Journal of Theoretical and Applied Information Technology
10th March 2015. Vol.73 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A FRAMEWORK TO SUPPORT EVALUATION OF PROJECT IN-HAND AND SELECTION OF SOFTWARE DEVELOPMENT METHOD

[1]DAYA GUPTA, [2]RINKY DWIVEDI
[1]Professor, Delhi technological University, Delhi, INDIA
[2] Research Scholar, Delhi Technological University, Delhi, INDIA

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.

Keywords - *Software Development Methodology, Impact Of Project Characteristics On Software Method Selection, Agile Software Development, Neural Network Approach For Methodology Selection.*

1. INTRODUCTION

It is never possible to construct a strong house without a strong foundation and a good architect. Similarly, in order to design and develop a reliable and usable software product that meets all the requirements of the stakeholders; a suitable development method is necessary. Selection of an appropriate software development method is very essential for failure free development of the software projects.

From ages non-agile methods are used for designing, developing and testing of projects in Information system domain. ER method is used to show detailed logical representation of data for an organization and uses three main constructs that is, data entities, relationships, and their associated attributes [1]. Process Flow Diagrams by Yourdon, are used to show the flow of data in a system. The UML provides languages for: visualizing, specifying, constructing, and documenting object oriented system [2, 3]. In some situations these approaches involve a significant overhead in planning, designing and documenting the system and were not found as an optimum solution for development, more time was spent on how the system should be developed than on the actual development.

Dissatisfaction in some areas with these heavyweight approaches led a number of software developers in 1990s to propose agile methods [4]. Agile methods are welcomed in industry by both managers and programmers [5].They allows the development team to focus on the software itself rather than on its design and documentation. Agile methods are intended to deliver working software quickly to customers and allow changes in requirements to be included in later iteration of the system [4]. Well known agile approaches include Extreme Programming [6], Scrum [7], Dynamic System Development Method [8] and Feature Driven Development [9]. For comparison between different agile approaches kindly refer [24].

Methods have their own limitations although very large systems like Facebook and some Google products are developed using agile methodologies still they are not well suited for development

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

systems with the development teams in different places and where there may be complex interactions with other hardware and software systems. Nor should agile methods be used for critical systems development where a detailed analysis and documentation of all of the system requirements is necessary to understand their safety or security implications.

Therefore, the need arises of a framework that supports the selection of an appropriate software development method.

In the present research we introduce a complete framework to assist software developers in selecting the appropriate software development method for the project-in-hand. The proposed framework is supported by decision support systems realized with the help of neural networks.

Project characteristics depend upon the situation in hand and will vary for project to project. For example, value of risk involved should be more for safety systems than for general purpose software. So, in our research we present measurement parameters to assign weight and values to each of these characteristics. Based on these, the proposed framework predicts the most suitable software development methodology for the project under consideration. Since process of evaluation is complex, neural network has been used for the evaluation process.

The outline of this paper is as follows. Section 2 foresees the related work on engineering various methodologies. Section 3 describes the framework and decision support system for selecting software development method. In Section 4 will discuss a practical case study for the proposed approach. In section 5 the implementation of the algorithm is shown with the help of feed-forward back propagation artificial neural network model.

2. RELATED WORK

Methods make the software development task easy, efficient, systematic and resourceful. But it is a fact that there is no universal method that can be applied to all projects since different projects have different characteristics and situations. This requirement creates home ground for Method Engineering (ME). Method Engineering has gained popularity with the first widely accepted definition by Brinkkemper. He has defined ME as a "discipline to design, construct and adapt methods, techniques and tools for an Information System Domain (ISD) project" [10]. There are number of proposals for developing project specific methods a good review of these can be found in [30].

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 [11]. 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 ones are Method Intension Architecture (MIA) [12] and Architectural Centric Method Engineering (ArCME)[13]. In these approaches there are problems like suitable style selection and further composition of these selected styles. Open Process Framework solves these issues due to their flexible nature but fails to address a wide variety of concepts like branching, situation cataloguing and evolution tracing [14]. For a detail description of these proposals and their comparison with other method engineering approaches kindly refer [15]. From last few years, Method configuration community has been working for techniques of choosing suitable methods based on project characteristics like complexity, risk involved, Programmer's Capability, Clarity and completeness of requirements, Business Risk, and many more[16,17].

Major limitations with these approaches were that they were mainly focussed on non-agile methods. Coherence of method formed or assembled is supported by Meta model that mainly models the non- agile methods.

As the time progressed, agile methods started gaining importance in industrial [18, 19] and in information system domain resulting in the need of strong frameworks, Meta concepts and Configuration environment for agile methods [20]. Qumer and hendersonsellers [21] proposed that it require five metrics i.e. flexibility, speed, leanness, learning and responsiveness for an agile method to be well defined. They further extend their proposal [22] by giving a framework to calculate agility in the given project. Dwivedi [23] gives configuration issues that need to be resolved before configuring an agile method and also the effort required for it.

