Design and Implementation of a Version Management System for Reference Modeling

Oliver Thomas
Institute for Information Systems (IWi)
at the German Research Center for Artificial Intelligence (DFKI),
Saarland University, Saarbruecken (Germany)
oliver.thomas@iwi.dfki.de

***Abstract*—The central idea in reference modeling is the reutilization of the business knowledge contained in a reference model. The user's task in reference model-based construction is the adaptation of the reference model. The derivation of specific models from reference models corresponds with the creation of reference model variants. Research on the design of such variant constructions generally assumes an unchangeable stock of reference models. The potential inherent in the management of these variant constructions, which reflect the changes in designed reference models through time and in doing so, their evolutionary development, has not yet been tapped into. The article at hand analyzes this problem and presents a concept for the version management of reference models as a solution. The task to be mastered with the proposed approach will be concretized using data structures and a system architecture and then prototypically implemented.**

***Index Terms*—Information model, information modeling, reference model, reference modeling, version, version management, reference modeling tool**

I. Introduction

The systems studied in the field of information systems research are extremely complex. It is difficult to describe their structure or predict their behavior in detail. By constructing models, the attempt is made to create abstracting artifacts which make the complexity of information systems manageable. The information models created thereby have a tradition of more than thirty years [3; 12; 22]. From today's perspective, these models have established themselves in information systems research as a medium for describing information systems [23; 26; 49; 51]. The application areas of information modeling range from software design to the configuration of ERP systems and business process reengineering. In information systems research it is widely accepted that conceptual models have a lasting influence on the quality of the software developed [e.g. 52]. Moreover, from a practical point of view, a current study from *Gartner* points out that the modeling of business processes with their processing times and respective responsibilities can lead to an increase in productivity of more than 12% [24].

The construction of information models is often connected with the demand to abstract from enterprise-specific attributes in order to make the models reusable. These so-called reference models, provide companies with an initial solution for organization and application system design [19; 34; 38; 48]. Examples in the scientific field are the reference model for industrial enterprises from Scheer [34], as well as the SAP R/3 reference model [14] resulting from commercial practice. On the one hand, the possibility of orienting oneself on the technical content of such reference models promises the model-users savings in time and costs, while on the other the quality of the model to be constructed can be increased by the use of a reference model.

One of the main problems is that the knowledge contained in these models is not unchangeable, so that reference models themselves are subject to change throughout time. These changes generally occur in two manners for models constructed within the two processes in reference modeling, development and usage. *First*, if within an evaluation, one notices during the development of a reference model that the model being constructed does not fulfill the defined requirements, then one must return to the preliminary phases. This generally results in revisional constructions, which replace the construction results evaluated. And *second*, revisional reference model constructions are also generated when reference models are used to derive specific models. These revisional constructions often differ only slightly from one another, depending on their use. Both of these aspects lead to a differentiation between version and variant constructions.

Reference modeling literature focuses on variant management [5; 6; 15; 29; 30; 32; 39; 44; 47] and although the terminological difference between variants and versions is alluded to in literature [16, p. 96; 45, p. 260; 50, p. 63], the design of a version management for reference models has only occurred in rudimentary form up to now. The task of the article at hand is to meet these concerns with the design and implementation of a version management tool for reference models.

This paper is based on 'Joint Reference Modeling: Collaboration Support through Version Management,' which appeared in the 'Proceedings of the 40th Annual Hawaii International Conference on System Sciences', © 2007 IEEE Computer Society Press, and 'Version Management for Reference Models' which appeared in 'Reference modeling', ISBN 978–3–7908–1965–6, © 2007 Physica.

II. Methodical Considerations and the Course of the Analysis

The goal of this analysis is the design and realization of an information system for the support of a version management for reference models. Established procedure models already exist for the system development required here. The task of these models is to secure the continuous description of the development process, from the business requirements to the technical implementation.

This analysis will use the phase model of the architecture of integrated information systems (ARIS) [35]. The ARIS phase model differentiates between the description levels 'requirements definition', 'design specification' and 'implementation'. In the requirements definition phase, the business concept to be supported is described in semi-formal languages. The design specification phase adapts the requirements description to basic constructs in information technology. Finally, in the implementation phase, the design specification is transferred to concrete information technical components.

The requirements definition is especially important for achieving our goal because *first*, it can be seen as a long-term bearer of business concepts (sub-goal 'design') and *second*, it acts as a starting point for further steps towards the technical implementation (sub-goal 'realization'). Because this significance is also generally reflected in information systems research, conceptual modeling is emphasized in modern literature as an independent research *method* [7; 47], in addition to its undisputed significance as an important research *topic* in information systems [20; 49; 51]. It has been selected as a method for the scientific sub-goal 'design'.

The results achieved through conceptual modeling should serve as a starting point for a consistent implementation in information technology. Due to the scientific understanding of information systems here, they result in a prototypical implementation. Because the design of prototypes is widely accepted in literature as an independent research method in information systems [9; 11; 27; 28; 33], it has been selected as a method for achieving the second scientific sub-goal 'realization'.

This results in the following outline for this article. Section 3 lays a foundation for the terms used here by first, explaining the terms 'information' resp. 'reference model' and then, differentiating between the terms 'variant' and 'version' in the context of reference modeling. Due to the methodical procedure selected for the design of the reference model versioning, a data model will be constructed in Section 4, which represents the version management of reference models on a conceptual level. In Section 5, this description will be adapted to general IT constructs in the form of a system architecture in order to then, in the implementation phase in Section 6, be transferred to IT components. In Section 7, the work discussed here will be distinguished from related work. The article ends with a critical discussion of its results and an outlook in Section 8.

III. Background

A. Information and Reference Models

This analysis supports a construction-oriented understanding of models. *Information models* are defined as purpose-relevant representations of an information system designed by way of a construction process. They are simply referred to as models. A *reference model*—to be precise: reference information model—is an information model used for the construction of other models. This analysis is therefore based upon a use-oriented reference model term which focuses on the use of reference models for the construction of enterprise-specific models [38; 47]. The reference model terms often found in information systems literature, based upon the attributes which characterize these reference models—in particular the attributes 'universality' and 'recommendation character' [45, pp. 31ff]—will not be used here. Every model resp. partial model that can be used to support the construction of another model can be seen, in this sense, as a reference model.

B. Reference Model Variants

The user's primary task in reference model-based construction, which can be supported by IT tools, is the adaptation of reference models. The derivation of a specific model from a reference model characterized by this term corresponds with the creation of variants of reference models. Thus, for example, the enterprise-specific models *information model product-oriented manufacturing enterprise* E_1 or *information model process-oriented manufacturing enterprise* E_2 could be derived as variants of the reference model *manufacturing*.

In analogy to industrial variant management [34], a variant *IM'* of an information model *IM* is understood as a model which differs from IM in only 'a few' features. In other words: *IM'* has the same feature presentation as *IM* with regard to at least one feature and a different presentation in regard to at least one other feature. Many features are conceivable for information models. In literature, these features, as well as their respective feature presentation, are used for the classification of information models. Examples of this are the distinctions between organization and application system models, as well as between structure (static view) and behavior models (dynamic view) [19].

The management of the variants derived from reference models is especially interesting on two counts [16, p. 94]. *First*, the storage of the variants together with the adaptation premises also being managed can speed up the development of future enterprise-specific models for comparable applications. And *second*, it allows a similarity analysis of the variants, the results of which can then be used for the development of new reference models.

Reference modeling languages should thus, be designed to support model variant management. Many of the approaches discussed in literature aim at maintaining alternative building blocks in the reference models, which the model user then keeps, supplements or removes, with

respect to enterprise-specific facts, during the adaptation. However, differing opinions exist on the question of which construction techniques should be used for reference model variant management. While for example, some authors link variant management to the construction technique of configuration and also refer to a variant as a configured service [e.g. 15; 32], VOM BROCKE generally arguments against the coupling of variant management with individual construction techniques and recommends aggregation, specialization, instantiation and analogy construction as further construction techniques [45, pp. 235ff; 46].

