Prediction of Suitability of Agile Software Development Methodology using Machine Learning

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

Master of Technology

In

Software Technology

By

Aditya Kumar Goyal

(Roll No: 2K15/SWE/03)

Under the guidance of

Prof. (Dr.) DAYA GUPTA

DEPARTMENT OF SOFTWARE TECHONOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
NEW DELHI

<u>DECLARATION</u>

==

I hereby declare that the thesis entitled, "**Prediction of Suitability of Agile Software Development Methodology using Machine Learning**", is a bona fide work done by me in partial fulfillment of requirements for the award of Master of Technology Degree in software technology at Delhi Technological University (New Delhi) is an authentic work carried out by her under my supervision and guidance. The matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma to the best of my knowledge.

____

Aditya Kumar Goyal

Department of Software Engineering

Delhi Technological University,

Delhi.

CERTIFICATE

==

DELHI TECHNOLOGICAL UNIVERSITY

BAWANA ROAD, 110042

Date: ____________

This is to certify that the thesis entitled, "**Prediction of Agile Suitability of Software Development Methodology using Machine Learning**", a bona fide work done by **Ms. Aditya Kumar Goyal (2k15/SWE/03)** in partial fulfillment of requirements for the award of Master of Technology Degree in software technology at Delhi Technological University (New Delhi), is an authentic work carried out by him under my supervision and guidance. The matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma to the best of my knowledge.

Prof (Dr.) Daya Gupta,

Department of Software Engineering,

Delhi Technological University, Delhi-110042

ACKNOWLEDGMENT

==

I would like to take this opportunity to express my gratitude and appreciation to all those who have helped me in any way towards the successful completion of this work.

I take this opportunity to thank my supervisor, **Prof. Daya Gupta**, for guiding me and providing me with all the facilities, which paved way to the successful completion of this work. This thesis work was enabled and sustained by her vision and ideas. Her scholarly guidance and invaluable suggestions motivated me to complete my thesis work successfully.

I have given sufficient time and research to complete my project under timeline defined by the university. This project has given me opportunity to explore the domain of Software Engineering.

Aditya Kumar Goyal

2K13/SWE/03

ABSTRACT

A Large numbers of software projects fail during their development phases due to high reliance of inappropriate software development methods. It is not advisable to start with a randomly chosen software development methodology for successful completion of a project within budget and target time. All the development methodologies whether belongs to agile or non-agile domain have their merits and demerits. Traditional plan-based software development methods works extremely well if the requirements are static whereas for frequently changing project requirements these methodologies are often considered as slow and insensitive. Agile methods on the other hand are considered as light-weight methods that don’t produce requirement and design documentation needs but requires intensive communication between the developers and users. Here, we present a complete framework, Development Method Selection Framework (DMSF) that provides an overall context in terms of software project parameters for exploration of project-in-hand and selection of software development method.

Contents

Chapter 1

1 Introduction

1.1Introduction & Motivation

For a strong house, we need a strong foundation. Similarly we need strong software development methodology to develop a strong and reliable software product which meets the needs of stakeholders. To ensure the durability and reliability of software, appropriate methodology is required.

From the very beginning, many non-agile methods have been used for software development. Developers used ER Diagrams to show detailed entity relationship model of an organization.

The main components of the ER diagrams were data entity, relationships and their associated attributes. To show the flow in the Information system, Process flow diagrams were used.

For visualization, specifying and documenting object oriented system UML provides modeling language. These approaches take significant time and cause overhead in the software development even in the situations where they are not required. Major time is spent on documenting rather than actually implementing the system.

Due to the drawbacks of the heavyweight traditional methods, agile methods are welcomed in the industry by both the developers and the managers. Agile methods focus on quick delivery of the software to the customer and modifying the software as per the requirements of the customer. There is a need of the constant communication among the development teams and the customer to provide feedback for the system so that it can be quickly adapted to new requirements and satisfy the customer needs. Many Agile methodologies are used in the industry such as DSDM (Data System Development Method), XP (Extreme Programming), RAD (Rapid Application Development), SCRUM, FDD (Feature Driven Development) etc. Refer to Chapter 2 for the overview of these methodologies. Large Software like Windows, Google Drive cannot be developed using Agile Methods as there is proper documentation required for the system being developed.

Due to these complicacies there is a need to tool that helps to determine appropriate software development method for the project in hand based on the characteristics of the projects. The value of the project characteristics varies according to projects like for mobile application technical risk is less, but for Aircraft system technical risk is more.

In this project, we assign parameters to assign weights to the characteristics and their value based on the project. Depending upon the value of the parameter our framework finds out the most suitable agile method for the project development. Since process of evaluation is complex, machine learning technique has been used for the evaluation process.

1.2 Related Work

Methods make the task of Software development easy and efficient. Due to which there is a need to adopt a suitable Software In response to this, BrinkKemper has defined Method Engineering as "*discipline to design, construct and adapt methods, techniques and tools for an Information System Domain (ISD) project*" development method. This necessity makes the task method engineering more important. Widely accepted the definition of Brink Kemper has made method engineering popular.

Main focus of the method engineering community is to select method based on the programmer's capability, size of the project, requirement of the project, complexity of the project etc. There are numerous proposals in the literature to develop method suitable for a particular project such as method assembly (Fragment-Based Approach, GOPRR Based Approach) or method generation(Contextual Approach for method generation, Engineering Using Rules), method architecture(Intension-Architecture based approach (MIA), Architecture-Centric Method Engineering Approach (ArCME))etc. Recently method engineering has moved to method configuration where a base method is configured to meet project requirements and method configuration has also been defined by Dwivedi [22, 3, 23].

(IEEE Std 610.12, 1990) defines method configuration as "The arrangement of a computer system or component, defined by the number, nature, and interconnections of its constituent parts".

In configurability, first we create a new model called a configurable model followed by selecting only those elements of the configurable model that are relevant to the user's requirement.

(Coplien et al., 98) define **commonality** as an assumption held commonly across a given set of objects whereas **variability** is an assumption that is true for only some elements of the set.

(Weiss and Lai) defines the variability **as** an assumption about “How members of a family may differ from one another: A configurable model identifies commonality and variability that can be exploited in developing a new system from the configurable model.

(Davenport) describes the process of configuration as a methodology performed to allow a business to balance their IT functionality with the requirements of their business.

Some proposals have been found that talks about configurability in Situation Method Engineering domain but none of them reach at its maturity level and got success to give a consistent and coherent solution. Rolland et al. [20] proposed method design as method families, these method families are additionally surfaced to form a method line that eventually brings about an configured method the proposition is in the outset stage and neglects to give the definite procedure of designed strategy development. These proposition address many issues, however issues like 'right granularity', 'fittingness of strategy being chosen' and 'explicit portrayal of changeability' are as yet open.

Method Configuration is a sub-discipline to SME and has been figured and proposed via Carlson et al in [22]. They characterized method configuration task as a "way to adjust a specific technique to different arranged variables. The attention is therefore on one method as a construct for configuration as opposed to with respect to configuration of strategies as a base for assembly". Further, in their later recommendations, they [47] characterized high-level configurable construct for their method configuration process. The configurable idea proposed by them did not have the issues of liberality and granularity. Since the past proposition on method configuration focused on the one single base method decreased the extent of the method generation for each new project.

Dwivedi also provide framework for method configuration of Agile as well as traditional methods [3, 22, 23].

Earlier approaches to method engineering focused mainly on non-agile configuration since from ages only non-agile methods were used. With the popularity and need of Agile Software Development methods, there is a necessity of Tool that provide framework, Meta Concepts and Configuration Environment for Agile Methods and provide the solution based on agile characteristics of the projects. Five metrics provided by Qumer and hendersonsellers for agile method to be well defined are: Flexibility, Speed, Leanness, Learning and Responsiveness. They have further provided proposal to find the agility in a project.

The development methodologies whether agile or non-agile have their merits and demerits. As agile methods have their own limitation of less documentation, large cooperation from client and limited, dedicated programmer they are not applicable to all project. In a recent research, Dwivedi has stated that large numbers of software projects fail due to the high reliance on inappropriate Software development paradigm.

So there is a need to draw some criteria that assist the software developers to select appropriate software development paradigm for the current project. Dwivedi also gives the framework to find software paradigm based on project characteristics [24]. Further in her work she has presented a generic framework for method engineering that include method configuration for agile paradigm. In this framework preprocessing is decision about agile methodology selection[24] but due to the some limitation of the proposed framework like the once the complexity of fuzzification and defuzzification, validity of fuzzy rules, it will produce the same output irrespective of changing organizational requirements or behavior.

Our solution for providing the method selection framework uses neural network, where the project database (weights of connecting lines) is enhanced every time the new query case arrives thus providing much more experience for the upcoming projects. Further we are configuring the selected agile method based on the project in hand.

For configuration Dwivedi[25] has proposed commonality and variability in an agile method. Here commonality needs to be always there but variability may or may not be kept in the final configured method.

Our solution for selecting the variability to be kept is by defining fuzzy rules based on project characteristic. Project requirements are taken from customer and variability of agile method can be computed.

The configuration is completed by taking common and variable functionalities of a particular agile method. We are proposing a set of fuzzy rules for choosing the variable functionalities of an agile method.

1.3 Problem statement

Software development methodology can be divided into two paradigm namely plan driven methodology and agile methodology.

Depending on project characteristics such as characteristic of requirement, characteristic of development team, user participation and project type & associated risk suitable paradigm is chosen. For example:

-If user participation is low, plan driven methodology is suggested

-If development emphasis is high then agile methodology is suggested

-If requirements are volatile then agile methodology is suggested

A decision system support (Using fuzzy logic[9], Cuckoo search[10]) has been developed by the authors (Rinki, Priyanka) to select a suitable paradigm.

If agile paradigm is selected, Rinki[3] has proposed organizational requirements such as Task extent, Group size, Progress approach, Code style, expertise environment, Physical environment, Industry customs and Abstraction mechanism. Based on fuzzy rules Rinki has developed a method selection framework as follows:

1. Gather organizational requirements such as Task extent, Group size, Progress approach, Code style, expertise environment, Physical environment, Industry customs and Abstraction mechanism.

