Configurative Method Engineering - On the Applicability of Reference Modeling Mechanisms in Method Engineering

Jörg Becker
Ralf Knackstedt
Daniel Pfeiffer

European Research Center for Information Systems

Christian Janiesch

Follow this and additional works at: http://aisel.aisnet.org/amcis2007

Recommended Citation
http://aisel.aisnet.org/amcis2007/56
Configurative Method Engineering –
On the Applicability of Reference Modeling Mechanisms in Method Engineering

Jörg Becker
European Research Center
for Information Systems (ERCIS)
becker@ercis.de

Ralf Knackstedt
European Research Center
for Information Systems (ERCIS)
knackstedt@ercis.de

Daniel Pfeiffer
European Research Center
for Information Systems (ERCIS)
pfeiffer@ercis.de

Christian Janiesch
European Research Center
for Information Systems (ERCIS)
janiesch@ercis.de

Abstract
The aim of this paper is to discuss the applicability of reference modeling adaptation mechanisms in method engineering. Reference models are generic conceptual models that formalize state-of-the-art knowledge of a certain domain. Adaptation mechanisms can assist to tailor reference models to specific business or project needs. (Situational) Method engineering is concerned with the preparation of methods that fit the characteristics of a situation. Due to the similar requirements, it seems feasible to investigate to what extent the existing mechanisms of reference modeling have already been reflected in the context of method engineering. The result of a literature review points out that the mechanism of configuration has not been sufficiently considered yet. We suggest transferring the concept of reference model configuration to the domain of method engineering. This approach is supported by an example case utilizing the H2 toolset in which a method is configured according to project specific needs.

Keywords
Method engineering, situational method engineering, adaptive reference modeling, configuration, theories.

On the Connection of Method Engineering and Reference Modeling
Methods describe systematic procedures to overcome problems. Problems can be characterized as the discrepancy between an as-is and a to-be situation. It is widely accepted that a universal method, which can be used without modification in all situations, is not feasible (Brooks 1987; Fitzgerald et al. 2003; Kautz 2004; Lindvall and Rus 2000; ter Hofstede and Verhoef 1997; Wistrand and Karlsson 2004). Rather, appropriate methods for problem solving must be chosen, adapted, or designed depending on the specific characteristics of a situation, such as qualification, number of employees, or available time. In the method engineering community, terms like domain specific method engineering (Kelly et al. 2005; Luoma et al. 2004) or situational method engineering (Brinkkemper 1996; Harmsen 1997; Kumar and Welke 1992) have been used to voice this special circumstance.

A method engineer is confronted with the following dilemma: Multi-purpose methods contain a comprehensive set of activities and constructs but provide no guidance how to adapt them to fit the specific characteristics of a project and its context. Domain specific methods, however, contain a problem adequate set of activities and constructs but are only applicable in their well-defined domain. Thus, a mechanism is required which allows for adding domain and context specific knowledge to multi-purpose methods. In the area of reference modeling this problem is addressed by so called adaptation mechanisms (Becker et al. 2006a; vom Brocke 2007). Since it is possible to document methods in form of models, it is reasonable to transfer these mechanisms into the domain of method engineering. The aim of this paper is to analyze prior
work on method engineering to explicate their mechanisms of reuse and adaptation in comparison to existing approaches in the domain of reference modeling. Based on this analysis, configuration is identified as a promising mechanism to reduce the effort of method engineering.

The paper is structured as follows: First, a classification of reference modeling mechanisms is introduced to provide the basis for the analysis. Subsequently, related work on method engineering is reviewed concerning their use of adaptation mechanisms. Analog to configurative reference modeling (Becker et al. 2006a), a concept for configurative method engineering is derived. The paper closes with a short summary of the main results and an outlook on future research perspectives.