Our present research work is one layer above these, that is before developing a development engine for engineering methods or before configuring methods (either agile or non-agile) one has to be clear with the selection of methods. In the reviewed literature we have found various useful proposals on agile and non-agile methods but the selection of software development method is still an

Journal of Theoretical and Applied Information Technology
10th March 2015. Vol.73 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

open issue for successful project completion. Selection of methodology has to be done prior to development of the project, we calculated project characteristics like complexity, modularization of task, business risk, technical risk, and programmer's capability to support our decision support system.

3. FRAMEWORK FOR IDENTIFICATION OF FACTORS LEADING TO THE SELECTION OF SOFTWARE DEVELOPMENT PROCESS MODEL

With the above objectives in mind we have developed a Development Method Selection Framework (DMSF) at a very abstract level. Fig. 1 depicts the component of the DMSF and their relationships.

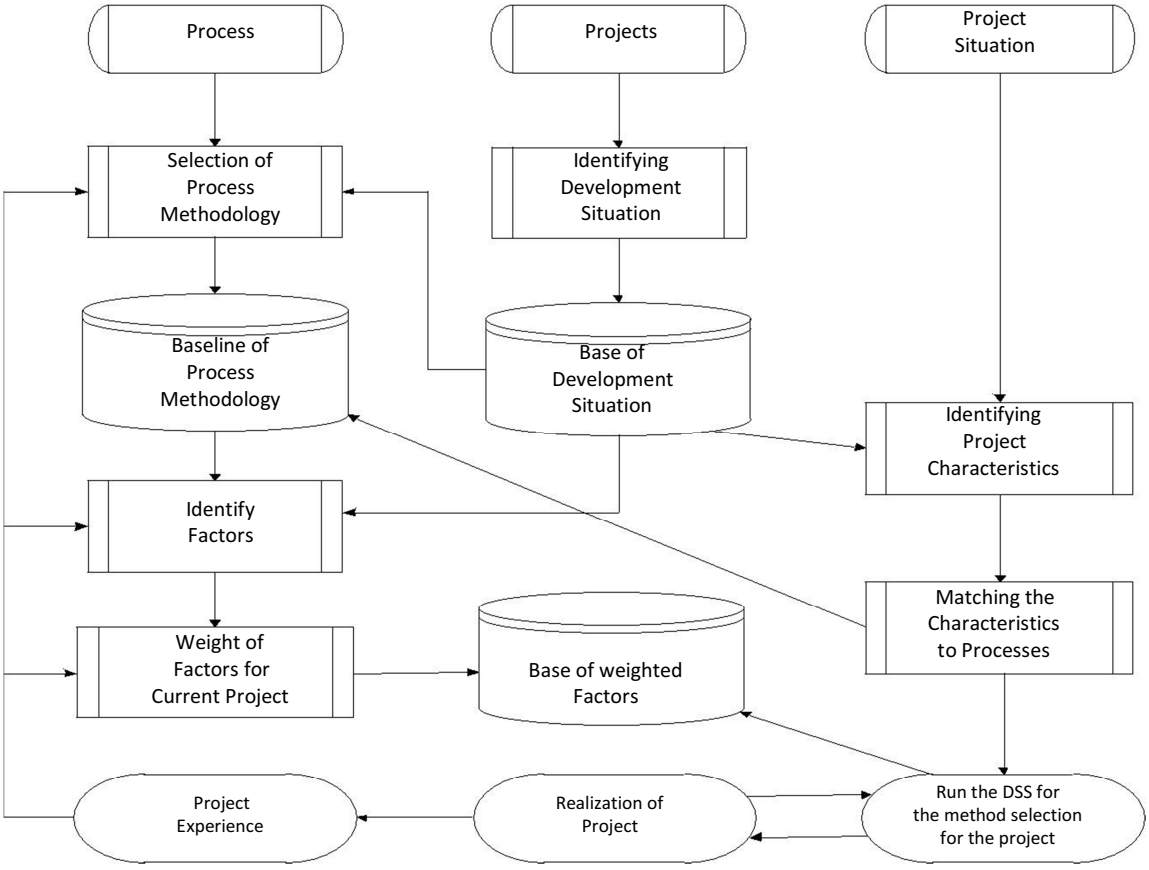

Fig. 1:- Framework For Selection Of Software Development Method.

Fig.1 shows a complete framework that depicts the sequence of operations in the project. There are three major dimensions in the framework: the process, the project and the project situation. The process as the name depicts deals with the current project-in-hand. The Projects are previously developed, fully explored and utilized work on projects stored for future project-configurations and project-reuse. The third important dimension is project situation or characteristics that are needed to be identified before staring a development process. These characteristics plays important role in method selection process thus needs to be explained in detail (see section 3.1).

The three data storage classes used for the storage of useful intermediate and final results for future reference are.

