

# **Prediction of Suitability of Software Development Methodology using Agile Configuration**

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

Master of Technology

In

Software Technology

By

**Priyanka Sharma**

**(Roll No: 2K13/SWT/11)**

Under the guidance of

**Prof. (Dr.) DAYA GUPTA**



DEPARTMENT OF SOFTWARE TECHNOLOGY  
DELHI TECHNOLOGICAL UNIVERSITY  
NEW DELHI

# **DECLARATION**

=====

I hereby declare that the thesis entitled, “**Prediction of Suitability of Software Development Methodology using Agile Configuration**”, 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.

---

**Priyanka Sharma**

**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 Suitability of Software Development Methodology using Agile Configuration**”, a bona fide work done by **Ms. Priyanka Sharma(2k13/SWT/11)** 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.

**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 am also thankful to the SAMSUNG who has provided me opportunity to enroll in the M.Tech Programme and to gain knowledge through this programme. This curriculum provided me knowledge and opportunity to grow in various domains of computer science.

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.

**Priyanka Sharma**

**2K13/SWT/11**

# Table of Contents

1 Introduction.....	6
1.1 Introduction & Motivation.....	6
1.2 Related Work .....	7
1.3 Problem statement .....	9
1.4 Scope.....	9
1.5 Organization of the thesis.....	10
2 Agile Modeling .....	11
2.1 Extreme Programming .....	11
2.2 Scrum .....	12
2.3 Feature Driven Development (FDD).....	14
2.4 Dynamic System Development Method .....	15
2.5 Adaptive Software Development (ASD).....	16
2.6 Agile Manifesto .....	16
2.7 Characteristics of Agile Methodologies .....	18
3 Method Selection Framework .....	20
3.1 Method & Method Engineering.....	20
3.2 Method Configuration Framework .....	23
3.3 Modification in Method Selection Framework.....	26
4 Proposed Method Prediction Process .....	28
4.1 Method Selection Block Diagram.....	28
4.2 Case Bases in Databases .....	29
4.3 Method Selection Tool.....	39
4.4 Details of the Implementation .....	43
4.5 Case Studies .....	48
5 Results, Contribution & Conclusion .....	53
References .....	55

# Chapter 1

## 1 Introduction

### 1.1 Introduction & 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 complications 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 method for the project development. Since process of evaluation is complex, Case based reasoning with Cuckoo search 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”.

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 configuration in the form of method families, these method families are further surfaced to form a method line

that ultimately results into a configured method the proposal is in infancy stage and fails to give detailed process of configured method construction. These proposals addresses many issues, however issues like ‘right granularity’, ‘appropriateness of method being selected’ and ‘explicit representation of variability’ are still open.

**Method Configuration** is a sub-discipline to SME and has been formulated and proposed by Karlsson et al in [22]. They defined method configuration task as a “means to adapt a particular method to various situated factors. The focus is thus on one method as a base for configuration rather than on a set of methods as a base for assembly”. Further in their later proposals they [47] defined a high-level configurable construct for their method configuration process. The configurable concept proposed by them lacked the issues of genericity and granularity. Since the previous proposals on method configuration centered on the one single base method, reduced the scope of the method generation for every 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 projects. 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 both traditional and agile paradigm. In this framework preprocessing is



decision about paradigm selection [24] but due to the some limitation of the proposed framework like the once the tool is trained from the data set, it will produce the same output irrespective of changing organizational requirements or behavior.

Our solution for providing the method selection framework uses case based reasoning, where the case base (Past Experience) is enhanced very time the new query case arrives thus providing much more experience for the upcoming projects. Further we check the outcome based on propositional logic, conditions necessary for Agile and Traditional Methods, to verify our output.

### **1.3 Problem statement**

As highlighted in the foregoing, there are proposals for Decision support system for methodology paradigm selection based on set of project characteristics. These project characteristics are selected based on their impact on SDLC and Neural Network is used to select software development paradigm. In this work a new learning algorithm is proposed for selection of paradigm

Problem statement for this thesis is as below:

**Develop a Decision support system to select a suitable paradigm for the project in hand.**

### **1.4 Scope**

First task is to identify the project characteristics like requirement, risk, time to market and many other and study their impact on the paradigm selection. Based on their impact for current project in hand, method paradigm is selected before method configuration.

Second task is assign weights to these characteristics based on their role in Information System Development paradigm. .

Since these weights and their value can vary depending upon the project in hand, we have taken a large case base of projects and apply **Case Based Reasoning(CBR)** to find the suitable method for the project in hand. Case bases store the previous projects experience and ouput the result for project in hand.

**Cuckoo search(CS)** is used to find the set of suitable (similar) case from the set of cases available. To find the solution, Pearsons's Correlation function is used to find the best

nest(case) among the set of nest that are most suitable(Selected according to similarity function).

Post finding the best nest(case) some propositional logic(set of conditions which are applicable for Agile and Traditional development method) is applied on the best nest to find if the method of the best nest is also suited for the query case(project in hand).

After the successful completion of the propositional logic on best nest, output is given to user is Agile, Traditional or Hybrid method is appropriate for the project in hand. Hence scope of the work can be summarized as:

- Identification of project characteristics for lifecycle selection and assigning weights based on their impact on the paradigm selection..
- Applying Cuckoo search in hybrid with Case based reasoning to select paradigm
- 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 provides the overview of methods, methods engineering process and method selection paradigm proposed by dwivedi[24] which is further extended in this project.

**Chapter 4:** This chapter presents the work done in the project to accomplish the task of method selection. Algorithm, Implementation Control flow and Output are analyzed. Software Case Studies are discussed and their outputs.

