December 2007

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

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. logics 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:

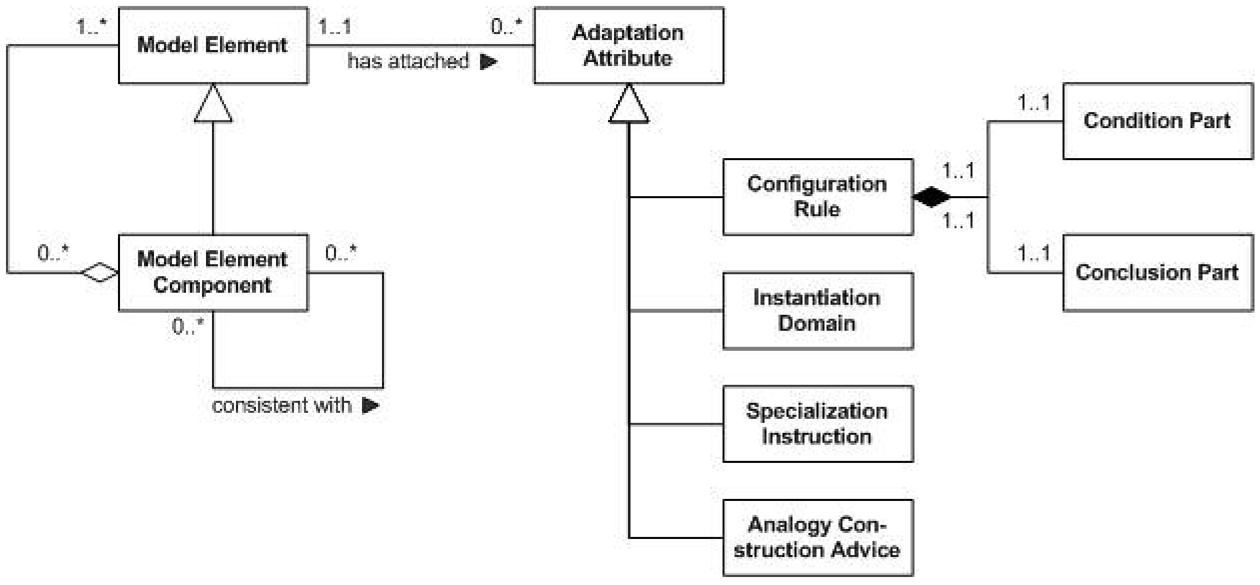


Figure 1. Meta model of the generic part of an adaptable reference modeling method

- *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).

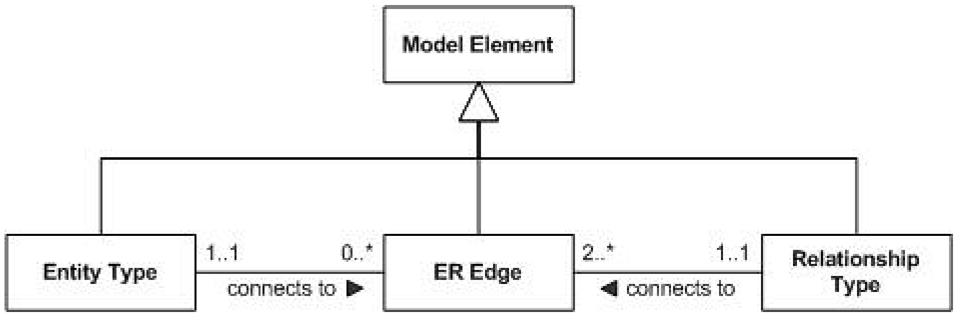


Figure 2. Meta model of a specific part of the reference modeling method

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).