Base of the Development Situations: - For large number of already developed projects, the development situations are identified and "Base of development situation" is stored in the database.

Baseline of process methodology: - It stores the process methodology to be followed for the current

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

process, that is obtained by consulting base of development situation and the process under consideration.

Base of weighted factors: - It stores the weighted factors of the project and provides these factors as input to the decision support system at runtime.

For a new project to be developed, the development situations and project characteristics are identified by consulting *base of development situation*. The identified project characteristics are matched to processes by consulting *baseline of process methodology*. Finally, the weighted factors stored in the *base of weighted factors* are made available for the Decision Support System (DSS) to complete its defined functionality and concludes the result for the current project. In this paper, we give the detailed process of decision support system and its realisation however construction of method bases is taken as a e-prequisite for the approach.

3.1 PROJECT CHARACTERISTICS

The **project characteristics** of Method Engineering can be conceptualized in many ways as descriptors [26], contingency factors [27] situation factors [28] and context type [29] and project manager/ method engineer can assign should be able to determine the situation to these factors by assigning values. These factors are complicated and tedious for an application engineer to understand, in our proposal we define the project characteristics under four major domains. The outcome of these characteristics is then used to categorize the suitability of software development methodology for the project- in-hand.

On characteristics of Requirements: -Working with project requirements is a challenge, for some project requirements are volatile, difficult to understand or initially not complete for others they may be complex or critical. The selection of the software development method is highly dependent on the characteristics of the Requirements gathered in the requirement phase.

On characteristics of Development team: - Method selection is not fully dependent on nature of the project only; it depends upon the characteristics of the development team as well. Some of the team members have less experience, some have experienced but little domain knowledge, and some have expertise in the project field but lacks in the familiarity with the technology being used in the project. Training is also a major factor to consider, as to what extent the team has to be trained and how much recourses it requires in terms of time and cost leaves a major impact on project development.

On Users participation: - Selection of software development method also depends on the involvement of user during the software development. Sometimes it requires and is possible for the user to be present during all phases of the development life cycle. But the situation is not always same.

On Project type and associated risk: - Project type and associated risk plays an important role in the software development life cycle selection. Attributes like project funding, strict deadlines, high reliability, risk in terms of money, people etc. have a great impact on the decision of software development method selection.

3.2 WEIGHT DISTRIBUTION FOR DIFFERENT METRICS

The Decision Support System described has been developed as one of the deliverables of the aforementioned framework of selecting appropriate software development life cycle. The aim of the system is to improve software development practices by supporting the framework and tools for software development methods. Weight distribution has been done based on the knowledge and the practical experience of various developers in leading software companies and the available literature. It depends on the weight of the input category of a large number of identified metrics. These weights are assigned to input parameters that are divided into five categories depending upon the role of these metrics on the software development. Table I describes the numerical equivalent value for different categories.

Table 1:- Numeric Values For Different Input Category

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

The numerical value assigned to weight is normalised with values ranging from 0.0 to 1.0.

3.2.1 Weight distribution parameter and selection of input category for metrics of project under consideration.

Based on the knowledge and rationale available on the methodologies, metrics are decided. The characteristics of each metrics play important role in weight assignment. For the available set of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

metrics, the metrics having more support for non-agile has been given more weight and the metrics having more support for agile has been given less weight.

All input metrics cannot be measured on the same scale. Thus, there should be different measurement scale for different metrics on the basis of their characteristics.

1. Volatility of requirements:-

For the projects with frequently changing requirements the value of the metric will be high and value will be very low if requirements are stable.

Table 2: Showing Criteria For Selection Of Values For "Volatility Of Requirement"

Percentage of volatile known requirement	Category
Less than 10%	Very Low
10-19%	Low
20-29%	Medium
30-39%	High
Greater than 39%	Very High

2. Complexity:-

The project that requires more effort and is large in size is considered to be complex project. The development process for these projects is very complex and requires a detailed analysis and a systematic way of development. More value of this metric strongly supports the high complexity in the project. This metric can be measured by object-oriented metrics given by Chidamber and Kemerer[25]. The metrics useful for measurement of Complexity are WMC, NOC and DIT.

WMC: - Weighted Methods Per Class.

NOC: - Number of methods defined in a class.

It is difficult to reuse classes with many methods. WMC is useful to predict the time and effort required to develop and maintain the class.

Table 3:- Values For Complexity Based On Wmc

Percentage of classes more than 24 methods	Category
Less than 10%	Very Low
10-14%	Low
15-19%	Medium
20-24%	High
Greater than 24%	Very High

DIT:- Depth of Inheritance Tree

The deeper a class is in hierarchy, the more methods and variables it is likely to inherit, making it more complex. A recommended DIT is 5 or less. Excessively deep class hierarchies are complex to develop.

