A Complete Method Configuration Process for Configuring Project-Specific Methods

Rinky Dwivedi*, Daya Gupta
Delhi Technological University, New Delhi, India.

* Corresponding author. Email: rinkydwivedi@gmail.com.
Manuscript submitted October 21, 2014; accepted March 8, 2015.
doi: 10.17706/jsw.10.5.599-615

Abstract: In this paper we are going to present our Method Configuration process in its generic form. A Method Configuration process with respect to the instantiation of a Configurable Meta model and Method configuration to the finest level of granularity has been envisaged in the present research. Emphasis is on a different selection of method elements in the method component. Hence, it can act as a bridge between the necessary focus on the method artifact and the project team during method engineering, which can improve the use of methods during projects.

The ultimate goal is to facilitate the user by simplifying the tedious task of method configuration. For information system domain, the knowledge of conceptual structures is combined for defining the essentialities in a method. However for practical applications, the observations and demands form the basis of defining the essentials in the domain. The customized method is formed by considering the purposes of the artifacts chosen for the situated method and dependencies that need to be satisfied in configuring the method.

For validation purpose, we use an atomic and a compound method as a case study object, but the principles described should apply to a wide range of systems engineering methods since what we discuss are general underpinning principles, not the method restricted as such.

Key words: Method configuration process, situational method engineering, variability, configurable meta model, project characteristics.

1. Introduction

In the Past years variability is gaining importance in every aspect of software development, the notion has also been extended in Situational Method Engineering (SME) science, as different method components are stored in a method base or repository and used in a given situation in function of the project characterization. The notion of variability is defined as the "ability of a software system to be changed, customized or configured to a specific context" [1].

In SME domain, the term coined for variability is Method Configuration [2]. Configurability is a sub-discipline to SME and has been formulated and proposed by Karlsson et al in [3]. 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 [4], [5] defined a high-level configurable construct for their method configuration process. The configurable concept proposed by them lacks the issues of genericity and granularity, to overcome these constraints they propose the idea of combining activity theory and method engineering [6]. Since the previous proposals on method configuration centred on the one single base method, reduced the scope of the method generation for every new project.

The task undertaken in this paper is to design a complete configuration methodology for information system domain methods. One such proposal on Method Configuration has been attempted in [7], [8]. The idea put forth and elaborated in this paper, in contrast with the previous approaches, that support one single base method for the process, we rely on a method base of configurable method components of distinct granularity. (By granularity we mean atomic methods or compound methods). The configurable method components are the pre-made method configurations and are an efficient way to achieve genericity and granularity. The difference between previous constructs and the proposed one is that it is capable enough to support Essentiality in a method.The Essentiality attribute can take two values either common or variable. Common are the non-configurable part of the method component and can be defined as the essentiality in the method component, which if omitted method will lose its identity.

We use Conceptual Structure knowledge-introduced in [10] and further extended for generic Meta model [11], as a deciding factor for the essentiality attribute in our proposal.

All the Meta concepts of configurable method component are supported by configurable Meta model that is specially designed to suppress details during the Method Configuration process and to emphasize the task of constructing the situated method. Our configuration process is supported by a tool support to make efficient use of this approach

The main contribution of the present research is to solve following issues

- The design of a Meta model needs to model the concepts of a configured method.
- The second issue is what 'good component' is and what the 'right granularity' is.
- The third is regarding the Selection of a configurable method component for the situation –in-hand.
- Lastly, the complete process of configuration to reach coherent desired method.

The proposed method configuration process is supported by a method base of configurable method components. The implementation of the process makes it possible to have an interactive method configuration process where method user can construct a suitable situational method.

The paper is organized as: Section 2 foresees the related work on method configuration. Section 3 describes the configurable Meta model consistent with the generic Meta model. In Section 4 the macro level architecture of Complete Configuration process is described. Section 5 proposes the design of a method base, capable of residing configurable and configured method. In section 6, steps for complete method configuration process are explained. The paper closes with the introduction of method extension process which may be required to form situated method.

2. Related Work

It has now been proved that there is no universal method that can be applied to all projects since different projects have different requirements [12], [13]. SME is a solution offered to the problem of the selection of the "most appropriate" methodology for an organization and/or its projects [14].

There are number of proposals for developing project-specific methods such as Fragment-based approach [15], [16], Contextual approach [16] and Graph, Object, Property, Role and Relationship (GOPRR) approach [17]. All these approaches require instantiation of a complex Meta-model, where the concepts of a method are made instances of meta-model concepts or inter-relationships. Gupta [9] had proposed a method requirement specification language rooted in a simple meta-model i.e. Method View Model (MVM) having only limited concepts thereby simplifying the task of instantiation.

In recent times, the SME has been extended to include Method Configuration to construct a project-specific method. During our research we come across some proposals on configurability in SME domain.

2.1. Method for Method Configuration (MMC)

The major proposal for method configuration is from Karlsson and Agerfalk [3], [4]; they proposed a meta-method named MMC – Method for Method Configuration for configuration process. MMC takes the Base method as method component or initial method for their process. The base method is subsequently configured with respect to the project requirements defined in the form of development situations and characteristics to form situation-specific method.

MMC approach focuses on configuring a base method with respect to Configuration package and Configuration template.

The process defined by them is as follows:-

1) Method component used in MMC framework is the base method. Base method is configured on basis of development situations and characteristics use to elicit project development requirements.
2) If multiple development requirements exist, configured method is formed by configuration template. Configuration template is generated by combining configuration packages.
3) The configured method is then adapted in accordance with the Project Situations to form Situational Method.

The criteria for selecting base method is external to this approach therefore the issues such as the selection of Base Method and the granularity of method component remain unanswered.