Reference Modeling Mechanisms

Reference models are generic conceptual models that formalize state-of-the-art or best-practice knowledge of a certain domain (Fettke and Loos 2003; Rosemann and van der Aalst 2007). They are of normative nature and can cover different domains such as industry sectors or functional areas (e.g., logistics or accounting). Well known reference models are for example the Retail-H (Becker and Schütte 2007), a reference model for retail enterprises, or the Y-CIM model (Scheer et al. 2007), a reference model for industrial enterprises. Consequently, reference models are created to be helpful in more than one situation. The objective of reference models is to simplify the creation of company specific models. They support this activity by providing a template from which knowledge about common business processes and organizational structures can be reused. To facilitate this reuse several methods have evolved, which utilize distinct mechanisms (Becker et al. 2006a; vom Brocke 2007). These mechanisms are specified in Figure 1:

- **Configuration**: Reference models can be designed as configurable artifacts. They are provided with specific attributes that contain configuration rules. Depending on the values assigned to the condition part of the rule it can be decided whether the conclusion part of the rule has to be executed. The conclusion part can imply consequences such as the elimination of model elements or the modification of their representation. With this mechanism model variants regarding application-specific characteristics can be created in an automated manner. The reference model is projected into a company-specific model based on the prior selection and annotation of configuration parameters. An important application for this mechanism is for example to tailor a reference model to a specific perspective such as a management view or software engineering view.

- **Instantiation**: When the mechanism of instantiation is used, reference models are equipped with placeholders. The placeholders are inserted within the construction phase of the reference model and annotated with an instantiation domain. When a specific model is created, the placeholders are filled with valid occurrences to the
particular circumstances. Depending on the properties of the application domain, numeric or alphanumeric values, distinct model elements, or even composed model clusters can be defined as instantiation domains.

- **Specialization**: Through specialization a specific model is derived from a more general model by adapting, extending and/or partially modifying the more general one. For this purpose, the model is annotated with *specialization instructions*. Reference models which support specialization have a higher level of abstraction than the resulting company specific models. These more abstract reference models offer only a relatively low number of model elements.

- **Aggregation**: Reference models, which support aggregation, are not available as monolithic blocks but rather contain independent parts, so called *model element components*. Within aggregation, a specific model is built by assembling these model components. Interface descriptions of the model components offer information on their general compatibility and how to combine them.

- **Analogy Construction**: As reference models are general artifacts they can be useful in domains they were not constructed for. The application of a reference model can lead to conclusions by analogy. Analogies can be drawn from any aspect of the reference model which can be indicated by the annotation of *analogy construction advices*. This approach is used especially for analysis or design pattern collections.

Reference models must be prepared to support the mechanisms described above. Figure 1 contains the generic part of a reference modeling method in which all adaptation mechanisms are specified. In order to apply a reference modeling method in a certain domain, a specific part must be defined also. Figure 2 contains a meta model of the Entity-Relationship Model (ERM) (Chen 1976).

![Diagram](image)

**Figure 2. Meta model of a specific part of the reference modeling method**

As all method constructs in Figure 2 inherit from *Model Element* they support the adaptation mechanisms from Figure 1. Subsequently, a reference model can be created by using the specified method. This model can then be annotated with *adaptation attributes* to prepare its transformation into a company specific model. An example for an annotated reference model is given in Figure 3 a). A company specific model is derived by applying the adaptation mechanisms to the reference model. A possible result is given in Figure 3 b). By following the configuration rules, certain parts of the model have been graphically emphasized as the perspective *Customer Support* was chosen. The placeholder for the cardinality has been filled with the value 0, n from the instantiation domain. A model element component has been attached to include the article information via aggregation. No specialization or analogy construction has been performed. Whereas the instantiation has required a manual intervention, the configuration could be performed in an automated manner but may imply the need for a manual semantic consistency check. The automation of aggregation strongly depends on model part interface descriptions. The example shows that all mechanisms can be used in conjunction as well as independent of each other (Becker et al. 2006a). There is no general rule on the sequential use of adaptation mechanisms. It is however sensible to regularly consider large and granular adaptations before small and more detailed ones. Thus, it depends on the conditions of specific project whether a model is for example aggregated first and configured later or contrariwise.

Common application scenarios for these mechanisms can be found in the adaptation, i.e. customization, of Enterprise Systems, which have to be fitted to the specific needs of a company (Soffer et al. 2003). Moreover, the adaptation of conceptual process models supports the customer specific design of organizational structures and operations. For further examples cf. (Becker et al. 2007; Dreiling et al. 2006; Janiesch 2007).
Mechanisms in Method Engineering

The research field of method engineering within the Information Systems (IS) discipline has been established in the 1990’s. Since then many suggestions for the design of methods and the combination of their components have been published. In this section the current state-of-the-art of method engineering research is examined and compared (cf. Table 1).