C. Reference Model Versions

In literature, the creative potential contained in a revision management and control tool connected with the creation of versions is primarily discussed from a technological perspective within the framework of configuration management [8; 17]. Configuration management has its origins in hardware development. This field of work generally deals with the consistent description of system components, as well as monitoring and controlling the changes made in these components. Since the beginning of the 1980ies the attempt has been made to transfer this concept to software processes under the term 'software configuration management' [8]. The term *configuration management*, according to the terminology of software engineering, can be understood as the development and usage of standards and methods for managing a system continuously under development [37, p. 651]. The methods for configuration management have been established: for example, how system changes are documented and processed, as well as which relation they bear to the system components and the methods used for their description [37, p. 651].

The development of methods and procedures for configuration management is seen as one of the central challenges in the field of software engineering [17, p. 279]. Most of the products for configuration management are based upon a core of concepts and mechanisms. One of these, is the concept of versioning, mentioned above, which goes back to the middle of the 1970ies [31].

Generally, the term *version* refers to the state of an object at a certain point in time [16, p. 96; 53, p. 9]. One can differentiate between three dimensions of versioning subject to the construction purpose: historical, logical and cooperative versioning [13, p. 240; 18, p. 122; 53, pp. 9f]:

1. *Historical versioning:* A version made to replace another version is called a *revision*. With the construction of a revision, the further development of the original version is abandoned in favor of this new version. In practice, the revision of a component is made by changing a copy of the last version. Old revisions are stored for maintenance and documentation purposes and form the version history of an object. Revisions have a predecessor/successor relationship to one another.
2. *Logical versioning:* In contrast to revisions, the adjustments made to an object to fit the specific circumstances of an application context (for example: the company adjustment of a standard software module or a software component with reduced functionalities for testing purposes) are referred to as *logical versions*. The term variant was already introduced for such logical versions. Both terms will be used in the following as synonyms. There is no predecessor/successor relationship between logical versions; they exist parallel to one another.
3. *Cooperative versioning:* Variants that are integrated with other variants resp. combined are referred to as *temporary variants.* Temporary variants are used for example, to change an old revision when the development of a new revision has not yet been concluded. Temporary variants are used primarily for supporting cooperative construction processes.

To illustrate this, the terms above have been arranged in a framework for reference modeling procedures (cf. Figure 1). Because reference models are special information models used to support the construction of other models, the framework considers both the fact that reference models are to be made using a modeling language (reference model development), as well as that reference models are made to be used (reference model usage). In comparison with existing procedure models for reference modeling, the framework emphasizes the aspects relating to the construction of versions and variants.

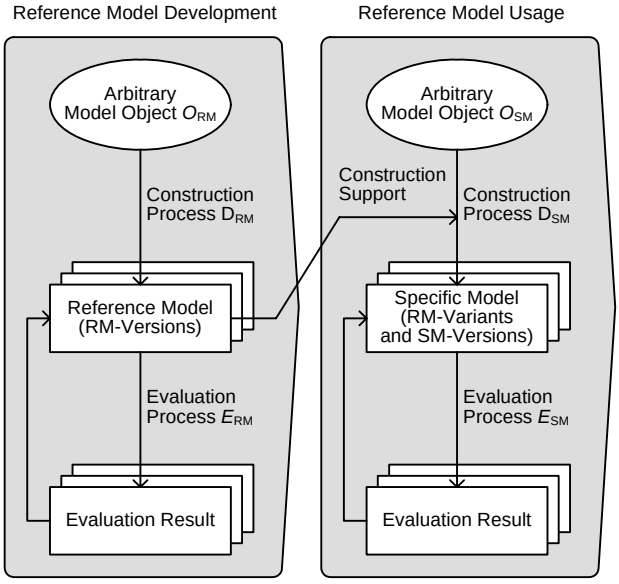

Figure 1. The arrangement of model versions and variants in the procedure for the development and usage of reference models

In analogy to software engineering projects, it is recommended that evaluations of a model be carried out before it is 'handed over' to the model management via a release mechanism. This applies to the reference models constructed (development phase), as well as to the variants derived from these (usage phase). To some extent, evaluation phases are taken into consideration in existing procedure models for the development and usage

of reference models, for an overview cp. [2]. The conception of the term ‘evaluation’ and the tasks to be carried out by project members during the evaluation phase differ however, in the works mentioned. In this article, the goal of a reference model evaluation—in the sense of the evaluation of construction results—consists in evaluating the reference model resp. determining the value of a reference model. The value of a reference model is understood here as the model’s share in achieving the reference modeling goals pursued. Evaluation processes are run through several times, because evaluation results can lead to a decision to return to preliminary construction phases, in order to improve a model.

The necessity of versioning construction results connected to this step-by-step improvement of the model, also known as *model evolution*, will be concretized in the following section using conceptual models.

IV. Requirements Definition for the Management of Reference Model Versions

A. Basis Model

The starting point for the conceptual design of the reference model versioning is the UML class model represented in Figure 2 (for the current specification of the Unified Modeling Language cf. http://www.uml.org/). We have left out operations and class attributes for reasons of simplicity.

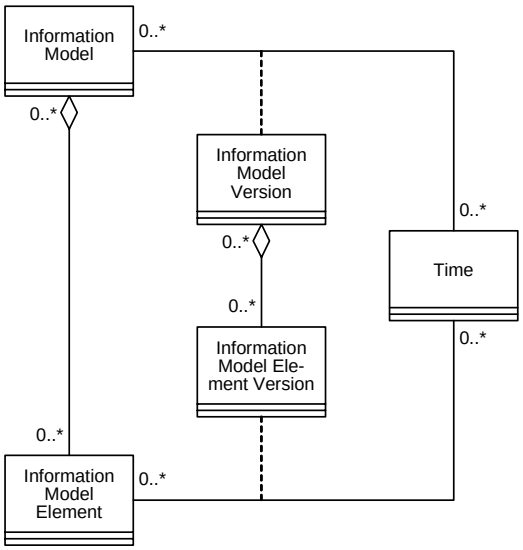

Figure 2. UML class model for managing model versions [35, p. 87]

This class model for the management of model versions takes into account that as a rule, revisional constructions in modeling projects are not exclusively carried out on the model as a whole. They are, in fact, carried out by project members on detail models—due increasingly to the trend towards the division of work in model development [41; 45]—and then combined to form an improved construction. This correlation is taken into consideration with the classes Information Model and Information Model Element, as well as the existing part-of-relationship between the objects of the classes involved. Each object in the sub-class Information Model Element can—at any time—be a component of several objects in the aggregate class Information Model, e.g. a function as part of a function hierarchy diagram, as well as part of an EPC model [36], and every information model can be composed of several information model elements. Therefore, there is no composition between both classes, but rather a (0.∗):(0.∗)-aggregation.

The information models, as well as the information model elements which compose a model, receive a ‘time stamp’ due to the (0.∗):(0.∗)-association with the class Time. This relationship between the models, resp. model elements and time, defines the model versions resp. the model element versions and is therefore formulated as the association class Information Model Version resp. Information Model Element Version. Within the framework of their approach to the configuration management of models Esswein, Greiffenberg, Kluge also allude to the fact that ‘a single record of the different developmental states of a model is insufficient’ [16, p. 96]. In fact, ‘in order to retroactively trace the changes in a model over time, […] a record of the development of individual model parts’ [16, p. 96] should be made. The said classes are also connected to each other via an aggregation, which signifies that an information model version is composed of the versions of the information model elements assigned to it. This was—analog to the association between the objects of the classes Information Model and Information Model Element—not constructed as a composition (no limitation of the cardinality which is annotated to the aggregate class to 1.1), so that the case that each information model element version can be a part of several information model versions is also taken into account. On the other hand, the (0.∗)-cardinality on the class Information Model Element Version indicates that a model version can be assigned to several versions of a model element. Minor revisional constructions on model elements do not necessarily lead to the designation of new information model versions. This is especially important for the management of versions of comprehensive (reference) models.

B. Extension of the Management of Model Versions Through Version Graphs

The graphic illustration of the development history of an object is done in software engineering using so-called *version graphs* [13, pp. 240ff; 53, pp. 10ff]. This article sees version graphs as a type of modeling language. The concept of version graphs is transferred to the representation of the development history of information models in the following section. A version graph model is represented in Figure 3 and will be used as a basis for the explanation of the basic language constructs of a version graph, as well as their representational forms.