Later on Karlsson along with Wistrand [5], [6] extended the proposal by defining a conceptual construct to facilitate the method engineer's task of method configuration and termed it as "Method Component". Method Components are composed from method elements and can be configured to achieve an overall goal on the basis of one or more project requirements. For future use a method rationale is also maintained corresponding to each transformation. Each artifact is capable of taking recommended inputs called <pre-requisites>. These inputs are then modified to deliver output. The modification is done on the specification of <outcomes> or goals.

The process defined is as follows:-

2.1.1. Define the Input/output of a method component

Method component consists of artifacts, each artifact has a value either prerequisite or outcome. Prerequisite are the inputs for the method component whereas output or deliverables are specified by the outcome artifact.

2.1.2. Define the operations performed by the method component

Method components are configured to perform some specific operations. These operational characteristics are defined by the Internal View of method component. Internal view defines the purposes of a method component in the form of Actions, Goals, Method Elements, Actor roles etc.

Recently Rolland [19] 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 gave solutions to many issues, however issues like 'right granularity', 'appropriateness of the method being selected' and 'explicit representation of variability' are still open.

Above proposals are centered on a single base method configured to form situated method, this reduces the scope of method generation for each new project and hence decrease the flexibility in the process.

2.2. Method Configuration In Respect to the Existing SME Approaches

To position the method configuration process within the existing SME research we will first describe the characteristics that can be used to compare different SME approaches. There are several aspects in which SME approaches differ, but most notably are:

- Meta Model and its underlying principles
- Reusable building blocks

2.2.1. Meta model and its underlying principles

Meta-model (Meta-modeling) is the principal technique used for understanding, comparing and evaluating methods. It is generally defined as a “model of models” [20]. Broadly, two types of information are required to develop a meta-model: the structure of products and the procedures to produce the products i.e. process. Reflecting the product-process dichotomy of methods, two types of meta-models have been developed. The first of these are meta-data models for the product aspects of methods. Such models introduce a system of concepts in which the static and data aspects of methods as well as constraints defined in them can be represented. The second kind of meta-models deals with the process aspects of methods; these meta-models are called meta-activity models and specify a system of concepts to define tasks and task transition criteria. Next the need comes for coupling the product and process aspects of methods, the contextual meta-model [17] and the decisional meta-model [9] have been proposed. The contextual meta-model, context is defined as <situation, decision>, where decision reflects the choice the application engineer makes in a situation. The decisional meta-model is process and paradigm independent, it views a method as a set of decisions and dependencies between them together with a mechanism for decision enactment. After these, the requirements moved towards the need of generic models, generic models abstract out the common properties of meta-models. This results in a three-layer framework consisting of the generic, meta-model and method layers capable enough of handling the link types between the architectures [11].

The configurable Meta model presented in this proposal is engineered from generic model hence making it an important proposal in the field. Since the generic concepts are entered in the configurable Meta model, it makes sense to model a number of methods from it and after customization; a complete and consistent situated method can be formed.

2.2.2.Reusable building blocks

SME reuses method components, which are the building blocks of development methods to create situational methods in SME. The methods are built from small parts of existing methods i.e. smaller pieces (e.g. fragments, patterns, components, chunks). Method fragment consists of process fragments and product fragments. The process fragment describes the stage, activities and task whereas the product fragment concerns the structure of a process product (diagrams, deliverables etc.). Further, method fragments defined as "standard building blocks based on a coherent part of a method” [21]. A situational method can be constructed by combining a number of method fragments.

A Pattern is described as “a problem which occurs over and over again in our environment and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing the same twice” [22]. It is necessary “to specialize the pattern and to fill the abstract solution with additional information in order to meet the specific conditions of the actual case” [23]. It is used to guide model based design of software. We can apply pattern for the issue at hand maps with the general problem specification in particular pattern. For example, Analysis pattern which contains the knowledge on how to appropriately represent a certain fact in requirement engineering.

A component provides the partial solution to a specific problem [25]. They are more concrete as compared to the patterns and can be used without modification. Process building blocks are an example for model components. Researcher observed that components may also be specialized or configured before or after aggregation. “Method components, the building blocks of SME, are development methods or any coherent parts of them” [24].

“A method chunk represents a reusable building block for situation-driven method construction or adaptation whereas a road-map represents a path in a method or a specific sequence of method chunks in a method” [26].

In addition to above, many other existing method modules have been used as an input for analysis. The aim of the analysis has been to determine the limitation of the current design and what changes are made in order to transform it as a configurable method component. Subsequently, the analyses have focused on configuring a configurable method component and adapt it for current organizational definition.

Our notion of configurable method is similar to the method chunk; it combines product and process perspective into the same modelling component. The configurable method can be atomic, compound, transformational and constructional besides these a new essentiality concept is added in the method that transforms a method into configurable method.

3. Meta Model Supporting Method Configuration Process

In this section we are looking to fulfil our requirement of a configurable Meta model, which has the properties of integrating the product and process aspects of methods and provides facility of ordering method features. Therefore the class of product Meta models, process Meta models and contextual model do not individually fit in to our requirement of the Meta model. Thus our choices are limited to decisional and generic models only. We choose to modify decisional Meta model to develop configurable Meta model since the decisional Meta model itself is an instantiation of generic model.

The fundamental part of the configurable meta-model is the configurable construct as a means to facilitate efficient and rationally motivated modularization of systems development method.The Configurable method supports a new attribute called Essentiality in the method. The Essentiality attribute can take two values either common or variable. Common are ones which if omitted; the method will lose its identity. So the variability in the proposal lies in the variable part of the method. Within a method, it is possible for method blocks to be either common or variable. This is shown in the Fig. 1 by the essentiality attribute of the method block.