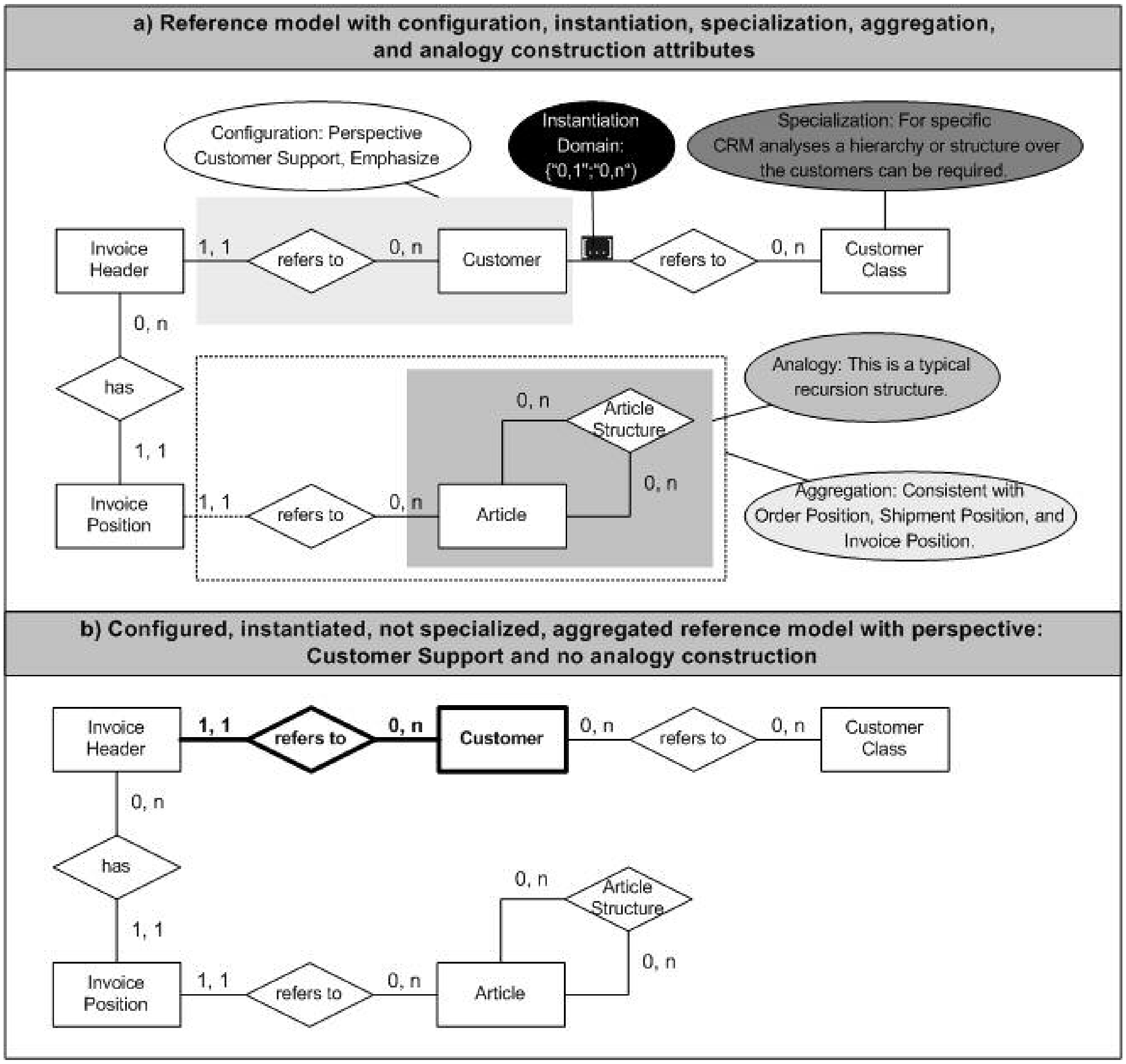

a) Reference model with configuration, instantiation, specialization, aggregation, and analogy construction attributes
Configuration: Perspective Customer Support, Emphasize
Instantiation Domain: {"0,1";"0,n"}
Specialization: For specific CRM analyses a hierarchy or structure over the customers can be required.
Invoice Header
1, 1
refers to
0, n
Customer
refers to
0, n
Customer Class
0, n
has
1, 1
Analogy: This is a typical recursion structure.
0, n
Article Structure
0, n
Aggregation: Consistent with Order Position, Shipment Position, and Invoice Position.
Invoice Position
1, 1
refers to
0, n
Article
b) Configured, instantiated, not specialized, aggregated reference model with perspective: Customer Support and no analogy construction
Invoice Header
1, 1
refers to
0, n
Customer
0, n
refers to
0, n
Customer Class
0, n
has
1, 1
0, n
Article Structure
0, n
Invoice Position
1, 1
refers to
0, n
Article