Table 4:- Values For Complexity Based On Dit

DIT	Complexity
1,2	Very Low
2,3	Low
4,5	Medium
6,7	High
Greater than 7	Very High

3. Business Risk:-

The metric is related to return on investment and customer satisfaction. For example, suppose customer is unsatisfied with the product after release and has no market value then organization should be able to release new version but the time consumption and cost factors will be raised up for the organization. More value of business risk supports for early and incremental release of the product, so any deficiency can be detected early.

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

4. Technical Risk:-

Technical risk involves the non-availability of developer or technology. During development, it may occur due to failure of tool or leaving of developer before completion of task. Technical risks are range from software malfunctions to electric failure to viruses that can completely shut down a firm's operation. These are serious risks that a firm must plan to face. Risk involved with installing new system also comes under technical risk. When a firm shifts to a new system without appropriate integration, the new system is not able to accomplish all that was promised. Sometimes it even performs poorer than the system it replaces. New system often requires employees to operate according to new processes. These may be difficult to learn, take training to execute correctly, or may even be outright resisted by employees who prefer the old way of doing business. So, the value of this metric should be chosen by considering all the above said factors wisely.

5. Operational Risk:-

This is the risk involved due to failure of some functionality of the project. If the impact of such failure is very high then, operational risk is high.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

For example, suppose in some safety system if any functionality fails then their impact will be very high so, operational risk is high.

Operational risk is the risk of loss resulting from inadequate or failed internal processes, people and systems, or from external events. Causes of operational risks include failure to address priority conflicts, failure to resolve the responsibilities, insufficient resources, no proper subject training, no resource planning and lack of communication in team. So, the value of this metric should be chosen properly.

6. Flexibility:-

Modifying the source code is very easy, but it is very difficult to manage the impact of changes on the other parts of the source code. Flexibility is the ease with which an operational program can be modified. Less weight is chosen for the methodologies with more flexibility.

Table 5: - Table Showing Criteria For Selection For "Flexibility"

5-9%	High
10-14%	Medium
15-19%	Low
Greater than 20%	Very Low

7. Modularization of Task:-

Modularization is very important for quick and easy software development. Modules can be developed in parallel and can be combined to deliver the final product. Less weight is assigned for the methodologies that support modularity to a great extent.

This metric can also be measured by object-oriented metrics given by Chidamber and Kemerer [25]. The CBO (Coupling between Object) is useful for measurement of this metric. Criteria for selection of values for "modularization of task" are given in table 9.

8. Time to Market:-

This metric signifies the time (in months) before that at least first phase (least functionality) of the product must be released. Less weight is assigned for the metrics that believes in incremental release of the product.

Table 6:- Table Showing Criteria For Selection Of Values For "Time To Market"

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

9. Amount of requirement known initially:-

For several projects initial exploration of all requirements is not possible due to lack of knowledge and many other factors. Some requirements are visible only after using the minimum workable (first release) of software. Less weight is assigned for the metric where the user involvement is high in subsequent phases.

Table 7:- Criteria For Selection Of Values For "Amount Of Requirements Known Initially"

Amount of requirements known initially	Category
Less than 20%	Very Low
20-39%	Low
40-59%	Medium
60-79%	High
Greater than 79%	Very High

10. Clarity and Completeness of requirement:-

It is a characteristic of requirement that defines well definiteness, unambiguous and clearly visible requirements. Clearly visible states that requirements documented are complete and will not require any further analysis.

For the requirements that are very well defined, clearly visible and does not require any further analysis are called as clear and complete.

Table 8:- Criteria For Selection Of Values For "Clarity And Completeness Of Requirements"

Amount of complete and consistent Requirements	Category
Less than 20%	Very Low
20-39%	Low
40-59%	Medium
60-79%	High
Greater than 79%	Very High

11. EXPANDABILITY: -

This metric is chosen based on the ease with the modifications made to the software at later stages. The more value of this metric has more support for agile methodology. The effort required in addition of new functionality to the already working software. The value of this metric can be selected based on the effort estimation for addition of new functionality.

12. COUPLING: -

Coupling is the degree of interdependence between classes. For good software, coupling should be as low as possible, a high value of coupling increases complexity and hence more value of this metric supports for complex methodologies.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The CBO (Coupling between Object) is useful for measurement of this metric.

CBO=Number of classes to which a class is coupled.

Two classes are coupled when methods declared in one class uses methods or instance variables defined by the other class. The value of CBO greater than 14 is too high. So, coupling will be very high for CBO greater than 14.

Table 9: - Table Showing Criteria For Selection Of Values For "Coupling"

Value for CBO	Modularisation of Task
Less than 2	Very Low
2,3	Low
4,5	Medium
6,7	High
Greater than 7	Very High

13. TOOL EXPERIENCE: -

This metric is chosen depending upon how much year of work experience the developer has on the tool to be used for development.