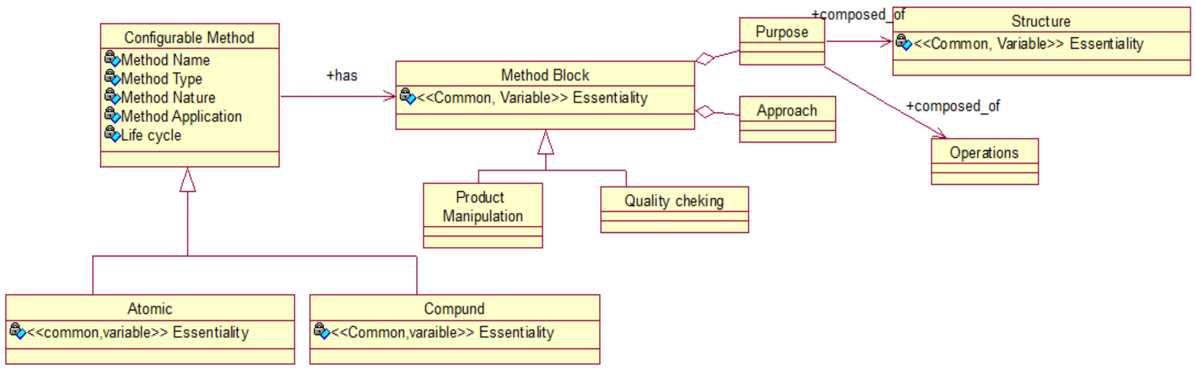

Fig. 1. Configurable meta model.

A method block is an aggregate of Purpose and Approach. For simplicity, let us ignore the notion of an approach. Thus a method block reduces to a purpose. Now, in a purpose, there is a structure part and an operation part.

Purpose= <structure, operation>

]

As we will see, the operation part is used to create, delete and modify the structure part and is given in the Meta model. Thus, they are not configurable. The only configurability lies in the structure part. This results in the purpose and consequently, the method block to be configurable.

In the rest of this section, we give a detailed description of Configurable Meta model. Configurable Meta model treats method as a triple <MB, Dep, E> where MB is a set of method blocks, Dep is a set of dependencies between these, and E is the enactment algorithm.

The Meta model is centered round MB and Dep. E is the procedure that exploits the given set of MB and Dep to produce the product. The set Dep establishes dependencies between instances of method blocks. Thus, if a method block is common then all dependencies in which it participates are relevant to the desired method. However, if a method block is a variable and not included in the desired method then all dependencies in which these variables participate are meaningless.

3.1. Structure

There are two kinds of structures, those whose instances can be created and destroyed by application engineers and those whose instances are pre-defined. The former are called conceptual structures and the latter are called fixed structures. Conceptual structures constitute the set of concepts in terms of which a product is expressed. Fixed structures are those that are defined once and all for by a method engineer. An example of fixed structure is a method constraint such as completeness and conformity which cannot be created or destroyed by the application engineer.

3.1.1. Conceptual structures

Conceptual structures are partitioned into two dimensions. The first dimension classifies them as either atomic or compound. The second dimension represents conceptual structures into disjoint classes of structures called constraint, definitional, constructional, link, and collection of concepts respectively.

Simple constructional structures cannot be decomposed into other components. Links are conceptual structures that are used to build collection of concepts from given concepts. For example ISA and aggregation are links, as they build abstraction hierarchies. Collections of concepts are constructed whenever constructional structures are connected by links. Aggregations, specialization hierarchies, and subtype hierarchies are examples of collection of concepts. A collection of concepts is complex if it is defined out of other collections. Definitional structures define the properties of conceptual structures. Constraints impose application-related constraints on conceptual structures. The presence of the attribute, essentiality, in configurable Meta model shows that conceptual structures are configurable.

3.1.2. Fixed structures

Fixed structures deal with the restrictions that are used to enforce quality of conceptual structures. They are defined by the method engineer to help the application engineer in creating well defined and well formed conceptual structures. In their simplest form, they are the method constraints of completeness, consistency, conformity and fidelity. Similarly there are compositional constraints which are specified between conceptual structures of the different simple component methods of a compound method. A structure of one of these cannot compose any arbitrary structure of the other. Such composition is governed by constraints that control the product resulting from the use of compound methods. The method engineer defines these constraints at the time the compound method is defined. For example in UML, operation in Class Diagram must be a use case in Use Case Diagram.

3.2. The Operation

The operations identify the set of process types that operate upon product types to provide product manipulation and verification capability to application engineers.

3.3. Dependencies

Method concepts (MCi) in a method are dependent upon one another. In generic model, these dependencies are defined on two main properties namely, Urgency and Necessity (As shown in table Urgency refers to the time during which the dependent method concept, MC2, is to be enacted. If MC2 is to be enacted immediately after MC1 is enacted then this attribute takes on the value Immediate. If MC2 can be enacted any time, immediately or at any later moment, after MC1 has been enacted, then urgency takes on the value Deferred.

Necessity refers to whether or not the dependent method concept MC2 is necessarily to be enacted after MC1 has been enacted. If it is necessary to enact MC2, then this attribute takes the value Must otherwise it has the value Can. Combining these two properties, four possibilities as shown in Table1.

Table 1. Types of Dependencies

Dependency Type	Urgency	Necessity	Abbreviation
1	Immediate	Must	IM
2	Immediate	Can	IC
3	Deferred	Must	DM
4	Deferred	Can	DC

In configurable Meta model, four kinds of dependencies are defined corresponding to the above mentioned dependencies of generic model.

