Applying Machine learning for configuring Agile Methods

Rinky Dwivedi and Daya Gupta

Delhi Technological University, New Delhi, INDIA
rinkydwivedi@gmail.com, daya_gupta2006@yahoo.in

Abstract

Situational method for satisfying organisation configuration definitions has become an important activity for many modern organisations. In the last few decades many situational method models have been developed but no model has proved to be successful at effectively deliver the customized light-weight methods fulfilling the organisation requirements. The paper introduces an Agile Method engineering approach to find the degree of suitability of agile methods for a particular situation. The introduced process uses associative clustering for finding the cluster of appropriate methods against the organisational requirements-in-hand. The specially designed fuzzy logic controller is used to extract the most appropriate methods from the cluster of appropriate methods. Fuzzy logic controller works in coordination with the databases that have been formed using the previous results and is being trained with the new knowledge. Finally two practical case studies have been discussed to describe how these concepts are applied in practice with industry specified requirements and results are being explored.

Keywords- Agile methods, method configuration, situational method for current organisational requirement

1. Introduction

Agile methods for software development emerged in mid 1990s [3], they intend to improve the software quality and responsiveness to changing customer requirements. Twelve golden principles have been defined in an agile alliance meeting conducted in 2001[4]. These principles provide support for the software development however; there is no guidance about how to configure agile methods for the situation-in-hand [19, 20]. This extends a need to develop a configuration process for agile methods that supports to select some practices of different agile methods for the current organisational requirements.

Method Engineering (ME) is a related stream of research that has been focused on tailoring of software development methods to the actual need of development context. ME has proved to be benefited for number of projects [14, 15]. In the present research we focus to introduce a Agile Method Engineering(AME) process to form situation specific agile method, the method thus formed is a blend of more than one configured method assembled together to form the desired method that fulfils the current organisation requirement.

AME process supports an Essentiality attribute for agile methods; this essentiality attribute can take two values either essential or variable. A similar type of proposal for traditional methods had been put forward in [9, 10] the proposal contained in that paper is static in nature (i.e., the methods stored in the method base are in the form of pre-made method configurations). Whereas, for agile methods, there is a requirement of a dynamic approach to configure (i.e., the essentialities within the method are decided dynamically for every organisational requirement) [11-13].

Given the above, we were interested in investigating the use and tailoring of agile methods in actual practice. Specifically, our research objective is to investigate
• How suitable agile methods are being selected (among the pool of all agile methods) for the current organisational requirement.
• Assigning weights to all suitable methods using fuzzy logic controller.
• Customizing highly weighted or most suitable agile methods found.
• How the individual customized methods needs to be assembled to form situated methods.

For the purpose, the proposed Agile Method Engineering process is divided into sub-processes that independently handle the task of method retrieval, finding suitable method and deciding the essential practices of the methods(or customizing the method) as per the situation-in-hand. The process flows as first finding the associative discovery cluster rules on the information stored in the databases, further retrieving the most suitable methods found, the retrieved methods are then assigned weights using the upper layer of fuzzy Logic Controller. The lower layer of Fuzzy Logic Controller is used to assign the weights to the practices of the methods; here the term ‘weight’ refers to the degree of suitability of the method for the specified set of requirements. The essence of the method customization process is to find essentialities in the methods that led to form organisational specific method.

The paper is organised in 5 sections, including this opening section. In the following section, we discuss basic concepts used in this paper, in order to understand the proposed Agile Method engineering process section three contains the step-by-step procedure required to form situated method. In fourth section we show the case studies utilizing the proposed Agile Method Engineering process. Finally the paper closes with the fifth section, containing a discussion and a conclusion.

**Motivation for the proposed agile method engineering process**

Although the introduction of agile development methods is an improvement over the traditional system development methods but since requirements of organisation vary and there is no single agile method that fulfils the entire set of requirements.

This motivates to create a blend of different agile methods based on the rich knowledge of the past usage of these methods under different requirement sets. The applicability of the method thus formed will be significantly improved than the existing methods because the assembled method thus formed contains the essentials of each constituent method.

**2. Basic Concepts of the Process**

The proposed Agile Method Engineering (AME) process is presented based on a set of concepts. These concepts build the foundation of the whole process. In the next subsections, the brief introductions of these concepts are given; the discussion provides the black-boxing of its contents. Internal view will be described in the subsequent sections.