Figure 3. Application of reference modeling mechanisms

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 1. Overview about mechanisms in method engineering approaches

Reference	Reference Modeling Mechanism(s)	Mechanism(s) of the Approach	Objects of the Mechanism
(Baskerville and Stage 2001)	Aggregation, Specialization	Accommodation	Method Fragments
(Bajec et al. 2007)	Configuration	Process Configuration	Base Method, Configuration Rules, Project Characteristics
(Brinkkemper 1996; Brinkkemper et al. 1998; Harmsen 1997)	Aggregation	Method Assembly	Method Fragments
(Cameron 2002)	Aggregation	Tailoring	Work Product Descriptions, Work Breakdown Structures
(Fitzgerald et al. 2003)	Aggregation	Method Tailoring	Original Formalized Methodologies
(Greiffenberg 2003b)	Aggregation	Creation of Meta Model	Concepts
	Specialization	Choice of Reference Meta Model Scope	Reference Meta Models
	Analogy Construction	Creation of Meta Model	Typing Patterns
(Gupta and Prakash 2001)	Aggregation	Method Assembly	Method Components
(Henninger et al. 2002)	Specialization	Refinement and Tailoring	Software Development Resources
(Karlsson 2005; Karlsson and Ågerfalk 2004; Karlsson and Wistrand 2006)	Configuration	Configuration Framework	Base Method, Configuration Package, Configuration Template
(Kumar and Welke 1992)	Aggregation	Methodology Engineering	Methodology Components
(Leppänen 2005)	Aggregation	Method Engineering Methodical Skeleton	Contextual Method Components
(Nuseibeh 1994)	Aggregation	Template Reuse	Viewpoint Templates
	Instantiation	Instantiation	Viewpoint Templates
	Specialization	Inheritance	Super Templates
(Odell 1995)	Specialization	Single Framework Modeling	Kernel Meta Model
(Patel et al. 2004)	Aggregation	Selection and Assembly	Method Fragments
	Specialization	Method Tailoring	Method Fragments
(Punter and Lemmen 1996)	Aggregation	Assembly	Method Fragments
(Mirbel and Ralyté 2006; Ralyté et al. 2003; Ralyté and Rolland 2001a; Ralyté and Rolland 2001b)	Aggregation	Assembly-based strategy	Method Chunks
	Specialization	Extension-based strategy, Paradigm-based strategy	Method Chunks, Meta Models
	Analogy Construction	Paradigm-based strategy	Meta Models
(Rossi et al. 2004; Tolvanen 1998)	Aggregation	Method Construction	Method Components
	Specialization	Method Refinement	Meta Models
(Saeki and Wenyin 1994)	Aggregation	Method Integration	Meta Models
(van Offenbeek and Koopman 1996)	Specialization / Analogy Construction	Fit	Scenarios

In their situational method engineering approach Harmsen (1997), Brinkemper (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.