- Requirement dependency: Requirement dependency says that when a certain manipulation purpose is performed, there must associate some constraints that have to be related with it. This corresponds to dependency type 3 of generic model.
- Removal Dependency: removal dependency is the inverse of requirement dependency. It says that when a certain manipulation purpose is performed then there are certain purposes that are not to be performed. This corresponds to dependency type 1 of generic model.
- Activate dependency: It says that a purpose activates other purpose. The activate dependency is of type 4 of generic model.
- Inactivate dependency: Inactivate dependency is the inverse of the activate dependency. It says that when a certain manipulation purpose is performed then there are certain manipulation purposes that cannot be performed. , inactivate dependency is of type 1.

As stated above, our Configurable Meta model supports methods as Configurable Methods. We define the configurable method "as an abstraction of method that identifies the essentiality of the method concepts and its relationships". The crucial part of this definition is the 'essentiality of the method concepts'. For our process, we accomplish this by using Conceptual structure knowledge. Conceptual structures are classified in seven categories viz simple definitional, complex definitional, simple constructional, complex constructional, simple collection of concepts, complex collection of concepts and links. The detailed discussion on conceptual structures has been presented in our previous research [8]. The simple definitional and simple constructions are basic building blocks of a method; hence considered them as common in our proposal. The rest of the structures are considered as variables. The identification is done on the knowledge and judgment of method engineers, who are practically working with many methods in different organizations moreover their feedback and knowledge of the method-in –action is used to enrich the method base.

Entity- Relationship(ER)[27] method expressed in configurable Meta model (Atomic Method)

3.4. Method Nature Part

Describes the method name and method characteristics

Method Name (12 characters) <*ER method*>

Method Type (Atomic/Compound) <*Atomic*>

Method Nature (Constructional/Transformational)

<*Constructional*>

Method Application (Data/Process/Behaviour Oriented) <*Data Oriented*>

Method life cycle (Requirement/Design/Testing/Complete life cycle)

<*Design Phase*>

3.5. Method Conceptual Model

Method conceptual model stores the method concepts, instantiation of conceptual structures and essentiality of each method concept in a method. Since atomic method belongs to one product model only, there is single conceptual model for each atomic method.

<Concept Name> : <Type> <Essentiality>

<Entity>: <Simple Constructional>

<Common >

<Relationship>: <Simple Constructional> <Common>

<Cardinality>: <Simple Definitional> <Common>

<Primary Key>: <Complex Definitional> <Variable>

<Attribute> : <Complex Definitional> <Variable>

<Multiplicity of Attribute>: <Complex Definitional><Variable>

<Role>: <Simple Definitional>

<Common>

<Functionality>: <Simple Definitional> <Common>

<N-ary Relationships>: <Complex Constructional> <Variable>

3.6. Method Purposes and Dependencies

Along with the method nature part and method conceptual part, purposes and dependencies in the method are also stored.

<Purpose>: <Basic life cycle, Relational, Constraint Enforcement>

<Dependencies>: <Activate, Requirement, Inactivate, Removal>

Unified Modeling Language [28,29] expressed in configurable Meta model (Compound Method)

3.7. Method Nature Part

Method Nature Part describes method name and method characteristics of the method. In compound methods, method nature part also defines the method components within the method

Method Name (12 characters) <*UML method*>

Method Type (Atomic/Compound) <*Compound*>

Method Components < *Class Diagram, Use Case Diagram, Sequence Diagram, Collaboration Diagram, State Chart Diagram, Component Diagram, Deployment diagram, activity diagram, object diagram*>

Method Nature (Constructional/Transformational)

<*Transformational*>

Method Application (Data/Process/Behaviour oriented)

<*Process Oriented*>

Method life cycle (Requirement/Design/Testing/Complete life cycle)

<*Design Phase*>

3.8. Method Component Model

Since compound methods consist of more than one product model, they need a separate component model to be well expressed. Method component model of compound methods stores the entire set of method components together with their essentiality. For each compound method a separate component model is designed.

<Component Name>: <Essentiality>

< Class Diagram> :< Common>

< Use Case Diagram>: <Common>

<Sequence Diagram>: <Variable>

< Collaboration Diagram> :< Variable>

< State Chart Diagram> : < Variable>

<Component Diagram>: <Variable>

<Deployment diagram>: <Variable>

<Activity diagram>: <Variable>

3.9. Method Conceptual Model(s)

Since compound methods composed of more than one method components, there is a separate conceptual model for each method component defined in the component model. Following is the method conceptual model for component Class Diagram.

<Concept Name> : <Type> <Essentiality>

<Class> :< Simple Constructional> < Common>

<Data_type :< Simple Definitional> <Common>

<Association>: <Complex Constructional > <Variable>

<Aggregation>: <Simple Collection of concepts > < Variable>

<Operation>: < Complex Definitional> <Variable>

<Generalization>: <Simple Collection of Concepts> < Variable>

<Generalization_link> : < Link> <Variable>

<Aggregation_link> : <Link > < Variable>

<Cardinality>: < Simple Definitional>

< Common>

<Degree of association>: <Complex Definitional > < Variable>

<Multiplicity>: < Complex Definitional>

< Variable>

UML has eight components in the method component model consequently eight conceptual models are needed to express complete UML in configurable Meta model

3.10. Method Purposes and Dependencies

Purposes and dependencies for each conceptual model are defined. In case of compound methods, together with the operations available in the atomic method, a separate class of operations is also defined. The class is named as Integration class and it deals with the structure belonging to different product models of the compound method.

<Purpose>: <Basic life cycle, Relational, Integration class, Constraint Enforcement, >

<Dependencies>: <Activate, Requirement, Inactivate, Removal>

4. Method Configuration Architecture

The Macro level architecture that gives the flavor of our whole process is shown in figure2.

