

Consequences of Meta-Model Modifications within Model Configuration Management

Jens Weller, Werner Esswein

Technische Universität Dresden
Information Systems, esp. Systems Engineering
01062 Dresden
{Jens.Weller|Werner.Esswein}@tu-dresden.de

Abstract: Today, conceptual models are intensively used in the information systems discipline. They can support the development and adaptation of software systems as well as the (re-)design of organizations. As conceptual models change during their lifetime, there is a need to manage different version of models. Thus in the past years, findings in software configuration management has been transferred to the conceptual modeling field.

In this paper, we will assign the experiences made in model configuration management to the meta-modeling field. We discuss consequences of meta-model modifications and analyze the process of migrating related conceptual models to a modified meta-model version.

1 Introduction

Conceptual modeling has been established in the past years. While in the beginning, conceptual models were mainly used in software engineering [DJM⁺00] [Flo01], today they are used for organizational issues as well [RSD05]. As conceptual models change during their lifetime, there can be different versions of a model, each valid for a different period of time. To support the administration of those versions, the knowledge of software configuration management has been transferred to the conceptual modeling field. Configuration management of models can increase the quality of conceptual models and enable team work within the modeling process [EGK02].

In contrast to the software development field, models used for organizational issues are mostly created using different, company specific modeling languages [BSH98]. To support the modeling process with different, individual modeling languages, meta-modeling tools have been widely established. Those tools enable the user to define meta-models, representing the language that will be used within the modeling project. Afterwards, the meta-models can be used for conceptual modeling instantly [Sem05] or for creating new modeling tools based on the defined language [KRT05].

As meta-models are models as well, different version of meta-models can exist too. Thus,

organizations must deal with different meta-model versions [Tol98] [Sae06]. As we will show in this paper, configuration management can be applied to the meta-modeling field to handle different versions of these meta-models. Furthermore, we analyze consequences arising from changes on meta-models that are administrated by such a system.

The paper is structured as follows. In section 2, we explain the fundamentals of conceptual models and configuration management. Afterwards in section 3, we discuss the main concepts of the configuration management of models and meta-models. Section 4 deals with consequences arising from meta-model changes and in section 5, we analyze the adaptation of conceptual models according to a modified meta-model. Section 6 summarizes our findings.

2 Background

2.1 Models and Meta-Models

A model is understood as the result of a construction process „... done by a modeler, who examines the elements of a system for a specific purpose ...“ [SR98], which is defined by the user of the model. Depending on the kind of system that is examined during the modeling process (the problem domain), models in the information systems field can be classified as design models or conceptual models. While design models represent software systems or parts of it, conceptual models describe real world phenomena [EW05] [DJM⁺00] [WW02]. The language used to express the model is called the modeling language. Different modeling languages are currently used, which comply with different problem domains as well as with different modeling purposes [BSH98]. Lastly, models are valid for a certain time interval leading to constant modifications of these models during their lifetime.

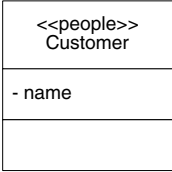
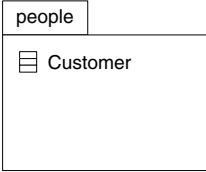
<i>model element content (uml class & package)</i>	<i>representation #1 (uml class diagram)</i>	<i>representation #2 (uml package diagram)</i>
<ul style="list-style-type: none"> + Customer (Type: Class) - name (Type: Attribute) - people (RelationTo: Package) + people (Type: Package) 	 <pre> classDiagram class Customer { - name } Customer --> people </pre>	 <pre> packageDiagram package people { class Customer } </pre>

Figure 1: Content and representation of model elements

In the information systems discipline, models are used to redesign organizations or to develop software systems [RSD05] [FL02]. Therefore, graphical models have been established as „...a medium to foster communication with prospective users...“ [Fra99]. For

reasons of simplicity, complex models often consist of several views showing just a section of the whole model [Sch98] [Str96]. The different views, however, are not separated but integrated using the same model elements within different views [Gre03] [SR98]. Thus we can argue that a model always consists of its content and one or more graphical representation, each of which illustrates a set of the model's elements and their relationships (see figure 1).