**Chapter 5:** This chapter is for the results and conclusion of the thesis.

# Chapter 2

## 2 Agile Modeling

Agile – devoting “the quality of being agile; readiness for motion; nimbleness, activity, dexterity in motion” as mentioned in the Oxford Dictionary – are the methods to offer provide solution to the business looking for faster and nimbler development process. All the agile methodologies acknowledge that high quality software and customer satisfaction can only be achieved by light weight methods only. Some of the most commonly used methods are explained below.

### 2.1 Extreme Programming

Extreme programming evolved due to problems of the traditional development models. The XP process is characterized by short development cycles, incremental planning, continuous feedback and evolutionary design. 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 programmer estimates the effort for implementation and customer decides scope and timing of release based on 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 build with unnecessary code and complexity removed.
- Refactoring – It is the restructuring of system by removing duplication, improving communication, simplifying and adding flexibility but without changing functionality of program
- Pair programming – Production code written by two programmers on same system.

- Collective ownership – No particular person is responsible for any code segment. Anyone 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 has to be available at all times with the development team.
  - Coding Standards – Coding rules exist and are followed by the programmers to bring consistence and improve communication among the development team.
- The lifecycle of an XP project, shown in Figure 4, is divided into six phases: Exploration, Planning, Iterations to release, 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.

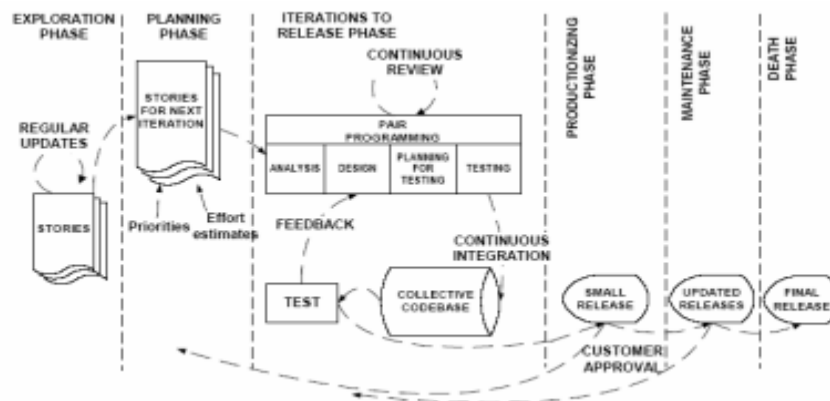


Figure 2.1 : Lifecycle of the XP process

## 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** - It is the list of all features and changes that need to be made to the system desired by multiple actors, such as customers, marketing and sales and project team. The Product Owner is responsible for maintaining the Product Backlog.
- **Sprints** - Sprints are 30-days in length. It is the procedure of adapting to the changing environmental variables and must result in a potentially shippable increment of software. The working tools of the team are Sprint Planning Meetings, Sprint Backlog and Daily Scrum meetings.
- **Sprint Planning meeting** – It is attended by the customers, users, management, Product owner and Scrum Team where a set of goals and functionality are decided on. Next the Scrum Master and the Scrum Team focus on how the product is implemented.
- **Sprint Backlog** – It is list of features for a particular sprint.
- **Daily Scrum** – It is daily meeting of around 15 minutes where obstacles and progress are discussed and tracked.

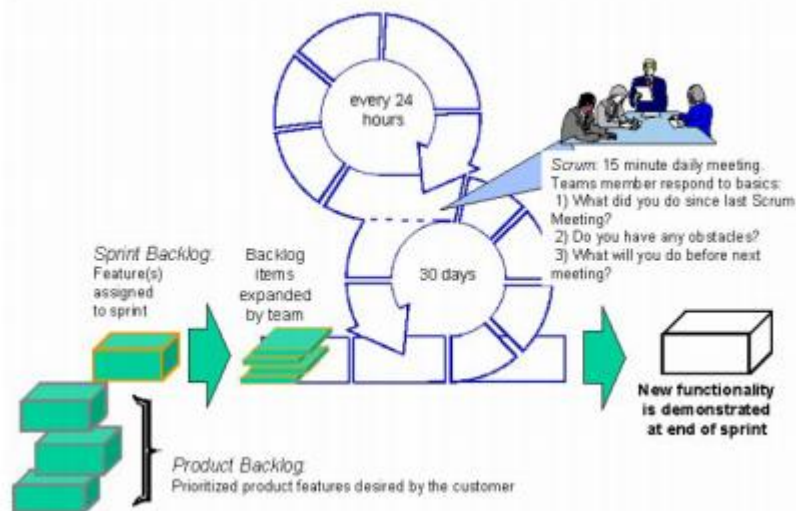


Figure 2.2 : Scrum Process

## 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 6), an explanation of the different roles and responsibilities if given below:

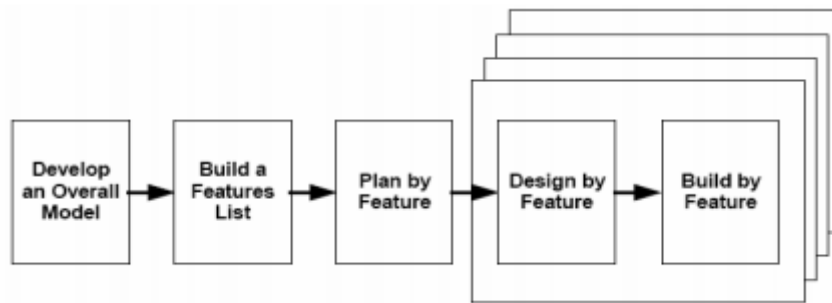


Figure 2.3: Feature Driven Development processes

- **Develop an Overall Model** – An overview of the system scope and its context is performed by the domain expert. Documented requirements such as use cases or functional specifications are developed.
- **Build a Features List** – List of features to support requirements is produced.
- **Plan by Feature** – Feature sets are ordered according to their priority and dependencies and assigned to chief programmers. The classes identified in the first phase are assigned to class owners. Schedules and milestones are set for the feature sets.
- **Design by Feature & Build by Feature** - Features selected from the feature set are developed by the class owners. Both these are iterative procedures during which the team produces the sequence diagrams for the assigned features. These diagrams are provided to the developers who implement the feature to support the design for a particular feature. The code developed

is then unit tested and inspected. After a successful iteration, the completed features are promoted to the main build.