The basic elements of the modeling language ‘version graph’ are nodes and edges. Nodes represent the versions of an information model. They are depicted in Figure 3 by shaded rectangles with rounded corners. The versions are marked with unique identifiers—also called *version*

numbers. In contrast to this type of versioning, referred to in literature as *extensional*, the identification of versions in the *intentional* manner occurs on the basis of attributes, which describe attributes of the versioned objects [4, pp. 102 ff]. The predecessor/successor relationships existing between versions express the fact that the successors were derived from their predecessors by way of revisional constructions. Arrows represent the corresponding edges.

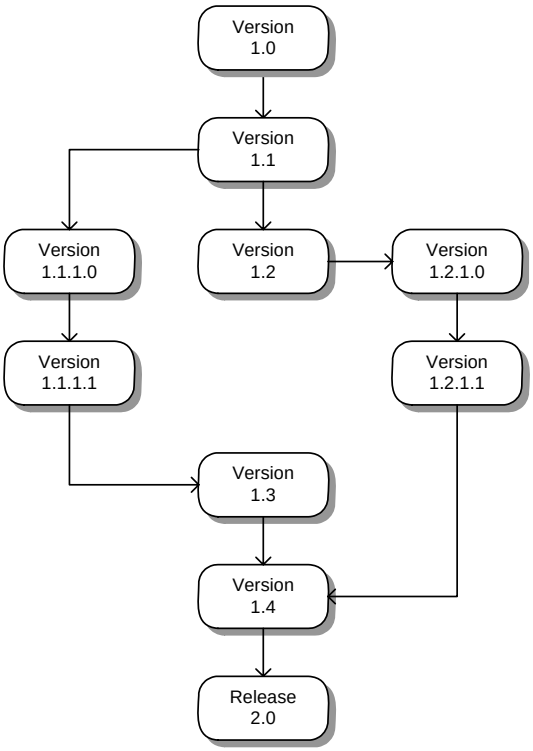

Figure 3. Example for a version graph model

Version graph models can take on different forms [13, pp. 240 f]. In the simplest case, they consist of a sequence of revisions where the versions can have a maximum of one predecessor and one successor. Figure 3 represents the fact that a version can have several predecessors, as well as various successors. According to software engineering terminology, a model version released and turned over to the model management—and perhaps even delivered to a customer—is referred to as a *release*.

Possible corresponding entity relationship models are shown in Figure 4. The models are based on facts that focus on the relationships between the services provided by an enterprise and its partners on the market. In a first step (Version 1.0), external orders for services that can be turned over to market partners or procured from them are defined. The entity relationship model is then detailed in a second step (Version 1.1), so that *first*, the network-like integration of services (for example, in the form of a bill of materials) is taken into consideration and *second*, the connections between services and market partners in the form of master data are recorded through conditions. The data structure is detailed further by using the construction operator ‘specialization’ in a final step (Release 2.0).

To transfer the concept of the version graph to information modeling, the UML class model for managing model versions must be extended. A version graph is described as a predecessor/successor relation of a set of versions. In Figure 3 two association classes were constructed for the representation of versions: Information Model Version and Information Model Element Version. Due to the part-of-relationship between these classes there are two possibilities for the integration of a version graph structure:

1. The definition of two structures, which are differentiated between according to whether they describe a relationship between models or model elements.
2. The definition of a structure, which can be related to both of the said classes.

The first case leads to a redundant description of a structure, which would exist on a model level, as well as a model element level. In the second case, this redundancy was abolished; however, no suitable language constructs are available in the UML class diagram—if the construction of further classes is not considered—capable of generating the corresponding assertion. Because of this, the part-of-relationship between the classes Information Model and Information Model Element, as well as the corresponding relationship between the association classes defined by the time stamp are ‘opened’ and replaced by the data structure in Figure 5.

The differentiation between complex and atomic models is the central thought in this revisional construction. While a complex information model can be broken down into model elements, this assumption does not apply to an atomic model. Examples for atomic models are, in the case of an EPC, a function or in the case of an ERM, an entity type. In this sense, an information model can be either complex or atomic, whereby a complex information model can, in turn, consist of several models. This is considered in Figure 5 by the specialized relationships plotted between the classes Information Model, Complex Information Model and Atomic Information Model. It is however, also given through the part-of-relationship, which guarantees the assignment of an information model to a complex model, whose component it represents, by way of a hierarchy between the objects of the classes involved.

The model versioning has now been dealt with on the model level, as well as on the model element level by way of the association class Information Model Version, since this relationship was inherited by the sub-classes Complex Information Model and Atomic Information Model of the super-class Information Model. The diagram-like management of versions using version graph models—which requires not only recording the respective construction results, but also recording the relations existing between these (cf. also Figure 3)—can now be annotated by a predecessor/successor relationship in the form of the recursive (0.∗):(0.∗)-association class Version Structure using the corresponding role on the class Information Model Version (cf. Figure 5).