**2.1 Defined Requirement**

The present method engineering approaches are anchored with several assumptions major ones are, it is possible for project members to explicitly specify the required situational method upfront and successfully communicate these requirements to the method engineer and these requirements do not change over the lifetime of a project[1,2]. However, in practice requirements are often ‘evolutionary’ in nature; a commonly cited method for these system requirements is the user-centred agile approach that helps in bridging between end users and systems developers. Another major problem with the requirements is ‘vagueness’-requirements are often vague and difficult to understand. To handle these, a support system is provided in the present research that converts the elicited vague requirements into a specific format that can be easily input to any tool support
designed for the method configuration. The entire set of converted requirements is termed as ‘defined requirements’ for our process. The process and the format of the defined requirements will be discussed in the white boxing of the process.

2.2 Method Configuration

Since projects differ in various dimensions, for example, with respect to development context, situation, complexity and granularity. Different proposals for how to create a situational method that fits the unique project have been put forth, ranging from formalized meta-methods [14] to architectural based [7] and further extended to more formal guidelines [8]. Present research reveals an Agile Method Engineering approach that uses method configuration process to configure individual agile methods, these configured agile methods assemble to satisfy the current organisational requirement definition. The process introduce the Essentiality attribute for agile methods, the essentiality attribute can take two values either essential or variable.

The method configuration process designed for agile method engineering is dynamic in nature, the presence of essential practices are purely dependent on the current requirements. In other words, Method configuration means to configure a particular agile method to various situated factors [16, 17, 18].

2.3 Situated Agile Method Formed

The heavily weighted or most suitable configured Methods are assembled to form the situated method. Situated method comprises of Method Part and Method Extension Part. 

Method part primarily contains the essentials of the first configured method, and may also contain the essentials of the other configured method (s), discussed in detail in later sections. The length of the method part is obtained by the MAX function applied on all the configured methods. The appended part that is, method extension part contains all the remnant essentials of the configured methods.

3. The Agile Method Engineering Process

The agile method engineering process is defined as a two part process (two sub-processes), this is introduced to make the process more understandable and less complex. Former is to obtain suitable methods for the situation-in-hand and later is to configure them by finding the essential practices for the most-suitable or highly weighted configured methods. The configured methods are put together to form the situated method.

The following section will discuss the white boxing of the process.

3.1. Selection Sub-Process

The first sub-process deals with finding the suitable methods and assigning weights to them, represented by Figure 1 and described as follows:
3.1.1 Gathering of the Requirements:

The process begins with the elicitation of the method requirements, the elicited requirements are often vague and difficult to understand. To handle this ‘vagueness’ requirements are converted into a standard format, specified by the framework. The formatted requirements are fed into the Database that serves two purposes: it creates a repository of requirements for future use and for generating the associative discovery clustering rules to find the cluster of suitable methods using some data mining tool like WEKA.

3.1.2 Format for Specifying the Requirements:

The elicited requirements are first represented in a format for further processing, these formatted requirements are then act as a keywords for method search or cluster operation. The format specifies that every keyword should have at least 2 words or should be split into 2 words where the first word should be an adjective and next one will be noun.

Like:
- Small followed by team : small team
- Complex followed by technology: complex technology

3.1.3 Finding the Suitable Methods

The step followed by gathering of the requirements is the methods retrieval and methods selection. For method retrieval operation, associative clustering is used to find the cluster of appropriate methods.

**Associative Clustering for method retrieval**

Clustering is basically unsupervised learning it finds “natural” grouping of instances given unlabeled data. Associative clustering is based on associative discovery rules that discover the presence of an object with respect to another object. It is used in market analysis basically where the vendors decide the placing of the items in the vicinity of the other items.

Association rules are if/then statements that help uncover relationships between
seemingly unrelated data in a relational database or other information repository. An example of an association rule would be "If a customer buys a dozen eggs, he is 80% likely to also purchase milk." An association rule has two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent. The project extends the concept of clustering, for generating the associations or relationships between the requirements and put forward the suitable methods from the cluster generated.

Example: If the set of requirements are ‘R1, R3, R4’, the suitable methods may be ’M2’, ’M6’ or ’M8’.