Table 10: - Criteria For Selection Of Values For Tool Experience

Developers experience on the tool to be used for project-in-hand. Time(in months)	Platform Experience
Less than 6 months	Very Low
6-12 months	Low
12-18 months	Medium
18-24 months	High
Greater than 24 months	Very High

14. PLATFORM VOLATILITY: -

How frequently the platform (Operating system on that product is being developed) is evolving. For example, if we are developing windows based project then it require frequent modification because Microsoft releases new version of windows frequently.

Table 11: - Criteria For Selection Of Values 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

15. APPLICATION EXPERIENCE: -

The work experience of the developer on the desired application (application may be java or c etc.).

Table 12: - Criteria For Selection Of Values For "Application Experience"

Time (in months)	Application Experience
Less than 12 months	Very Low
12-24 months	Low
24-30 months	Medium
30-36 months	High
Greater than 36 months	Very High

16. PROGRAMMER'S CAPABILITY:

This metric depends upon how much capable the programmer is for development of the project in terms of knowledge, vision and dedication towards the work.

We cannot define any specific range for this metric. It will depend on the capability of the development team and can be categorized on the basis of current situation.

17. ADD-ON FUNCTION:-

The percent of functions to be developed as Add-on functions. Add-on functions are fancy functions and don't directly contribute to the main application they are added to make the customer happy.

Table 13: - Criteria For Selection Of Values For "Add-On Function"

Percentage of functions developed as add-on functions	Category
Less than 20%	Very Low
20-39%	Low
40-59%	Medium
60-79%	High
Greater than 79%	Very High

18. Necessary Functions:-

These are essential functions that should be developed in a defined manner.

Table 14: - Criteria For Selection Of Values For "Necessary Function"

Percentage of functions to be developed as necessary functions	Category
Less than 20%	Very Low
20-39%	Low
40-59%	Medium

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

60-79%	High
Greater than 79%	Very High

19. Reuse of existing code:-

The amount of code reused, it is an important metrics and is very useful in calculating time and cost of the current project.

Table 15:- Criteria For Selection Of Values For "Reuse Of Existing Code"

Percentage of code reused	Category
Less than 20%	Very Low
20-39%	Low
40-59%	Medium
60-79%	High
Greater than 79%	Very High

20. Develop for reuse:-

If a project is to be developed as a base project then it should be developed in a defined way and should be well documented. Quality of such product should be very high.

Table 16: - Criteria For Selection Of Values For "Develop For Reuse"

Purpose of the project	Time to market
Developed as a base project (developed only for reuse)	Very High
Probability of being used in other project is very high	High
It may require additional functionalities at a later stage	Medium
No visible project found, that requires code of present project-in-hand	Low
Never be reused	Very Low

21. Platform experience:-

This parameter deals with how much work experience developers have on the platform to be used for current project**.**

Table 17: - Criteria For Selection Of Values For Platform Experience

Developers experience on the platform to be used for project-in-hand. Time(in months)	Platform Experience
Less than 6 months	Very Low
6-12 months	Low
12-18 months	Medium
18-24 months	High
Greater than 24 months	Very High

22. Team Cohesion:-

Ease of communication and interaction among team members is known as team cohesion. The value for this metric can be chosen on the basis of current situation of the organization and team members.

3.2.2 Proposed Algorithm

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

Step 1:- Assign input values to each metric for a given project from the five possible values.

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

Step 2:- Calculate $S= (W_i * P_i)$ 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} matrices that is variable (project specific).

Step 3:- Select the suitable methodology for the given project on the basis of the value of S obtained in step2.

The output will range from 1 to 8. For the values between1 to 4 agile methods are seems to be good for development and for 5 to 8 non-agile methods will do well. For some projects values lies between 4 and 5 indicating hybrid methodology may be used for the project-in-hand. The algorithm is demonstrated in the next section.

4. DEMONSTRATION OF ALGORITHM USING CASE STUDY

The DSS algorithm has been tested for four live projects to check the working of decision support system. Mobile App. Development (MAD), Air Traffic Control (ATC), ERP implementation for small and medium enterprises (SME's) and Banking application development.

Mobile Application Development: - Input values are assigned to each of the metric and the product of each input with their respective weight is calculated. Further, the sum of all these products will act as the parameter for decision making. For mobile application development the output is 2.71, indicating that agile methodology will do well for the development. Similarly, for Air traffic control the output is 6.13, for banking application its 4.41 and for ERP for SMEs development is 4.17. The complete weight distribution metrics for all the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

above projects can be taken from the authors.