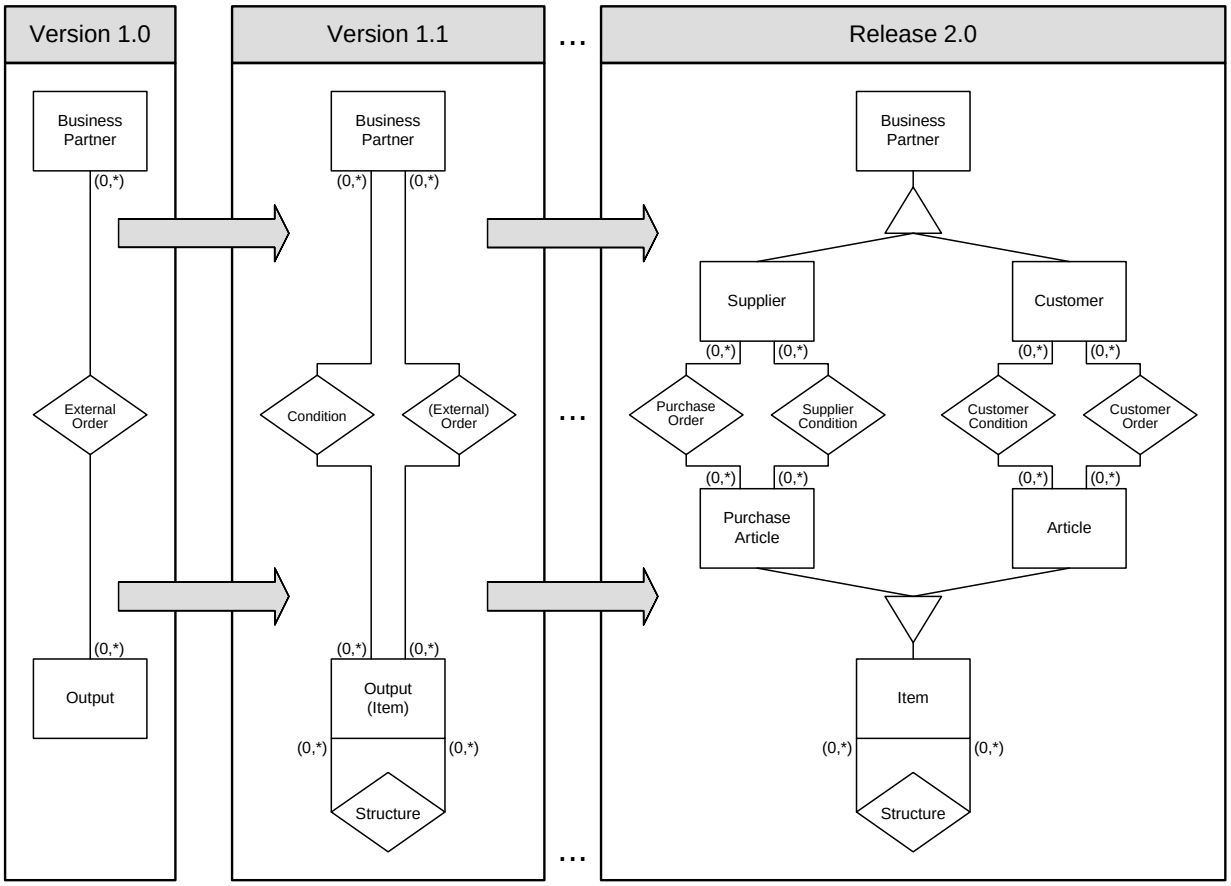

Figure 4. Versions of an entity relationship model

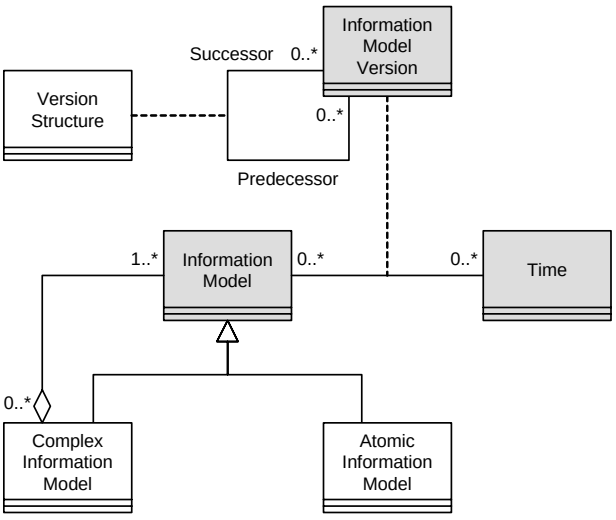

Figure 5. Extension of the management of model versions through version graphs

The essential requirements on the conception of system functionality for the management of information model versions as a sub-aspect of reference model development and usage are herewith defined. In addition to the construction results represented by models, information must be recorded about revisional constructions carried out on a model within the framework of a modeling project such as content, reason, time, those responsible, etc. The chronicle of these revisional constructions and the reasons for making them allows the retrospective analysis of the same and provides information for decision making processes in future developments [16, p. 93].

This applies not only to reference model development and usage, but also to the implementation phase of a constructed to-be model in an enterprise or the implementation of an application system, which both potentially follow these processes. If for example, within the framework of a reference process model adaptation, the required processes are selected resp. unnecessary processes removed using typological features of an enterprise and an automatic control system, then the connection between the requirements used and the process structures selected is lost in the process. It is however, exactly this information that is needed for further revisional constructions, as well as for the remodeling of a process already being carried out in an enterprise. Information about the reasons for the structure of a certain process, as well as for its modification—the 'Why'—should therefore be saved with the model, in addition to the processes themselves—the 'What'—and the rules for the selection of the required processes—the 'How'. The recording of this information is guaranteed by the data structure represented in Figure 5.

V. Design Specification for the Version Management of Reference Models

A. System Architecture

The primary technical aspect of the tool for versioning reference models refers to the definition of the technological platform, the identification of the IT components, as well as the description of their DP logical relationships. The architecture of the system, which will be referred to in the following as the *reference model management system* (RMMS), is illustrated in Figure 6.

The system architecture of the RMMS is a client/server architecture. Due to the multitude of RMMS system elements these are 'classically' structured in three layers—the data management, application and presentation layers.

The data management layer of the RMMS system architecture is divided up into database and file management. While the structured data (human resource and customer data, as well as as-is and reference models) is managed in relational databases, the weakly structured data (text documents, spread sheets, presentation graphics, images, video and audio files, as well as links to further documents) is stored in a file system.

The data management layer differentiates between four databases—an enterprise-wide human resource database, an enterprise-wide customer database, an as-is model database and a reference model database. The reference model database in particular, is a systematized collection of reference models (reference model library). It stores the reference model constructs, as well as their structural relationships, model attributes such as name, identification number, type of model (for example: EPC or ERM), description, time of creation, originator, last modification or last processor. The customer model database is also a model database, as is the case with the reference model database. It contains documented as-is models, i.e. sections of the customer's enterprise structure interpreted by the creator of the model at the time of modeling. It makes no difference whether the customer is internal or external.

The external databases in Figure 6 are represented as a logical unit for purposes of simplicity, which however, as a rule, consist physically of several distributed databases. For example, the reference model database could consist of several external databases. This is the case when, in modeling projects, reference models from different modeling tools are used and each manage the models in their own databases.

The application layer comprises the server services and data (RMMS repository), which are used to carry out the technical tasks. The programs in this layer receive the user's (clients) instructions and carry them out on the relevant data. By using a client/server architecture, several applications and users can access the same database at the same time and process it.

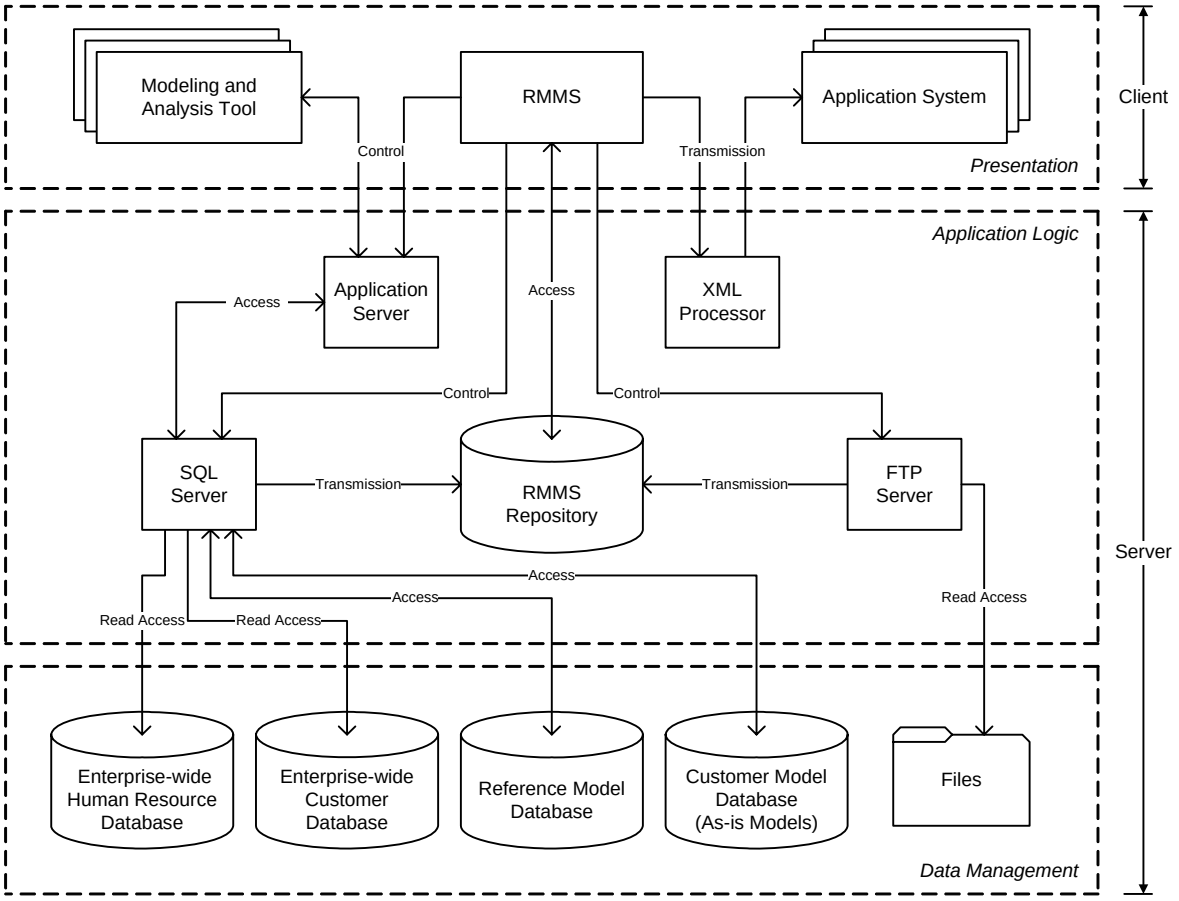

Figure 6. RMMS system architecture

The components of the RMMS, with which the user has contact, are assigned to the presentation layer. They make the input and output user-friendlier and are represented by a graphic interface. The operational concept of the RMMS and its graphic user interface should be adapted to the interface design of established modeling and analysis tools. This way, for the user, the separate systems appear to be a logical entity—from the technological point of view. This also makes access to the RMMS easier for users familiar with modeling tools.

