From Situational to Functional Method Engineering

S.B. Goyal, Supervisor: Prof. Naveen Prakash

goyalsb@yahoo.com

GCET, 1 Knowledge Park Phase II, Greater NOIDA, U.P. INDIA 201306

Abstract The problem of selecting a good candidate method is a big issue in situational method engineering. We believe that the solution to this can be found not in the structure of a method itself but in the work to be done, the task to be performed by the method. We propose a method development life cycle with requirements, design and construction engineering stages. For the first, we propose the notion of a method intention, an abstraction of method functionality. At the Design stage we introduce the notion of method functionality and propose ‘method architecture’ to represent this functionality. Thus, method engineering becomes Functional Method Engineering (FME). A functionally close method architecture is selected, adapted, enhanced, restricted as needed. The task of construction handles the putting together of method features and structuring the method.

Keywords: Method, Meta-model, Fragment, Component, Method Engineering

1 Introduction and the research question of the research work

Many Information Systems Development Methods exist. Since no ISDM exists that is best in all situations, situational method engineering (SME) has been proposed [Bri96] for developing or tailoring ISDMs for specific projects. Harmsen [Har97] defined SME as “… directed towards the controlled, formal and computer-assisted construction of situational methods out of method fragments”.

Several SME approaches have been described in the literature. The approach of Brinkkemper [Bri98] relies on the experience and knowledge of the method engineer in ensuring well selected components and building the required method. Grundy [Gru96] proposed an integrated facility for carrying out method development from scratch, method modification and method reuse. Gupta [Gup01] proposed a instantiation algorithm that formed the basis for method development from scratch, by modification and by assembly.

Ralyte [Ral03] suggests that method engineering is facilitated if the intention of the method can be determined and raises some questions: a) how can assurance be provided that the method to be enhanced, extended, or restricted is a good candidate method? b) What are the chances that at the method engineering intention stage, the method shall have to be discarded because its adaptation is very difficult? c) Should not some more exploratory work be done before committing to setting up method adaptation intentions?
Our research is directed towards answering these questions. Our belief is that the answers to the questions raised can be found not in the structure of a method itself but in the work to be done, the task to be performed. Whether method, M is a good candidate for adaptation, enhancement, restriction can be determined if it does similar work to that of the one being engineered. Further, the closer the similarity the less are the chances of discarding M. Thus, for us, the notion of method requirements is the key to the solution of the problem.

Broadly, our solution is to treat method engineering as having a full cycle consisting of Requirements, Design and Construction Engineering. Our view is that current state of the art in situational method engineering addresses the construction engineering phase. The other two stages help us in doing further ‘exploratory work’ referred to by Ralyte. At the Design stage we introduce the notion of method functionality and propose ‘method architecture’ as an abstraction of this functionality. Thus, method engineering becomes Functional Method Engineering (FME). A functionally close method architecture is selected, adapted, enhanced, restricted as needed. The task of construction handles the putting together of method features and structuring the method. Thus, we see a difference between structural method engineering and functional method engineering. This shall be elaborated later.

The Requirements Engineering stage is upstream to Design Engineering. At this level, we introduce the notion of a method intention. Thus, to the ISDM requirements engineer, method engineering is intentional in nature. Once methods with similar intentions to the one to be engineered is found, a menu of methods to be adapted, enhanced, restricted is determined. This is further refined in the Design stage where architecture matching occurs. Again, a residue of methods is found and at this stage the architecture of the new method emerges as a set of functions connected together. Finally, this architecture is engineered from building blocks taken from the residue.

It can be seen that progressive selection in the Requirements and Design stages has a) the potential to give us the assurance that the method to be enhanced, extended, or restricted is a good candidate method b) that inappropriate methods, those having dissimilar intentions and dissimilar architectures are rejected before the actual construction stage and therefore reduces the possibility of rejection and, c) enough exploratory work is done before committing to method features.

We can now state the aim of the thesis. We wish to move to intentional method engineering so as to explore the context of situational method engineering more fully. As a result, method selection for adaptation shall be more appropriate and give assurance that the SME task is progressing purposefully. The chance of method rejection at later stages shall be considerably reduced.

To do this we develop a 3-stage life cycle for intentional method engineering as shown in the next section. We treat this life cycle in two parts, the bottom two layers constitute the functional level of method engineering dealt with here. Once this is developed we expect to put on top of the functional level an intentional level that shall
still further raise the abstraction in terms of which method requirements shall be expressed.

2 Preliminary Results

In this section we present our method development life cycle to bring out the notion of intentional method engineering. In this life cycle we introduce a method architecture matching phase that corresponds to our view of functional method engineering. Thereafter, the notion of method architecture is explained through a meta-model and a set of operations is defined that enables architecture matching. Details of this work can be found in [Pra07, Pra08].