As mentioned above, models are created using a specific modeling language. Modeling languages itself can be described by models as well. Models which represent a modeling language are called meta-models [Str96] [Ham99]. According to Strahringer [Str96], models describing the process how to create another model can be understood as meta-models too. In this paper, however, we concentrate on meta-models representing modeling languages only. Meta-models are mainly used in the method engineering field. To support the use of company dependent modeling methods, tools have been established to assist the user with the creation and use of meta-models [Sem05] [KRT05] [Sae03].

2.2 Configuration Management

Configuration management (CM) has been intensely discussed in theory and established in practice for many years. Norms like ISO 2000:9000 demand it to increase process and product quality [Int00] and it is therefore also suggested by process maturity models [PCCW93] [Kne03]. In general, configuration management is defined as an „...activity that applies technical and administrative direction over the life cycle of a product, its configuration items, and related product configuration information.“ [Int03]

While configuration management is an activity, a configuration management system is a socio-technical system consisting of people, organizational rules, tools and their relationships realizing configuration management. Configuration management tools support one or more of the CM activities [Est00].

The configuration item is the basic element in configuration management. *Configuration items* are the elements (source code file, requirements document, model) being under control of the CM [Int03] [CW98]. To document all modifications of an item, every modification leads to a new state of the item which are called *versions* [Zel97] [BEGW06]. According to the purpose of the modification, versions can be characterized as follows:

- Versions that are created to replace an older version (e. g. for the purpose of development or maintenance) are called historical versions or *revisions* [Zel97] [Tho06].
- In contrast, versions created with the intention to coexist are called *variants*. Variants are usually created to support different user purposes, e. g. a car component for different user groups [Tho97]. The process to create variants can be subsumed under the term of logical versioning [Zel97].
- Versions can also be created to support parallel working. Thereby, changes made on configuration items are done in different workspaces. A workspace is an „...individual area of a developer, isolating him from changes made by others...“ [Zel97].

Coexisting versions for the purpose of the development in different workspaces are called *temporary variants* [BEGW06].

A set of all versions of exactly one configuration item is called a *version family*. Version graphs are established to illustrate version families and the different kinds of versions existing within a version family [CW98]. Figure 2 shows an example of such a graph.

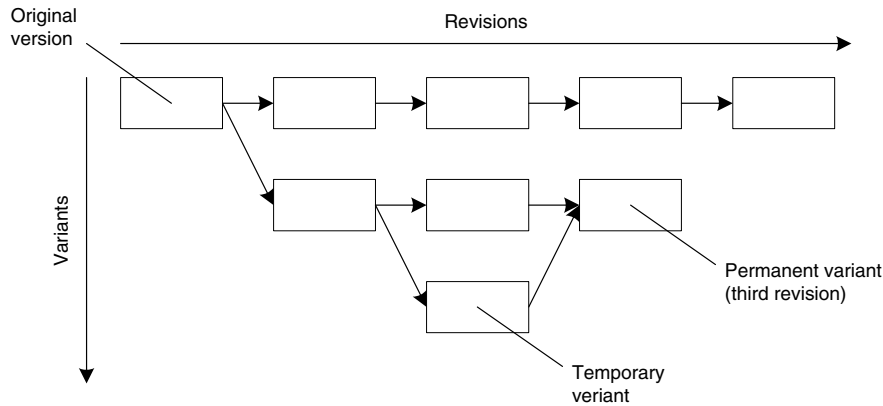


Figure 2: Example of a version graph [Zel97]

As the versions of different configuration items are independent from each other [Est00], there has to be a container keeping them all together. This container is called the configuration. A *configuration* is a bundle of versions of configuration items, which represent a complex product [Int03]. It is itself a configuration item. Thus, a new version of a configuration is created, when modifying an item that belongs to the configuration.

3 Configuration Management of Models and Meta-Models

Most research on configuration management is done in the software development discipline [CW98] [ELC⁺02] [Zel97]. However, in the past years CM has also been applied to the information systems field. Esswein et al. used configuration management to support the creation and maintenance of models [EGK02]. The so called model configuration management enables the tracking of changes made on a model and supports cooperative development of models. The model CM has been enhanced by Greiffenberg [Gre03] and was finally implemented in a modeling tool [Sem05]. A similar approach has been developed by Saeki and Oda [SO05] [Sae06]. The authors discussed the configuration management of methods (and their meta-models respectively) to embed CM into a CAME¹ tool. Further research has been done by adopting configuration management of models to the reference modeling field [BEGW06] [Tho06].