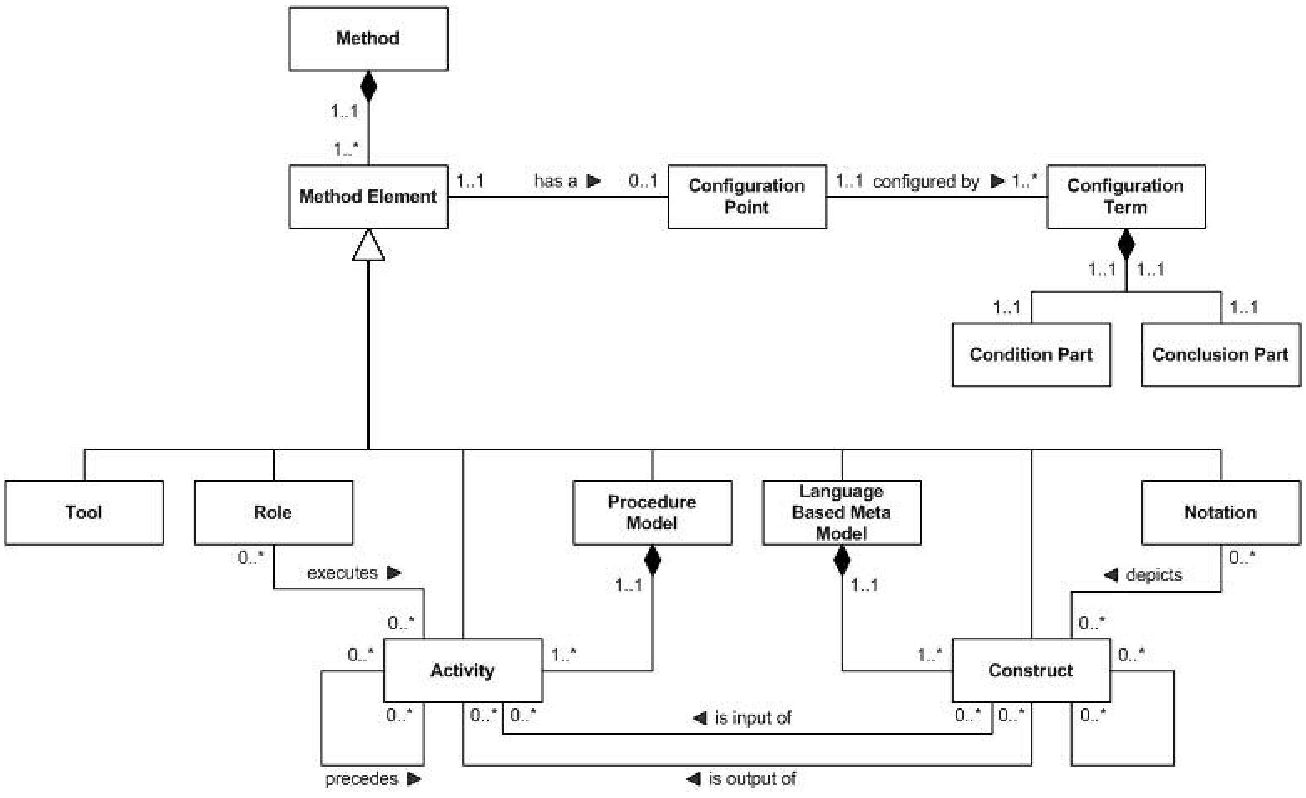


Figure 4. Meta model of a configurable method (cf. Braun et al. 2005; Karlsson 2005)

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.