While the RMMS components are used for processing information important for the development and usage of reference models, the creation, editing and deletion of information models remains the task of the modeling and analysis tool. Several different modeling and analysis tools may be used here. In order not to focus on the integration capability of modeling tools [25], the use of only *one* modeling and analysis tool will be assumed, because it is not the interchange between several modeling tools, but rather the general exposure to reference models, their versions and associated information objects, which are the subject here.

The version management for reference models is created using the structure and transformation processes 'inside' the RMMS repository, which was dealt with up to now as a black box.

B. RMMS Repository as a Central Component for Model Versioning

The RMMS repository consists of four database components: a user, a customer, an RMMS model and a project database. These databases are managed by the server plotted in Figure 6 and show relations to the external databases described above, as well as to the external file system. The data access and transfer of the server has already been discussed. For purposes of clarity, only the relations between the components of the repository and the components of the data management layer have been accentuated graphically in Figure 7.

The system users are created in the user database and are authenticated with it. The user database is a database derived from the enterprise-wide human resource database. Beyond the 'business card' managed in the enterprise-wide human resource database, the user database of the RMMS contains the personal profile of the user (for example: start and standard settings of the RMMS user interface, technical interests), as well as authorizations given to the user regarding the manipulation of data. Basic rights of disposal are the reading, creation, modification and deletion of objects.

The customer database is based on the external enterprise-wide customer database. The fact that customers appear as users of the RMMS is indicated by the relation between the customer database and the user database.

In addition to the 'pure' model data, the RMMS model database also manages information about the construction of versions during the project. On the one hand, it adopts models from the external reference model database and on the other hand, the consideration of already existing information systems for reference model usage requires accessing the customer's external as-is model database (reverse engineering).

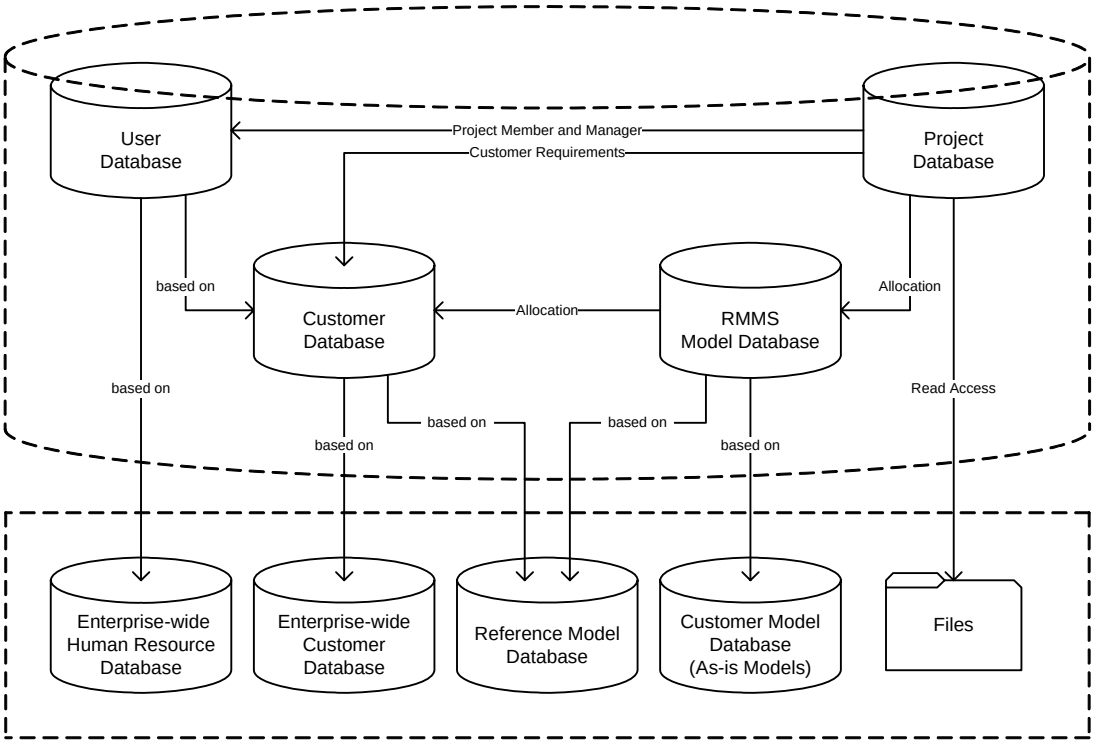

Figure 7. RMMS repository and databases

The transfer of the model data into a logical structure allows the simple connection to further external model databases in the system architecture. It also allows the RMMS user to carry out the same functions on all models. This comprises not only the version management of the models, but also searching resp. navigating in the model databases.

Because the information models are not created and processed with the RMMS, but rather with the modeling tool, it appears expedient to allow the RMMS read access to only the external databases.

The project database is at the center of the RMMS repository in Figure 7. It manages internal and external reference modeling project tasks between organizational units by storing data, such as the project name, type, goal, period, status or progress. Project documents, such as the project commission, structure plan, and schedule, proceedings from meetings, status reports or requirement specifications are not stored directly in the RMMS repository. These documents are created by the users locally and managed in an external file directory.

The project database also supports the project management by managing the number, type and logical sequence of measures with which a reference modeling effort should be realized, as well as by storing model histories (version management). With the help of relations to the user database, each reference modeling project is assigned a project leader and a group of project employees. Associations to the customer database take service-specific customer requirements into account. The project-related new or revisional construction of reference models and the documentation of changes in the knowledge basis require access to the reference model database.

VI. Implementation of the Reference Model Version Management Tool

A. Selecting a Basis Modeling Tool

Up to now, the concept introduced for the version management of reference models was developed independent of modeling languages, methods and tools. This applies to the construction of the conceptual models in Section 4, as well as to the design of the system architecture and the RMMS repository in Section 5.

Because established products exist in the field of modeling [10], a complete new development of the RMMS is not necessary, but rather an extension of the existing systems. The functionalities necessary for the development and usage of reference models, which for example, make the revisional construction of models possible, have already been implemented in the respective tools. Functionalities which, on the other hand, serve the documentation of the construction process or a certain procedure in reference model usage must be re-implemented as necessary. The RMMS is therefore implemented as an integrated component of a professional tool for modeling business processes. The ARIS-Toolset from IDS Scheer, Inc. has been selected as a basis modeling tool. The ARIS-Toolset is a software system for the analysis, creation and navigation of business processes [1]. It is based on the research of the Institute for Information Systems (IWi) in Saarbruecken, Germany. The following are factors decisive for the selection of the ARIS-Toolset as the basis modeling tool:

1. IDS Scheer, Inc. and the Institute for Information Systems (IWi) are both located in Saarbruecken. This naturally facilitates the intensive dialogue between employees and software developers on the topic of reference modeling, as well as the support of reference modeling with tools as seen by the user.
2. Since 1994, IDS Scheer, Inc. has provided several reference models created with the ARIS-Toolset. These were made available by the company and used for testing purposes within the research project that forms the foundation of this article.

B. Graphic Representation of the Models

The work area of the RMMS is divided up into an explorer and a viewer (cf. Figure 8). All of the information models managed by the RMMS are displayed in the explorer. This applies to the reference models constructed in development projects, as well as enterprise-specific models created in application projects.

The index card system of the RMMS serves the management of important information for the development and usage of reference models. The information models managed by the RMMS are characterized on the index card ‘Overview’. The other index cards serve the graphic model representation (‘Graphic’), the representation of model attributes (‘Attributes’) and the support of distributed construction processes (‘Collaboration’). In Figure 8, the index card ‘Graphic’ is activated. It gives users access to the versioning functionalities. The functionalities that support the graphic representation of the models in the RMMS will therefore be discussed first.

While the graphic representation of the information model is displayed in the left part of the index card, the attributes of the model components selected by the user are displayed on the right side. For navigation within the graphic, the user is given different functionalities. In the example in Figure 8, the version 1.2 of a reference model framework for event management which, due to its form, is referred to as ‘Event-E’, is selected. The project, which serves the development of a reference model for the application domain ‘event management’, was carried out at the Institute for Information Systems. Here, we will abstract from the functional aspects of this reference model. The event management reference model is documented in [40].