¹CAME = Computer Aided Method Engineering

The motivation for developing a CM for models instead of using existing configuration management systems from the software engineering field are the differences between the configuration items administrated by these systems. In source code oriented CM systems configuration items are text files and differences between those files are recovered by comparing the files line by line. „Since we use diagram documents..., we should manage the changes on the diagrams, not in the granularity of a line, but of a logical component...“ [SO05], such as a model element. Thus, within model configuration management configuration items represent model elements. A complete model is represented as a configuration [EGK02] [BEGW06].²

According to the principles of configuration management to store all changes as revisions, in model CM it is possible to go back and forth in the revisions graph to restore an older version of a model. Additionally, the creation of variants and temporary variants is possible within a model CM. The latter enables the cooperative development of models. „In this case, multiple developers work in parallel on different versions.“ [CW98] Thus, all modifications on a model are made in workspaces [EGK02]. Models are interchanged between the workspaces using a central repository. Different CM operations exist to transfer models from the repository to a user’s workspace and vice versa (see [BEGW06] for an overview).

As meta-models are models as well, model configuration management can also deal with meta-models [Sae06] [Gre03]. In this case a configuration represents a meta-model and configuration items represent meta-model elements. While the administration of meta-models is the same as for models, the dependencies between meta-model versions and the versions of its instances (the models) have to be analyzed. Thus, when a meta-model is changed, we might need to check if the existing models are still consistent with the changed meta-model [Sae06].

This fact, however, has attained only little attention in literature. None of the existing approaches discuss in which cases the above mentioned consistency check is necessary. While Saeki demands in [Sae06] the evaluation of derived models to guide the modeler, Greiffenberg changes existing models automatically [Gre03]. There might be cases, however, where it is not necessary or not desirable to adapt the models. Furthermore, changes on the graphical representation of meta-model elements are mostly disregarded in current analysis [Sae06]. Thus, in the next section, we discuss the consequences of meta-model changes within a configuration management system that administrates both, models and their meta-models. Thereby, we will consider the mentioned deficiencies of existing approaches.

4 Meta-Model enhancement

As pointed out in the previous section, models are valid within a given period of time. As basic elements within method engineering projects, meta-models also change during

²While Saeki and Oda use the term ‘product’ for a complete model [SO05], we use the term ‘configuration’ to comply with the definitions given by configuration management in general.

their lifetime with respect to the modelers individual needs [HBO94] [Tol98]. To analyze the consequences arising from a meta-model modification, we will first describe different situations models and meta-models are used in and discuss different ways how to handle meta-model changes within these situations. Afterwards, we highlight the relationships between models and meta-models within model configuration management and discuss the feasibility of different ways from a technical perspective.

4.1 Situations of meta-model modifications

The decision about how to handle meta-model modifications depends on various factors:

1. The philosophy concerning the modeling language used in the company – standardized vs. situation dependent
2. The status of the modeling project – running vs. completed projects
3. The relevance of the meta-model changes for a modeling project – affect of the modifications on the project

According to these factors it might be preferable that existing models follow the changes made on the corresponding meta-model. Thus, there are two possible ways how to handle meta-model changes: Keep the relationships between the old version of the meta-model and related models or migrate related models to the new version.

4.1.1 Modeling language

Using a given modeling language is sometimes difficult when creating models of a specific domain, the language was not explicitly made for [BSH98]. In this case modelers might want to add new concepts or rules that are not covered by the original modeling language. The situational method engineering approach [HBO94] [Tol98] solves that problem. Following that approach, meta-models used within a modeling project are adapted permanently according to the needs of the modelers. To use the enhancements of the new meta-model version within the project, an adaptation of appropriate models is strongly desired.

There can be, however, organizational or contractual reasons for using a standardized method within the modeling projects of a company. In these cases models of a project always refer to a defined meta-model and models are not adapted to changes made on (successor versions of) the meta-model.

4.1.2 Project status

As it might be useful to adapt models within a running project (see previous section) it is not useful to adapt models that document situations within already completed projects.

Firstly, the adaptation causes costs and secondly, it might be confusing for documenting reasons when existing project results (the models) are changed. Thus, we recommend to leave those models untouched.