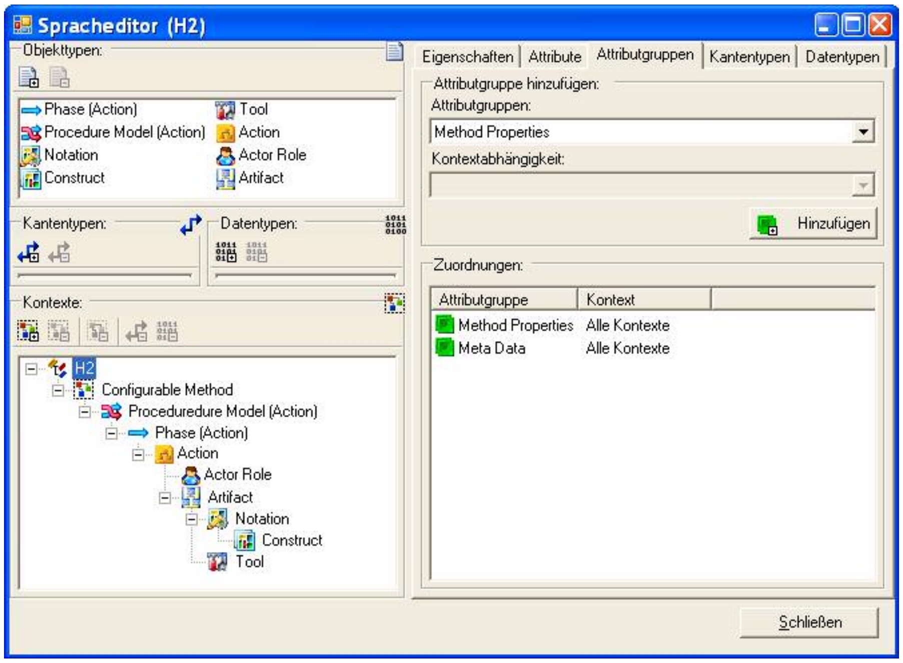
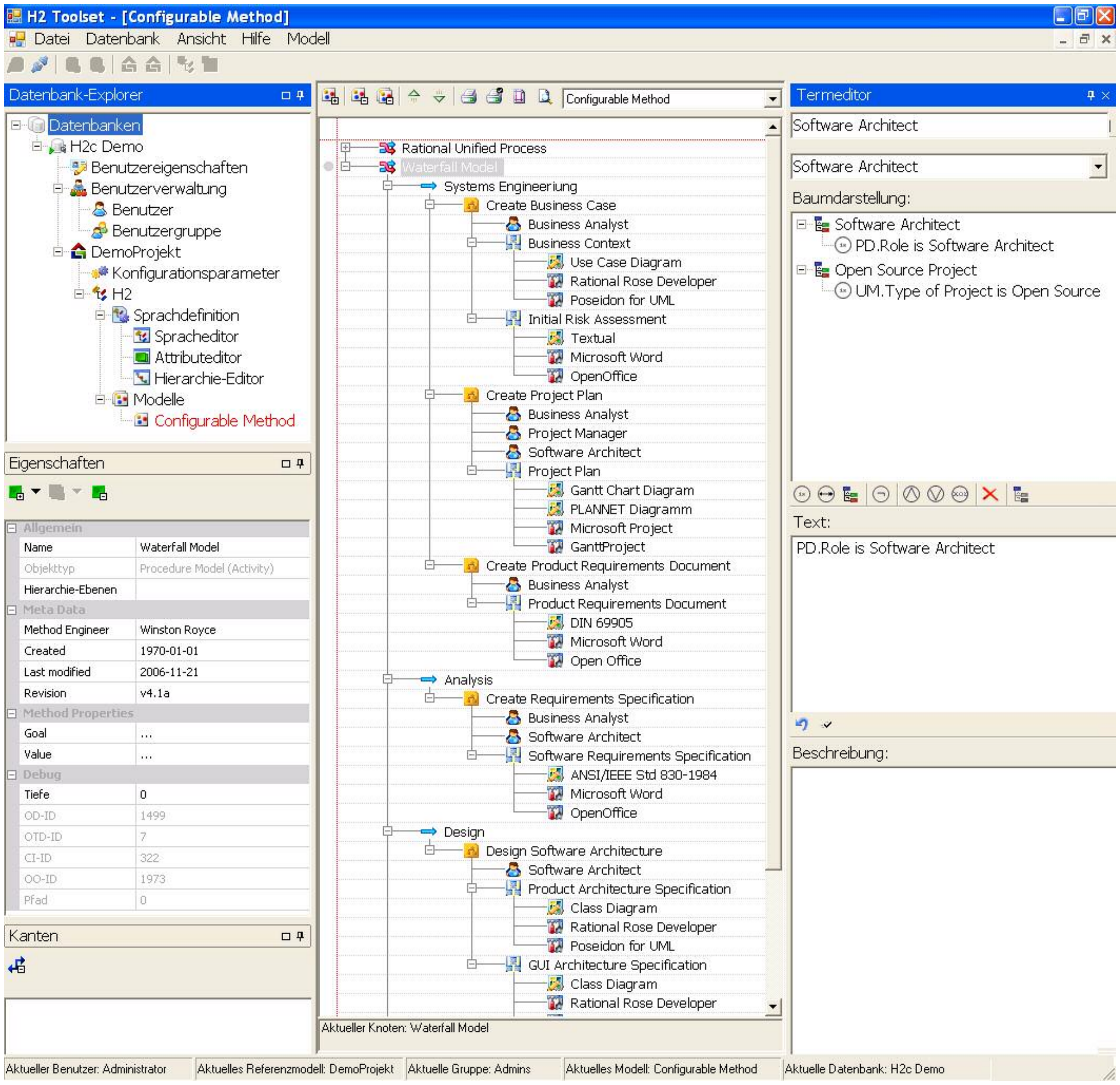