## 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 7):



Figure 2.4 : DSDM process diagram

- Feasibility Study – In this phase a decision is made whether to use DSDM or not based on the type of project, organizational and people issues.
- Business Study – In this phase workshop is organized to understand the business phase of project. Key Outputs from this section are System architecture definition and an Outline prototype plan.
- Functional Model Iteration –This phase involves analysis, coding and prototypes of project. The key output of this phase is a functional model which consists of the prototype code and analysis models.
- Design and Build Iteration –The design and functional prototypes are reviewed by the users and further development of the system is based on the users’ comments.

- Implementation – In this final phase the system is given to the users. Training is provided. In DSDM, the iterative and incremental nature means that maintenance can be viewed as continuing development. Instead of finishing the project in one cycle, the project can go to any of the phases, Design and Build phase, Functional Model Iteration, or even Feasibility phase so that previous steps can be refined. There is an emphasis on high quality and adapts to the changing requirements. Like other agile methods, DSDM approaches iterations as short cycles of between two and six weeks.

## 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 – This is to define the project mission, make clear the realization about what is unclear.
- Collaborate – It highlights the importance of teamwork for developing dynamic systems
- Learn – This phase is to learn from the mistakes, and that requirements may well change during development.

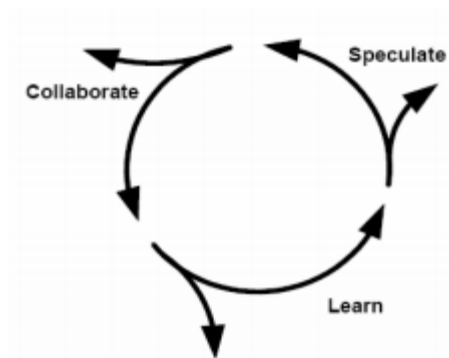


Figure 2.5 : ASD Lifecycle

## 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, that 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 highquality, 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 than 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.

**Balancing Flexibility and Planning** –Planning is better but future of the projects cannot be predicted in advance. Good planning strategy is to make detailed plans for the next few weeks, rough plans for the next few months, and extremely crude plans beyond that. In this view one of the main sources of complexity is the irreversibility of decisions. Agile designers need to think about how they can avoid irreversibility in their decisions. Rather than making the right decision now, look for a way to either put off the decision until later or make the decision in such a way that can be reversed later without too much difficulty.

**Empirical Process** – The Agile methods develop software as an empirical (or nonlinear) process. Processes are either defined or empirical. In other words, defined process is one that

is started and allowed to run to completion producing the same results every time it runs. Defined process cannot be considered in software development because too much change occurs when the team is developing the product.

**Decentralized Approach** – Decentralized management style can severely impact a software project because it could save a lot of time than a centralized management process. Decision making is distributed among the developers. Developers do not take on the role of management. Management is still needed to for the progress of project. However management recognizes the expertise of the technical team to make technical decisions without their permission.

**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 is 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”

# Chapter 3

## 3 Method Selection Framework

This chapter presents the overview of Methods, Methods Engineering, complete generic framework of the process of paradigm selection. Method selection framework has been proposed by Dwivedi [6, 24]. Further Dwivedi[22, 3, 23] provided towards the method configuration process to configure the method for the project in hand as per the project requirements.

### 3.1 Method & Method Engineering

#### 3.1.1 Methods

Many definitions for the methods can be found in different literature:

(Brinkkemper, 1996): described a method as *“an approach to perform a systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities with similar development products”*.

(Prakash, 94): proposed a method as *“a collection of tools and techniques, product and process models, guidelines, checklists, heuristics, etc. that help an application engineer to build a suitable product”*.

(Iacovelli et. al., 2008): describes *“A method is based on models (systems of concepts) and consists of some task/activities/steps, which should be performed, in particular, order”*.-

(Smolander et. al., 1990): A method is an *“a predefined and organized collection of techniques and a set of rules that state by whom, in what order, and in what way the techniques are used to achieve or maintain some objectives.”*

Methods Play a very important role in Information System Development as they provide efficient way for developing the system for the successful completion of project. Methods Provide us:

- Best way to develop the system
- The guidance to use the steps in the development process.

There are two aspects of a method mentioned below:

**Product Aspect-** The product aspect ensure product standard and provides features for product development. Examples of product model are – *ER diagram, OOA, OMT*, etc. The product aspect of method provides:

Functional features- Functional features provide the set of building blocks so that complex concepts can be built from simpler concepts. Final product structure can be created as an combination of simple and complex concepts.

Non-functional features- These are the quality constraints some of which are mandatory and others are desirable. -

(Prakash, 97) has classified mandatory constraints as:

*Consistency Constraint:* To maintain consistency, if something holds then opposition does not hold.

*Completeness Constraint:* All the components necessary for the concept should be well structured are defined and put together. For example, an *entity* in ER diagram is complete, if it has, at least, *one attribute* and *one primary key* associated with it.

1. Process Aspect – The process aspect provides the route to be followed to ensure the efficiency of product development. For example, In (Coad and Yourdon, 91) *DFD must be completed before construction of design begin.*

### **3.1.2 Method Engineering**

No single developed method can be useful for all projects. Therefore organizations need to study the project characteristics and the organization characteristics to find out which method is most suited for the project in hand for the successful completion of project on time and within budget of the organization. This requirement necessitates ME.