If models of a completed project, however, will be reused within another project adaptation can be an option within the new project.

4.1.3 Relevance of changes

There might be changes within the meta-model that do not effect a project. This is, when the concepts added or modified are neither currently used within the dependent models nor will be used in the future. To use the example given by [SO05]: If a method engineer adds a timing constraint to a UML sequence diagram this might be useful for real time projects but it might not for other ones.³ In this case we recommend to keep the old meta-model version for the current modeling situation.

4.2 Relationships between model and meta-model

The meta-model contains meta-model elements – and relationships – representing modeling concepts that will be used within the model. Each element of the model can, therefore, be assigned to exactly one meta-model element [Gre03]. Thus, we can say that every model element depends on exactly one meta-model element.

Considering configuration management, model and meta-model will be controlled by a configuration management system. As mentioned in section 2.2, changes on a configuration item do not overwrite older versions, but lead to a new state of the item [Dar91]. Thus, when changing a meta-model, the old meta-model version still exists and corresponding models are still consistent with that version.

As pointed out in the previous section, it might be useful that existing models follow the changes made on the corresponding meta-model. From the technical viewpoint of the configuration management this means creating new versions for all affected models and creating new versions for all model elements whose meta-model elements has been changed. The created model element versions will refer to the new version of the meta-model element and the new model version to the meta-model version respectively [Gre03] [Sae06].

4.3 Conclusion

As we have shown, different use cases exist for both ways, staying with the old meta-model and migrating to the new version. Thus, there is no preferred way how to handle

³As the authors in [SO05] follow the situational method engineering approach, the meta-model is relevant for one projects anyway. It might be possible, however, that standardized methods are enhanced in a similar way.

meta-model changes. Instead, the modeler shall decide whether or not she or he wants to use the new meta-model.

If the modeling process is supported by modeling tools with configuration management support, the tools must not migrate automatically to the new meta-model version. Nevertheless, those tools should support the adaptation process, if the user demands it. While we have discussed the process from the configuration management perspective already, there are more things to consider. Newly created models versions that refer to the new meta-model version must be evaluated and if needed be adapted to ensure their consistency with the meta-model [SR98] [LSS94] [SO05] [Sae06]. Thus, we will discuss the kind of changes made on meta-models and the consequences for the consistency of dependent models arising out of these changes in the following chapter.

5 Model adaptation

Two different techniques must be carefully distinguished according to the amount of modifications to the meta-model. Firstly, if there are major modifications (like from UML 1.5 to UML 2.0) and the involved meta-models differ strongly from each other, a model migration is necessary. Secondly, in the case of small meta-model changes as they are necessary within projects following the situational method engineering approach an evaluation and possibly an adaptation of the models is required.

The migration of models from one meta-model to another one has already been discussed in literature. The problem firstly arose in the database field within the migration from hierarchical to relational to object-oriented database schemas [AHCM94] [FV95]. Later on the findings have been adopted to the modeling discipline too [OMG03]. In general, the migration process consists of two main phases. Firstly, similarities of the two meta-models (the languages respectively) has to be analyzed and secondly, the model will be „copied“ concerning the similarities [PG05].

In our paper we will not discuss such a model migration. Instead we describe the adaptation process arising from small meta-model changes. Thus, we do not discuss questions of how to recognize meta-model similarities. As both meta-models are administrated by a configuration management system and both are part of the same version family (and their elements respectively), we assume that meta-model elements that are changed still have the same meaning. To migrate a model from an old meta-model version to a new one, we have to create a successor version of that model referring to the new meta-model version (see section 4.2). Because of consistency reasons, we have to evaluate the model according to the meta-model after the modification. Thereby, the following cases have to be considered [Zel97]:

- The new meta-model element version might have changed (general case).
- The meta-model element might have been deleted (e. g. construct removed).
- There might be new configuration items in the meta-model (e. g. representing newly added constructs).

Because there are several differences between the adaptation of the model content and its representation, we analyze content and presentation separately in the following sections.

5.1 Model content

5.1.1 Adding new meta-model elements

Adding new elements to the meta-model has no direct effect on the dependent models [Sae06]. E. g., if the method engineer added new constructs to the meta-model these constructs are available after the adaptation process. It depends, however, on the modeler to use the new constructs according to their domain-world meaning.