In IS literature, a method is not considered as a single monolithic block. It rather consists of a set of fragments (e.g. Brinkkemper et al. 1998; Harmsen 1997; Punter and Lemmen 1996; Saeki and Wenyin 1994), also called chunks (e.g. Ralyté and Rolland 2001a) or components (e.g. Karlsson and Wistrand 2006; Song 1997). These fragments can have a very different granularity and they can describe the product (which is created) as well as the process aspect (how is it created) of a method. The fragments can comprise a single activity or construct but they can also contain a complete method. Hence, a method engineering project can start with a set of atomic method fragments which must be assembled as well as an existing method which has to be modified (Ralyté et al. 2003). Method engineering research has mainly focused on the first strategy so far (Bajec et al. 2007). Contrary to that, reference modeling mechanisms primarily focus on an existing model and only additionally consider its design by the aggregation of model element components.
<table>
<thead>
<tr>
<th>Reference</th>
<th>Reference Modeling Mechanism(s)</th>
<th>Mechanism(s) of the Approach</th>
<th>Objects of the Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Baskerville and Stage 2001)</td>
<td>Aggregation, Specialization</td>
<td>Accommodation</td>
<td>Method Fragments</td>
</tr>
<tr>
<td>(Bajec et al. 2007)</td>
<td>Configuration</td>
<td>Process Configuration</td>
<td>Base Method, Configuration Rules, Project Characteristics</td>
</tr>
<tr>
<td>(Brinkkemper 1996; Brinkkemper et al. 1998; Harmsen 1997)</td>
<td>Aggregation</td>
<td>Method Assembly</td>
<td>Method Fragments</td>
</tr>
<tr>
<td>(Cameron 2002)</td>
<td>Aggregation</td>
<td>Tailoring</td>
<td>Work Product Descriptions, Work Breakdown Structures</td>
</tr>
<tr>
<td>(Fitzgerald et al. 2003)</td>
<td>Aggregation</td>
<td>Method Tailoring</td>
<td>Original Formalized Methodologies</td>
</tr>
<tr>
<td>(Greiffenberg 2003b)</td>
<td>Aggregation</td>
<td>Creation of Meta Model</td>
<td>Concepts</td>
</tr>
<tr>
<td>Specialization</td>
<td>Choice of Reference Meta Model Scope</td>
<td>Reference Meta Models</td>
<td></td>
</tr>
<tr>
<td>Analogy Construction</td>
<td>Creation of Meta Model</td>
<td>Typing Patterns</td>
<td></td>
</tr>
<tr>
<td>(Gupta and Prakash 2001)</td>
<td>Aggregation</td>
<td>Method Assembly</td>
<td>Method Components</td>
</tr>
<tr>
<td>(Henninger et al. 2002)</td>
<td>Specialization</td>
<td>Refinement and Tailoring</td>
<td>Software Development Resources</td>
</tr>
<tr>
<td>(Kumar and Welke 1992)</td>
<td>Aggregation</td>
<td>Methodology Engineering</td>
<td>Methodology Components</td>
</tr>
<tr>
<td>(Leppänen 2005)</td>
<td>Aggregation</td>
<td>Method Engineering Methodical Skeleton</td>
<td>Contextual Method Components</td>
</tr>
<tr>
<td>(Nuseibeh 1994)</td>
<td>Aggregation</td>
<td>Template Reuse</td>
<td>Viewpoint Templates</td>
</tr>
<tr>
<td>Specialization</td>
<td>Instantiation</td>
<td>Viewpoint Templates</td>
<td></td>
</tr>
<tr>
<td>Specialization</td>
<td>Inheritance</td>
<td>Super Templates</td>
<td></td>
</tr>
<tr>
<td>(Odell 1995)</td>
<td>Specialization</td>
<td>Single Framework Modeling</td>
<td>Kernel Meta Model</td>
</tr>
<tr>
<td>(Patel et al. 2004)</td>
<td>Aggregation</td>
<td>Selection and Assembly</td>
<td>Method Meta Models</td>
</tr>
<tr>
<td>Specialization</td>
<td>Method Tailoring</td>
<td>Method Fragments</td>
<td></td>
</tr>
<tr>
<td>(Punter and Lemmen 1996)</td>
<td>Aggregation</td>
<td>Assembly</td>
<td>Method Fragments</td>
</tr>
<tr>
<td>(Mirbel and Ralyté 2006; Ralyté et al. 2003; Ralyté and Rolland 2001a; Ralyté and Rolland 2001b)</td>
<td>Aggregation</td>
<td>Assembly-based strategy</td>
<td>Method Chunks</td>
</tr>
<tr>
<td>Specialization</td>
<td>Extension-based strategy, Paradigm-based strategy</td>
<td>Method Chunks, Meta Models</td>
<td></td>
</tr>
<tr>
<td>Analogy Construction</td>
<td>Paradigm-based strategy</td>
<td>Meta Models</td>
<td></td>
</tr>
<tr>
<td>(Rossi et al. 2004; Tolvanen 1998)</td>
<td>Aggregation</td>
<td>Method Construction</td>
<td>Method Components</td>
</tr>
<tr>
<td>Specialization</td>
<td>Method Refinement</td>
<td>Meta Models</td>
<td></td>
</tr>
<tr>
<td>(Saeki and Wenyin 1994)</td>
<td>Aggregation</td>
<td>Method Integration</td>
<td>Meta Models</td>
</tr>
<tr>
<td>(van Offenbeek and Koopman 1996)</td>
<td>Specialization / Analogy Construction</td>
<td>Fit</td>
<td>Scenarios</td>
</tr>
</tbody>
</table>