Figure 5. Definition of configurable method meta model in H2

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 super ordinate 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.



```
Datenbanken
  H2c Demo
    Benutzereigenschaften
    Benutzerverwaltung
      Benutzer
      Benutzergruppe
    DemoProjekt
      Konfigurationsparameter
      H2
        Sprachdefinition
          Spracheditor
          Attributeditor
          Hierarchie-Editor
        Modelle
          Configurable Method
```

Allgemein	
Name	Waterfall Model
Objekttyp	Procedure Model (Activity)
Hierarchie-Ebenen	
Meta Data	
Method Engineer	Winston Royce
Created	1970-01-01
Last modified	2006-11-21
Revision	v4.1a
Method Properties	
Goal	...
Value	...
Debug	
Tiefe	0
OD-ID	1499
OTD-ID	7
CI-ID	322
OO-ID	1973
Pfad	0

Configurable Method

```
Rational Unified Process
Waterfall Model
  Systems Engineerung
    Create Business Case
      Business Analyst
      Business Context
        Use Case Diagram
        Rational Rose Developer
        Poseidon for UML
      Initial Risk Assessment
        Textual
        Microsoft Word
        OpenOffice
    Create Project Plan
      Business Analyst
      Project Manager
      Software Architect
      Project Plan
        Gantt Chart Diagram
        PLANNET Diagramm
        Microsoft Project
        GanttProject
    Create Product Requirements Document
      Business Analyst
      Product Requirements Document
        DIN 69905
        Microsoft Word
        Open Office
  Analysis
    Create Requirements Specification
      Business Analyst
      Software Architect
      Software Requirements Specification
        ANSI/IEEE Std 830-1984
        Microsoft Word
        OpenOffice
  Design
    Design Software Architecture
      Software Architect
      Product Architecture Specification
        Class Diagram
        Rational Rose Developer
        Poseidon for UML
      GUI Architecture Specification
        Class Diagram
        Rational Rose Developer
```

Aktueller Knoten: Waterfall Model

Software Architect

Software Architect

Baumdarstellung:

```
Software Architect
  PD.Role is Software Architect
Open Source Project
  UM.Type of Project is Open Source
```

Text: PD.Role is Software Architect

Aktueller Benutzer: Administrator | Aktuelles Referenzmodell: DemoProjekt | Aktuelle Gruppe: Admins | Aktuelles Modell: Configurable Method | Aktuelle Datenbank: H2c Demo

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

Bajec, M., Vavpotič, D., and Krisper, M. "Practice-driven approach for creating project-specific software development methods," *Information and Software Technology* (49:4) 2007, pp. 345-365.

Baskerville, R., and Stage, J. "Accommodating Emergent Work Practices: Ethnographic Choice of Method Fragements," IFIP TC8/WG8.2 Working Conference on Realigning Research and Practice in IS Development: The Social and Organisational Perspective, Boise, ID, 2001, pp. 12-28.

Becker, J., Delfmann, P., and Knackstedt, R. "Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models," Reference Modeling Conference (RefMod2006), Passau, 2006a.

Becker, J., Janiesch, C., and Dreiling, A. "A Framework for Interdependent Configuration of Enterprise Systems," (Pre-)ICIS Inaugural Workshop on Enterprise Systems Research in MIS, Milwaukee, WI, 2006b.

Becker, J., Janiesch, C., Seidel, S., and Brelage, C. "A Framework for Situational and Evolutionary Language Adaptation in Information Systems Development," 15th International Conference on Information Systems Development (ISD 2006), Budapest, 2006c.

Becker, J., Kugeler, M., and Rosemann, M. (eds.) *Process Management: A Guide for the Design of Business Processes*. Springer, Berlin, 2007. To appear.

Becker, J., and Schütte, R. "A Reference Model for Retail Enterprises," in: *Reference Modeling for Business Systems Analysis,* P. Fettke and P. Loos (eds.), Idea Group, Hershey, USA, 2007, pp. 182-205.

Braun, C., Wortmann, F., Hafner, M., and Winter, R. "Method Construction: A Core Approach to Organizational Engineering," 20th ACM Symposium on Applied Computing, Santa Fe, NM, 2005, pp. 1295-1299.

Brinkkemper, S. "Method engineering - engineering of information systems development methods and tools," *Information and Software Technology* (38:4) 1996, pp. 275-280.