There have been a number of proposals for developing project-specific methods like Fragment-based approach [9, 10], Contextual approach [28], Method generation using rules and GOPRR approach [19]. These approaches require instantiation of a Meta-model where the concepts of a method are made, instances of meta-model concepts or inter-relationships. Since instantiation is a tedious and time-consuming task, Gupta [8] had proposed a method

requirement specification language based on a simple meta-model i.e. Method View Model (MVM) which requires only limited concepts which mitigated the problem of instantiation to a great extent. The task of method engineers is complex in nature. In order to facilitate them there are proposals to provide a rich set of rules and guidelines to form a coherent method known as Architecture based software engineering. These proposals are analogous to architectural based software engineering domain proposals. They provide for the task to be performed in a more disciplined and cohesive way. The major approaches in Architecture based software engineering are Method Intension Architecture (MIA) and Architectural Centric Method Engineering (ArCME). These approaches have problems like suitable style selection and further composition of these selected styles. OPEN Process Framework(OPF) [32] solves these issues due to their flexible nature because it has rich collection of method fragments but fails to address a wide variety of concepts like branching, situation cataloguing and evolution tracing [6].

In recent times, the SME has moved to method configuration to construct a project-specific method. Method Configuration is a sub-discipline to SME and has been formulated and proposed by Karlsson et al.

The task of configurability in the method configuration is to first create a new model called a configurable model and then selecting only those parts of the configurable model which are relevant to the user's requirement. Configurable models use semantics of commonality and variability.

Coplien et al[15] define **commonality** as an assumption which is held uniformly across a given set of objects whereas **variability** is an assumption which is true for only some elements within a set.

In [16] the **variability is defined 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 [17] describes the process of configuration as a methodology performed to allow a business to balance their IT functionality with the requirements of their business.

Soffer et al. [18] consider configuration as an alignment process of adapting the enterprise system to the needs of the enterprise.

[19] Proposes a method that systematically develops requirements using commonality and variability in product line approaches.



Fig: 3.1 Generic Method Configuration Framework

### 3.2.1 Phases of Framework

The different phases of the Configuration framework are defined below:

**Phase 1:** This phase selects the appropriate method for the project in hand.

**Phase 2:** This phase in the framework retrieves the method configuration definitions in regards to the organizational characteristics. These method configurable definitions have a definite set of method concepts or practices along with an essentiality attribute.

**Phase 3:** This final phase is to refine the retrieved configurable definition for the project situation.

**3.2.2 Paradigm Selection Framework:** The work done in this thesis is the modification of phase 1 proposed by Dwivedi. She defined set of 22 project characteristics as input. The project characteristics are mentioned as below:

Metrics
Volatility of requirement
Complexity
Add-on Function
Necessary Function
Flexibility
Modularization of task
Time to Market
Amount of Requirement Known Initially
Clarity & completeness of requirements
Expandability
Coupling
Business Risk
Technical Risk
Operational Risk
Programmer's Capability
Application Experience
Reuse of existing code
Develop for future use
Platform volatility
Platform experience
Tool Experience
Team cohesion

Table 3.1 Project Characteristics defined by Dwivedi [24].



The input values were fed to the neural network. The paradigm selection was implemented using the neural networks.

**3.2.2.1 Algorithm:** The Proposed Algorithm to solve the stated problem is as follows:

**Step 1:-** Each Characteristic was assigned value from the five values.

Category	Very Low	Low	Medium	High	Very High
Value	1	2	3	5	8

Table 3.2 Input values for the Project Characteristics

**Step 2:-** Calculated  $S = (W_i * P_i)$  for  $i=1$  to 22 Where,  $W_i$  is the weight assigned to the  $i$ th metrics that is fixed (constant) and  $P_i$  is the input values chosen for  $i$ th characteristic that is variable (project specific).

**Step 3:-** Suitable method was selected on the basis of  $S$  calculated in Step 2. Output can range from 1 to 8. For output 1 to 4, Agile is best suited for project. For output 4 -5, Agile, traditional or combination of both could be used. For output 5 to 8, traditional method is best suited.

### Process using Neural Networks

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 twenty two neurons at input layer, three neurons at hidden layer and one output neuron. Please refer below figure.

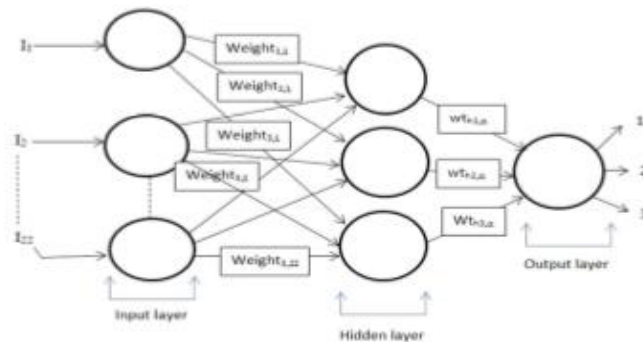


Fig 3.2 Problem Mapped as Neural Networks [6]

Output of the neural network was divided into three categories i.e 1, 2 and 3.

The output '1' of the network indicated that agile methodology is the best suited for given project parameters, output '2' indicated that either agile or non-agile or a combination of both methodologies can be used for the given project parameters. A '3' at the output indicated that non-agile methods are the best suited for the development of the project in-hand.

### **3.3 Modification in Method Selection Framework**

The work done in the thesis is the modification of phase 2 of Method Configuration Framework suggested by Dwivedi. The Method Selection Framework presented in this thesis make use of **Case Based Reasoning(CBR)** and **Cuckoo search** to find the suitable method for the project in hand. The framework and functionality of the proposed paradigm selection process have been elaborated in next chapter. Instead of making use of the 22 characteristics proposed by Dwivedi[6], we make use of 21 characteristics based which are selected based on below mentioned requirements kept in mind.