2.1 Method Development Life Cycle

We have developed method development life cycle, MDLC [Pra07] for method development. As shown in Table 1, the Requirements Engineering stage consists of Intention Matching. First, the intention of the method To Be is elicited. The intention matching process uses synonym matching to identify intentionally similar methods that reside in the method repository. These methods become candidates for the second stage of the MDLC.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Process</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Intention</td>
<td>Intention of the method To Be obtained from</td>
<td>Intentionally similar methods to the method</td>
</tr>
<tr>
<td>Engineering</td>
<td>Matching</td>
<td>Interviews etc.</td>
<td>To Be</td>
</tr>
<tr>
<td>Design</td>
<td>Architecture</td>
<td>Architectures of intentionally similar methods</td>
<td>Architecturally similar methods</td>
</tr>
<tr>
<td>Engineering</td>
<td>Matching</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Construction</td>
<td>Organization</td>
<td>Organizations of architecturally similar method</td>
<td>Method To-Be</td>
</tr>
<tr>
<td>Engineering</td>
<td>Matching</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: The Method Development Life Cycle

In the Design Engineering stage, the method engineer retrieves the architecture of each candidate from the method repository. That subset of these components and inter-relationships is selected which best meets the broad functional needs of the method To-Be. Such selections are made from all the candidate methods and are synthesized together into the architecture of the desired method.

In the Construction stage the architecture is populated with instances of the method features needed in the method. These features are in accordance with the meta model used, fragment, contextual, decisional et.

2.2 Method Architecture

We have defined method architecture [Pra07] as an abstraction of the method that identifies its components and inter-relationships to highlight the externally visible functionality of the method. We use the class abstraction as a way of formally defining method architecture as follows:
Method architecture = \{\text{method} | \text{method performs Function F}\}

A method architecture [Pra08] is named and the name reflects the function performed by the class of methods abstracted in the architecture.

The method architecture meta model can be summarized as follows. An architecture is implemented as method organisation and this organisation shows the features of the method and their inter-relationships. An architecture can be atomic or complex and architectures can be related to one another by links. These links form a successor-predecessor relationship between architectures. Links are labeled by their execution properties, Urgency and Necessity respectively. The full meta-model and its impact can be found in [Pra08].

2.2.1 Method Architecture Matching Process
In design engineering stage, the method engineer retrieves the method architecture of each candidate method. As shown in Table 1, these candidates are obtained from the intentional level. The new desired method architecture is built by matching the functional needs of this method with the functional expressions of the candidate methods, selecting and adapting those that are needed and discarding the useless ones. The following operations have been proposed to do this:
- i) Given a named architecture, rename it,
- ii) Create a new architecture,
- iii) Delete an existing architecture,
- iv) Nest, N architectures within another one,
- v) Un-nest architectures so that a nested architecture becomes visible at a higher level of nesting,
- vi) Change a link type,
- vii) Make a sequence of architectures by introducing an edge between them and defining their link type,
- viii) Eliminate a sequence.

2.3 Functional Method Engineering
Let us bring out the difference between SME and FME being proposed here. In the fragment based SME proposal [Bri98], we have two fundamental elements a) product and their structures b) procedures and their execution order to develop the products. It is clear from (a) that interest is the structure of products. Similarly, since the structure of a process is largely determined by the order of execution, interest is in process structure. Therefore, we can conclude that SME is centered around the structural aspects of methods.

This focus on engineering the structure of methods de-emphasizes what the method does, what task it is good for. In fact, the determination of whether the method structure can carry out the project task at hand is based on the experience of the method engineer. In other words, the method engineer determines the task of the project by some ad-hoc means selects the appropriate method structures and then assembles these together.

FME puts method structure subordinate to method functionality. FME asks for an explicit determination and representation of method functionality in the form of method architectures. It is only after the architecture has been built that the issue of method structure is to be considered. In this sense, SME occupies the, downstream, construction engineering stage of our life cycle.
3 Ongoing and Future Works

A method organization [Pra07, Pra08], represents method features and their interconnections. Interest here is in the method concepts, inter-relationships between concepts, constraints, heuristics, guidelines and other such features of a method. It can be seen that method organisation represents the structural aspects of methods. Alternatively, it defines the input to be given to a CAME tool to engineer/implement the required method. We have selected the generic method model for representation of method organizations.

In order to finalize the construction-design stage interaction, we are also developing a set of operations for performing organisation matching. This will allow us to adapt method organizations determined by architecture matching to our structural needs.

Thereafter we propose to develop the intentional level. Method Intention refers to the goal that the method fulfils. We shall develop the method intention meta-model and provide a precise definition of an intention. We aim to associate an intention with each method and as for, architecture matching, develop the method intention matching operations. Finally, the link between the intentional and architecture levels shall be defined. Thus, the entire life cycle of Table I shall be covered. Once tool support is available, we shall experiment with our technique to establish its usefulness.

References