Brinkkemper, S., Saeki, M., and Harmsen, F. "Assembly Techniques for Method Engineering," 10th International Conference on Advanced Information Systems Engineering (CAiSE 1998). Lecture Notes in Computer Science, Springer, Pisa, 1998, pp. 381-400.

Brooks, F.P. "Essence and Accidents of Software Engineering," *IEEE Computer* (20:4) 1987, pp. 10-19.

Cameron, J. "Configurable Development Processes," *Communications of the ACM* (45:3) 2002, pp. 72-77.

Chen, P.P.-S. "The Entity Relationship Model - Toward a Unified View of Data," *ACM Transaction on Database Systems* (1:1) 1976, pp. p. 9-36.

Davis, G.B. "Strategies for information requirements determination," *IBM Systems Journal* (21:1) 1982, pp. 4-30.

Dreiling, A., Rosemann, M., van der Aalst, W., Heuser, L., and Schulz, K. "Model-Based Software Configuration: Patterns and Languages," *European Journal of Information Systems* (15:6) 2006, pp. 583-600.

Fettke, P., and Loos, L. "Classification of reference models—a methodology and its application," *Information systems and e-business management* (1:1) 2003, pp. 35-53.

Fitzgerald, B., Russo, N.L., and O'Kane, T. "Software Development: Method Tailoring at Motorola," *Communications of the ACM* (46:4) 2003, pp. 65-70.

Greiffenberg, S. "Methoden als Theorien der Wirtschaftsinformatik," 6. Internationale Tagung Wirtschaftsinformatik (WI 2003), Physica, Dresden, 2003a, pp. 947-968.

Greiffenberg, S. *Methodenentwicklung in Wirtschaft und Verwaltung* Verlag Dr. Kovac, Hamburg, 2003b.

Gupta, D., and Prakash, N. "Engineering Methods from Method Requirements Specifications," *Requirements Engineering* (6:3) 2001, pp. 135-160.

Harmsen, A.F. "Situational Method Engineering," Utrecht, 1997.

Henninger, S., Ivaturi, A., Nuli, K., and Thirunavukkaras, A. "Supporting Adaptable Methodologies to Meet Evolving Project Needs," Joint Conference on XP Universe and Agile Universe, Chicago, IL, 2002, pp. 33-44.

Janiesch, C. "Implementing Views on Business Semantics: Model-based Configuration of Business Documents," 15th European Conference on Information Systems (ECIS 2007), St. Gallen, 2007.

Janiesch, C., Dreiling, A., and Seidel, S. "Document Variant Management: Facilitating Enterprise System Definition, Configuration, and Interoperability," 17th Australasian Conference on Information Systems (ACIS 2006), Adelaide, 2006.

Karlsson, F. "Method Configuration: Method and Computerized Tool Support," Linköping, 2005.

Karlsson, F., and Ågerfalk, P.J. "Method configuration: adapting to situational characteristics while creating reusable assets," *Information and Software Technology* (46:9) 2004, pp. 619-633.

Karlsson, F., and Wistrand, K. "Combining method engineering with activity theory: theoretical grounding of the method component concept," *European Journal of Information Systems* (15:1) 2006, pp. 82-90.

Kautz, K. "The Enactment of Methodology: The Case of Developing a Multimedia Information System," 25th International Conference on Information Systems (ICIS 2004), Washington, D.C., 2004, pp. 671-683.

Kelly, S., Rossi, M., and Tolvanen, J.-P. "What is Needed in a MetaCASE Environment?," *Enterprise Modelling and Information Systems Architectures* (1:1) 2005, pp. 25-35.

Kumar, K., and Welke, R.J. "Methodology Engineering: A Proposal for Situation-specific Methodology Construction," in: *Challenges and Strategies for Research in Systems Development,* W.W. Cottermann and J.A. Senn (eds.), John Wiley & Sons Ltd., Chichester, 1992, pp. 257-269.

Leppänen, M. "An Ontological Framework and a Methodical Skeleton for Method Engineering: A Contextual Approach," Jyväskylä, Jyväskylä, 2005.

Lindvall, M., and Rus, I. "Process Diversity in Software Development," *IEEE Software* (17:4) 2000, pp. 14-18.