The user has selected the sector ‘Event Strategy’ in the framework. The attributes of this sector can be viewed in the attribute window that can be navigated though using a vertical scrollbar. In addition to the general attribute group and the attribute group on the model status, which can be seen in Figure 8, further attributes exist to characterize the model components such as, creator, date created, inspector, date of inspection, person responsible for release, date of release, validity period, etc.

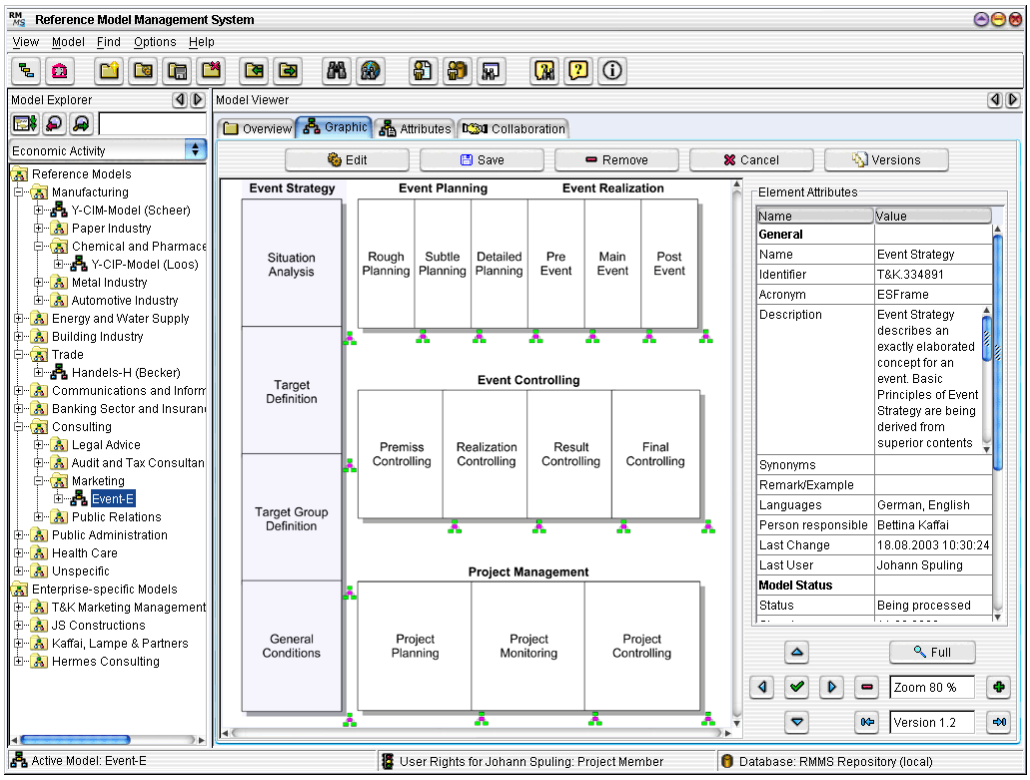

Figure 8. Graphic model representation

C. Interaction Design with the Basis Modeling Tool

If the user wishes to carry out changes on a model version he must first open the modeling tool. This can be done by clicking the button ‘Edit’. This command opens the file assigned to the information model component marked on the ‘Graphic’ card. In Figure 8, the user has marked the component ‘Event Strategy’ of the reference model framework ‘Event-E’. Using the ‘Edit’ button he can open the process model assigned to ‘Event Strategy’ in its current version. This ‘jump’ to the modeling tool is represented in Figure 9. In addition to reading, changing or deleting models and model elements, the user can now use further functionalities of the modeling tool. This pertains for example, to the graphic arrangement and grouping of model elements, the creation of model elements and element attributes, the placing of attributes or the connection of a model with OLE objects, such as for example, text documents or slides.

D. Managing Model Versions

The management of the model and model element versions made in the course of a reference modeling project (model history) is carried out via the dialog reachable using the button ‘Versions’ of the graphic index card (cf. Figure 8). In addition to the most important model data such as name, type or time of creation and modification, those responsible for the change, the type, reason, priority and status of the model changes, as well as the associated project activities are recorded. After pressing the button ‘Versions’, the version management dialog opens for the model displayed in the ‘Graphic’ window resp. for the model element selected in the representation window. Figure 10 shows an open version management dialog. The case represented here is the following: the user ‘Johann Spuling’ wishes to retrace the history of the model ‘Event Strategy’ after confirming the adoption of the modifications he had made to the EPC model ‘Event Strategy’. For this purpose, he has opened the corresponding path in the model explorer and called up the ‘Version Graph’ dialog.

The graphic displayed in the dialog ‘Version Graph’ in Figure 10 represents the structural relationships stored in the RMMS database between the versions of the active EPC model in the ‘Graphic’ window. It can be navigated using the vertical and horizontal scrollbar. With a simple mouse click, the user selects the version element constructs. The attributes of the marked version element are displayed in the right side of the window (‘Attributes’). By double-clicking, one can display the model assigned to the version construct in the ‘Graphic’ window. This way, the user can retrace the complete developmental path of the constructed information models. Using the toolbar at the top of the ‘Version Graph’ window, the user can create new models or model element versions (‘New’), save attribute changes (‘Save’), remove marked version constructs (‘Remove’) or reject respective changes and close the window (‘Cancel’). The version numbers are automatically generated by the RMMS. They can however, be changed by the user at a later point in time.