In their situational method engineering approach Harmsen (1997), Brinkkemper (1996) and Brinkkemper et al. (1998) focus in particular on the recombination of method fragments and, thus, are using the mechanism of aggregation. They describe rules in the context of the aggregation to guide the assembly of method fragments. Comparable approaches were for example published by Fitzgerald et al. (2003), Gupta and Prakash (2001), Punter and Lemmen (1996), and Saeki and Wenyin (1994). Kumar and Welke (1992) handle methodology components similarly but also stress the disassembly of old methods prior to the assembly of new methods. Cameron (2002) puts so called work products together and chooses their temporal order to define a specific development process. Baskerville and Stage (2001) as well as Patel et al. (2004) emphasize the need to adapt
a method after the aggregation by means of deletion, addition and/or modification. This so called method accommodation is considered an application of the specialization mechanism.

Greiffenberg (2003a) has published a comprehensive approach for the development of modeling languages. Greiffenberg specifies a meta modeling language, a reference meta model, a set of meta modeling patterns as well as a process model for method engineering. Following Greiffenberg, the construction of a meta models is based on concepts. For this purpose the mechanism of aggregation is used. Furthermore, Greiffenberg includes the mechanism of specialization for selecting from the reference model. Analogy construction is taken into consideration by the application of meta modeling patterns.

Henninger et al. (2002) and van Offenbeek and Koopman (1996) base their methodology on existing scenarios or available resources. They gather contextual factors such as risk to guide the adaptation process of a method. The procedure results ultimately in a refined method that is a specialized version of the original or that is analog to the original. The refinement process is strongly depending on the fit of original model to the specific problem. A configuration in terms of specific adaptation points or parameters is not in focus.

Karlsson and Ågerfalk (2004), Karlsson (2005), and Karlsson and Wistrand (2006) are one of the few authors in method engineering who directly addresses the mechanism of configuration. Adaptations of methods are performed by the use of configuration packages which rest upon a base method. To manage complex situations with a number of characteristics, configuration packages are combined to configuration templates. Based on the characteristics of a project, an acceptable configuration template is chosen and applied on the base method. Thus, the base method is configured according to the project needs. The configuration of the method focuses only the procedure model. Due to this fact, the modeling language and the resulting products are only taken into consideration indirectly. A comparable approach that also focuses the process of the method is suggested by Bajec et al. (2007).

Leppänen (2005) also makes use of method components that can be aggregated to form situation-specific methods. He, however, focuses on forming an ontology that assists the selection and combination, i.e. integration, of these components. He provides comprehensive interfaces that explicate the compatibility of method components. However, his ontology is not intended to be the basis for any further configuration and, thus, only provides a means to perform method aggregation.