- Characteristics of Requirements
- Characteristics of Development team
- User's participation
- Project type and associated risk
- Based on Agile Configuration

#### **3.3.1 Merits of Proposed Algorithm**

To make the method selection framework proposed in [6] more efficient, there is proposal for new method selection framework. Merits of the proposed paradigm selection framework are mentioned below:

**1) Dynamic Nature of Database:** Since we use case based reasoning for method selection, the database of case bases grows as the new method is selected for some project in hand.

a) As a method is selected for a project, after the successful completion of the project, the database of the method selection tool is modified to contain the current project data as well to use this project experience for the upcoming projects.

b) Also the project success after applying the method selection would be analyzed if the project was successful or not. In case the project is not successfully with the outcome of the method selection tool, the database is modified accordingly for the experience for upcoming projects.

Update db

**2) Verification of output by Propositional Logic:** The output of the process is verified using propositional logic on the query case. With the help of propositional logic, the output is verified if it satisfies the requirements of Agile or Traditional Information System Development.

# Chapter 4

## 4 Proposed Method Prediction Process

This chapter provides the proposed method selection block diagram, projects characteristics, Components of Proposed solution, Case Studies. The project characteristics are identified based on which the method selection decision has to be carried out. The proposed method selection process identifies the project characteristics, process them using Case based reasoning and cuckoo search with the project base we have.

### 4.1 Method Selection Block Diagram

The Block diagram for the proposed method selection process is given below:

Figure 4.1 Framework for Selection of Software Development Method

The major components of method selection process are

- 1) **Past Information:** Past information contains the database of previous experience projects case bases.
- 2) **Present Information:** Present information refers to the query case of the project in hand for which the method has to be selected.
- 3) **Tool:** Tool is the method selection tool implemented in this project which provides the output as the method for the project in hand.
- 4) **Prediction:** Prediction refers to the output or result from the tool regarding which method should be adopted for the current project.

## 4.2 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 indentified 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.

### 4.2.1 Weight Distribution and Input values

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:

**Table 4.1:** Values for “volatility of requirement”

<b>% of volatile known requirement</b>	<b>Category</b>
< 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:

**Table 4.2:** Values for “Complexity”

<b>Initial Time</b>	<b>Category</b>
< 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 functionality 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).

**Table 4.3:** Value for “Flexibility”.

<b>Ease of modification</b>	<b>Category</b>
< 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:

**Table 4.4:** Metrics for “Modularization of task”.

<b>Extent to which the project can be made modular</b>	<b>Value</b>
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:

**Table 4.5:** Value for “Time to market”

<b>Time before the first release</b>	<b>Time to market</b>
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:

**Table 4.6:** 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.

**Table 4.7:** Value for “coupling”

Value for NCM	Modularisation 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:

**Table 4.8:** 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:

**Table 4.9:** 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:

**Table 4.10:** 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:

**Table 4.11:** Values for “Existing Code Reuse”

<b>Reuse of Existing Code</b>	<b>Category</b>
< 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):

**Table 4.12:** Value for “Develop as base code”

<b>Purpose of the project</b>	<b>Time to market</b>
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 average 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:

**Table 4.14:** Value for “Reuse of existing code”

<b>Customer Interaction with development teams</b>	<b>Category</b>
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:

**Table 4.15:** Value for “Testing Support”

<b>Reuse of Existing Code</b>	<b>Category</b>
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:

**Table 4.16:** 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:

**Table 4.17:** 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

**Past Information:** For our project the past project data is stored in a text file named “**sample\_input\_test\_cases.txt**” which consist of all the cases from the previous project experience. All the project characteristics and their values have been stored in this database.

**Present Information:** Present information refers to the current project situation and input metrics values for the current project. For our project this data is stored in “**query\_input.txt**”.

### 4.3 Method Selection Tool

Proposed tool takes the project characteristics, apply the proposed algorithm on the input query and output the result. There is omnipresence of Case based reasoning and Cuckoo search in the algorithm.

#### 4.3.2 Case Based Reasoning

**Case-based reasoning (CBR)** is the process of solving new problems based on the solutions of similar problems faced in past. For example, an auto mechanic who fixes an engine by recalling another car that exhibited similar symptoms is using case-based reasoning. So for an engineer copying working elements of nature, is like treating nature as a database of solutions to problems.

It has been argued that case-based reasoning is not only a powerful method for computer reasoning, but also a common behaviour in everyday human problem solving. We can say that all reasoning is based on past cases personally experienced. This view is related to prototype theory, which is most deeply explored in cognitive science.

##### 4.3.2.1 Process

Case-based reasoning has been formalized for purposes of computer reasoning as a four-step process:

1. **Retrieve:** Given a target problem, retrieve from memory cases relevant to solving the given problem. A memory case consists of a problem, its solution, and annotations about how the solution was derived.
2. **Reuse:** Map the solution from the previous memory cases to the target problem. This may involve adapting the solution as needed to fit the new situation with additional requirements.
3. **Revise:** After mapping the previous solution to the target situation, test the new solution in the real world and, if necessary, revise.
4. **Retain:** After the solution has been successfully adapted to the target problem and results are correct, store the resulting experience as a new case in memory.

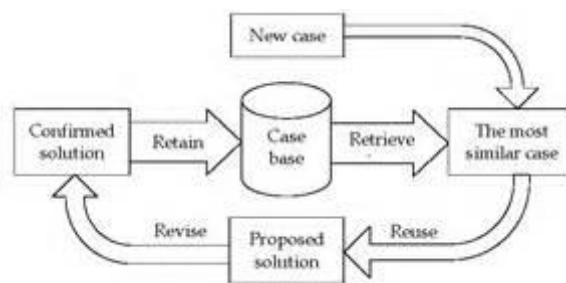


Figure 4.2: CBR Process

### 4.3.3 Cuckoo Search

**Cuckoo search (CS)** is an optimization algorithm developed by Xin-she Yang and Suash Deb in 2009. It was inspired by the obligate brood parasitism of some cuckoo species of laying their eggs in the nests of hosts of some other species. Some host birds can identify the cuckoo and involve in direct conflict with the intruding cuckoos.