2. Apply fuzzy rules to select suitable base method from method base, which consists of popular agile methods.

3. Finally based on project characteristics such as technology used, task extent, team experience, physical environment, documented requirements and progress approach.

Since neural network outperforms fuzzy rules, thus neural network can be applied for selecting suitable method from method base of popular agile methodologies.

Since an agile methodology consists of many practices which may not be usable for all kind of projects. Intel Shannon, IBM, Nokia has been tailoring the base method based on particular project characteristics. Fuzzy rules help in selecting applicable practices based on project requirements.

Problem statement for this thesis is as below:

Develop a Decision support system to select a suitable base method from method base of popular agile methods and develop fuzzy rules to choose usable practices based on the project charactoristics.

1.4 Scope

Method base consists different agile method like Extreme programming, Scrum, Feature driven development, Dynamic system development model, Adaptive development model and Crystal.

First task is to identify the project characteristics like task extent, group size, code style and many other and study their impact on the agile methodology selection. Based on their impact for current project in hand, agile method is selected from method base of agile method.

Since these weights and their value can vary depending upon the project in hand, we have taken a large database of projects and apply **neural network** to find the suitable method for the project in hand. Database (weights of neurons) store the previous projects experience and output the result for project in hand.

Neural network is used to process the data in hand. Learning is initial phase, in this phase the data is used to update weights of neural network. For learning feed-forward methodology is used as shown in the algorithm.

Post assigning the weights to neural network, there comes a back-propagation phase in which error calculation is done and it is back propagated to update the previously assigned weights by the previous dataset.

After the successful completion of the learning phase, there comes the suggestion phase. In this phase requirements are gathered for the project in hand. These requirements are given as input in the neural network and output weights are calculated. The methodology with highest weight is chosen.

Post suggesting a suitable agile method, practice selection framework is proposed using fuzzy rules to configure project specific method. Project requirements are gathered after selecting a suitable agile method for configuring the same for particular project.

Hence scope of the work can be summarized as:

- Identification of organizational requirements for agile method selection.

- For the selected method commonality and variability are defined and weights are assigned to them.
- A set of fuzzy rules for base process has been developed in this thesis.
- Work out case studies taken from Industry visit and previous

1.5 Organization of the thesis

The structure of the thesis in terms of the contents of its various chapters is as follows.

Chapter 2: This chapter provides the overview of Agile Development methodology along with different Agile Methods Available and the characteristics of Agile Methods.

Chapter 3: This chapter presents the work proposed by rinki for paradigm selection and showing comparison between cuckoo search and neural networks

Chapter 4: This chapter presents the work done in the project to accomplish the task of agile method selection. Algorithm, Implementation Control flow and Output are analyzed.

Chapter 5: Implementation and Case Studies are discussed and their outputs.

Chapter 6: This chapter is for the results and conclusion of the thesis.

Chapter 2

2 Agile Modeling

agile– giving "the nature of being agile; preparation for movement; agility, action, skill in movement" as specified in the Oxford Dictionary – are the techniques to offer give an answer for the business searching for quicker and nimbler advancement process. All the agile techniques recognize that amazing programming and consumer loyalty must be accomplished by lightweight strategies as it were. The absolute most ordinarily utilized techniques are explained beneath.

2.1 Extreme Programming

Extreme programming advanced because of issues of the conventional development models. The XP procedure is portrayed by short advancement cycles, incremental planning, consistent feedback and evolutionary plan. Due to these qualities, XP programmers respond to change with much efficiency. Term “Extreme” comes from the applying following practices to extreme level.

• Planning – The developer evaluates the effort for implementation and client choose scope and timing of release in light of estimate.

• Small/short releases – Small and frequently release are provided. New version can be released monthly or daily.

• Metaphor – There is a set of metaphor between customer and developer that defines how the system will work.

• Simple Design – Simple solution is built with unnecessary code and complexity removed.

• Refactoring – It is the rebuilding of framework by evacuating duplication, enhancing correspondence, disentangling and including flexibility bet without changing usefulness of program.

• Pair programming – Production code composed by two developers on same system.

• Collective ownership – No particular individual is in charge of any code segment. Anybody can change any part.

• Continuous Integration – Newly developed code is integrated and tested as soon as it is integrated.

• 40-hour week –A maximum of 40-hour working week for the developers.

• On-site customer – Customer must be accessible constantly with the advancement group.

• Coding Standards – Coding rules exist and are trailed by the software engineers to bring consistency and enhance correspondence among the development group. The lifecycle of an XP project appeared in Figure 2.1, is isolated into six stages: Exploration, Planning, Iterations to discharge, Production, Maintenance and Death. In the Exploration phase, the customer writes out the story cards to be included in their program. Then comes the Planning phase where a priority order is set to each user story and a schedule for the first release is developed. From iteration to release phase, development team develop and integrate their code. In production phase, extra testing and performance is done. Ideas and suggestions found at this phase are documented for later implementation in the updated releases made at the Maintenance phase. The Death phase is when user has no story and there are no changes to be made to system.

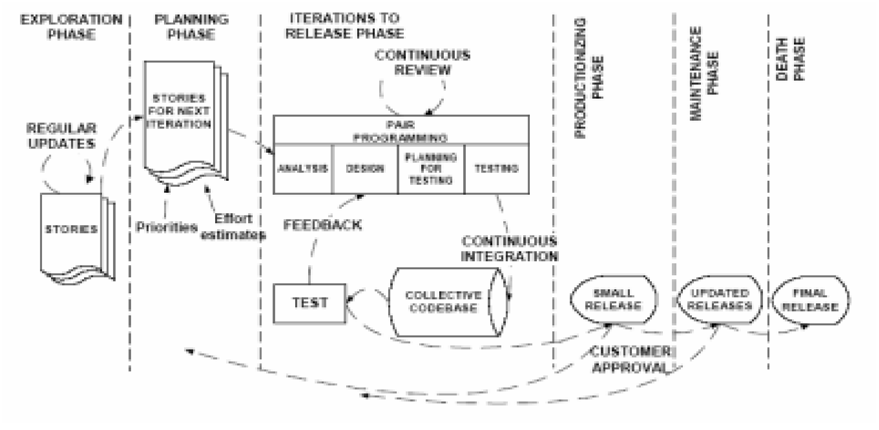


Figure 2.1: Lifecycle of the XP process

2.2 Scrum

Scrum is an iterative and incremental process for developing system in continuously changing environment. Every iteration produces set of functionality. The term “scrum” is derived from the strategy of the rugby game of “getting an out-of-play ball back into the play”. Scrum does not define any particular method to be used but it requires certain management practices and tools to avoid chaos by unpredictability and complexity. Key practices of Scrum are as follows:

• Product Backlog

• Sprints

• Sprint Planning meeting

• Sprint Backlog

• Daily Scrum

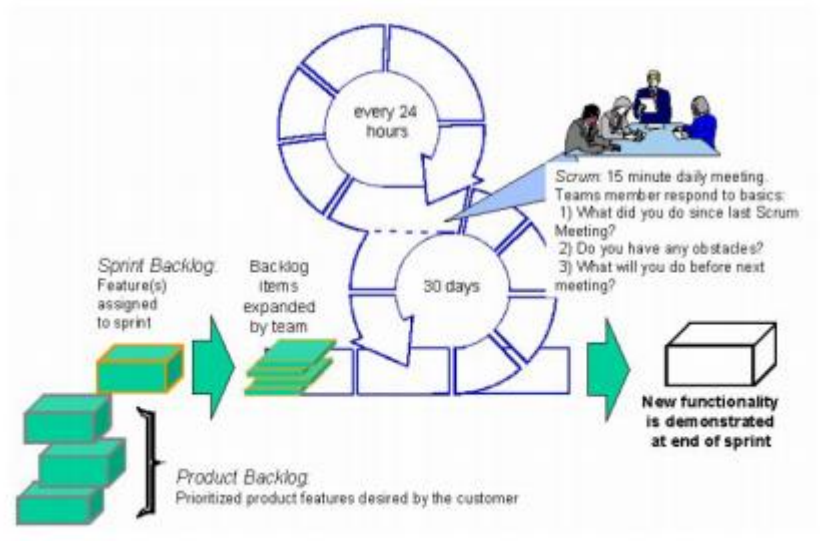

every 24 hours
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
30 days
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

Figure 2.2 : Scrum Process

2.3 Feature Driven Development (FDD)

FDD was first used for the development of a large and complex banking application project in the late 90's.Unlike the other methodologies, the FDD approach only focuses on the design and building phases.

The first three phases are done at the beginning of the project and the last two phases are the iterative part which supports the agile development with quick adaptations to late changes in requirements and business needs. This approach includes frequent deliverables, along with accurate monitoring of the progress of the report. FDD consists of five sequential steps (Figure 2.3), an explanation of the different roles and responsibilities if given below:

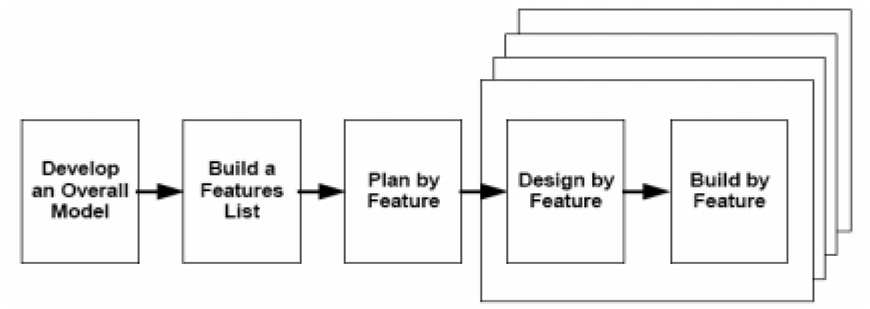


Figure 2.3: Feature Driven Development processes

• Develop an Overall

• Build a Features

• Plan by Feature

• Design by Feature & Build by Feature

2.4 Dynamic System Development Method

The Dynamic System Development Method was developed in the United Kingdom in the mid-1990. Martin Fowler, one of the writers of Agile Manifesto, believes, "DSDM is notable for having much of the infrastructure of more mature traditional methodologies, while following the principles of the agile methods approach".