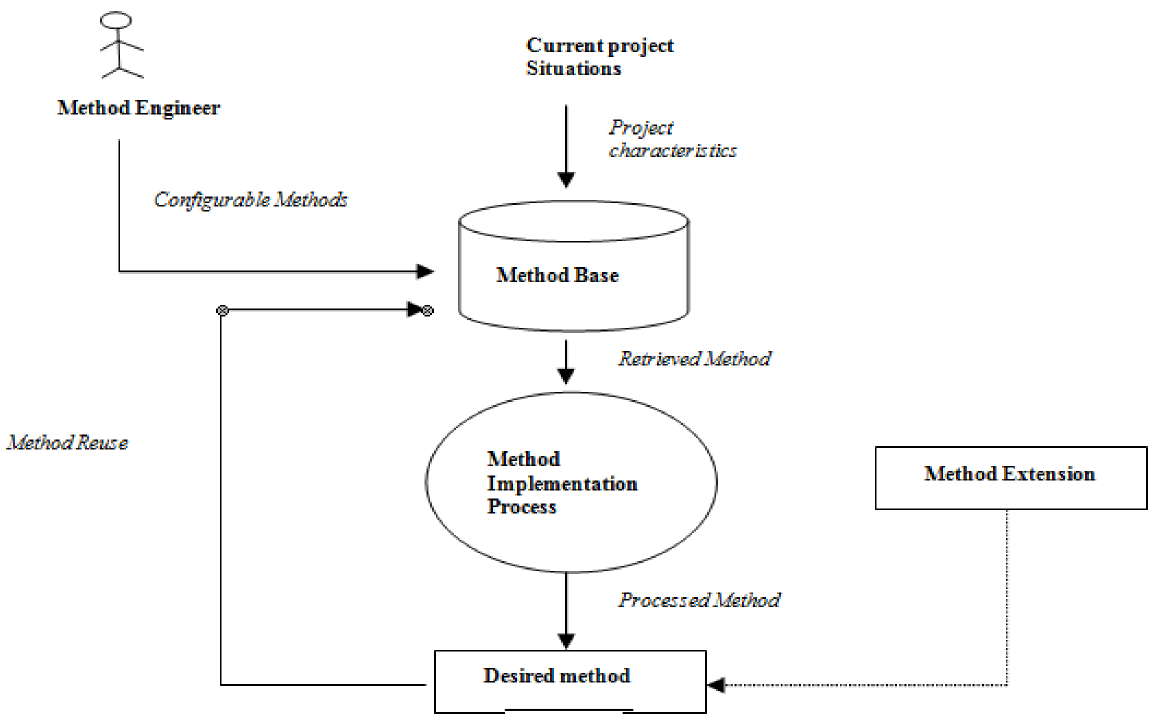

Fig. 2. Macro level architecture of method configuration process.

Our idea behind Method configuration process is as follows: The desired method is built around each project-in-hand. The project characteristics are gathered from current project situations. These project characteristics are used to retrieve methods from the method base. In our process, method base is formed by collecting configurable methods. The Configurable method has property of essentiality which consists of two attributes i.e. common and variable. The configurability in our proposal lies in the variable attribute of the method. In the method implementation stage the variables meeting the project requirements will be kept and redundant variables will be omitted resulting in formation of desired method. If the configured method formed is not the method to-be, the Method Extension activity is required to be carried out. Therefore in the Method configuration process we propose the method extension as an optional activity.

5. Method Base

Method configuration process proposed by us is centered on method base. Availability of a rich method base is probably the most important prerequisite for the method configuration process. It is a formal representation of how a configurable method is stored. The construction of a method base is a crucial activity as it presents the foundation for creating the desired methods and thus has to be done prior to the process starts.

5.1. Design of the Method Base

The method base is designed in two parts Method Configurable Part (MCP) and Method Implementation Part (MIP). Configurable methods are stored in MCP; from MCP these methods are retrieved and configured to form desired method. The method structure stored for both atomic and compound methods in MCP is described in Section 3. Here we will focus on Method Implementation Part, MIP stores the configured methods formed from actual methods for future reuse.

Method Implementation Part for Atomic methods

Following is the configured method formed (ER_{conf}) from ER method having only *Single valued attribute and Binary relationships.* To form ERconf the product entities <n-ary relationship> and <multivalued

attribute> will remove from the Method conceptual Model of ER and the purposes and dependencies are modified accordingly. *ER_{conf}* is stored in MIP as

ER_{conf} stored in Method Implementation Part

Method Conceptual Model

<Entity>: <Common>

<Relationship> :< Common>

<Cardinality>: <Common>

<Primary Key>: <Variable>

<Attribute> : <Variable>

<Role>: <Common>

<Functionality>: <Common>

Method Purposes and dependencies:

In the configured method, some method concepts have been neglected and are not becoming the part of the desired method, consequently, purposes and dependencies of the original conceptual model shall be engineered to form the coherent method. The method formed is then stored in the MIP of the method base. Engineering process for atomic methods will be discussed in Section 6.

Method Implementation Part for Compound methods

Following is the configured method formed *UML_{conf}* from UML method having only two atomic methods *Class Diagram and Use Case Diagram*. *UML_{conf}* is stored in MIP as

Method Component Model

< *Class Diagram>*

< Use Case Diagram>

Method Conceptual Model(s)

<Conceptual Model of class diagram>

<Conceptual Model of Use Case Diagram>

Method Purposes and dependencies:

In the configured method, some method components have been neglected and are not becoming the part of the desired method, consequently, purposes and dependencies of the original component model shall be engineered to form the coherent method. The method formed is then stored in the MIP of the method base. Engineering process for compound methods will be discussed in section 6.