Luoma, J., Kelly, S., and Tolvanen, J.-P. "Defining Domain-Specific Modeling Languages: Collected Experiences," 4th OOPSLA Workshop on Domain-Specific Modeling (DSM 2004), Vancouver, 2004.

Mirbel, I., and Ralyté, J. "Situational Method Engineering: Combining Assembly-based and Roadmap-driven Approaches," *Requirements Engineering* (11:1) 2006, pp. 58-78.

Nuseibeh, B.A. "A Multi-Perspective Framework for Method Integration," London, 1994.

Odell, J. "Meta-modelling," OOPSLA'95 Workshop on Metamodelling in OO Austin, TX, 1995.

Patel, C., de Cesare, S., Iacovelli, N., and Merico, A. "A Framework for Method Tailoring: A Case Study," 2nd OOPSLA Workshop on Method Engineering for Object-Oriented and Component-Based Development, Vancouver, 2004.

Pérez, G., El Emam, K., and Madhavji, N.H. "Customising Software Process Models," 4th European Workshop on Software Process Technology (EWSPT 1995), Noordwijkerhout, The Netherlands, 1995, pp. 70-78.

Punter, T., and Lemmen, K. "The MEMA-model: towards a new approach for Method Engineering," *Information and Software Technology* (38:4) 1996, pp. 295-300.

Ralyté, J., Deneckère, R., and Rolland, C. "Towards a Generic Model for Situational Method Engineering," 15th International Conference on Advanced Information Systems Engineering (CAiSE 2003), Klagenfurt/Velden, Austria, 2003, pp. 95-110.

Ralyté, J., and Rolland, C. "An Approach for Method Reengineering," 20th International Conference on Conceptual Modeling (ER 2001), Yokohama, Japan, 2001a, pp. 471-484.

Ralyté, J., and Rolland, C. "An Assembly Process Model for Method Engineering," 13th International Conference on Advanced Information Systems Engineering (CAiSE 2001). Lecture Notes in Computer Science, Interlaken, 2001b, pp. 267-283.

Rosemann, M., and van der Aalst, W.M.P. "A configurable reference modelling language," *Information Systems* (32:1) 2007, pp. 1-23.

Rossi, M., Ramesh, B., Lyytinen, K., and Tolvanen, J.-P. "Managing Evolutionary Method Engineering by Method Rationale," *Journal of the Association for Information Systems* (5:9) 2004, pp. 356-391.

Saeki, M., and Wenyin, K. "Specifying software specification & design methods," 6th International Conference on Advanced Information Systems Engineering (CAiSE 1994), Utrecht, The Netherlands, 1994.

Scheer, A.-W., Jost, W., and Güngöz, Ö. "A Reference Model for Industrial Enterprises," in: *Reference Modeling for Business Systems Analysis,* P. Fettke and P. Loos (eds.), Idea Group, Hershey, USA, 2007, pp. 167-181.

Shneiderman, B. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, (4th ed.) Addison-Wesley Amsterdam, 2004.

Soffer, P., Golany, B., and Dori, D. "ERP modeling: a comprehensive approach," *Information Systems* (28:6) 2003, pp. 673-690.

Song, X. "Systematic integration of design methods," *IEEE Software* (14:2) 1997, pp. 107-117.

ter Hofstede, A.H.M., and Verhoef, T.F. "On the Feasibility of Situational Method Engineering," *Information Systems* (22:6/7) 1997, pp. 401-422.

Tolvanen, J.-P. "Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence," M. Sakkinen (ed.), Jyväskylä, 1998.

van Offenbeek, M.A.G., and Koopman, P.L. "Scenarios for System Development: Matching Context and Strategy," *Behaviour & Information Technology* (15:4) 1996, pp. 250-265.

vom Brocke, J. "Design Principles for Reference Modelling – Reusing Information Models by Means of Aggregation, Specialisation, Instantiation, and Analogy," in: *Reference Modeling for Business Systems Analysis,* P. Fettke and P. Loos (eds.), Idea Group Publishing, Hershey, 2007, pp. 47-75.

Wistrand, K., and Karlsson, F. "Method Components – Rationale Revealed," 16th International Conference on Advanced Information Systems Engineering (CAiSE 2004), Riga, Latvia, 2004, pp. 189-201.