The fundamental idea is to fix time and resources, and then adjust the amount of functionality accordingly rather than fixing the amount of functionality and then adjusting time and resources to reach that functionality. DSDM consists of five phases (Figure 2.4):

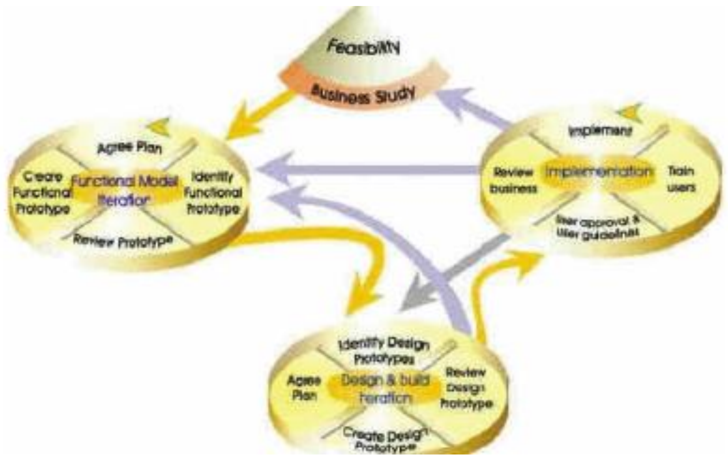


Figure 2.4: DSDM process diagram

- Feasibility study
- Business Study
- Functional Model Iteration
- Design and Build Iteration
- Implementation

2.5 Adaptive Software Development (ASD)

ASD offers an agile and adaptive approach to high-speed and high-change software projects. It becomes difficult to plan successfully in a fast moving and unpredictable business environment. In ASD, instead of static plan-design life cycle is there is dynamic speculate-collaborate-learn life cycle. ASD focal point is on three non-linear and overlapping phases (Figure 8).

- Speculate
- Collaborate
- Learn

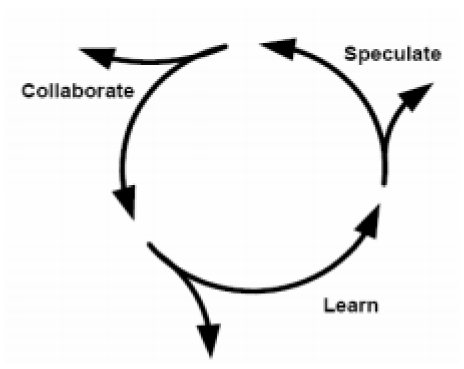


Figure 2.5:SD Lifecycle

2.6 Agile Manifesto

In February 2001, seventeen representatives from the different agile methods decided to form an Agile Alliance to promote their views which gave birth to Agile 'Software Development' Manifesto. Agile techniques have already been used by developers but after the manifesto the techniques have been grouped together in workable framework.

The values defined in the manifesto are as below:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

The 12 principles of the Agile Software development made by the Agile Manifesto:

• Highest priority is to satisfy the customer through early and continuous delivery of valuable software.

• Address dynamic requirements, which come even late in development.

• Deliver working software frequently, from a couple of weeks to a couple of months, in a short timescale.

• Users and developers must work together throughout development.

• Project developed among motivated individuals. Provide them the environment and support they need, and trust them to get the job done.

• To efficiently and effectively convey information to and within a development team focus on face-to-face conversation.

• Provide working software instead of prototype or design.

• The sponsors, developers, and users should be able to maintain a constant pace in software development indefinitely.

• Continuous attention to technical excellence and good design enhances agility.

• The best architectures, requirements, and designs emerge from self-organizing teams.

• At regular intervals, the team reviews its performance and work on improving the efficiency.

2.7 Characteristics of Agile Methodologies

According to Highsmith and Cockburn , “what is new about agile methods is not the practices they use, but their recognition of people as the primary drivers of project success, coupled with an intense focus on effectiveness and maneuverability. This yields a new combination of values and principles that define an agile world view.” Highsmith further describes the definition of agility: “Agility... is a comprehensive response to the business challenges of profiting from rapidly changing, continually fragmenting, global markets for high quality, high-performance, customer-configured goods and services.”

The following characteristics of agile methodologies are seen as the main differences between agile and traditional methods:

People Oriented- Agile methods consider people – customers, developers, stakeholders, and end users – as the most important factor of software methodologies. If the people on the project are good enough, they can use almost any process and accomplish their assignment. If they are not good enough, no process will repair their inadequacy

Adaptive – The developer are not afraid of change. They welcome changes to the requirements because they mean that the team has learned more about the market needs. Today the challenge is how to better handle changes that occur throughout a project.

Conformance to Actual – Agile methodologies value conformance to the actual results rather that conformance to the detailed plan. As Highsmith states, “Agile projects are not controlled by conformance to plan but by conformance to the business value". Each iteration adds business value to the ongoing product.

Simplicity – Agile teams always prefers the simplest to achieve their goals, so that it will be easy to change the design if needed on a later date. Never produce unnecessary and never produce documents attempting to predict the future as documents will become outdated.

Collaboration –The customer of the software works in collaboration with the development team, providing frequent feedback on their efforts. Due to its decentralized approach, there is a need of discussion to get the feedback on the software provided by the development teams.

Small Self-organizing teams – Agile team is a self-organizing. Responsibilities are communicated to the team and the team finds out the best way to fulfill them. Agile teams discuss and communicate together on all aspects of the project, that's why agility works well in small teams. As mentioned by Alistair Cockburn and Jim Highsmith, "Agile development is more difficult with larger teams. The average project has only nine people, within the reach of most basic agile processes. Nevertheless, it is interesting to occasionally find successful agile projects with 120 or even 250 people"

2.8 commonality and variability practices of agile methods

An agile method is not always perfectly suitable for a project. Some changes can be made in the agile practice for making it suitable for the particular project.

For defining the changes Rinky[25] has proposed commonality and variability of every agile method, which defines the common and variable practices of an agile method.

Here common practices have to be there in the final configured practice but the variable components can be further analyzed for their existence in the configured method.

The common and variable practices can be defined in the table below:

Here C: common

V: variable

Agile Values	Responding to change over following a plan		Customer Collaboration over contract negotiation		Working Software over comprehensive documentation		Individuals and Interactions over processes and tools	
XP	1. Metaphor 2.Simple Design 3.Refactoring 4. Coding standard	C V C C	1.The Planning Game 2.On-Site Customer	C V	1.Testing 2.Short releases 3.Continuous Integration	C V V	1.Pair Programming 2.Collective Ownership 3.On-Site Customer 4.The planning game	C V V C
Scrum	1. Sprint Planning meeting 2. Sprint Review 3. Sprint Retrospective 4. Scrum of Scrums	C V V V	1.Sprint planning meeting 2 .Product Backlog	C V	1. Sprint 2.Sprint Review	C V	1.Scrum Teams 2.Daily Scrum Meeting 3.Sprint Planning meeting	C V V
FDD	1. Domain Object Modelling. 2.Configuration Management	C V	1.Domain Object Modelling	C	1.Developing By Feature 2.Inspection 3.Regular Builds 4.Reporting/Visibility of results	C V V V	1. Domain Object Modelling. 2.Individual Class Ownership 3.Feature Teams 4.Inspection	C V V V
ASD	1. Adaptive Cycle Planning 2.Customer Focus group reviews	C C	1.Adaptive Management Model 2.Joint Application Development	C V	1.Developing by Components 2.Software Inspection 3.Project Post mortem	C V V	1.Adaptive Management Model 2.Collaborative teams 3.Joint Application Development by independent agents 4.Customer Focus Group reviews	C V V V
DSDM	1. Reversible Changes	C	1.Collaboration and Cooperation among stakeholders 2.Requirements are baseline at a high level	C V	1.Frequent Product Delivery 2.Iterative and Incremental development 3.Integrated testing	C V V	1. Empowered Teams. 2.Active User Involvement	C V
Crystal	1. Reflection workshops 2.Methodology Tuning	C V	1.Staging 2.User Viewings	C V	1. Monitoring of a progress. 2.Revision and Review	C V	1.Holistic Diversity and Strategy 2.Parallelism and Flux 3.User Viewings	C V V

Table 2.1: Common and Variable practices

Chapter 3

Paradigm selection

3.1 Paradigm selection

Software development methodologies can be divided into two paradigm namely plan driven methodology and agile methodology.

Plan driven (or traditional) methodologies consists a series of steps namely requirement definition, planning, building, testing and deployment. First the customer elaborates the requirements and the requirements are documented to the full extent. Then project development starts based on these requirements. Then comes various testing and deployment. Basic idea of plan driven methodology is to visualize the finished project prior the development starts.

Agile methodologies have already been described in detail in chapter 2.

Depending on project characteristics such as characteristic of requirement, characteristic of development team, user participation and project type & associated risk suitable paradigm is chosen. For example:

-If team is experienced then plan driven methodology is suggested

-If requirements are volatile then agile methodology is suggested

-If same type of project have been developed then plan driven methodology is suggested

-If associated risk is high then agile methodology is suggested

A decision system support (Using fuzzy logic[9], Cuckoo search[10]) has been developed by the authors (Rinki, Priyanka) to select a suitable paradigm.

Since cuckoo search outperforms neural network, hence cuckoo search is used. Cuckoo search is based on case base reasoning which is explained below:

3.1.1 Case Based Reasoning

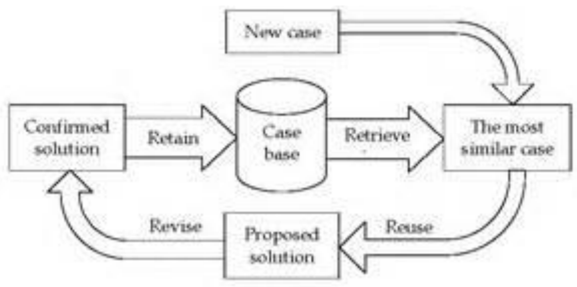
Case-based reasoning (**CBR**) is used to find solution of problems using the experience of same type of problems faced in the past. A database is maintained of all previous problems and their respective solutions. Using this database a learning process is performed which collects the data and classifies it on the basis of problems. For a new problem it finds the best class and suggest a solution. For example a doctor prescribe a disease based on symptoms that occurred in previous patients.