In some cases, however, new meta-model elements cause inconsistencies between the model and the meta-model. This is the case, if the meta-model prescribes that an existing element has to be connected with at least one instance of the new element (see figure 3). To solve this conflict, a new instance has to be created in the model for each instance of the related construct to ensure model consistency.

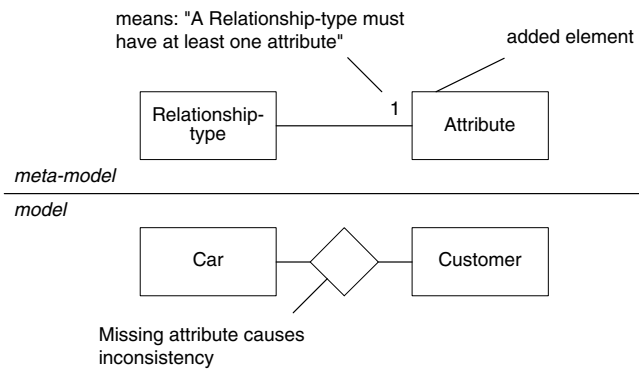


Figure 3: Inconsistency caused by adding meta-model elements

As modeling is a modeler dependent construction process, adding instances of the new meta-model element cannot be done automatically. Instead, highlighting of erroneous elements should be done within the models to help the modeler with the identification of inconsistent model elements.

5.1.2 Removing meta-model elements

Removing an element from the meta-model means that this element cannot be used in the adapted models anymore. Therefore, all instances of the element must be deleted from the dependent models to ensure model consistency. While this causes no problems in general, removing an meta-model element can also lead to inconsistencies when using

special meta-model notations.

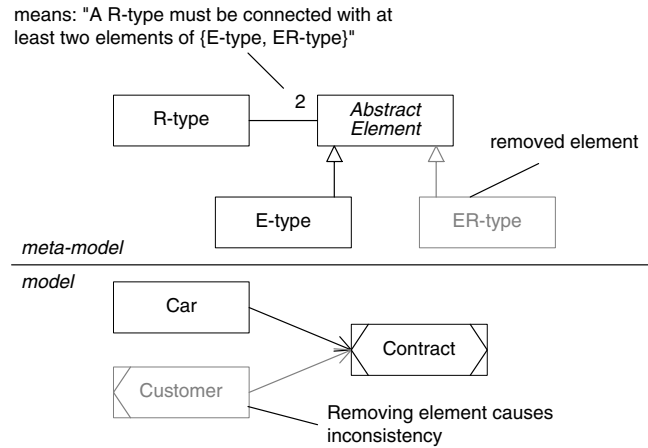


Figure 4: Inconsistency caused by removing meta-model elements

Meta-model elements like constraints or inheritances [Gre03] [OMG04] can force the creation of meta-model instances (see figure 4). As described in the previous section, only the modeler can solve this conflict.

5.1.3 Changing meta-model elements

A simple case is changing the name of a meta-model element. Because meta-models express the grammar of a modeling language and not the meaning of their constructs, the reason for changing meta-model elements remains unknown in the meta-model. As both meta-model element versions (new and old one) belong to the same version family, we imply that they still have the same meaning. Therefore, changing meta-model elements has no effect on dependent model elements, as they still refer to the same construct represented by the meta-model element.

A more complex kind of change in the meta-model is modifying the relationships between meta-model elements. Changing cardinality between constructs can lead to inconsistencies as described in section 5.1.1 and 5.1.2. When removing relationships, related elements in dependent models have to be disconnected.

5.2 Model representation

In the previous chapter we discussed modifications on the meta-model content. In section 2.1 we proposed, however, that models do not only consist of its content but also of at least one graphical representation. We discuss aspects regarding the graphical representation in the following sections.

5.2.1 Removing model elements

As discussed in section 5.1.2, removing a meta-model element results in removing all instances of that element. Thus, in the model representation, the representation of the elements has to be removed as well. As illustrated in figure 5, this can lead to 'broken' models.