Table 19:- Weight Distribution, Input Values, Their Product And Total Sum For Mobile Application Development

	Metrics	Weight	Input Values (Mobile App. Development)	Product of weight and input values
1.	Volatility of requirements	0.02	Medium(3)	0.06
2.	Complexity	0.1	Very low(1)	0.1
3.	Add-on function	0.02	Medium(3)	0.06
4.	Necessary Function	0.09	Medium(3)	0.27
5.	Flexibility	0.02	High(5)	0.1
6.	Modularisation of task	0.02	High (5)	0.1
7.	Time to Market	0.02	High(5)	0.1
8.	Amount of requirement known initially	0.05	Medium(3)	0.15
9.	Clarity and Completeness of Requirements	0.05	High(5)	0.25
10.	Expandability	0.02	High(5)	0.1
11.	Coupling	0.09	Very Low(1)	0.09
12.	Business Risk	0.03	Very High(8)	0.24
13.	Technical Risk	0.08	Very Low	0.08
14.	Operational Risk	0.1	Low(2)	0.2
15.	Programmer's Capability	0.03	Medium(3)	0.09
16.	Application Experience	0.02	Medium(3)	0.06
17.	Reuse of Existing Code	0.03	Medium(3)	0.09
18.	Develop for Reuse	0.07	Medium(3)	0.21
19.	Platform Volatility	0.02	Low(2)	0.04
20.	Platform Experience	0.04	Low(2)	0.08
21.	Tool Experience	0.03	Medium(3)	0.09
22.	Team Cohesion	0.05	Medium(3)	0.15
	Total(Sum of products)			**2.71**

5. REALIZATION OF THE ALGORITHM THROUGH NEURAL NETWORKS

To identify the suitable software development method, software project parameters need to be estimated. For this purpose we used neural networks to realise the same.

Why use neural network

A neural network is a processing device, either an algorithm or an actual hardware, whose design was inspired by the design and functioning of animal brains and components thereof. The computing world has a lot to gain from neural networks, also known as Artificial Neural Networks (ANN) or neural net. The neural networks have the ability to learn by example that makes them very flexible and powerful. Neural networks works like black-boxes that is to perform a specific task, there is no need to understand the internal mechanism of that particular task. This simplifies the work of the user to a great extent [32].

5.1 PROBLEM MAPPED AS ARTIFICIAL NEURAL NETWORK (ANN) MODEL.

The decision support system is simulated by three layer feed-forward back propagation neural network having input, hidden and output layer. Neural network tool available in Matlab [31] is used for training and simulation. Network consists of twenty two neurons at input layer (number of inputs), three neurons at hidden layer and one neuron at output l

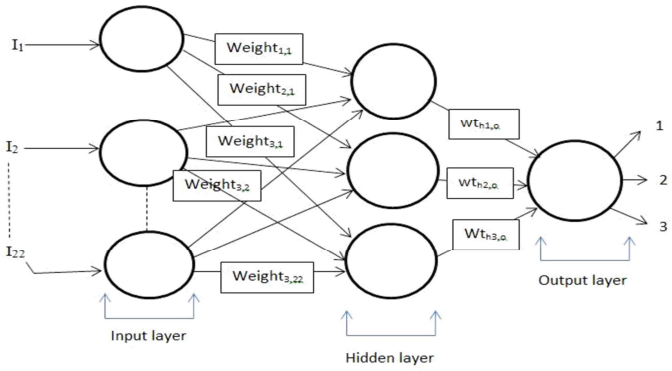

Fig. 2: Problem Mapped As Neural Network

Output is divided into three categories 1, 2 and 3. Here the mapping of the previous output range and present output range of the network is done as follows: - the output range from 1 to 4 is mapped as output '1', the output range from 4 to 5 is mapped as output '2' and the output range from 4 to 8 is mapped as output '3'. The output '1' of the network indicates that agile methodology is the best suitable development methodology for given project parameters, output '2' indicates 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 indicates that non-agile methods are the best solution for the development of the project-in-hand.

The ANN model

The model of ANN is specified by the three basic entities:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- The model's synaptic interconnections;
- The training or the learning rules adopted for updating and adjusting the connection weights;
- Their activation functions.

Connections or create networks:-An ANN consists of highly interconnected processing elements (neurons) such that for each processing element output is found to be connected through weights to other processing elements or to itself. The arrangement of neurons to form layers and the connection pattern formed within and between layers is called network architecture. There exist a number of neural network architectures for our research we used the Feed-forward Back-propagation Neural Network. This is very popular neural network architecture, because it can be useful in several different tasks. The first term, "feed-forward" describes the way patterns are processed and recalled by this neural network. Neurons are connected only foreword in a feed-forward neural network. There are connections from each layer of the neural network to the next layer (for instance, there are connections from the input to the hidden layer), but there are no such backwards connections exist. The word "back-propagation" defines the way the neural network is trained. The form of training used by Back-propagation is supervised training. In such a scenario the network requires sample inputs and estimated output to be provided. The estimated outputs being provided are then compared with the actual outputs for given set of input. Then the back-propagation algorithm for training takes a calculated error using the estimated outputs, after which weights of various layers are adjusted backwards i.e. from output layer to the input layer.