5.2. Operations on Method Base

The basic notions of storage and retrieval from the method base are formalized by defining two operators: the *storage* operator (for storing the configurable and configured methods in the method base) denoted by 'store' and the *retrieve* operator for the retrieval of methods denoted by 'retrieve'. The former, as its name implies, stores a method M_i in the method base. The method stored in the method base exists in two forms:

- Store a new configurable method: Here a new configurable method is stored in the Configurable method part of the method base. The new method stored is completely defined by method nature part, conceptual and component model and purposes and dependencies of the method.

Store (M_i) = MCP of Method Base.

- Store a configured method: After the method configuration process, the configured method formed will again store in the Method Implementation part of the method base. The configured method is an implementation of configurable method stored in the CMB of the method base, the method structure of the configured method comprises of modified conceptual and component model along with the modified set of purposes and dependencies.

Store (M_{iconf}) = MIP of Method Base.

The 'retrieve' operator used to retrieve methods from method base based on the project characteristics of the project in hand.

Retrieve (Project Characteristics) =Method M_i from method base.

The retrieved method is in the form of configurable method, and has to be configured to form situated method. In the next section we will the detail process of method configuration.

6. Developing Configured Method

The process of Method Configuration is depicted in the Fig. 3

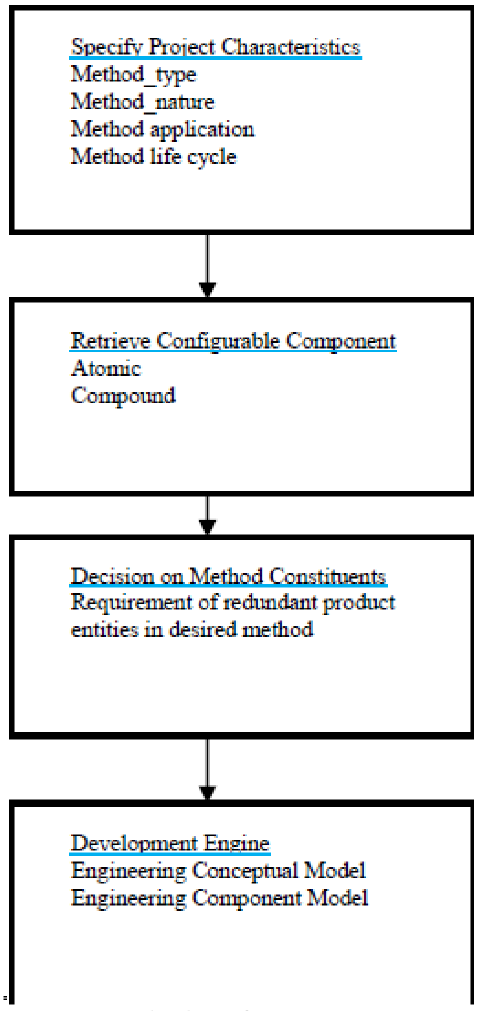
Specify Project Characteristics
Method_type
Method_nature
Method application
Method life cycle
Retrieve Configurable Component
Atomic
Compound
Decision on Method Constituents
Requirement of redundant product entities in desired method
Development Engine
Engineering Conceptual Model
Engineering Component Model

Fig. 3. Method configuration process.

The situation of SME can be conceptualized in many ways, as descriptors [16], contingency factors [30], project factors [15], context type [25] and project type [31],[32] and project manager/ method engineer can assign should be able to determine the situation to these factors by assigning values. All these factors are considered to define the project situation or development situation. In the step 1, the situation is gathered in the form project characteristics defined by any of the above mentioned process. For example, the project characteristics are specified as method type=atomic, Method Nature=Constructional, Method Application=Data Oriented and Method Life Cycle=Design Phase. The proposal contained in this paper has been used to design a computerized tool to capture project requirements, retrieve configurable method and to facilitate the method-user to perform method configuration, hence bridging the gap between end users understanding and method engineer's expectations from a situational method.

The tool has been developed, and has proved useful with an understanding that it is possible for the method users to actively participate and construct suitable situational methods if they were provided with appropriate high-level modeling concepts, such as configurable method component. The project characteristics are used to retrieve configurable method by matching the method characteristics stored in the method nature part of the method base (refer Section 3) in the second step. The method so retrieved can be atomic or compound. The atomic methods by definition consist of only one method component and belong to one product model only. On the other hand, compound method composed of other atomic methods or method components.

As stated above, method requirements are complex in nature and there cannot exist only one set of requirements, rather several sets of requirements exists anchored in the views of different stakeholders with different interests. In step 3, method constitute are decided, decision is done on the basis of the requirement of the redundant product entities for the project-in-hand. The method retrieved from step 2 can be atomic or compound, decision on constitutes varies accordingly. In atomic methods the method concept can have the property of essentiality or redundancy at its conceptual level that is the lowest level of granularity. Whereas in a compound method, the essentiality is defined at two levels of granularity: The method component level and the method conceptual level. This simply reflects the change in the composition of the selected method. However this negotiation of the composition is done in the method configuration process. The choices are indicated through the selection of the variables, either to add to or suppress them from situational method. Suppose, from the above configurable method, method user need to build ERconf having only single valued attribute and only binary relationship. The configurable method is varied accordingly, and the set of purposes and dependencies are modified, and the modified method is build automatically.

Since, some product entities have been neglected and do not constitute the final product, the Purposes and Dependencies of the original method shall be engineered to form the desired coherent method. For our configuration process, the operation is performed by Development Engine part shown. Again, the complexity of process depends upon the method type. In an atomic method, method conceptual model is engineered to form desired method. On the other hand in compound methods both method component model and method conceptual models are engineered to form desired method.