It has been contended that case-based reasoning is a capable technique for computer reasoning, as well as a typical behavior in regular human problem solving. We can state that all thinking depends on past data present by experience. This view is identified with a model hypothesis, which is most profoundly investigated in psychological science.

3.1.1.1Process

Case-based reasoning can be defined by a flow chart. The flow chart contains four steps used in for computer reasoning.

1. **Retrieve:** For a given problem, fine the problems same as the target problem from database. This group of problems can be used to suggest an appropriate solution by finding respective solution of these problems.
2. **Reuse:** Using the solutions from the database, this step finds the appropriate solution for the problem in hand. This step combines the solution of all matching problems to form the best solution.
3. **Revise:** After finding the best solution using '**Reuse**', this solution is tested on the given problem. If it doesn't fit to the problem find the next solution using previous step.
4. **Retain:** After the solution has been successfully tested on the problem and it is giving the desired result, then the database is updated with a set of new problem and its solution.



6 Figure 3.1:BR Process

3.2 Case Bases in Databases

Databases refer to the collection of data that is used in the proposed method selection tool. The data collected is the past experience of the projects developed using the methodology, Agile, Traditional or Hybrid (1, 2 or 3 respectively). The data is collected from different organizations, developers, managers or the organizations. In the block diagram it refers to the past information required in the tool to find out the method for the current project.

The data consist of values of project characteristics. The project characteristics are identified based on below mentioned categories:

1. **Characteristics of Requirements:** Project requirements can be complex to work on. Requirements can be known in the beginning, volatile in nature, complex, misunderstood.
2. **Characteristics of Development team: -** Method selection can depend on development team experience, hands on technology, communication with each other and skills of the team, whether training is provided to the team or the scope of training.
3. **User's participation: -** How the user can communicate with the developers and provide feedback on the software being developed plays an important role in method selection. It provides clarity, completeness of requirement.
4. **Project type and associated risk: -** Project type and risks involved play an important role in method selection.Technical Risk, Operational Risk , Business Risk characteristics cover this need.
5. **Based on Agile Configuration:** Since there are 4 manifestos in Agile Software Development, their characteristics are very important to analyze if agile has to be followed or not. Communication among developer and user, time to market and team size are some of the characteristics taken in account to address this need.

3.2.1 Weight Distribution and Input values for paradigm selection

According to the importance of the characteristic to the project in hand, the characteristics are assigned weight and input values for different projects.

The project characteristics are mentioned below, to get better understanding the weight distribution criteria are also explained. The numerical value assigned to the weights ranges between 0.0 - 0.1.The total weights of characteristics for the projects should be 1.

Project Characteristic 1: Volatility of requirements

This characteristic can be handled in Agile methods as they known for the volatility of requirement as we saw in chapter 2. Low weight is assigned to it as this is addressed in Agile Method (0.02).Values can be assigned as follows:

*2**Table 3.1**: Values for "volatility of requirement"*

% of volatile known requirement	**Category**
< 10%	Very Low
10 to 19%	Low
20 to 29%	Medium
30 to 39%	High
> 39%	Very High

Project Characteristic 2: Complexity

More the complex project less the chances of using agile configuration, hence it is given high weightage (0.12).Values can be assigned as follows:

*3**Table 3.2**: Values for "Complexity"*

Initial Time	**Category**
< one week	Very Low
< 15 days	Low
< one month	Medium

< six months	High
> six months	Very High

Project Characteristic 3: Business Risk

Business risk is associated with customer satisfaction, being on time, company image and returns on investment. If customer is not satisfied, organization should be able to provide new version immediately, also there is competition in the market, hence company should be able to provide issue fixes, support at any time. Hence this characteristic is given less weight of 0.03.

Business risk can be influenced by various factors, including government regulations, sales volume, per-unit price, input costs, overall economic climate and competition. So, the value of this metric should be chosen by considering all the above-said factors.

Project Characteristic 4: Technical Risk

Technical risk involves tool failure, non-availability of developer, non-availability of Tool in the development phase. This property is addressed in Traditional Method, hence the weight is more 0.08 for this.

Technical Risk can range from system failure, software malfunction, virus that can destroy the software. If company is shifting from one platform to another, it is not necessary that it can work better. There can be a possibility that is worse from the previous one. These points should be kept in mind and hence assign the value of the characteristic.

Project Characteristic 5: Operational Risk

Operational Risk involve the failure of funtionality of some or any component of the system. This is a very major issue and this type of systems need to be developed using traditional method in a systematic and properly defined way, hence assigned more weight(0.1).

Th value of this characteristic should be chosen depending upon the impact of failure or any component on the system.This can vary from project to project. Project having high impact of failure should

Project Characteristic 6: Flexibility

Flexibility refers to the ease with which a system can be modified. Agile is suitable for the systems that are flexible to provide easy and quick delivery of the product at each iteration, hence less weight for this (0.02).

4*Table 3.3*: *Value for "Flexibility".*

Ease of modification	Category
< 5%	Very High
5 to 9%	High
10 to 14%	Medium
15 to 19%	Low
> 20%	Very Low

Project Characteristic 7: Modularization of Task

Modularization is very essential for quick and secure software development. It will be very easy to develop the modules in parallel for quick release if the task is divided into modules. Agile methodology is suitable for development of modular code. Results very less weight i.e. 0.03 for this. Values can be assigned as follows:

5*Table 3.4*: *Metrics for "Modularization of task".*

Extent to which the project can be made modular	Value
Complete project	Very Low
More than half functionality	Low
Half of the functionality	Medium
Less than half	High
Very minimum amount of modules	Very High

Project Characteristic 8: Time to Market

This characteristic signifies the time (in months) before which at least first phase (least functionality) of the product must be released. Agile methodology delivers in very short sprints to the user. Results very less weight i.e. 0.02 for this. Values can be assigned as follows:

6***Table 3.5***: *Value for "Time to market"*

Time before the first release	**Time to market**
2 months	Very Low
4 months	Low
6 months	Medium
8 months	High
Greater than 8 months	Very High

Project Characteristic 9: Clarity and Completeness of requirement

This characteristics defines how well defined, clearly visible requirements are and require minimum further analysis. These types of requirements can be addressed by both traditional and agile. Results in a medium weight i.e. 0.05 for this. Values can be as follows:

7***Table 3.6***: *Value for "clarity and completeness of requirements"*

Amount of complete and consistent Requirements	**Category**
< 20%	Very Low
20 to 39%	Low
40 to 59%	Medium
60 to 79%	High
> 79%	Very High

Project Characteristic 10: Coupling

Coupling defines the degree of dependency between functionalities. More coupling, more is the complexity and hence more value to it supports for traditional methodology. Results very high weight i.e. 0.1 for this. Values can be assigned as follows:

NCM: Number of Couples modules.

*8**Table 3.7**: Value for "coupling"*

Value for NCM	Modularization of Task
< 2	Very Low
2 to 3	Low
4 to 5	Medium
6 to 7	High
> 7	Very High

Project Characteristic 11: Tool Experience

The year of work experience the developer has, on the tool to be used for the development. Since, agile development supports simple and automated tools to a large extent. Results a low weight of 0.03 for this. Values can be assigned as follows:

*9**Table 3.8**: Metric for "Tool experience"*

Developers experience on the tool to be used for project-in-hand.	Platform Experience
< 6 months	Very Low
6 to 12 months	Low
12 to 18 months	Medium
18 to 24 months	High
> 24 months	Very High

Project Characteristic 12: Platform volatility

How frequently the projects is required to adapt the platform changes. Agile methodology creates self-contained modules that are developed to accept technical changes. Results a low weight i.e. 0.02 for this. Values can be assigned as follows:

*10**Table 3.9**: Metrics for "Platform volatility"*

Types of changes in platform	**Platform volatility**
Likely to evolve from one platform to another having different architectures (windows to Linux)	Very High
Likely to evolve from one platform to another having same architectures (Red hat to Ubuntu)	High
Likely to evolve from one platform to another having different version (windows XP service pack 2 to windows XP service pack 3) Or (Ubuntu 10 to Ubuntu 11)	Medium
No visible change found, but may require at later stage	Low
Never evolve	Very Low

Project Characteristic 13: Developer Experience

This is the work experience of the developer on the desired application. Since, agile supports collaborative and cooperative environment for the development. Collective ownership is also there; that provides the group support to the developers at each level. Results a low weight i.e. 0.02 for this. Values can be assigned as follows:

*11**Table 3.10**: Input Metrics for "Developer experience"*

Time (in months)	**Application Experience**
< 12 months	Very Low
12 to 24 months	Low
24 to 30 months	Medium
30 to 36 months	High

> 36 months	Very High

Project Characteristic 14: Programmer’s capability

This characteristic defines the programmer’s capability of understanding. In agile, not only the developer but the complete group work together to achieve a common goal, hence getting a better understanding of the project. Results a low weight i.e. 0.03 for this. The input value for this characteristic depends on the programmers vision, knowledge, interest , dedication for the project and can vary accordingly.

