 (6), 74–78.
[16] Sutherland, Jeff, et al. "The scrum papers: Nuts, bolts, and origins of an agile process." (2007).
[17] Lindvall, Mikael, et al. "Empirical findings in agile methods." *Extreme Programming and Agile Methods—XP/Agile Universe 2002*. Springer Berlin Heidelberg, 2002. 197-207.
[18] Agile Modeling Home Page. Effective Practices for Modeling and Documentation. [Online] Retrieved 17th March 2016. Available at: www.agilemodeling.com.
[19] 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.
[20] Caruana, R., Niculescu-Mizil, A., An Empirical Comparison of Supervised Learning Algorithms, Proceedings of the 23rd International Conference on Machine Learning, 2006.
[21] M. E. Porter, Competitive Advantage: Creating andSustaining Superior Performance, Free Press, NY, 1998.
[22] R. J. Offen and R. Jeffery, "Establishing Software Measurement Programs," IEEE Software, vol. 14, 2, pp. 45-53,1997.
[23] Hareton Leung, Zhang Fan, “Software Cost estimation”,Hong Kong Polytechnic University, 2002.
[24] C. Ravindranath Pandian, Software Metrics A Guide toplanning, Analysis and Application, India, 2004.
[25] A. Albrecht, “Measuring Application Development Productivity”, in Proceedings of Joint SHARE/GUIDE/IBMApplication Development Symposium, October 1979.
[26] Desharnais, J.-M.; St-Pierre, D.; Maya, M.; Abran, A., “Full Function Points: Counting Practices Manual - Rules and Procedures”, Montréal ,Université du Québec à Montréal , 1997.
[27] 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).
[28] Roger S. Pressman, “Software Engineering, A Practitioner’s Approach” Sixth Edition, McGraw-Hill, NY,2005.
[29] M. JORGENSENA, “Review of Studies on Expert Estimation of Software Development Effort”, Journal of Systems and software,70(1-2):37–60, 2004.

[30] Caruana, R., Niculescu-Mizil, A., An Empirical Comparison of Supervised Learning Algorithms, Proceedings of the 23rd International Conference on Machine Learning, 2006.
[31] Caruana, R., Niculescu-Mizil, A., An Empirical Comparison of Supervised Learning Algorithms, Proceedings of the 23rd International Conference on Machine Learning, 2006
[32] Yang, X-S. and Deb, S. (2009) 'Binary Cuckoo Searchvia Levy flights', Proc. of World Congress on Nature & Biologically Inspired Computing (NaBIC), India, IEEE Computer Society,pp.210–214.
[33] Rajabioun, R. 2011. Cuckoo optimization algorithm. Applied Soft Computing 11(8):5508–5518.

Appendix -1

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

INDEX

LIST OF FIGURES

LIST OF TABLES