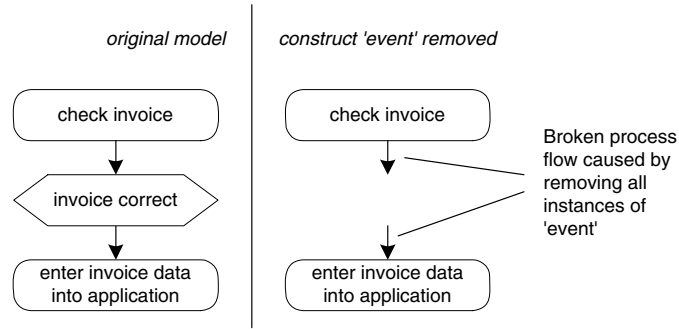


Figure 5: Visual problems caused by removing meta-model elements

Thus, graphical connections between models have to be removed and rearranged. As there can be different types of relations between model elements (in UML associations and inheritances, e. g.), a solution for the problem strongly depends on the meaning of the connection type. As meta-models do not contain any information about the meaning of its constructs, the process of repairing 'broken' models cannot be automated on the basis of these meta-models only.

5.2.2 Changing meta-model element representation

While meta-models describe constructs and their relationships, they also contain information about the graphical representation of these constructs [Gre03]. According to the purpose of modeling, it can be necessary to change this representation [BDK04]. Thus a meta-model element can also be modified according to its graphical representation. In this case, all instances of the meta-model element have to be adapted according to the new graphic.

While in general this is not a problem, a different size or shape of the new representation, however, can cause visual discrepancies that negatively effect the quality of resulting models [SR98]. If the size of the graphic increased, other model elements might overlap with the new element (see figure 6). To avoid such side effects of the adaptation process, we recommend to adapt the size of the new model element's representation according to the size of the old representation.

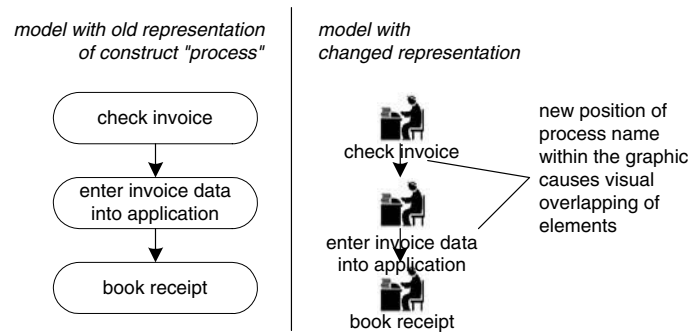


Figure 6: Visual problems caused by changes of meta-model representation

6 Discussion

As we have discussed in the previous chapters, configuration management of meta-models is similar to the configuration management of related models. In this case, both are administrated by a configuration management system and changes on the models as well as meta-model modifications lead to new versions.

Using new meta-model versions for the creation of new models is unproblematic in general. Several problems arise, however, when adapting the meta-model modifications to existing models. We have firstly shown, that this adaptation is not always necessary. In the positive case, however, it is important to know the consequences arising from the adaptation. Thus, we discussed several kind of meta-model modification. While configuration management can ease the comparison between the meta-model versions and most of the adaptation process can be automated, there is still need for manual review activities. As tools can support a adaptation process and, therefore, decrease the effort, we demand the development of appropriate modeling tools.

Our further work will concentrate on examining the problems mentioned in the paper within the method engineering e³-method [Gre03]. As there is already a configuration management system included in that method, we can evaluate that CM system as well. Additionally, existing modeling tools will be evaluated according to their possibilities to support the adaptation process and depending on the evaluation results, we will afterwards start with designing and implementing an appropriate modeling tool.

References

- [AHCM94] Rateb Abu-Hamdeh, James Cordy, and Patrick Martin. Schema Translation Using Structural Transformation. In *CASCON '94, IBM Centre for Advanced Studies*, pages 202–215, 1994.
- [BDK04] Jörg Becker, Patrick Delfmann, and Ralf Knackstedt. Konstruktion von Referenzmodellierungssprachen: Ein Ordnungsrahmen zur Spezifikation von Adaptionsmechanismen

für Informationsmodelle. *WIRTSCHAFTSINFORMATIK*, 46(4):251–264, 2004.