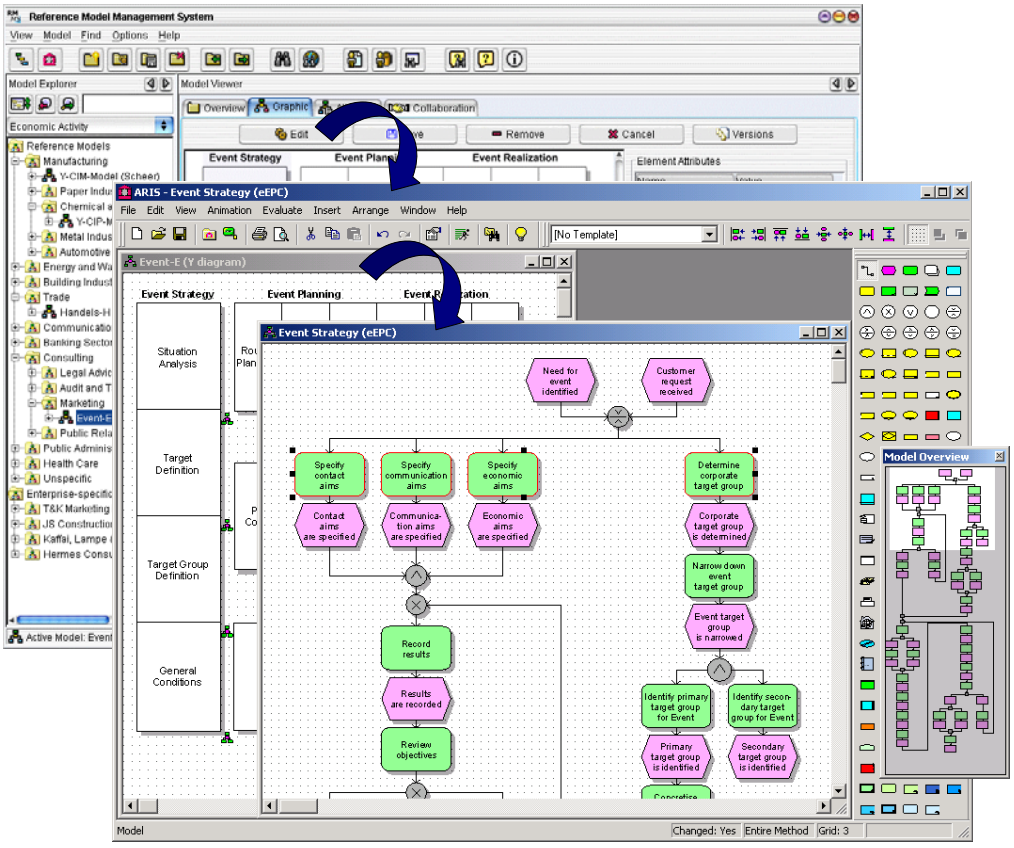

Figure 9. Interaction design between RMMS and the ARIS-Toolset

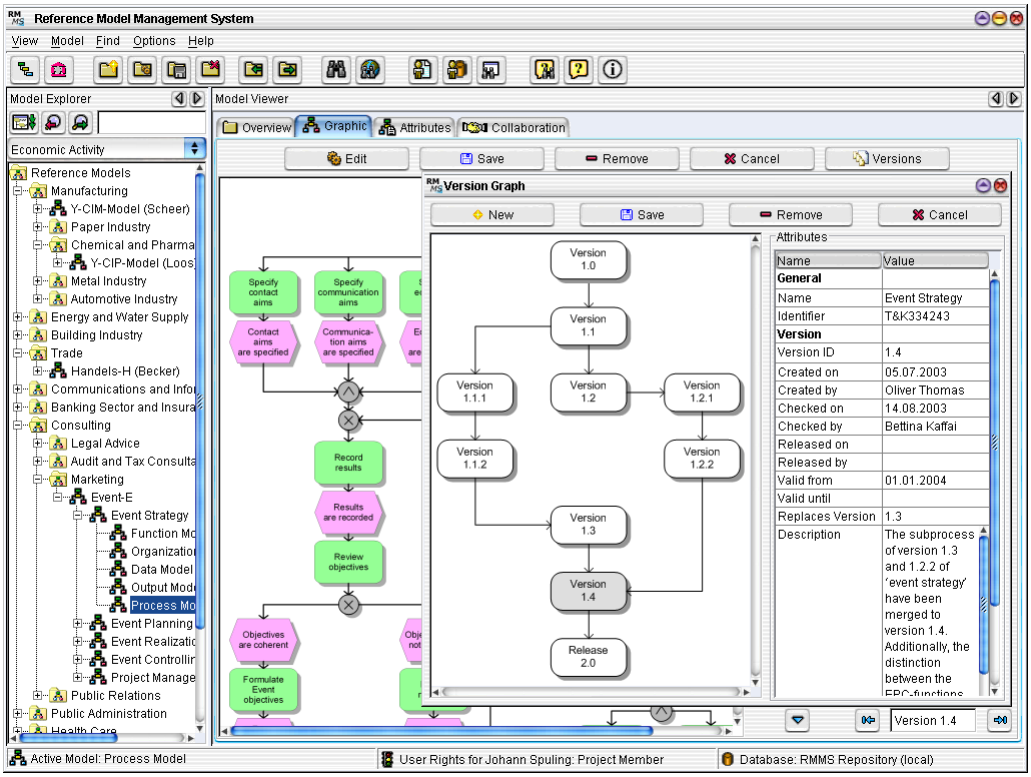

Figure 10. Managing model versions

VII. Related Work

Within the framework of a study on method engineering GREIFFENBERG et al. introduced an approach for the configuration management of information models [16; 21]. On the basis of available standards, as well as the corresponding experience reports ('best practices') from software development, the authors systematized the requirements on a model versioning tool. The natural language depictions are represented by diagrams of the so-called E^3-method [21, pp. 99ff]—and thus, transferred to a system specification for the versioning of information models. Which requirements are necessary for the integration of the approach in professional modeling tools was not dealt with by the authors.

VOM BROCKE differentiates between new constructions, version constructions and variant constructions in his studies on the design and distribution of construction processes according to the technical construction theory [45; 46; 46; 47]. Furthermore, the author emphasizes the versioning of reference models and enterprise-specific models, on the basis of so-called reference model components, as noteworthy in comparison to 'conventional' methods for the construction of reference models [45, pp. 262f]. He relates this to the documentation of the evolution of a stock of models achievable with the version relationships between constructions in all levels of the construction process. However, he does not say how these relationships are to be fashioned.

WARNECKE et al. also allude, within the framework of an evolution concept for reference models, to the fact that 'the increasing range of reference models, as well as the multitude of versions to be managed […] complicate the use of a reference model' [50, p. 63] and state that 'in addition […] all versions of a reference model must be managed and available at any time' [50, p. 63]. The authors do not however, answer the questions as to how the versioning of information resp. reference models can be designed.

Further related work generally deals with the systematization of reference models, whereby it is, in fact, the tabulation of reference models that is meant here and not so much the survey-like textual description of the actual stock of reference models found in literature [19; 42; 43].

It is indisputable that a model management, based on the cataloging of reference models, is very useful for the developers and users of reference models. It systematizes and facilitates access to the models and is suitable for supporting the search and selection of reference models. The said form for the model management is however based, as a rule, on the assumption of a given stock of models. Which advanced requirements must be made to model management, when the changes in a model over time (model evolution) are also to be considered, is not dealt with in the said studies.

VIII. Discussion of the Results and Outlook

Procedure models for reference modeling predominantly give recommendations for an incremental construction, i.e. the creation of a model step by step, whereby the development of the model progresses with each step—they are quasi constructed as a sequence of 'increments'. Despite this connection, few studies exist that deal with the problem of model versioning connected with this incremental development of reference models. The article at hand has accommodated this fact by designing an information system that supports the version management of reference models. The illustrated approach corresponds to an evolution concept for reference models, in which know-how from previous modeling tasks can be 'conserved', in order to make it available within the framework of other problems or tasks. The concept was designed on the basis of data structures and a system architecture and implemented in the form of an application system.

It became clear in the design phase, that the creative potential in the revisional management and control of the development and usage of reference models is justified from the perspective of a basic information model term, as well as from the perspective of a specific reference model term. Due to the use-oriented understanding of reference models in this article, as well as the generally heterogeneous stock of models, the formation of the concept and data structures had to be geared toward the versioning of information models, instead of solely to reference models. Based on this, the version management could be designed as a function whose performance in modeling projects is *first* permanently carried out, *second* serves the evolution of the results to be produced in these projects (information and reference models) and *third* supports the development, as well as the usage of reference models.

The central thoughts in the prototypical implementation were *first*, the implementation of the versioning as an 'integrating' component in a professional modeling tool and *second*, the use of version graphs taken from the field of software engineering. It is especially these graphs that form a basis for the navigation of model versions, enriched thanks to their graphic representations. The result is a prototype that allows system users to retrace model histories.

It must however, be criticized that the existing relationships between the information models and model elements resp. information model versions and element versions which were used, are ultimately a simplified interpretation of the relationships existing between a model and the elements of a model. They provide no information as to how model elements resp. element versions are to be aggregated to form a model resp. model version. The author sees a future challenge in the extension of the approach to the construction techniques currently "discussed" in the field of reference modeling, as well as in the embedding of other forms of version graphs, such as for example, sequences, trees and acyclic graphs.

ACKNOWLEDGMENT

This work has been funded by the German Research Foundation as part of the research project "Fuzzy-Customizing" (SCHE 185/25–1).