A special set of database is designed for the process that contains the set of keywords and the methods that suit the combination of those keywords, this database is a standard which has been made out of exhaustive study and research on various set of requirements and is always enriched with updated data with the usage of system.

**Database Schema**

**Database 1:**

**Keywords:** Permutations and combinations of the keywords

**Cluster of Methods:** The method names suitable for the combination of the keywords.

3.1.4 **Method Selection-selecting Suitable Methods**

Out of all the selected methods which are lined up after the previous step, the task is to select the most suitable ones-this is being carried out using fuzzy logic controller. It assigns membership to each of the methods depicting the degree of perfectness of the particular method for the defined requirements. In progress the highly weighted methods are selected for further processing.

Fuzzy logic controller assigns the membership degree to the agile methods (found from clustering operation) for the specified set of requirements. It takes the defined requirements and the suitable methods found, as its input and assigns the weight to the method. Rule Base here will be in the form of IF-Then Rules .Say if the set of keywords is “X” and method for this is “Y”, and then Value is “High”. Value tells the suitability of the respective method for the defined requirements. In progress the highly weighted methods are selected for further processing.

The value will be one of the Keywords: Low, Average, and High. The Range for keywords has been decided by the experts. Here the input will directly be in the fuzzified form that is in the linguistic terms. The rule base is applied to the fuzzified terms that fall under the work of inference engine. The output of the inference engine will be defuzzified to assign the crisp value.

One important thing is that it is not necessary that every time the same value is assigned to a method and further to its practices. As there is a range that is being fed into the fuzzy inference system so for the same set of defined requirements value may vary in the specified range.

3.2 **Configuration Process**

The next sub-process describes the procedure for assigning the weights to the practices of the highly weighted methods. Specifically it categorises the essential practices for the current organisational requirement.
3.2.1 Assigning Weights to the Practices

Finding the essential practices is the basis for our method configuration process. The framework provides the space for finding the membership of the practices for the most-suitable agile methods found for the situation-in-hand, using the lower layer of FLC. The process filters out the unwanted methods by selecting the highly weighted methods for further operation for example the methods with highest weights are discovered and are given as input with the set of requirements to find the membership degree of the practices of the method. The outcome is then fed into the Database 2 - a repository that can be used in future if the similar set of defined requirements came across and these databases will be updated periodically.

**Database schema**

**Database 2:**
- **Keywords:** Permutations and combinations of the keywords.
- **Weighted Methods:** Retrieved Agile methods with their respective weights.
- **Weighted Practices:** Weighted practices of the highly-weighted agile methods for the situation in hand.

The output of the FLC is stored in the second database; third column stores the weights of the practices for the highly weighted methods. FLC will be updated after every usage.

Example: If the set of defined requirements is \( X=\{ \} \) and Method is \( Y \) with weight \( A \) then Practice 1 is “Low”, Practice 2 is “Average”, Practice 3 is “Low”, Practice 4 is “High”. These “Low”, “High”, “Average” keywords will be assigned numerical values after the process of defuzzification.

3.2.2 Finding Essentiality in the Method for the Set of Defined Requirements

This subpart describes the operations performed on the weighted practices to find the essentiality pattern. For each highly weighted method, the mean of weight for all the practices is calculated separately. This is executed in order to find the essentiality pattern for the practices. Essentials are those that are mandatory to consider for the situated agile method. Assign ‘1’ to those practices, whose weights are greater than the nominal value, these are termed as essentials, ‘0’ to remnant practices whose weights are lesser than the nominal value these are termed as variables.

3.2.3 Assigning the Colour Scheme

In order to differentiate between the highly weighted configured methods, a different colour is assigned to each method. Say ‘red’ colour to method 1 and ‘green’ colour to method 2 and so on.
3.2.4  Assembling the Individual Highly Weighted Configured Methods

This process requires to perform logical OR operation but with an exception that if there are two 1’s of different colour both will be considered and will be appended to the result. If the numbers of practices are not same a ‘don’t care condition(X)’ is applied. The obtained outcome is the final situated method formed by assembling the individual configured method.

Consider '1' as representation for essential practices of method 1 and '1' for essential practices of method 2. A 0 represents the variable practice of either method.