Project Characteristic 15: Existing Code Reuse

It is the amount of code that can be taken from existing code. In agile, deliverables are independent module which can be inherited from other projects and hence results a low weight (0.03) for this. Values can be assigned as follows:

*12**Table 3.11**: Values for “Existing Code Reuse”*

Reuse of Existing Code	**Category**
< 20%	Very Low
20 to 39%	Low
40 to 59%	Medium
60 to 79%	High
> 79%	Very High

Project Characteristic 16: Develop as base code

Product developed for reuse should be well documented. Product quality for these project should be very high and hence support traditional method. Therefore more weight is assigned to the characteristic(0.07):

*` 13**Table 3.12**: Value for “Develop as base code”*

Purpose of the project	**Time to market**

Developed only as a base project	Very High
Probability of being used in another project is very high	High
Project may require additional functionalities at a later stage	Medium
No open project found that needs the code of present project-in-hand	Low
Never to be reused	Very Low

Project Characteristic 17: Team cohesion

It represents the ease of communication among the team members and how well they can interact. This is needed for both traditional and agile methods and hence given avrage weight of 0.04. Values can be assigned as follows:

Project Characteristic 18: Customer Collaboration

This refers to the level of interaction among user and the development team or the ease with which customer can provide its feedback to the developer. If customer is not able to interact with developer, than traditional method is more suited because agile sprint/module cannot be provided without interaction, hence giving more weight (0.08) to this characteristic. Values can be assigned as follows:

*14**Table 3.13**: Value for "Reuse of existing code"*

Customer Interaction with development teams	**Category**
Less than 20%	Very High
20-39%	High
40-59%	Medium
60-79%	Low
Greater than 79%	Very Low

Project Characteristic 19: Testing Support

Testing is required for both traditional and agile method, since agile produces more versions of the product hence requires more frequent testing. Average weight of 0.05 is assigned to this characteristic. Values can be assigned as follows:

15***Table 3.14***: *Value for "Testing Support"*

Reuse of Existing Code	**Category**
Less than 20%	Very Low
20-39%	Low
40-59%	Medium
60-79%	High
Greater than 79%	Very High

Project Characteristic 20: Requirements known initially

It is not possible to know requirements in the beginning, they evolve with iterations. Also the customers are able to provide requirement only after using the product and should be able to interact with the developers hence giving less weight (0.02) to the characteristic. Values can be assigned as follows:

16***Table 3.15***: *Value for "amount of requirements known initially."*

Amount of requirements known initially	**Category**
< 20%	Very Low
20 to 39%	Low
40 to 59%	Medium
60 to 79%	High
> 79%	Very High

Project Characteristic 21: Platform experience

Platform experience is the experience of developer needs on the platform. Since it is needed for both, it is assigned average weight of 0.04. Values can be assigned as follows:

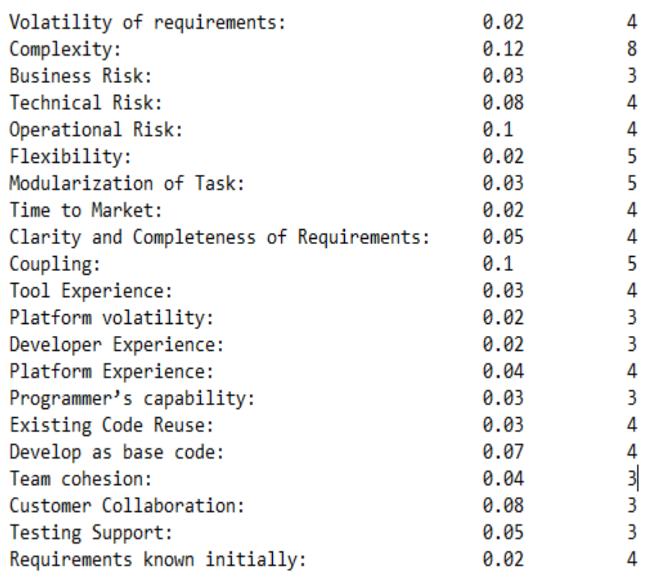
*17**Table 3.16**: Values for "Platform experience"*

Developer's exp. on the platform used (in months)	**Platform Experience**
< 6 months	Very Low
6 to 12 months	Low
12 to 18 months	Medium
18 to 24 months	High
> 24 months	Very High

3.3 Case study: Banking application

By cuckoo search:

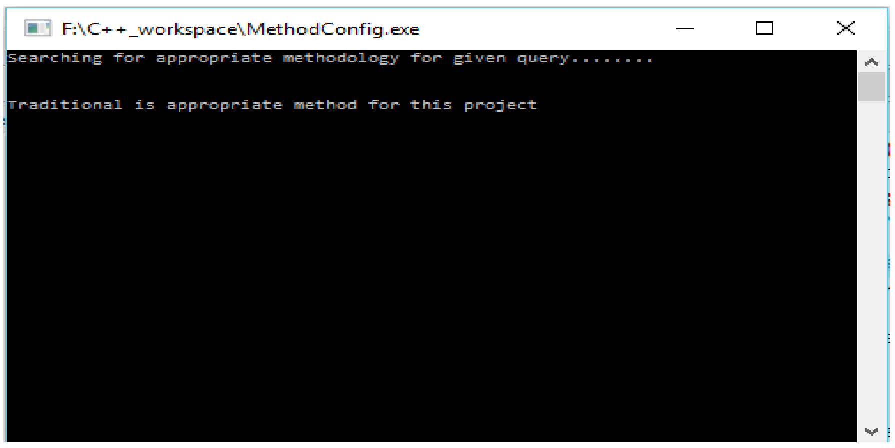
We gathered the requirements from the developers of the banking application. Problem statement was gathered and the value of different metric was induced to find out the method for the application. Problem statement for the banking application can be defined as
Open an account for a customer (savings or cheque), Deposit, Withdraw, Display details of an account, Change LOC, Produce monthly statements. Current and savings account transactions, up to 15 months (excluding MySave), credit card statements, your last statement and any transactions made since it was issued, mortgage transactions, up to 3 years loan balance and amount outstanding. The managers at the bank are consulted for the criticality of the project, market need, team size, known requirement of the system to be developed and other factors that are necessary to find the method suitable for the project.
Input Query to the Tool is as below:



```
Volatility of requirements:                 0.02      4
Complexity:                                 0.12      8
Business Risk:                              0.03      3
Technical Risk:                             0.08      4
Operational Risk:                           0.1       4
Flexibility:                                0.02      5
Modularization of Task:                     0.03      5
Time to Market:                             0.02      4
Clarity and Completeness of Requirements:   0.05      4
Coupling:                                   0.1       5
Tool Experience:                            0.03      4
Platform volatility:                        0.02      3
Developer Experience:                       0.02      3
Platform Experience:                        0.04      4
Programmer's capability:                    0.03      3
Existing Code Reuse:                        0.03      4
Develop as base code:                       0.07      4
Team cohesion:                              0.04      3
Customer Collaboration:                     0.08      3
Testing Support:                            0.05      3
Requirements known initially:               0.02      4
```

From the ouput we can see that the Project with the maximum corelation used Traditional project developement method and hence the output from the most corelated case is given. Traditional Method is appropriate method for this project:

By cuckoo search:



F:\C++_workspace\MethodConfig.exe

```
Searching for appropriate methodology for given query........

Traditional is appropriate method for this project
```

Output

3.4 Neural network v/s fuzzy logic

Fuzzy logic and neural network both are very popular algorithms in the field of machine learning. Fuzzy logic works on fix fuzzy rules made by the data available from previous projects. These rules helps in fuzzification and de-fuzzification of the newer problem.

Neural network works on a circuit of neurons which are connected by interconnection lines. Every neuron is consisting of a particular weight which gets updated after every learning cycle.

Properties	**Fuzzy logic**	**Neural network**
Computation cost	Low	High
Complexity	High	Low
Efficiency	Low	High
Adaptability	Low	High
Amount of data can handle	Low	High
Maintenance	High(due to fizzification and Defuzzificaion)	Low

18Table 3.17 : comparison between fuzzy logic and neural network

Considering all above advantages I have chosen neural network as a classification methodology for selection of suitable agile methodology.

Chapter 4

4 Method Configuration Process

This chapter provides the proposed method configuration process (fig 4.1), project characteristics, method selection algorithm and implementation. The project characteristics are identified. Based on these characteristics method selection has to be carried out. The proposed method selection process identifies the project characteristics and process them using neural network.

4.1 Method Selection Block Diagram

The generic method configuration process is depicted using a flow diagram (fig 4.1) which consists of seven major steps from gathering organizational requirements to configuring the agile method. Each step is described below.

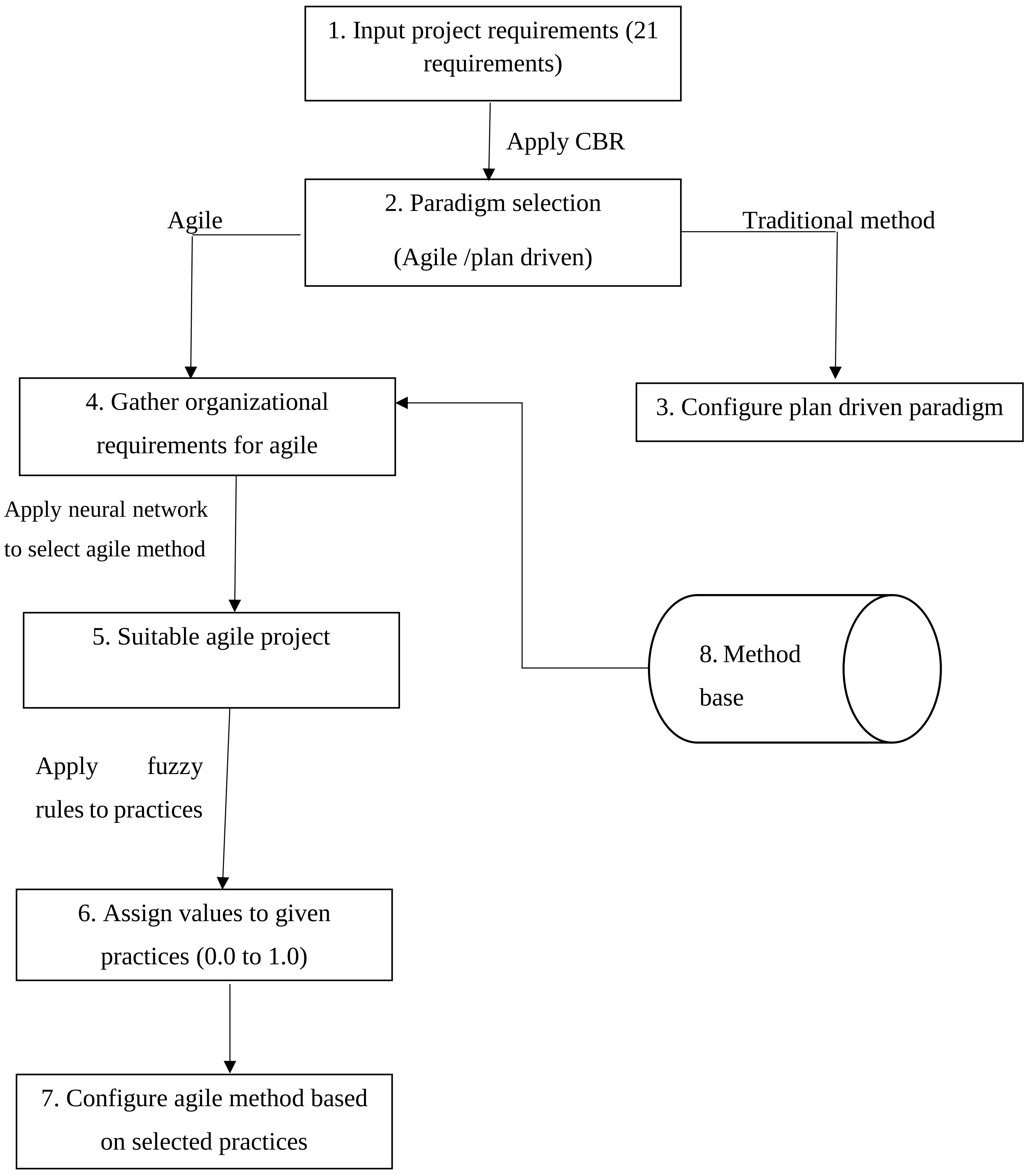
1. Input project requirements: This is the initial phase of this framework. Here project requirements affecting paradigm selection, namely volatility of requirement, Complexity, Business Risk, Technical Risk, Operational Risk etc. are gathered