- [BEGW06] Robert Braun, Werner Esswein, Andreas Gehlert, and Jens Weller. Configuration Management for Reference Models. In Peter Loos and Peter Fettke, editors, *Reference Modeling for Business Systems Analysis*. IDEA Group, Hershey, 2006.
- [BSH98] Sjaak Brinkkemper, Motoshi Saeki, and Frank Harmsen. Assembly Techniques for Method Engineering. pages 381–400, 1998.
- [CW98] Reidar Conradi and Bernhard Westfechtel. Version models for software configuration management. *ACM Computing Surveys*, 30(2):232–282, 1998.
- [Dar91] Susan Dart. Concepts in configuration management systems. In *Proceedings of the 3rd international workshop on Software configuration management*, pages 1–18, New York, NY, USA, 1991. ACM Press.
- [DJM⁺00] Oscar Dieste, Natalia Juristo, Ana M. Moreno, Juan Pazos, and Almudena Sierra. Conceptual Modelling in Software Engineering and Knowledge Engineering: Concepts, Techniques and Trends. In *Handbook of Software Engineering and Knowledge Engineering*, number 1, pages 733–766. World Scientific Publishing Company, 2000.
- [EGK02] Werner Esswein, Steffen Greiffenberg, and Christian Kluge. Konfigurationsmanagement von Modellen. In E. J. Sinz and M. Plaha, editors, *Modellierung betrieblicher Informationssysteme - MobIS 2002*, pages 93–112, Nürnberg, 2002.
- [ELC⁺02] Jacky Estublier, David Leblang, Geoff Clemm, Reidar Conradi, Walter Tichy, Andre van der Hoek, and Darcy Wiborg-Weber. Impact of the research community on the field of software configuration management: summary of an impact project report. *SIGSOFT Software Engineering Notes*, 27(5):31–39, 2002.
- [Est00] Jacky Estublier. Software configuration management: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 279–289, New York, NY, USA, 2000. ACM Press.
- [EW05] Joerg Evermann and Yair Wand. Ontology based object-oriented domain modelling: fundamental concepts. *Requirements Engineering*, 10(2):146–160, 2005.
- [FL02] Peter Fettke and Peter Loos. Klassifikation von Informationsmodellen - Nutzenpotenziale, Methode und Anwendung am Beispiel von Referenzmodellen. Working Papers of the Research Group Information Systems & Management 9, Johannes Gutenberg-Universität Mainz, Lehrstuhl für Wirtschaftsinformatik und BWL, Mainz, 2002.
- [Flo01] Christiane Floyd. Das Mögliche ermöglichen: Zur Praxis der Realitätskonstruktion am Beispiel der Softwareentwicklung. In Albert Müller, Karl H. Müller, and Friedrich Stadler, editors, *Konstruktivismus und Kognitionswissenschaft: Kulturelle Wurzeln und Ergebnisse*, pages 115–134. Springer Verlag, Wien, 2001.
- [Fra99] Ulrich Frank. Conceptual Modelling as the Core of the Information Systems Discipline - Perspectives and Epistemological Challenges. In D.W. Haseman, S. Nazareth, and D. Goodhue, editors, *Proceedings of the Fifth America's Conference on Information Systems (AMCIS 99)*, pages 695–697, Milwaukee, 1999.
- [FV95] Christian Fahrner and Gottfried Vossen. Transformation relationaler Datenbank-Schemas in objekt-orientierte Schemas gemäß ODMG-93. In Georg Lausen, editor, *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), GI-Fachtagung*, pages 111–129. Springer, 1995.