There may arise 4 cases during the OR operation:
1. 1 OR 0: output will be 1 in method part.
2. 1 OR 0: output will be 1 in method part.
3. 0 OR 0: output will be 0 in method part.
4. 1 OR 1: output will be 1 in the method part and 1 in the method extension part.

3.2.5  Method Representation

Final output of the above step is represented into two parts: method part and method extended part. Method part includes the actual part that comes out of the OR operation whose length is the length of the maximum of the two method representations M1 and M2 and rest of the appended part which is called method extension part that falls under the 4th case of OR operation discussed in later sections, where it is required to keep the essential practices of both the methods, so this part contains the essentials of the second practice as essentials of the first method had already been included in the method part.

4  Empirical Grounding: The Illustrations

In order to evaluate the usefulness of the proposed methodology, case studies are used as a research method. Two case studies have been chosen for the same. These case studies are based on two projects carried out by two different organisations.

Case Study 1: A large software project developed by a software company to produce a stock market investment and analysing tool for analysing the fluctuations in the stock market and intelligently studies the areas for the investment in the market. It involves a huge team for its development which are further isolated into small teams with a self-organising feature. The project uses the complex technology for the implementation. The average duration of the project was 1 year. The software involves the changing requirement nature from the customer’s side. There is a need for documented requirements, to track the progress of the project and further to help during the testing phase.

Extracting the Requirements and converting into a specific format. Converting into the specified format (Table 1):

Table 1. Extracted Requirements for Case Study 1

<table>
<thead>
<tr>
<th>Number</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Large Software</td>
</tr>
<tr>
<td>R2</td>
<td>Changing Requirements</td>
</tr>
<tr>
<td>R3</td>
<td>Complex Technology</td>
</tr>
<tr>
<td>R4</td>
<td>Small teams</td>
</tr>
<tr>
<td>R5</td>
<td>Isolated teams</td>
</tr>
<tr>
<td>R6</td>
<td>Documented Requirements</td>
</tr>
<tr>
<td>R7</td>
<td>Iterative Developments</td>
</tr>
<tr>
<td>R8</td>
<td>Self-Organising Team</td>
</tr>
</tbody>
</table>
Feed these requirements to the database as a query, and it will give set of the best possible methods. Here set of the methods means more than one set; it will contain those methods also that are used only 10% percent of the times and such methods will be filtered out using FLC. For fresh requirements, rules will be generated using associative clustering. The general format will be “If ‘combination of keywords’ is X, then method is Y.

Further, the requirements and the retrieved methods are fed to the fuzzy logic controller to find their membership degree for the situation-in-hand. For the current situation, the two most suitable methods (method with high membership degree) found are shown in Table 2:

<table>
<thead>
<tr>
<th>Number</th>
<th>Method</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>Feature Driven Development</td>
<td>0.8</td>
</tr>
<tr>
<td>M2</td>
<td>Scrum</td>
<td>0.7</td>
</tr>
</tbody>
</table>

The retrieved suitable methods and set of requirements are again fed to FLC, to find the weights of the practices of these methods for the current organisational requirement. The output will be shown in Table 3 and 4:

For Feature Driven Development:

<table>
<thead>
<tr>
<th>Number</th>
<th>Practice</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Domain object modelling</td>
<td>0.8</td>
</tr>
<tr>
<td>P2</td>
<td>Developing by Feature</td>
<td>0.8</td>
</tr>
<tr>
<td>P3</td>
<td>Inspection</td>
<td>0.4</td>
</tr>
<tr>
<td>P4</td>
<td>Individual class ownership</td>
<td>0.7</td>
</tr>
<tr>
<td>P5</td>
<td>Feature teams</td>
<td>0.9</td>
</tr>
<tr>
<td>P6</td>
<td>Regular builds</td>
<td>0.3</td>
</tr>
<tr>
<td>P7</td>
<td>Configuration Management</td>
<td>0.2</td>
</tr>
<tr>
<td>P8</td>
<td>Progress report</td>
<td>0.4</td>
</tr>
</tbody>
</table>

For Scrum:

<table>
<thead>
<tr>
<th>Number</th>
<th>Practice</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Product Backlog</td>
<td>0.8</td>
</tr>
<tr>
<td>P2</td>
<td>Sprint Backlog</td>
<td>0.4</td>
</tr>
<tr>
<td>P3</td>
<td>Effort Estimation</td>
<td>0.9</td>
</tr>
<tr>
<td>P4</td>
<td>Sprint</td>
<td>0.8</td>
</tr>
<tr>
<td>P5</td>
<td>Daily Scrum meeting</td>
<td>0.3</td>
</tr>
<tr>
<td>P6</td>
<td>Sprint Review meeting</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Since, the method configuration process, is centred around the ‘identification of essential practices’ the next step is to find the mean (nominal) weight of the practices of individual method. The mean weight is considered as the nominal value-above which the practices are considered as essentials for the current situation-in-hand.