Cuckoo search idealized such breeding behavior, and hence can be applied for various optimization problems. It looks like it can outperform other meta heuristic algorithms in applications.

Cuckoo search (CS) uses the following representations:

Each egg in a host nest represents a solution, and a cuckoo egg represents a new solution. The aim is to use the new and potentially much better solutions (cuckoo) to replace a not-so-good solutions (eggs) in the nests. In the simplest form of cuckoo search, each nest has one egg.



#### 4.3.4 Proposed Algorithm:

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: Agile/Hybrid/Traditional method

Initialize count\_case\_base to 1 Initialize total\_cases to 100. /\*

While (count\_case\_base <= total\_cases)

{

    Consider the current case from the case base (Say j).

    Evaluate Similarity Function( $F_i, F_j$ ) =  $\sum_{i=1}^n F(f_i, f_j) * w_i$ , where  $F(f_i, f_j) = 1$  if  $f_i == f_j$  else 0,  $n = 21$  – Project Characteristics,  $f_j$  = Problem Case's  $i$ th project characteristic input metric //for all factors

    Store the similarity value of  $F_j$  in another data structure named diffArr[]

    count\_case\_base = count\_case\_base + 1

}

Store the K cases in final\_case\_base having the maximum similarity values from temp\_case\_base //K is taken to be 10 here

Keep all these maximum values (best solutions)

ignore\_cases = total\_cases – final\_case\_base

Fractions (ignore\_cases) which are worst nests are abandoned to carry on next generation.

Rank the solutions / cases in final\_case\_base

{

    Find a correlation between the problem cases and the cases in final\_Case\_base//Correlation is found using the below mentioned Pearson's Correlation formula

    The maximum value of correlation function is ranked as best solution.

}

The case with highest correlation value is considered.

Post Process results {

    Apply all possible propositional logic condition(Propositional Logic defined next)

    Depending on the condition the outcome can be {Agile, Hybrid, Traditional}

}

Store the query case base in the memory case base for future reference.

End

Correlation between the query case and best case is found using Pearson' relation

So if we have one dataset  $\{x_1, \dots, x_n\}$  containing  $n$  values (for our algorithm,  $n$  is 21) and another dataset  $\{y_1, \dots, y_n\}$  containing  $n$  values (one from the query set and other from the nest(case) then that formula for  $r$  is:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

### **Propositional Logic:**

#### **Agile Projects:**

If the output from the best nest is Agile method, based on the manifesto of Agile Software Development following propositional logic should be successfully passed for the appropriate output.

**If<Amount of Requirement Known initially> "less than or equal to" Medium**

**If<Criticality of project> "greater than or equal to" Medium.**

**If<Time to market> "greater than or equal to" Medium.**

**If<Communication among the customer and the developer> "less than or equal to" Medium**

**If<Adaption to changes> "less than" Medium**

#### **Traditional Projects:**

If the output of the best solution found from the best nest(case) from the cuckoo search, then it has to successfully pass the following propositional logic to satisfy the traditional method approach for the project.

**If<Amount of Requirement Known initially> "greater than or equal to" High**

**If<Criticality of project> "greater than or equal to" Medium.**

**If<Time to market> "greater than or equal to" Medium.**

**If<Communication among the customer and the developer> "greater than or equal to" High**

**If<Adaption to changes> "greater than or equal to" Medium**

## 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 case base has been developed using the experience of [24]. Around 50 test cases have been collected in the case base. For which project the method is being selected, that is also added to the case base which helps to enhance the case base set 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:

### Data Structures:

//To retrieve the project characteristics, their values from the case base.

```
struct node{
    char name[50];
    float weight;
    short value;
};
```

//To track the maximum similarity case from the case base.

```
struct diff{
    int idx;//index of the case in case base
    float diff; //similarity of cases from query case
};
```

### Modules:

**CBR();**//This module is to process the query case.

**findSimilarity(int i);**//This method find the similarity between the ith case base (nest) and the query case.

**findBestNests();**//This method find the best nests(cases) based on their similarity value. The cases with maximum similarity are selected as best nests for current solution. The remaining cases are discarded for the current problem with probability  $p_a = \text{Ignore\_case}/\text{Total\_case}$ .

The discarded cases can be used for another problem.

**findCorelation();**//This method find the correlation between the best solutions and the query case. Corelation is found using Pearsons's Corelation formula mentioned in section 4.3.4.

This control flow of the tool is illustrated below:

Agile

Traditional

## Detailed Steps of the Algorithm:

*Step 1:* Here we have taken the case base which consist of 50 cases for our tool.

*Step 2:* Now we take input from the user for his query in the form of a text file, where he mentions all the input metric for the query case. The values of all the project characteristics are mentioned by the user.

*Step 3:* One counter is required to keep track of cases in case base, hence we maintain a counter name count\_case\_base to 1. This counter points to the current case in the case base.

*Step 4:* Here we keep the count of total cases in case base hence initialize a counter named total\_case to 50.

*Step 5:* For loop is maintained to encounter all the cases in case base.

*Step 5.1:* Point the counter count\_case\_base to the first case in the database (say j).

*Step 5.2:* Evaluate the objective function  $Sim_1^N f(F_i, F_j) = i$

Where  $F_i$  is the problem case,  $F_j$  is the case stored in case base.  $N= 21$ (Project characteristics). The function  $f(F_i, F_j) = 1$  if  $F_i = F_j$  else 0 and  $w_i$  is the weight of the attribute.