- [Gre03] Steffen Greiffenberg. *Methodenentwicklung in Wirtschaft und Verwaltung*. Dr. Kovac, Hamburg, 2003.
- [Ham99] Christoph Hammel. *Generische Spezifikation betrieblicher Anwendungssysteme*. Shaker Verlag, Aachen, 1999. Dissertation Otto-Friedrich-Universität Bamberg.
- [HBO94] Frank Harmsen, Sjaak Brinkkemper, and J. L. Han Oei. Situational method engineering for information system project approaches. In A. A. Verrijn-Stuart and T. William Olle, editors, *Methods and associated tools for the information systems life cycle, Proceedings of the IFIP Working Conference*, pages 169–194. IFIP, Elsevier Science B.V. (North-Holland), 1994.
- [Int00] International Organization for Standardization (ISO). *Quality management systems: Requirements (ISO 2000:9001)*. Beuth Verlag GmbH, Berlin, 2000.
- [Int03] International Organization for Standardization (ISO). *Quality management systems: Guidelines for configuration management (ISO 10007:2003)*. Beuth Verlag GmbH, Berlin, 2003.
- [Kne03] Ralf Kneuper. *CMMI: Verbesserung von Softwareprozessen mit Capability Maturity Model Integration*. dpunkt, 2003.
- [KRT05] Steven Kelly, Matti Rossi, and Juha-Pekka Tolvanen. What is Needed in a MetaCASE Environment. *Journal of Enterprise Modelling and Information Systems Architectures*, 1(1):25–35, 2005.
- [LSS94] Odd Ivar Lindland, Guttorm Sindre, and Arne Slyberg. Understanding Quality in Conceptual Modeling. *IEEE Software*, 11(2):42–49, 1994.
- [OMG03] OMG Object Management Group. MDA Guide Version 1.0.1. <http://www.omg.org/cgi-bin/doc?omg/03-06-01>, 2003.
- [OMG04] OMG Object Management Group. UML 2.0 Superstructure Specification. <http://www.omg.org/cgi-bin/doc?formal/05-07-04>, 2004.
- [PCCW93] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber. Capability Maturity Model, Version 1.1. *IEEE Software*, 10(4):18–27, 1993.
- [PG05] Daniel Pfeiffer and Andreas Gehlert. A Framework for Comparing Conceptual Models. In Jörg Desel and Ulrich Frank, editors, *Enterprise Modelling and Information Systems Architectures: Proceedings of the Workshop in Klagenfurt*, number P-75 in Lecture Notes in Informatics, pages 108–122, Bonn, 2005. Köllen Druck + Verlag GmbH.
- [RSD05] Michael Rosemann, Ansgar Schwegmann, and Patrick Delfmann. Vorbereitung der Prozessmodellierung. In Jörg Becker, Martin Kugeler, and Michael Rosemann, editors, *Prozessmanagement: Ein Leitfaden zur prozessorientierten Organisationsgestaltung*, chapter 3, pages 45–103. Springer, 2005.
- [Sae03] Motoshi Saeki. CAME : The First Step to Automated Method Engineering. In *OOPSLA 2003: Workshop on Process Engineering for Object-Oriented and Component-Based Development*, pages 7–18, Sydney, Australia, 2003. Centre for Object Technology Applications and Research.
- [Sae06] Motoshi Saeki. Configuration Management in a Method Engineering Context. *Lecture Notes in Computer Science*, (4001):384–398, 2006.
- [Sch98] August-Wilhelm Scheer. *ARIS – Vom Geschäftsprozeß zum Anwendungssystem*, volume 3. Springer-Verlag, Berlin Heidelberg New York, 1998.

- [Sem05] Semture GmbH. *Cubetto Toolset*, <http://www.semture.de/cubetto>, Download: 15.07.2005.
- [SO05] Motoshi Saeki and Takafumi Oda. A Conceptual Model of Version Control in Method Engineering Environment. In O. Belo, J. Eder, J. Falcao e Cunha, and O. Pastor, editors, *Proceedings of the CAiSE05 Forum*, pages 89–94. Faculdade de Engenharia da Universidade do Porto, 2005.
- [SR98] Reinhard Schuette and Thomas Rotthowe. The Guidelines of Modeling: An Approach to Enhance the Quality in Information Models. *Lecture Notes in Computer Science*, 1507:240–254, 1998.
- [Str96] Susanne Strahringer. *Metamodellierung als Instrument des Methodenvergleichs: Eine Evaluierung am Beispiel objektorientierter Analysemethoden*. Shaker, Aachen, 1996.
- [Tho97] S. M. Thompson. Configuration management - keeping it all together. *BT Technology Journal*, 15(3):48–60, 1997.
- [Tho06] Oliver Thomas. Version Management for Reference Models: Design and Implementation. In *Multikonferenz Wirtschaftsinformatik 2006 (MKWI '06)*, 2006.
- [Tol98] Juha-Pekka Tolvanen. *Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence*. PhD thesis, University of Jyväskylä, 1998.
- [WW02] Yair Wand and Ron Weber. Research Commentary: Information Systems and Conceptual Modeling—A Research Agenda. *Information Systems Research*, 13(4):363–377, 2002.
- [Zel97] Andreas Zeller. *Configuration Management with Version Sets: A Unified Software Versioning Model and its Applications*. PhD thesis, Braunschweig Technical University, 1997.