For Feature Driven Development:
Mean= (0.8+0.8+0.4+0.7+0.9+0.3+0.2+0.4)/8= 4.5/8= 0.5625 rounded off to 0.6.  

For Scrum:
Mean=(0.8+0.4+0.9+0.8+0.3+0.5)/7= 37/7=0.528 rounded off to 0.5

A ‘1’ in the configured method representation, shows that particular practice is essential and it should be included in the situated method whereas, a ‘0’ represents the variability in the configured method. Separate colour schemes are assigned to the methods.
for clear identification say, RED colour to the method Feature Driven Development and Green to the method Scrum.

Method M1 (Feature Driven Development) representation after all the calculations is:

```
1 1 1 1 0 0 0 0
```

Method M2 (Scrum) representation after all the calculations is:

```
1 1 1 1 0 0 1 X
```

In the above representation, notice a ‘don’t care’ condition is appended at the end of the second one, the purpose is to make equal length of considered methods. The final situated method is formed by performing the ORing operation on the considered methods.

There may arise 4 cases during the OR operation:

1. 1 OR 0: output will be 1 in method part.
2. 1 OR 0: output will be 1 in method part.
3. 0 OR 0: output will be 0 in method part.
4. 1 OR 1: output will be 1 in the method part and 1 in the method extension part.

Output after ORing is

```
1 1 1 1 0 0 1 0
```

There is no colour assign to ‘0’ in the situated method formed because of the fact that neither of the practices of the suitable methods has been selected in the output sequence.

Understanding the output

After all the calculations the final method representation shows the practices of the methods that should be considered according to the set of requirements given by the customer.

Considering the above illustration’s output

```
1 1 1 1 0 0 1 0
1 2 3 4 5 6 8 1 2
|------Method part--------|----Method Extension part------|
```

Second row represents the position of every bit in the output sequence. The output method has two parts namely:

1. Method part
2. Method extended part

**Method part** includes the actual part that comes out of the OR operation whose length is the length of the maximum of the two method representations M1 and M2, which is 8 here and rest of the appended part which is called **method extension part** that falls under the 4th case of OR operation, where it is required to keep the essential practices of both the methods, so this part contains the essentials of the second practice as essentials of the first method had already been included in the method part.
The Red colour represents the practices of method 1 and green colour represents the practices of the method 2. Only important point to consider is sequencing of the practices.

The above output can be explained as:
Position 1st and 2nd of the method part in the situated method signifies the presence of 1st and 2nd practices of the method 1; consider the value at position 3rd of the method part, which implies that 3rd practice of the method 2 has to be considered. Position 4 of the method part represents the presence of practice number 4th of the method 1. Then there are ‘0’s till position 8 that represents the variables-those practices which may or may not become a part of situated method. Further in the method extension part, the position number again starts from 1 and followed till the required length according to the OR operation. So 1 in green colour at position 1 of the method extension part represents the presence of 1st practice of the method 2 and similarly 1 in green colour at position 2 of method extended part represents the 2nd number practice of method 2.

Case Study 2: Case study 2 is a project that involves the upgrading of the existing code and it a large software project. It was being developed for a University which has many colleges located at various different places and each college administration used the software for the academy management placement management of the students. It involved the iterative and incremental development of the software, but on the other side it was a rapid development project. The project had seen an active user involvement during the development of the project because of the ever changing requirements of the customer. Since it was needed by colleges at different locations so teams were also spread at different locations for the software development so it was a distributed development project.

Extracting and converting the requirements into a specified format as shown in Table 5.