6.1. Engineering Configured Method from Atomic Methods

The algorithms for engineering the atomic and compound methods are relatively simple. It starts with the all elements in the method conceptual model and ends when there is no link left that would connect the current excluded variable with any other element.

```
PROCEDURE engineering atomic method (cm, $s_i$)
// cm-conceptual model, $s_i$-set of elements in cm, $s_i$=<$E_i$ or $V_i$>
// $E_i$-method concepts with essentiality as common.
// $V_i$ -method concepts with essentiality as variable.
//Exclusion of a product entity $V_i$ requires the following deletion to be done in the chunk of purposes and dependencies generated at the time of method creation.
Begin:
Find all the purposes ($p_i$) where $V_i$ participates
For all pi
        delete < $V_i$, forward purpose> and delete < $V_i$, inverse purpose>
Find all fixed structure ($f_i$) where $V_i$ participates as a sub-concept or as a super concept
```

```
        delete < $V_i$, {completeness, conformity and fidelity}>
// when a project specific method is created using the algorithm above, the dependency list should be checked for the completeness and coherency of the method formed.
For each deleted purpose $p_i$
Check
{       Deletion of a purpose deactivates its inverse.
Deletion of a purpose requires the deletion of all its method constraint and completeness purposes for all its super-concept as well as the method constraint fidelity purposes for all its sub-concept.
            }
For each deleted variable $V_i$
Check
           {
If a product entity is deleted, all the purposes activated as a result of its creation become inactivated.
          }
END
```

6.2. Engineering configured method from Compound methods

The process is defined for compound methods. Compound methods are composed of method components. These components along with their essentiality are defined in method component model. The input to the process is the method component model that contains the set of method components along with essentiality property.

```
PROCEDURE engineering compound method (mc, $s_i$)
// mc-component model, $s_i$ -set of elements in mc, $s_i$=< $E_i$ or $V_i$ >
// $E_i$ -method components with essentiality as common.
// $V_i$ -method components with essentiality as variable.

// Method components in a compound method communicate with each other through the operations defined in the Integration class.
/*These operations are represented by a triplet, < $V_i$, $MC_i$, O> where, $V_i$ is the variable method component to be excluded, $MC_i$ is the method component to which $V_i$ is communicating and O is the communicating operation. */

Begin:
//Disable all communication coming-In and Out from the dispensable method component.

For each deleted variable $V_i$
Disable
{
< $V_i$, $MC_i$, export> and its inverse < $V_i$, $MC_i$, withdraw>
 < $V_i$, $MC_i$, import> and its inverse < $V_i$, $MC_i$, dump>
 < $V_i$, $MC_i$, correspond> and its inverse < $V_i$, $MC_i$, seperate>
 < $V_i$, $MC_i$, convert> and its inverse < $V_i$, $MC_i$, deconvert>
}
END
```

7. Method Extension

Our approach for method configuration consists of one basic process i.e. method configuration and one extended process i.e. method extension. We propose that the configuration process is the essential to the process and method extension should be attempted only when configurability fails to deliver the desired method. The failure can happen if a method concept is missing to make a complete desired method.

During method extension, the method engineer selects and integrates the missing product entities to form the desired method. For method extension, we consider the external view of the method that represents only the functionality of the method.

The new product entity must enter into the chunk of purposes i.e. Basic life cycle purpose, Relational purpose, Constraint Enforcement Purpose and Integration class purposes. Since the set of purposes get modified the dependencies and the constraints have to mutate accordingly.

8. Discussion and Conclusion

Presently the methods used for software development are not situation specific and does not adapt to the requirements of given project. This issue has been raised for a long time leading to several proposed solutions to make the methods more situation specific and suiting to requirements of the project. The most of these solutions are proposed by method engineers after continued research. However, very few of them have been practically implemented in bits and pieces. Our current research presents an approach for situation specific software development method popularly known as Method configuration approaches. Though our approach is based on established method engineering principals, the effort is to reduce the impact of difficulties.

Method configuration in this paper has been treated as a specific kind of situational method engineering. A Method Configuration process with respect to the instantiation of a Configurable Meta model and Method configuration to the finest level of granularity has been envisaged in the present research. The fundamental part of the configurable meta-model is the configurable construct as a means to facilitate efficient and rationally motivated modularization of systems development method. The benefits of using the configurable method component is that the process can be performed more efficiently since pre-configured components are available and can be used over and over again. Hence, there is no need to perform a complete configuration for each new project.

The Configurable method supports a new attribute called Essentiality in the method. The Essentiality attribute can take two values either common or variable. Common are the non-configurable part of the method component and Commonality can be defined as the essentiality in the method component which if omit method will lose its identity.

The research work is based on the assumptions that it is possible to elicit situational characteristics for the situation in hand by means of any of the previous approaches proposed.

Since the proposed configuration process does not rely on a complex query language for administrating and retrieval of methods. The decreased complexity improves possibilities to perform method configuration process interactively with method users. The reusability advantage is obvious since pre-made configurations can be used over and over again. Hence, there is no need to perform a complete method configuration for each new project.

Our future work will concentrate on method extension process -How the individual configured methods can be assembled to form situated methods? Basically, it will show the storage and retrieval of the new product entities. The procedure for engineering new extended configured method formed and structures it as a desired coherent method.

References

[1] Van, G. J. (2000). Variability in software systems the key to software reuse. Licentiate Thesis, University of Groningen, Sweden.

[2] Henderson, B. S., & Ralyte, J. (2010). Situational method engineering: State-of-the-art review. *Journal of Universal Computer Science*, *16(3)*, 424-478.