2. Paradigm selection: Software development methodology can be divided into two paradigms namely plan-driven methodology and agile methodology. Case-based reasoning is applied for selecting the best suitable paradigm.

3. Configure plan driven paradigm: Plan driven paradigm is configured in case of selection of traditional. It is illustrated in [7], which is beyond the scope of this thesis.

4. **Elicitation of organizational requirements for agile method selection**: In the case of selection of agile methodology, First task is to identify the project characteristics namely group size, code style, expertise environment, task extent, Physical Environment, Industry customs and Abstraction mechanism and study their impact on the agile methodology selection. Based on their impact for the current project in hand, an agile method is selected from method base of agile method.

Method base consists different agile method namely Extreme Programming, Scrum, Feature-driven development, Dynamic system development model, Adaptive development model and Crystal.


1. Input project requirements (21 requirements)
Apply CBR
2. Paradigm selection (Agile /plan driven)
Agile
Traditional method
3. Configure plan driven paradigm
4. Gather organizational requirements for agile
Apply neural network to select agile method
5. Suitable agile project
Apply fuzzy rules to practices
6. Assign values to given practices (0.0 to 1.0)
7. Configure agile method based on selected practices
8. Method base

7Figure 4.1: Method selection and configuration framework

5. Suitable agile project: Since these weights of characteristics can vary depending upon the project in hand, we have taken a large database of projects and apply neural network to find the suitable method for the project in hand.

The method base for the tool is developed using the projects of different organizations. The experts from different organization provided the data and the project characteristic values based on their experience on the projects and method being used for the projects. Projects characteristics and their values are defined. Further, the database of around 50 projects has been developed using the experience of [24]. Around 50 test cases have been collected to train the network. For which project the method is being selected, that is also added to the database which helps to enhance the suggestion of the agile methodology of the project.

6. Assign values to given practices (0.0 to 1.0): Since an agile methodology consists of many practices which may not be usable for all kind of projects. Intel Shannon, IBM, Nokia has been tailoring the base method based on particular project characteristics. Fuzzy rules help in selecting applicable practices based on project requirements.

Project characteristics namely Task extent, Technology complexity, Teams experience, Physical environment, Requirements documentation, Progress approach are defined for the particular project.

Using these project characteristic, Common and variable practices (defined in Table 3.1) have been assigned weights (0.0 to 1.0) by applying fuzzy rules.

7. Configure agile method based on selected practice: Select the essential practices with weights greater then a particular threshold value. Practices with less than the threshold value can be made absolute.

4.2 Weight Distribution and Input values for agile method selection

According to the importance of the characteristic to the project in hand, the characteristics are assigned weight and input values for different projects.

Project Characteristic 1: Task Extent

This characteristic defines the length of the project. We are classifying the length into four parts. The length of project plays a major role in choosing of a decent software development methodology.

Values can be assigned as follows:

19***Table 4.1****: Values for “****Task Extent****”*

Task Extent	Suggested method
Small	XP, SCRUM, FDD, DSDM, Crystal
Medium	XP, SCRUM, FDD, Crystal
Large	FDD, ASD
Complex	ASD

Project Characteristic 2: Group Size

Group size defines the number of people working on the project. Large number of people need a methodology with good communication strategy among developers.

Preferred method is shown in the table:

20***Table 4.2****: Values for “****Group Size****”*

Group Size	Suggested method
Less than 10	XP, SCRUM, Crystal
Multiple teams	SCRUM, DSDM, Crystal

Project Characteristic 3: Progress Approach

Progress approach is defined on the basis of frequency of review by the customer.

In rapid development methodology a prototype is made prior to actual development to ensure the requirements by user. In iterative development methodology review is done after every cycle of development. In distributive development the developing team is not at the same place, that’s why the methodology should be supportive of communication among developers.

21***Table 4.3****: Value for “****Progress Approach****”.*

Progress Approach	Suggested method

Iterative	XP, SCRUM, FDD, ASD, DSDM, Crystal
Rapid development	XP, SCRUM, ASD, DSDM, Crystal
Distributive Development	ASD

Project Characteristic 4: Code Style

Code style defines the type of language to be used for development. The code style may or maynot be object oriented or platform oriented. Considering these factors code style is categorized into two types weather the code is simple or not.

Values can be assigned as follows:

*22**Table 4.4**: Metrics for "**Code Style**".*

Code Style	**Suggested method**
Clean and Simple	XP
Not specified	SCRUM, FDD, ASD, DSDM, Crystal

Project Characteristic 5: Expertise Environment

This characteristic signifies the time (in months) before which feedback will be given by the client. In development feedback plays a major role since it is the only way of knowledge about the customer's expectation from the project.

Values can be assigned as follows:

*23**Table 4.5**: Value for "**Expertise Environment**"*

Expertise Environment	**Suggested method**
Quick Feedback	XP
Not Specified	SCRUM, FDD, ASD, DSDM, Crystal

Project Characteristic 6: Physical environment

This characteristics defines the location of development team, weather it is located at the same place or at different places. The teams at same place is called co-located teams. Teams at different places are called distributed teams. There is also option of not specifying this characteristic, this is useful when the team is not decided for the project.

Values can be as follows:

24*Table 4.6*: *Value for "Physical environment"*

Physical environment	Suggested method
Co-located teams	XP, ASD, Crystal
Distributed Teams	XP, ASD
Not specified	SCRUM, ASD, DSDM

Project Characteristic 7: Industry Customs

Industry cunstom is very essential for quick and secure software development. It will be very easy to develop the modules if collaborative customs are there.

XP and DSDM Agile methodologys are suitable in case of collaborative customs.

Values can be assigned as follows:

25*Table 4.7*: *Metrics for "Industry Customs".*

Industry Customs	Suggested method
Collaborative and Cooperation	XP, DSDM
Not specified	SCRUM, FDD, ASD, Crystal

Project Characteristic 8: Abstraction Mechanism

This defines the type of abstraction being used in the project. Abstraction can be done using two ways, by using object oriented technique or by using component oriented technique.

Values can be assigned as follows:

26*Table 4.8*: *Metrics for "Abstraction Mechanism".*

Abstraction Mechanism	**Suggested method**
Object Oriented	XP, SCRUM, FDD, ASD, DSDM, Crystal
Component Oriented	ASD, DSDM

Project Characteristic 9: Project requirements

It is not possible to know requirements in the beginning, they evolve with iterations. Also the customers are able to provide requirement only after using the product and should be able to interact with the developers.

Values can be assigned as follows:

27*Table 4.9*: *Metrics for "Project requirements".*

Project requirements	**Suggested method**
Adaptive	XP, SCRUM, FDD, ASD, DSDM
Predictive	Crystal

Project Characteristic 10: Developer Experience

This is the work experience of the developer on the desired application. Since, agile supports collaborative and cooperative environment for the development. Collective ownership is also there; that provides the group support to the developers at each level. Values can be assigned as follows:

28*Table 4.10*: *Input Metrics for "Developer experience"*

Time (in months)	**Application Experience**	**Suggested method**
< 12 months	Very Low	XP, SCRUM, FDD
12 to 36 months	Low	ASD, DSDM
> 36 months	Medium	Crystal

4.3 Method Selection Algorithm

The proposed method selection algorithm, to form project specific method, is explained here. Firstly project characteristics and method base are gathered. This method base is fed to neural network to start the learning process and find the weights of neurons. Then project characteristics are fed to updated neural network to find the most suitable base agile methodology based on particular project characteristics.

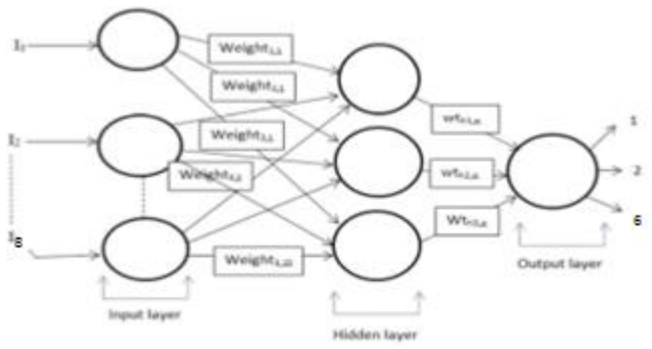
Variable practices of the base agile method are fed to fuzzy logic to find weights of all practices. Here weight refers to the essentiality of the practice for defined requirements. The practices with 'high weights' or 'most suitable practice' are chosen.