*Step 5.3:* For all cases their similarity function is evaluated and the function values is stored in a smaller database. This database is named as "diffArr". It contains all the function value when compared between problem case and cases of case base.

*Step 5.4:* Increment Count\_case\_base by 1.

*Step 6:* diffArr contains all cases similarity values. Now we need to find out which cases are quality solutions(most suitable). The most suitable cases are stored in final\_case\_base. From these final nest, cuckoo has to find out the best solution for the query case.

*Step 7:* The nests having maximum similarity value are considered as best solutions. All these best solutions/quality solutions are stored.

*Step 8:* Solutions not kept in the final case base are find out. Ignore count is calculated as total\_case-final\_Case\_base.

*Step 9:* The ignore cases are the worst nests. These nests are worst nests and abandoned for current cuckoo. These nests can be used for later cuckoos. Ratio can be found out as below:

$$pa = \text{Ignore\_cases} / \text{Total\_case}.$$

*Step 10:* Here we find the best solution for the current cuckoo. We rank the quality solutions so that the best nest can be chosen and selected as solution. To find out the current best following steps are performed

*Step 10.1:* Take each cases stored in final\_case\_base and find correlation of the cases with the query case. The nest from final\_case\_base which correlates maximum with the problem case is considered as the best solution.

*Step 11:* This is the final step of cuckoo search. Post processing is performed on the best solution found previously. Here the best cuckoo undergoes some port processing. Based on the outcome, the method for the query case is selected. Quite a few number of propositional logic is framed but only 2 of them is given below:

i)  $M = \text{"Requirement known initially are low"}$

$N = \text{"Criticality of the project is low"}$

$(M \wedge N) \rightarrow \text{"Agile"}$

*Step 11.1:* The conditions for Agile, Traditional methods are mentioned in section 4.3.4. They are applied on the outcome of the best solution.

*Step 11.2:* If condition is met following outcome is possible.

i) Agile

ii) Traditional

iii) Hybrid

*Step 12:* End of program.

### **Significance of Case Based Reasoning and Cuckoo Search:**

**Case Based Reasoning** came into when people were using experience to draw solution for new problems. Set of cases is developed which uses past experience (case base in our algorithm). This past experience is used to depict the solution for new problem. The tool fetches the most suitable case from the case base and uses the retrieved case for the solution of the problem.

As mentioned in the **Cuckoo search** algorithm, for finding the solution to a problem, quality nests are selected and worst nests are discarded with probability  $p_a = \text{Ignore\_Case} / \text{Total\_Case}$ .

Similarity function which calculates the similarity between the problem case and the case base cases is used to determine the quality solutions. The quality nests which have the maximum similarity with the problem case are selected for the current problem case and the remaining cases which are worst cases for the problem are discarded and can be used for other problems.

From the quality nests (solutions), the best nest (case) is derived using correlation which provided the solution for the problem case. Cuckoo Search procedure in the proposed algorithm is shown below:

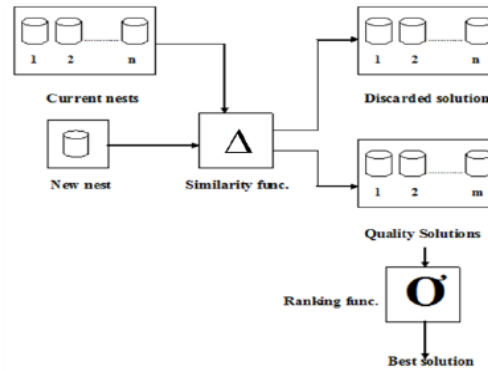


Figure 4.3: CS Procedure

**Components:** Components of the cuckoo search for the tool are elaborated below:

**Current Nest:** Case Base of our tool which.

**New Nest:** Query Case for which the method has to be selected which is provided as input to the tool.

**Similarity Function:**  $= \sum_1^n (F(f_i, f_j) * w_i)$ , where  $F(f_i, f_j) = 1$  if  $f_i == f_j$  else 0,  $n = 21$  (project characteristics). findSimilarity gives the similarity of all cases.

**Discarded Solutions:** Discarded Solutions are the solution which are discarded for the current query case and the number is given by  $Ignore\_case = Total\_case - Final\_case\_base$ .

**Quality Solutions:** Solutions with the maximum similarity function among all the cases in the case base are referred as quality solutions. These are determined by function findBestNest() in our tool as described above.

**Ranking Function:** Ranking function is the co- relation function (Pearson's Correlation) which finds the rank of the all the Quality Solutions and provide the case with the highest ranking in all the quality solutions. This case is our best solution and solution for the query input is the method used for this case. findCorelation() API gives the ranking to each case in case base and provide the case with highest ranking.

## 4.5 Case Studies

In this section, we present the case studies for Agile, Traditional and Hybrid method, 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:

Characteristic	Weight	Value(Project specific)
1Volatility of requirements:	0.02	2
2Complexity:	0.12	2
3Business Risk:	0.03	8
4Technical Risk:	0.08	1
5Operational Risk:	0.1	2
6Flexibility	0.02	8
7Modularization of Task	0.03	5
8Time to Market:	0.02	2
9Clarity and Completeness of Requirements:	0.05	3
10Coupling:	0.1	3
11Tool Experience	0.03	3
12Platform volatility	0.02	2
12Developer Experience	0.02	3
14Platform Experience	0.04	3
15Programmer's capability	0.03	5
16Existing Code Reuse	0.03	5
17Develop as base code	0.07	3
18Team cohesion	0.04	5
19Customer Collaboration	0.08	2
20Testing Support	0.05	3
21Requirements known initially	0.02	2



Similarity	Method used for the project (1 Agile, 2 Traditional, 3 Hybrid)
0.950000	1
0.950000	1
0.950000	1

The correlation among the best similar project in the database for the input query are found as below:

Plot of the similarity function with case base is shown as below:

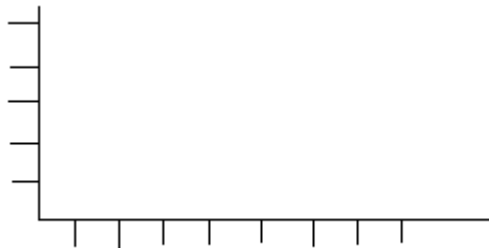


Figure 4.4: Plot for Similarity with cases

Corelation of Best Nests (Pearsons' Corelation)	Method used for the project (1 Agile, 2 Traditional, 3 Hybrid)
0.728326	1
0.662172	1
0.537367	1

Plot for Corelation with cases

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:

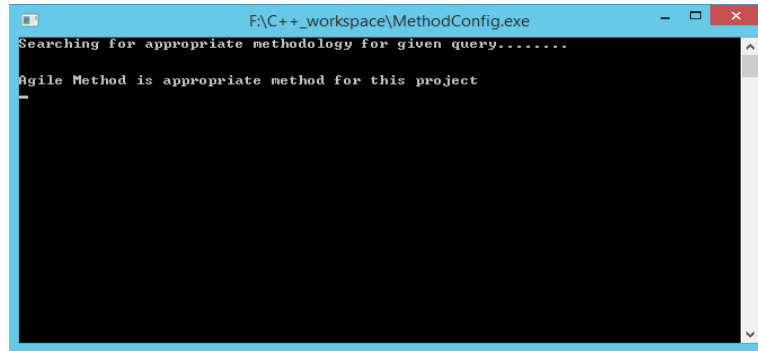


Figure 4.5: Output

## Case Study 2: Banking application

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

The correlation among the best solutions in the database for the input query are found as below:

<b>Similarity</b>	<b>Method used for the project (1 Agile, 2 Traditional, 3 Hybrid)</b>
0.400000	2
0.400000	3
0.400000	2
0.400000	2

Plot of the similarity function with case base is shown as below:

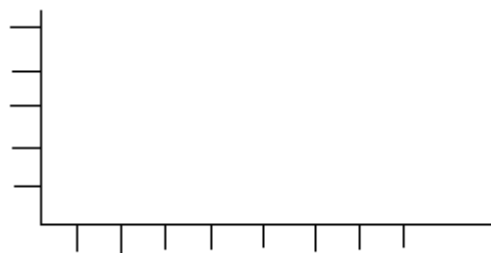
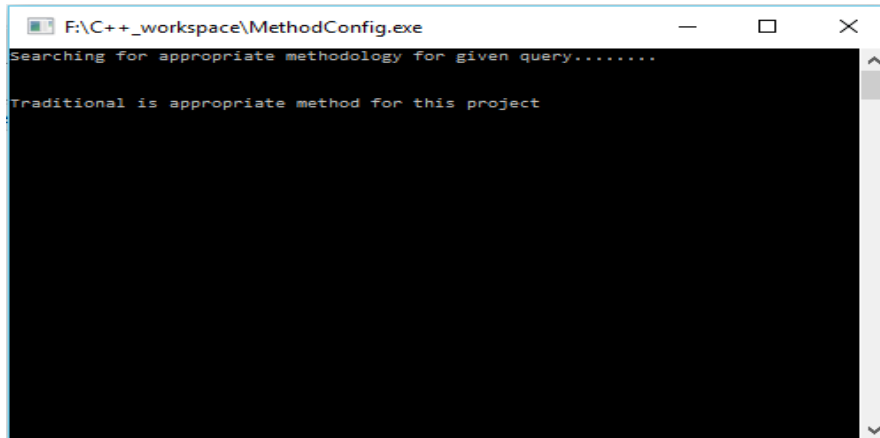


Figure 4.6: Plot for Similarity with cases

<b>Corelation of Best Nests (Pearsons' Corelation)</b>	<b>Method used for the project (1 Agile, 2 Traditional, 3 Hybrid)</b>
0.667636	2
0.497991	3
0.538636	2
0.432862	2

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:



```
F:\C++_workspace\MethodConfig.exe
Searching for appropriate methodology for given query.....
Traditional is appropriate method for this project
```

Figure 4.7: Output

# Chapter 5

## 5 Results, Contribution & 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 [21].

It shows an example of "College Management System" 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 choosing method paradigm important for successful project completion. Also one needs to configure project specific method for implementing quality implementation. As shown in research Dwivedi and all method paradigm selection is preprocessing of method configuration.

### **Contribution in the thesis:**

In my thesis, I have tried to overcome the drawbacks of the framework proposed by Dwivedi[24]. With the below mentioned benefits of proposed framework, I have tried to bring a better and more reliable framework for method selection for any project.

- Evolution of data base with every case study.
- Propositional Logic on the result.

A decision support system has been provided in this thesis to provide paradigm selection system. Project characteristics are identified which can be used to determine the method for a particular project based on characteristics like user participation, agile characteristics, project type and risk and characteristics of development team.

Input metrics to these characteristics are provided which helps the authorities to find out the suitable method for any project.

With the use of Case base, the tool provides the result based on the past experience of the developers and experts on the project. Also with the use of Propositional Logic, we validate our output for the input case.

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

**Conclusion:**

*"For the conclusion of the thesis, we found that there is a requirement to address the software paradigm selection problem and provide a better tool to solve this problem."*

The performance of the tool depends heavily on the repository of the case bases. The past experience or the knowledge should be as much more as possible to provide a better solution. If few cases are used, the chances of accurate output are less. Hence the requirement for the tool to work properly depends on the cases bases. Although we have used around 50 cases in the case base, this can increased for better output.

**Results:**

Results have shown that the proposed method is almost 100% accurate in providing the software development paradigm for the project. It is useful in providing the correct method for the project in hand.

## References

- [1] Coad, P., LeFebvre, 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](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.