In our present problem we form our network consisting of twenty two neurons at input layer, two neurons at hidden layer and one neuron at output layer. The next step is to train the network through rules or through for self-updating and self-adjusting the weights allotted for the connections

Train Network:- For training, the large number of inputs and their corresponding targets are provided, the data is generated from the four projects that is, Mobile Application Development (MAD), Air Traffic Control (ATC), Enterprise Resource Planning Implementation in SMEs and Banking System. The epoch or maximum number of iterations that the network can perform during training chosen here is 1000.

Figure 3 shows a screenshot of the same in Matlab Software.

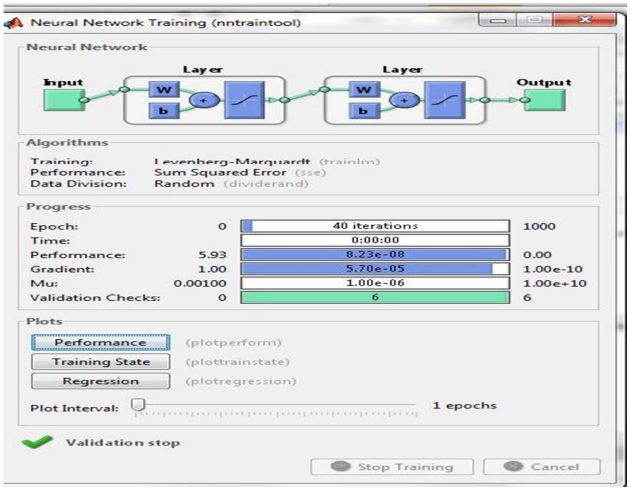

Fig. 3: Training Of Network

The Activation functions in a neural network:-The activation function in a neural network species the output of a neuron to a given input. Neurons are switches that output a '1' when they are sufficiently activated and a '0' when not. There are a number of common activation functions in use with neural networks. For our research, the activation function used is tangent sigmoid function. The equation for this function is tansig (n) = 2 / (1 + exp (-2*n)) - 1.

Table 20:-Final Weight Adjusted By Neural Network

	Metrics	Weight for input to hidden neuron 1	Weight for input to hidden neuron 2	Weight for input to hidden neuron 3
1.	Volatility of requirements	0.19376	0.89165	-0.40272
2.	Complexity	0.51028	-0.030383	-0.68257
3.	Add-on function	-0.29233	-0.099261	0.067508
4.	Necessary Function	0.58891	-1.0546	0.41018
5.	Flexibility	0.30301	0.29065	0.3696
6.	Modularisation of task	-0.48088	-0.042034	0.42978
7.	Time to Market	0.17812	0.41011	0.44175
8.	Amount of requirement known initially	-0.3488	0.21002	0.30634
9.	Clarity and Completeness of Requirements	0.29833	0.4106	0.2556
10.	Expandability	-0.14357	0.27188	0.1635
11.	Coupling	0.33175	-0.90402	-0.070217
12.	Business Risk	-0.57239	-0.60949	0.597
13.	Technical Risk	0.31599	-0.38916	0.38757
14.	Operational Risk	-0.34957	-0.35567	-0.29739
15.	Programmer's Capability	0.3903	-0.11019	-0.20646
16.	Application Experience	0.059763	0.21941	0.23257
17.	Reuse of Existing Code	0.20259	0.4015	-0.44519
18.	Develop for Reuse	0.29662	0.31654	-0.60989
19.	Platform	-0.27396	-0.035285	0.15748

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

	Volatility			
20.	Platform Experience	-0.27396	-0.035285	0.15748
21.	Tool Experience	0.36862	-0.1564	-0.29208
22.	Team Cohesion	0.31846	0.26222	0.2154

Since the network architecture used is feed-forward back propagation neural network, it adjusts the weights of the various layers from the output layer to the input layer. Final weight of the network after training and adjustments are:-

Bias to hidden layer neuron:-

[-1.3528; -0.40062; -1.2345]

Weight from hidden layer to output layer: -

[0.69954 -1.4034 -0.88563]

Bias to output layer neuron:

[0.11863].

6. CONCLUSION AND DISCUSSION

We have described here an extension process in the software development process by means of a Development Method Selection Framework, focussing on the selection of appropriate methodology for the software development.

The Development Method Selection Framework (DMSF) can be used to predict the methodological domain for the project-in-hand by gathering project characteristics in the form software parameters or metrics. Weights are assigned depending upon the role of these metrics on the software development. The process is helpful for the organizations to save on huge losses incurred upon the failure of projects caused due to wrong selection of process models. Proposed framework is supported by the decision support system realized with the help of feed-forward back propagation neural network architecture of artificial neural networks. In the future we intend to improve the selection process using different machine learning algorithms like Random Forest, Logistic Regression and Decision Tree to ensure more accurate and precise results.

REFERENCES

[1] P.P. Chen "A preliminary framework for entity relationship diagram", *Information modelling and analysis*, Elsevier science publisher 1983.