Proposed algorithm takes method base and project characteristics as input and apply the algorithm on the input query to suggest most suitable method.

4.3.1 Neural network

An Artificial Neural Network (ANN) is a data processing framework that is based on biological nervous systems, for example, how the brain collect and store the processed information. The key component of neural network is the novel structure of the data handling system. It is made out of neurons which have high interconnectivity among them working as one to tackle particular problem. Since neurons classifies the information, they can also be termed as processing elements. ANNs, similar to individuals, learn by examples. An ANN is designed for a particular application, for example, image processing or to classify data, through a learning procedure. Learning in natural frameworks includes updating the values of interconnecting associations that exist between the neurons. This is exactly what happens with ANN also.

Paradigm selection process was simulated by three layer feed forward back propagation neural network which contains input, hidden and output layer. Neural network tool is available in MATLAB is used for training and simulation. Network used in the process was having eight neurons at input layer, three neurons at hidden layer and six at output layer. Please refer below figure.



8Fig 4.2 Problem Mapped as Neural Networks [6]

Output of the neural network is divided into six categories i.e 1, 2 … , 6.

The output '1' of the network indicated that 'extreme programming' is the best suited for given project parameters, output '2' indicated that SCRUM can be used for the given project parameters. A '3' at the output indicated that 'feature driven development' can be used for the given project parameters. The output '4' of the network indicated that 'adaptive software development' is the best suited for given project parameters, output '5' indicated that 'dynamic software development method' can be used for the given project parameters. A '6' at the output indicated that 'crystal' is the best suited for the development of the project in-hand.

4.3.2 Proposed Algorithm: For agile method selection using neural network:

Input in the query which consist of the input values for the project characteristics

```
Input:  User query //Based on the factor values determine the methodology to be used.
Output:  XP/SCRUM/FDD/ASD/DSDM/Crystal
Initialize times to 5000 which defines the 'number of times' a data to be learned
Initialize values of input layer neurons to 8, hidden layer neurons to 3, output layer neurons to 6
Initialize loop to 0
```

LEARNING

```
While (loop <= times)
{
        "Compute output of hidden layer"
                Compute input of every neuron of hidden layer
                --for every hidden layer neuron 'Hi'
                Find sum of multiplication of output of input layer neuron (Ni) with
        weights of edge (Hi-Ni)
        Apply sigmoid function on sum to find output of neuron
    “Compute output of output layer”
        Compute input of every neuron of output layer
        --for every Output layer neuron ‘Oi’
                Find sum of multiplication of output of hidden layer neuron (Hi) with
        weights of edge (Hi-Oi)
        Apply sigmoid function on sum to find output of neuron
}
```

BACK-PROPOGATION:

```
“Calculate error for every neuron of output layer”
For every neuron of Output layer O (i)
    Error (I) = (Expected Output (i) – Actual output (i))*
              (Actual output (i))*
              (1 – Actual output (i));
“Calculate error of every neuron of hidden layer”
For every neuron of hidden layer H (i)
    For every neuron of output layer O (j):
        Find sum of multiplication of error(O(j)) with weight(O(j),H(i)) and assign it to
        errtemp(i)
    Error(i) = errtemp(i)*
               Actual output(i)*
              (1 – Actual output(i));
```

WEIGHT UPDATION:

```
    For every output layer neuton o[i]
        For every Hidden layer neuron H[i]
            Weight[i][j]+= Learning rate *
                           Error of output layer neuron*
                           Output of Hidden layer neuron
    For every Hidden layer neuton o[i]
```

```
        For every Input layer neuron H[i]
            Weight[i][j]+= Learning rate *
                        Error of hidden layer neuron*
                        Output of Input layer neuron
```

PREDICTION:

```
"By taking project requirements:
    "Compute output of hidden layer"
        Compute input of every neuron of hidden layer
        --for every hidden layer neuron 'Hi'
            Find sum of multiplication of output of input layer neuron (Ni) with
        weights of edge(Hi-Ni)
        Apply sigmoid function on sum to find output of neuron
    "Compute output of output layer"
        Compute input of every neuron of output layer
        --for every Output layer neuron 'Oi'
            Find sum of multiplication of output of hidden layer neuron (Hi) with
        weights of edge(Hi-Oi)
        Apply sigmoid function on sum to find output of neuron
```

SIGMOID FUNCTION:

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

4.3.4 Configuring the agile method:

In previous step an agile method is selected but the chosen agile method does not always fit to the project requirement. For developing a good software the development methodology need to be customized. Rinky have proposed commonalities and variability practice for each agile method.

In extend of that we are proposing fuzzy rules for choosing the essential practices for the project in hand.

There are some project characteristic which will affect practice selection. These characteristics are defined below:

Characteristic	Values	practice support (essential)	practice support (not essential)
Task extent	Large	Sprint	daily scrum
	Small	daily scrum	
Technology	Complex	Scrum team	
	Simple		Scrum team
Teams	Experienced		sprint retrospective
	Not experienced	scrum of scrum	Scrum
Physical environment	Distributed teams	Sprint review	sprint retrospective
	Co-located teams		
Requirements	Documented	product backlog	scrum of scrum
	Not documented	sprint review	
Progress approach	Iterative development	1.product backlog 2.sprint planning	

29Table 4.11: Project characteristics affecting essentiality of practices

Using these project characteristics fuzzy rules have been defined for practice selection.
Fuzzy rules for SCRUM:

if <size> is <small> then <daily scrum> is <essential> to select

if <size> is <large> then <daily scrum> is <not essential> to select

if <size> is <large> then <sprint> is <essential> to select

if <requirements> is <documented> then <product backlog> is <essential> to select

if <technology> is <complex> then <scrum teams> is <essential> to select if <team> is <experienced> then <sprint retrospective> is <not essential> to select

if <development> is <iterative> then <sprint planning meeting> is <essential> to select

if <technology> is <simple> then <scrum teams> is <not essential> to select

if <team> is <not experienced> then <scrum of scrum > is <essential> to select

if <physical environment> is <distributed> then <sprint retrospective> is <not essential> to select

if <physical environment> is <distributed> then <sprint review> is <essential> to select

if <development> is <iterative> then <product backlog> is <essential> to select

if <requirements> is <not documented> then <sprint review> is <essential> to select

if <requirements> is <documented> then <scrum of scrum> is <not essential> to select

if <teams> is <co-located> then <sprint planning meeting> is <not essential> to select

if <team> is <not experienced> then <scrum> is <not essential> to select

4.4 Details of the Implementation

This section describes the details of the implementation from creating the case base to the output.

4.4.1 Developing the Case Base: The case base for the tool is developed using the projects from the organization. The experts from different organization provided the data and the project characteristic values based on their experience on the projects and method being used for the projects. Projects characteristics and their values are defined in **Section 4.2**. Further the database of around 50 projects has been developed using the experience of [24]. Around 50 test cases have been collected to train the network. For which project the method is being selected, that is also added to the database which helps to enhance the suggestion of the agile methodology of the project.

4.4.2 Tool Development: The algorithm mentioned in **section 4.3.4** is implemented using C language.

The different data structures and modules of the implementation are mentioned below:

Modules:

1. Sigmoid(int x); //This module is to compute sigmoid function.

2.void suggest(double w[InputN][HN],double v[HN][OutN]) : This function is to predict the methodology to be used using the trained neural network and project requirements given by the customer.

3. Training: This module train the network using the database of previous projects. Here back propagation methodology is applied to train the network. This takes previous project database as input and trained neural network is presented as output.

4. Suggestion: The trained network is further used to find the suitable method based on new project characteristics. This takes project characteristics and trained neural network as input and suggested method is presented as output.

Chapter 5

5. Case study

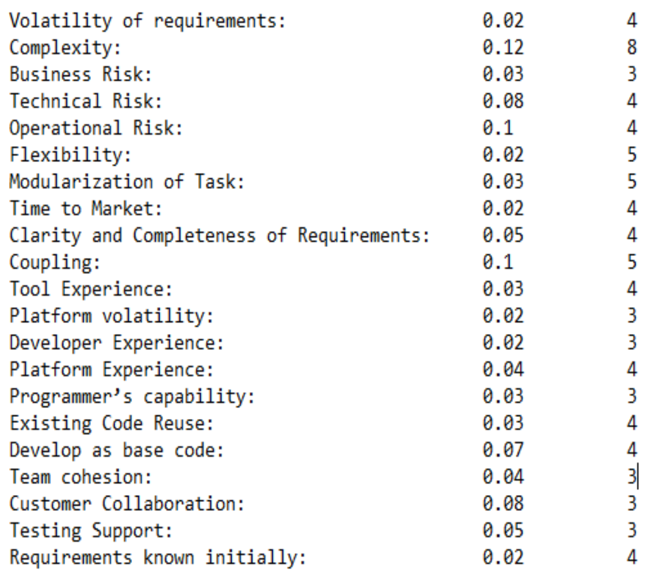
In this section, we present the case studies for XP, SCRUM, FDD, ASD, DSDM or Crystal, how the algorithm applies to these case studies and how the output is retrieved from the from the tool.

Case Study 1: Cab booking application project:

Project is to develop an application for cab booking. After consulting the developers and Managers at the organization developing the application project, we have found the below mentioned characteristics of the project.

In this project, there are a set of initial requirements to provide basic functionality to the user. There is a feedback option to the user, so that he can give his feedback any time to improve the application. Since there are already many cab booking applications and services running in the city or town, business risk is high if we do not develop the application on time.