The approach of Nuseibeh (1994) focuses on the multi-perspectivity of software development with the ViewPoint Framework. Due to the abstraction of viewpoints to viewpoint templates, this can be understood as a method engineering approach. New methods can be designed by the combination of viewpoint templates. On the basis of an abstract super template, specialization (inheritance) results in a more specific sub template. By applying the instantiation mechanisms on a viewpoint template, a viewpoint is created.

Odell (1995) suggests a basic vocabulary for describing modeling methods, which is called kernel meta model. With the aid of the kernel meta model, which is based on the mechanism of specialization, it is possible to derive specific concepts of meta models. For example, the concept relation from the kernel meta model can be specialized as is super type of.

Ralyté et al. (2003) provide a generic process model for situational method engineering. With a map notation they describe different strategies to construct a method that meets the contingencies of a project situation. The assembly-based strategy reuses method chunks from a repository and compiles them by applying the aggregation mechanism (Ralyté and Rolland 2001b). The extension-based approach uses the specialization mechanism on an existing method and employs patterns to provide novel additions to it (Ralyté and Rolland 2001a). The paradigm-based approach takes a meta-model that belongs to a certain theoretical framework as starting point. By analogy construction and specialization the meta model is adapted to the specific needs. Mirbel and Ralyté (2006) include the a detailed refinement of a project specific method by each project member. They describe the aggregation of method chunks for the development of project specific methods. For adapting the method with respect to the necessity of every project member, they suggest the adaptation of the procedure (roadmap) of the project specific method by the mechanism of specialization; e.g. the user is able to choose between a prosaic way and use case diagrams to document a requirements analysis depending on his expertise.

Tolvanen (1998) and Rossi et al. (2004) are highlighting the iterative, incremental aspects of method engineering. They assume that due to the inadequate acknowledgement of the application domain at the beginning of the method engineering project, only a part of the method specification is possible. Thus, an iterative process for evaluation and refinement of the method is necessary to reach an adequate description level for the method. This includes adaptation (specialization) and addition of missed constructs (aggregation).

This literature review has shown that the adaptation mechanisms of aggregation, instantiation, specialization and analogy construction are widely used by method engineering researchers. Especially aggregation and specialization are included in many approaches. However, the configuration of modeling methods and here especially the product and language aspects
have not been sufficiently addressed yet. Hence, the aim of this work is to transfer the mechanism of configuration into method engineering research.

**Configurative Method Engineering**

**Configuration of Conceptual Modeling Methods**

Constituent elements of methods are actions, artifacts, constructs, notations, and actor roles (Braun et al. 2005; Karlsson 2005). Figure 4 puts these elements into relation. Actor roles carry out actions. Actions are related to each other in form of a network and can be nested to represent phases or, on a higher level of abstraction, procedure models. The output of an action, an artifact, can be the input for a subsequent action. Within an action, artifacts, such as class diagrams or activity diagrams, are produced and utilized. Artifacts are described by notations. Constructs represent the contextual dimension of a notation and provide the formal vocabulary to describe a certain domain. Also, constructs can be nested to form modeling languages and their views. These method elements (as well as parts thereof) as well as their relations (not described here) can be subject to configuration. Methods and their corresponding configuration mechanisms can be supported by tools that implement them.

![Figure 4. Meta model of a configurable method (cf. Braun et al. 2005; Karlsson 2005)](image)

Similar to configurable reference models also configurable methods can be annotated with *configuration rules*. Configuration rules describe in which way methods can be adapted to specific-application contexts. The rules are composed of a condition part and a conclusion part. The *condition part* defines the application contexts by logical combination of configuration parameters. These parameters can include but are not limited to objectives of the method’s use, application systems or respectively organizations to be designed, roles, qualifications, and other attributes of the method users as well as financial, time, personnel, and other restrictions. Contingency factors for methods adaptations can be found for example in Davis (1982), Offenbeek and Koopman (1996), or Pérez et al. (1995). The *conclusion part* of the configuration rule describes the kind of adaptation that is performed on the method. It specifies the method elements of a configurable method which either persist within the organization specific method or have to be eliminated. The specific *configuration points* at which configuration rules can be annotated may differ depending on the method to be made configurable. Common configuration points for modeling methods are at the levels of constructs, actions or roles. Furthermore, deviations in notation and terminology can be incorporated.
Configurable methods comprise knowledge about all situation-specific methods, which can be derived from them. Therefore, the configurable methods are, similar to reference models, very comprehensive. They can contain method elements which are used only occasionally or that exhibit very different levels of granularity. This is necessary to support rule-based modifications for very particular cases as well as to be able to get project-specific or company-specific methods with diverse granularities.