[2] G. Booch , *Object Oriented analysis and design,*2nd ed., Pearson Education, 2006.

[3] J. Rumbaugh, I. Jacobson and G. Booch, *the Unified Modelling Reference Manual*, 1999.

[4] Agile Manifesto, 2001. Manifesto for agile software development.

[5] K. Petersen and C. Wohlin, "A Comparison of Issues and Advantages in Agile and Incremental Development between State of the Art and an Industrial Case", *Journal of Systems and Software*, Vol. 82, No. 9, 2009, pp. 1479-1490.

[6] K. Beck and C. Andres, *Extreme programming explained: embrace change,* 2nd ed., Addison-wesley 2004.

[7] K. Schwaber and M. Beedle, *Agile software development with scrum*, Nouvelle editions, 2002.

[8] DSDM consortium, Dynamic System Development Method Limited.

[9] J.Hunt, "Feature Driven Development", *Agile software construction, Springer,* 2006, pp. 161-182.

[10] S. Brinkkemper, "Method Engineering-Engineering of Information Systems development Methods and Tools" *Information and software technology*, vol. 38, no. 4, 1996, pp.275-280.

[11] D.Gupta and N. Prakash (2001) "Engineering Methods from their Requirements Specification", *Requirements Engineering Journal*, vol.6, 2001, pp.135 – 160.

[12] N.Prakash and S.B.Goyal, "Towards a Life Cycle for Method Engineering", *In Proceedings Eleventh International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'07)*, 27-36, 2007. Trondheim, Norway, on June 11-15, 2007.

[13] Ahamed et al.(2008)Performing Assembly-Based Method Engineering by Architecture-Centric Method Engineering Approach, *In Second UKSIM European Symposium on Computer Modeling and Simulation, IEEE-2008.*

[14] B.Henderson-Sellers, R. France, G. Georg and R. Reddy, "A method engineering approach to developing aspect-oriented modelling processes based on the OPEN process framework", *Information and software technology*, vol.49, 2007, pp. 761-773.

[15] D. Gupta, R. Dwivedi, "Configuration from Situational Method Engineering", *ACM SIGSOFT, Software Engineering Notes*, vol. 37, No.3, 2012, pp. 1-11.

[16] 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.

[17] D. Gupta, R. Dwivedi, "A Step towards

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Method Configuration from Situational Method Engineering", *Software Engineering: An International Journal (SEIJ),* Vol. 2, No. 1, 2012, pp. 51-59.

[18] J. Cameron , "Configurable development processes", *Communications of the ACM*, vol.45,no. 3, 2002, pp. 72-77.

[19] B.Fitzgerald , G.Hartnett and K.Conboy "Customizing agile methods to software practices at Intel Shannon" *European Journal of Information Systems*, vol. 15, no.2, 2006, pp. 197-210.

[20] F.Karlsson F. and P.J.Ågerfalk, "Exploring Agile Values in Method Configuration". *European Journal of Information Systems*, vol.18, no.4, 2009, pp.300-316.

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

[22] 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.

[23] R. Dwivedi "Configuration Issues and Efforts for Configuring Agile Approaches-Situational based Method Engineering" *International Journal of Computer Applications*, vol. 61, no.17, 2013, pp. 23-27.

[24] D. Gupta, R. Dwivedi, *"Domain specific priority based implementation of mobile services- an agile way" International Conference on Software Engineering Research and Practice*, Las Vegas, USA, July 2012.

[25] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design", *IEEE transactions on software engineering*, vol. 20, no. 6, 1994, pp 476-493.

[26] Rolland C and Prakash N, *A proposal for context-specific method engineering, Proceedings of the IFIP TC8,WG8.1/8.2 Working Conference on Method Engineering*, 26-28 August, USA, Chapman & Hall, 191-208, (1996).

[27] Slooten, K. van, and S. Brinkkemper, *A Method Engineering Approach to Information Systems Development*. In Information System Development Process, 167-186, (1993).

[28] Harmsen, F., I. Lubbers, G. Wijers, and *Success-driven Selection of Fragments for Situational Methods: The S3 model. In: Pohl, K., and P. Peters (Eds.), Proceedings of the Second International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'95)*, Aachener Beiträge zur Informatik, Band 13, pp. 104-115, Aachen, (1995).

[29] Rebecca Denecker, Elena Kornyshova, Bruno Claudepierre, *Contextualization of Method Components, 4th IEEE International Conference on Research Challenges in Information Science (RCIS'10)*, Nice, France, p.p. 235-246, (2010).

[30] B.Henderson-Sellers. and J. Ralyte , "Situational Method Engineering: State-of-the-Art Review", *Journal of Universal Computer Science*, vol.16, no.3, 2010, pp. 424-478.

[31] MATLAB, http://www.mathworks.com/

[32] S.N. Sivanandam and S.N. Deepa, *Principles of Soft Computing*, 2nd ed., Wiley-India, 2012.