Based on the statement from the concerned persons at the organization, the values for the metrics have been derived. Provide the method that can be used for this project. Input query to the tool is:



Volatility of requirements:	0.02	4
Complexity:	0.12	8
Business Risk:	0.03	3
Technical Risk:	0.08	4
Operational Risk:	0.1	4
Flexibility:	0.02	5
Modularization of Task:	0.03	5
Time to Market:	0.02	4
Clarity and Completeness of Requirements:	0.05	4
Coupling:	0.1	5
Tool Experience:	0.03	4
Platform volatility:	0.02	3
Developer Experience:	0.02	3
Platform Experience:	0.04	4
Programmer's capability:	0.03	3
Existing Code Reuse:	0.03	4
Develop as base code:	0.07	4
Team cohesion:	0.04	3
Customer Collaboration:	0.08	3
Testing Support:	0.05	3
Requirements known initially:	0.02	4

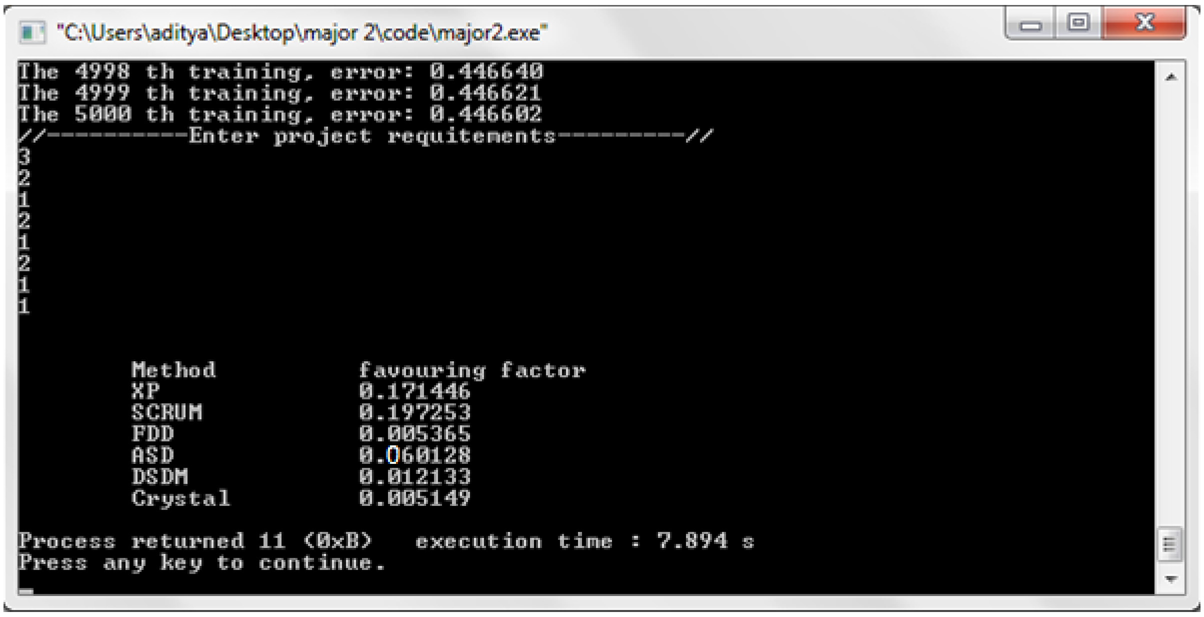
30***Table 5.1:*** *Requirements for cab booking application*

By applying cuckoo search agile methodology is found to be most appropriate methodology. After this more requirements are gathered for choosing best agile methodology among six agile methodologies.

31***Table 5.2***: *Values for "cab application system"*

Characteristics	**Values (Project specific)**
Task extent	Large
Group size	Multiple teams
Progress approach	Iterative
Code style	Not specified
Expertise environment	Quick feedback
Physical environment	Distributed teams
Industry customs	Collaborative and corporation
Abstraction mechanism	Object oriented
Project requirement	Adaptive
Developer experience	Very low

From the ouput we can see that the Projects with the maximum corelation used similar project developement method(Agile) and hence the output from the most corelated case is given.Agile Method is appropriate method for this project:


"C:\Users\aditya\Desktop\major 2\code\major2.exe"

```
The 4998 th training, error: 0.446640
The 4999 th training, error: 0.446621
The 5000 th training, error: 0.446602
//-----------Enter project requitements----------//
3
2
1
2
1
2
1
1



        Method          favouring factor
        XP              0.171446
        SCRUM           0.197253
        FDD             0.005365
        ASD             0.060128
        DSDM            0.012133
        Crystal         0.005149

Process returned 11 (0xB)   execution time : 7.894 s
Press any key to continue.
```

Output

Since SCRUM is with highest favoring factor, so it is chosen for cab application development. Identify characteristics for case study 1

*32**Table 5.3**: Values for "cab application system"*

Number	Requirements
R1	Large software
R2	Complex technology
R3	Experienced teams
R4	Distributed teams
R5	Documented requirements
R6	Iterative development

*33**Table 5.4**: Weighted practices of scrum for case project 1*

Number	Practice	Weight
P1	Product backlog	0.8
P2	Sprint review	0.4
P3	Scrum teams	0.9
P4	Sprint	0.8
P5	Daily scrum meeting	0.3
P6	Sprint planning meeting	0.6
P7	Sprint retrospective	0.0
P8	Scrum of scrum	0.2

These weighted practices will provide support to the method engineer to select variable attributes in the method.

A threshold value is chosen on the basis of the amount of attributes which can be kept in the configured agile development method. Here we are choosing 0.5.

Considering the threshold value, these variable practices **product backlog, scrum teams, sprint** and **sprint planning meetings** are chosen.

Chapter 6

6 Results & Conclusion

It was found that the organizations do not spend quality time on determining which method to adopt for their project, which results in the failure of the project. One example of the fail case without using method configuration can be found in case study 1[21].

It shows an example of "cab booking application" which was earlier developed without using any proper methodology and they faced many problem of:

- Team members did not participate actively.
- Too much documentation was needed
- System became complex
- Could not satisfy the evaluators
- Project could not be completed on time.

When the same project was developed using Agile Method, the output was satisfactory and project was developed perfectly with no problems.

From the example, it is clear that the method configuration is very important for successful project completion.

In my thesis, I have tried to develop a tool for properly evaluating the method for project in hand. With the use of neural network, the tool provides the result based on the past experience of the developers and experts on the project. Also commonality and variability are defined for all the agile methods.

We are also defining some fuzzy rules for selection of variability practices based on project requirements.

The query case is again stored in the database to increase our case base for future evaluation of method for any project.

"To sum up, the thesis found that there is a requirement to address the agile method selection problem and to develop a method configuration process for the project in hand."

References

[1] Coad, P., LeFebre, E. and DeLuca, J. (2000). Java Modeling in Color with UML: Enterprise Components and Process, Prentice Hall, Inc., Upper Saddle River, New Jersey.

[2] F.Karlsson and P.J.Ågerfalk, "Method Configuration: Adapting to Situational Characteristics While Creating Reusable Assets", Information and Software Technology, vol.46, no.9, 2004, pp.619-633.

[3] D. Gupta, R. Dwivedi, "A Step towards Method Configuration from Situational Method Engineering", Software Engineering: An International Journal (SEIJ), Vol. 2, No. 1, 2012, pp. 51-59.

[4] A. Qumer and B. Henderson-Sellers, "Crystallisation of agility-back to basics" ICSOFT 2, 2006, pp. 121-126.

[5] A. Qumer and B. Henderson-Sellers, "A framework to support the evaluation, adoption and improvement of agile methods in practice" Journal of systems and software, vol. 81, 2008, pp. 1899-1919.

[6] D. Gupta, R. Dwivedi, "A framework to support evaluation of project in-hand and selection of software development method "

[7]Moaven S., Habibi, J. and Ahmadi, H. (2008). Towards an Architectural-Centric Approach for Method Engineering. *In IASTED conference on Software Engineering*, Austria, (pp. 74-79).

[8]Stapleton, J. (1997). *Dynamic system development method- the system in practice*. Addison Wesley.

[9] D. Gupta, R. Dwivedi, "Configurable method model of agile methods - for creating project-specific methods."

[10] D. Gupta, R. Dwivedi, "Applying Case based reasoning in Cuckoo search for the expedition of Groundwater Exploration".

[11] Agile Manifesto (2001) *Manifesto for Agile Software Development*, [online]

http://www.agilealliance.org/the-alliance/the-agile-manifesto/ (accessed 14 March 2005).

[12] Avison, D. E., (1996). Information Systems Development Methodologies: A Broader Perspective. *In Method Engineering. Principles of Method Construction and Tool Support. Procs. IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering, 26-28, Atlanta, USA, S. Brinkkemper, K. Lyytinen, R.J. Welke, Eds. Chapman & Hall, London,* (pp. 263-277).

[13] Brinkkemper, S. (1996). Method engineering: Engineering of information systems development methods and tools. *Information & Software Technology*, 38(4), (pp. 275-280).

[14] Qumer, A. and Henderson-Sellers, B. (2008a). A framework to support the evaluation, adoption and improvement of agile methods in practice. *The Journal of Systems and Software*, 81(11), 1899–1919.

[15] Rizwan, M. and Qureshi, J. (2012). Agile software development methodology for medium and large projects. *IET Software*, 6(4), 358–363.

[16] http://www.unf.edu/~broggio/cen6940/ComparisonAgileTraditional.pdf

[17] http://www.mannaz.com/en/insights/when-is-agile-project-management-best-suited

[19] http://www.allaboutagile.com/is-agile-development-right-for-your-project/

[20] http://users.jyu.fi/~jpt/doc/thesis/ime-5_1.html

[21]S. Pathak, P. Pateriya, "A Case Study on Software Development Projects in Academic Knowledge Centers using SCRUM"

[22] Gupta D. and Dwivedi R. "*Method Configuration from Situational Method Engineering*" ISSN No. 0163-5948, Vol. 37, No. 3,pp. 1-11, May 2012.

[23] Dwivedi R. and Gupta D. "A *Complete method configuration process for configuring project-specific methods*" in **Journal of Software,** ISSN 1796-217X Vol. 9(3), pp. 29-40, (2015).

[24] Dwivedi R. and Gupta D. "*Applying machine learning for configuring agile methods*" in **International Journal of Software Engineering and its Application**, ISSN 1738-9984 vol.9, No.3(2015), pp. 29-40.

[25] Dwivedi R. and Gupta D. “Configurable method model of agile methods – for creating project specific methods” in **International journal of software engineering,** SERP’16 , pp. 86-91