The annotation of the configuration rules has to be performed by the method engineer prior to making the method available for configuration to method users. In this way, the engineer provides a consistent reference method or method master that can be adapted to the specific situational needs of a company, domain, purpose, or project. The challenge of a method engineer is to bear in mind the interdependencies between method elements and configuration rules (cf. similarly Becker et al. 2006b). In this way, the set of terms for any configuration parameter occurrence results in a consistent method variant.

**Tool Support for Method Configuration**

With respect to the corresponding adaptation concepts of reference modeling, a tool for configurable method engineering has to support two phases: In the first phase, configurable methods are specified by annotating elements with configuration rules and the definition of a condition and a conclusion part. In the second phase, methods are configured based on the selection of situation specific parameters. By matching the situation parameters with the condition part of the configuration rules it is decided whether the conclusion part is executed and method elements are eliminated or modified.

The H2 Toolset (Becker et al. 2006c) represents a proof-of-concept for configurative method engineering as presented above. H2 is a meta modeling tool designed to create hierarchically structured modeling methods and to construct models based on the defined methods. In H2 both models and modeling methods can be defined and modified in a hierarchical form. For the purpose of this paper H2 is not used to model a certain application domain but to model a method. Hence, the H2 tool is applied one meta level above its regular use. H2 supports both phases of configurative method engineering, the annotation of a method with attributes and its adaptation.

Figure 5 contains the specification of a meta model of a configurable method in H2. It comprises important elements of the meta model from Figure 4 in an hierarchical representation. It includes the elements action, artifact, notation, construct, actor role, and tool. To clarify the structure of actions, procedure model and phase have been added. Based on this specification it is now possible to define a configurable method.

![Figure 5. Definition of configurable method meta model in H2](image)

Figure 6 contains as an example a simplified waterfall model. The configuration process in H2 is roughly the following: Firstly, the configuration rules are assigned to model elements and configuration parameters are added to the condition part.
This is done with the configuration customizer and the term editor (cf. right tab of Figure 6). Secondly, the conclusion part is defined that enables the engineer to either dismiss elements from the method or to include them explicitly.

The configuration itself is checkbox-based and performable by the method users. A simple example reads as follows (again cf. Figure 6 for illustration): By selecting the role of a software architect and open source as a type of project, the configuration of the method would result in the omission of all activities and their superordinate phases that are not performed by the software architect and the dismissal of any software tool that is not open source such as the products by Microsoft and Rational. This represents only a very simple configuration task as configuration can take place on many other configuration points as introduced in the previous section.

**Figure 6. Exemplary configuration of a method in H2**

**Summary and Further Research**

Methods provide a means for structured problem solving. The determining factors of their application change from project to project and require situation-specific adaptations of the method. Reference models are adapted to company-specific models by the use of configuration, specialization, aggregation, instantiation, and analogy construction. It is reasonable to transfer the mechanisms of reference modeling to the domain of method engineering. An analysis of the body of knowledge of method
engineering revealed that the mechanism of configuration – as it is applied in reference modeling – was hardly recognized in the context of method engineering. This absence marked the starting point for an operationalization of configuration for method engineering, which was exemplified by employing the H2 Toolset.

The definition of configuration rules enables their empirical evaluation and fosters theory building. This, however, requires that configurable methods are available. Paramount to further research is to enhance current methods with appropriate configuration rules. In addition to that, it is necessary to modify modeling tools to assure software-based support for method configuration. This step is also necessary to evaluate our approach. Furthermore, the design of procedures is required that assist the configuration of complex methods. Since configuration intents to hide complexity from the user, criteria for the design of user interfaces (Shneiderman 2004) as well as high level configuration approaches (Janiesch et al. 2006) have to be taken into consideration.

With configuration mechanisms for methods available, the actual definition of configuration rules and their parameters moves into the focus of interest. In particular, a combination of the presented proposal with the approach of Karlsson (2005) and the contextual parameters of Leppänen (2005) seems to be promising. Apart from configuration also the mechanism of instantiation is only sporadically considered by the method engineering community. It is due to further research to extend this approach.

References