<table>
<thead>
<tr>
<th>Number</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Upgrading code</td>
</tr>
<tr>
<td>R2</td>
<td>Large project</td>
</tr>
<tr>
<td>R3</td>
<td>Active user-involvement</td>
</tr>
<tr>
<td>R4</td>
<td>Iterative Development</td>
</tr>
<tr>
<td>R5</td>
<td>Changing requirements</td>
</tr>
<tr>
<td>R6</td>
<td>Rapid development</td>
</tr>
<tr>
<td>R7</td>
<td>Distributed development</td>
</tr>
<tr>
<td>R8</td>
<td>Object-oriented approach</td>
</tr>
<tr>
<td>R9</td>
<td>Incremental development</td>
</tr>
</tbody>
</table>

Take the two most suitable (or three best methods) with the highest weights for further processing.

<table>
<thead>
<tr>
<th>Number</th>
<th>Method</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>Dynamic System Development Method (DSDM)</td>
<td>0.9</td>
</tr>
<tr>
<td>M2</td>
<td>Feature Driven Development (FDD)</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Weighted method practices for the current situation-in-hand.

For DSDM
Table 7. Weighted Practices of Retrieved Method 1

<table>
<thead>
<tr>
<th>Number</th>
<th>Practice</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Active User involvement</td>
<td>1.0</td>
</tr>
<tr>
<td>P2</td>
<td>Empowered team</td>
<td>0.9</td>
</tr>
<tr>
<td>P3</td>
<td>Frequent delivery of products</td>
<td>0.8</td>
</tr>
<tr>
<td>P4</td>
<td>Fitness for business purpose</td>
<td>0.7</td>
</tr>
<tr>
<td>P5</td>
<td>Iterative and incremental delivery</td>
<td>1.0</td>
</tr>
<tr>
<td>P6</td>
<td>Reversible changes</td>
<td>0.5</td>
</tr>
<tr>
<td>P7</td>
<td>Requirements are base lined at high level</td>
<td>0.4</td>
</tr>
<tr>
<td>P8</td>
<td>Integrated Testing</td>
<td>0.3</td>
</tr>
<tr>
<td>P9</td>
<td>Collaborative and cooperative approach shared by stakeholders</td>
<td>0.9</td>
</tr>
</tbody>
</table>

For FDD

Table 8. Weighted Practices of Retrieved Method 2

<table>
<thead>
<tr>
<th>Number</th>
<th>Practice</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Domain object modelling</td>
<td>0.8</td>
</tr>
<tr>
<td>P2</td>
<td>Developing by feature</td>
<td>0.9</td>
</tr>
<tr>
<td>P3</td>
<td>Inspection</td>
<td>0.4</td>
</tr>
<tr>
<td>P4</td>
<td>Individual class ownership</td>
<td>0.4</td>
</tr>
<tr>
<td>P5</td>
<td>Feature teams</td>
<td>0.5</td>
</tr>
<tr>
<td>P6</td>
<td>Regular builds</td>
<td>0.9</td>
</tr>
<tr>
<td>P7</td>
<td>Configuration management</td>
<td>0.4</td>
</tr>
<tr>
<td>P8</td>
<td>Progress reporting</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Identification of essential practices - for DSDM:

Mean=1.0+0.9+0.8+0.7+1.0+0.5+0.4+0.3+0.9=6.5/9=0.72 rounded off to 0.7

For FDD:

Mean=0.8+0.9+0.4+0.4+0.5+0.9+0.4+0.8=5.1/8=0.635 rounded off to 0.6

Method representation after all the calculations is

Method M1 (DSDM)

Method M2 (FDD)

Performing OR operation on the two method equivalents.

Output:

5. Conclusion

An agile method engineering approach has been developed to find the degree of veracity of agile methods for the specified set of requirements. This evolutionary
approach opens the paths to utilize the revolution brought by the concept of agility. Such a method can be used under the circumstances when a single agile method does not fulfils the complete set of requirements of the customer, so there is a need for the blend of parts of more than one agile method, which the introduced method process helps to achieve. The process supports to specify the requirements in laymen language and finds the suitable agile methods for the same with the practices that need to be followed. The aim is to deliver situation specific agile method for the current organisation requirement.

In the introduced method, there is a dependency on the sequencing of the practices and if there is any change in their sequencing, the situated method formed will turn out to be useless. It can be improved to remove this dependency in the future and make this method more flexible to work upon.

References


Copyright © 2015 SERSC