[3] Karlsson, F. & Ågerfalk, P. J. (2004). Method configuration: Adapting to situational characteristics while creating reusable assets. *Information and Software Technology*, *46(9)*, 619-633.

[4] Karlsson, F., & Agerfalk, P. J. (2012). MC Sandbox: Devising a tool for method-user-centered method configuration. *Information and Software Technology*, *54*, 501-516.

[5] Wistrand, K., & Karlsson, F. (2004). Method components–Rationale revealed. *Proceedings of the Advanced Information Systems Engineering 16thInternational Conference*.

[6] Karlsson, F., & Wistrand, K. (2006). Combining method engineering with activity theory: Theoretical grounding of the method component concept. *European Journal of Information Systems*, *15*, 82-90.

[7] Gupta, D., & Dwivedi, R. (2012). Method configuration from situational method engineering. *Software Engineering Notes*, *37(3)*, 1-11.

[8] Gupta, D., & Dwivedi, R. (2012). A step towards method configuration from situational method engineering. *Software engineering: An International Journal (SEIJ)*, *2(1)*, 51-59.

[9] Gupta, D., & Prakash, N. (2001). Engineering methods from their requirements specification. *Requirements Engineering Journal*, *6*, 135–160.

[10] Prakash, N. (1997). Towards a formal definition of a method. *Requirements Engineering Journal*, *2(1)*, 23-50.

[11] Prakash, N. (2000). On generic method models. *Requirements Engineering Journal*, *11(4)*, 221-237.

[12] Glass, R. L. (2000). Process diversity and a computing old wives'/husbands. *IEEE Software*, *17(4)*, 128-127.

[13] Glass, R. L. (2004). Matching methodology to problem domain. *Communications of the ACM*, *47(5)*, 19-21.

[14] Ralyté, J., Brinkkemper, S., & Sellers, B. H. (2007). Situational method engineering: Fundamentals and experiences. *Proceedings of the IFIP WG 8.1 Working Conference*.

[15] Harmsen, F., & Brinkkemper, S. (1995). Description and manipulation of method fragments for method assembly. *Proceedings of the Workshop on Management of Software Projects*.

[16] Harmsen, A. F., Brinkkemper, S., & Oei, H. (1994). Situational method engineering for information systems projects. *Proceedings of the IFIP WG8.1 Working Conference Cris*.

[17] Rolland, C., & Prakash, N. (1996). A proposal for context-specific method engineering. *Proceedings of IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering*.

[18] Kelly, S., Lyytinen, K., & Rossi, M. (1996). Metaedit+: A fully configurable multi-user and multi-tool case and came environment. *Proceedings of the 8th Conf. on Advanced Information Systems Engineering*.

[19] Rolland, C. (2009). Method engineering: Towards methods as services. *Software Process Improvement and Practice*, *14(3)*, 143-164.

[20] Rony, F. (2002). Metamodeling in EIA/CDIF-meta-metamodel and metamodel. *ACM Trans. Model. Comput. Simul*, *12(4)*, 322-342.

[21] Ågerfalk, P. J., Sjaak, B., Cesar, G. P., Brian, H. S., Fredrik, K., Steven, K., & Jolita, R. (2007). Modularization constructs in method engineering: Towards common ground, situational method engineering.

[22] Tran, H. N., Boulette, B., & Dong, B. T. (2005). A classification of process patterns. *Proceedings of the International Conference on Software Development*.

[23] Becker, J., Knackstedt, R., Pfeiffer, D., & Janiesch, C. (2007). Configurative method engineering-on the applicability of reference modeling mechanisms in method engineering. *Proceedings of the Americas*

Conference on Information Systems (AMCIS 2007).
[24] Anat, A., Iris, R. B. (2011). Semi-automatic composition of situational methods. *Journal Database Manag, 22(4)*, 1-29.
[25] Denecker, R., Kornyshova, E., & Bruno, C. (2010). Contextualization of method components. *Proceedings of the 4th IEEE International Conference on Research Challenges in Information Science* (pp. 235-246).
[26] Iacovelli, A., Carine, S., & Rolland, C. (2008). Method as a Service (MaaS).
[27] Chen, P. P. (1983). *A preliminary framework for entity relationship diagram, Information modelling and analysis*. Elsevier science publisher.
[28] Booch, G. (2006). Object oriented analysis and design.
[29] Rumbaugh, J., Jacobson, I., & Booch, G. (1999). The unified modelling reference manual.
[30] Slooten, K. V. (1995). Situated methods for systems development, dissertation. University of Twente.
[31] Bucher, T. *et al*. (2007). Situational method engineering–On the differentiation of "context" and "project Type".
[32] Bucher, T., & Winter, R. (2008). Dissemination and importance of the "method" artifact in the context of design research for information systems. *Proceedings of the third International Conference on Design Science Research in Information Systems and Technology*.

Rinky Dwivedi has received the B.Tech. degree in computer science from Guru Gobind Singh Indraprastha University, Delhi, India and received the M.E. degree from Delhi College of Engineering, Delhi, India. She is now pursuing PhD from Delhi Technological University, Delhi India. Her areas of interests include method configuration, software methodologies and Agile Methods.

Daya Gupta is a professor in the Department of Computer Engineering, Delhi Technological University New Delhi, India. She has done M.Sc. Post M.Sc. (computers Sc.) from IIT, Delhi, and PhD from Delhi University. She is a senior member of IEEE and a life member of CSI. Her research interests are requirement engineering, method engineering, information security, image characterization and software estimation. She has published several research papers in international journals and conferences and has chaired sessions and delivered invited talks at many national and international conferences.