REFERENCES

[1] IDS Scheer AG (ed.), *ARIS Design Platform : Whitepaper*, Saarbruecken: IDS Scheer AG, 2007
[2] Fettke, P. and Loos, P. (eds.), *Reference Modeling for Business Systems Analysis*, London: Idea Group Publishing, 2007
[3] Abrial, J.-R., "Data Semantics," in: Klimbie, J. W. and Koffeman, K. L. (eds.), *Data Base Management : Proceeding of the IFIP Working Conference Data Base Management, Cargèse, Corsica, France, 1–5 April, 1974*, Amsterdam: North-Holland Publ., 1974, pp. 1–60
[4] Asklund, U., Bendix, L., Christensen, H. B., and Magnusson, B., "The Unified Extensional Versioning Model," in: Estublier, J. (ed.), *System configuration management : 9th international symposium ; SCM–9, Toulouse, France, September 5–7, 1999*, Berlin: Springer, 1999, pp. 100–122
[5] Becker, J., Algermissen, L., Delfmann, P., and Niehaves, B., "A Web Based Platform for the Design of Administrational Reference Process Models," in: Zhou, X., Su, S., Papazoglou, M. P., Orlowska, M. E., and Jeffery, K. G. (eds.), *Web information systems – WISE 2004: 5th International Conference on Web Information Systems Engineering, Brisbane, Australia, November 22–24, 2004 ; proceedings*, Berlin: Springer, 2004, pp. 159–169
[6] Becker, J., Delfmann, P., Dreiling, A., Knackstedt, R., and Kuropka, D., "Configurative Process Modeling – Outlining an Approach to Increased Business Process Model Usability," in: Khosrowpour, M. (ed.), *Innovations through information technology: 2004 Information Resources Management Association International Conference New Orleans, Louisiana, USA ; May 23–26, 2004*, Hershey: Idea Group Publ., 2004, pp. 615–619
[7] Becker, J. and Niehaves, B., "Epistemological perspectives on IS research: a framework for analysing and systematizing epistemological assumptions," *Information Systems Journal* 17 (2007), no. 2, pp. 197–214
[8] Bersoff, E. H., Henderson, V. D., and Siegel, S. G., *Software configuration management : An investment in product integrity*, Englewood Cliffs: Prentice Hall, 1980
[9] Bischofberger, W. and Pomberger, G., *Prototyping oriented software development : concepts and tools*, Berlin: Springer, 1992
[10] Blechar, M. J. and Sinur, J., *Magic Quadrant for Business Process Analysis Tools*, Stamford: Gartner Research, 2006
[11] Budde, R., *Prototyping : an approach to evolutionary system development*, Berlin: Springer, 1992
[12] Chen, P. P.-S., "The entity-relationship model – toward a unified view of data," *ACM Transactions on Database Systems* 1 (1976), no. 1, pp. 9–36
[13] Conradi, R. and Westfechtel, B., "Version models for software configuration management," *ACM Computing Surveys* 30 (1998), no. 2, pp. 232–282
[14] Curran, T. A., Keller, G., and Ladd, A., *SAP R/3 business blueprint : Understanding the business process reference model*, Upper Saddle River: Prentice Hall PTR, 1998
[15] Delfmann, P. and Knackstedt, R., "Towards Tool Support for Information Model Variant Management – A Design Science Approach," in: Österle, H., Schelp, J., and Winter, R. (eds.), *Proceedings of the Fifteenth European Conference on Information Systems*, University of St. Gallen, 2007, pp. 2098–2109
[16] Esswein, W., Greiffenberg, S., and Kluge, C., "Konfigurationsmanagement von Modellen," in: Sinz, E. J. and Plaha, M. (eds.), *Modellierung betrieblicher Informationssysteme : MobIS 2002, 9.–11. September 2002, Nürnberg*, Bonn: GI, 2002, pp. 93–112 (in German)
[17] Estublier, J., "Software Configuration Management: A Road Map," in: Finkelstein, A. (ed.), *The future of software engineering 2000 : [part of the] 22nd International Conference on Software Engineering*, New York: ACM Press, 2000, pp. 279–289
[18] Estublier, J. and Casallas, R., "Three Dimensional Versioning," in: Estublier, J. (ed.), *Software Configuration Management, ICSE SCM–4 and SCM–5 Workshops, Selected Papers*, London: Springer, 1995, pp. 118–135
[19] Fettke, P. and Loos, P., "Classification of Reference Models – A Methodology and its Application," *Information Systems and e-Business Management* 1 (2003), no. 1, pp. 35–53
[20] Frank, U., "Conceptual Modelling as the Core of the Information Systems Discipline – Perspectives and Epistemological Challenges," in: Haseman, W. D. and Nazareth, D. (eds.), *Proceedings of the Fifth Americas Conference on Information Systems (AMCIS 1999) : August 13–15, 1999, Milwaukee, Wisconsin*, Atlanta: AIS, 1999, pp. 695–698
[21] Greiffenberg, S., *Methodenentwicklung in Wirtschaft und Verwaltung*, Hamburg: Kovac, 2004 (in German)
[22] Grochla, E., Garbe, H., Gillner, R., and Poths, W., "Grundmodell zur Gestaltung eines integrierten Datenverarbeitungssystems : Kölner Integrationsmodell (KIM)," in: Grochla, E. and Szyperski, N. (eds.), *Arbeitsberichte des BIFOA an der Universität zu Köln*, no. 71/6, Köln: WISON Verl., 1971 (in German)
[23] Hirschheim, R., Klein, H. K., and Lyytinen, K., *Information Systems Development and Data Modelling : Conceptual and Philosophical Foundations*, Cambridge: Cambridge Univ. Press., 1995
[24] Melenovsky, M. J., *Business Process Management's Success Hinges on Business-Led Initiatives*, Stamford: Gartner Research, 2005
[25] Mendling, J. and Nüttgens, M., "EPC Markup Language (EPML) – An XML-Based Interchange Format for Event-Driven Process Chains (EPC)," *Information Systems and e-Business Management* 4 (2006), no. 3, pp. 245–263
[26] Mylopoulos, J., "Information Modeling in the Time of the Revolution," *Information Systems* 23 (1998), no. 3/4, pp. 127–155
[27] Noorani, R., *Rapid prototyping: principles and applications*, Hoboken: Wiley, 2006
[28] Ram, S. and Ramesh, V., "Collaborative Conceptual Schema Design: A Process Model and Prototype System," *ACM Transaction on Information Systems* 16 (1998), no. 4, pp. 347–371
[29] Recker, J., Rosemann, M., van der Aalst, W. M. P., Jansen-Vullers, M., and Dreiling, A., "Configurable Reference Modeling Languages," in: Fettke, P. and Loos, P. (eds.), *Reference Modeling for Business Systems Analysis*, London: Idea Group Publishing, 2007, pp. 22–46
[30] Recker, J., Rosemann, M., van der Aalst, W. M. P., and Mendling, J., "On the Syntax of Reference Model Configuration – Transforming the C-EPC into Lawful EPC Models," in: Bussler, C. and Haller, A. (eds.), *Business Process Management Workshops : BPM 2005 International Workshops, Nancy, France, September 5,*

2005 ; Revised Selected Papers, Berlin: Springer, 2006, pp. 497–511
[31] Rochkind, M. J., "The Source Code Control System," *IEEE Transactions on Software Engineering* 1 (1975), no. 4, pp. 364–370
[32] Rosemann, M. and van der Aalst, W. M. P., "A configurable reference modelling language," *Information Systems* 32 (2007), no. 1, pp. 1–23
[33] Scheer, A.-W., "A Software Product is Born," *Information Systems* 19 (1994), no. 8, pp. 607–624
[34] Scheer, A.-W., *Business Process Engineering : Reference Models for Industrial Enterprises*, 2nd ed. Berlin: Springer, 1994
[35] Scheer, A.-W., *ARIS – business process frameworks*, 2nd ed. Berlin: Springer, 1998
[36] Scheer, A.-W., Thomas, O., and Adam, O., "Process Modeling Using Event-driven Process Chains," in: Dumas, M., van der Aalst, W. M. P., and ter Hofstede, A. H. M. (eds.), *Process-aware Information Systems : Bridging People and Software through Process Technology*, Hoboken: Wiley, 2005, pp. 119–145
[37] Sommerville, I., *Software Engineering*, 6th ed. Munich: Pearson, 2001
[38] Thomas, O., "Understanding the Term Reference Model in Information Systems Research: History, Literature Analysis and Explanation," in: Bussler, C. and Haller, A. (eds.), *Business Process Management Workshops : BPM 2005 International Workshops, Nancy, France, September 5, 2005 ; Revised Selected Papers*, Berlin: Springer, 2006, pp. 484–496
[39] Thomas, O., Adam, O., and Loos, P., "Using Reference Models for Business Process Improvement: A Fuzzy Paradigm Approach," in: Abramowicz, W. and Mayr, H. C. (eds.), *Business Information Systems : 9th International Conference on Business Information Systems (BIS 2006) ; May 31-June 2, 2006, Klagenfurt, Austria*, Bonn: Köllen, 2006, pp. 47–57
[40] Thomas, O., Hermes, B., and Loos, P., "Towards a Reference Process Model for Event Management," in: *10th International Workshop on Reference Modeling at the 5th International Conference on Business Process Management (BPM 2007), 24–28 September 2007, Brisbane, Australia*, 2007 (to appear)
[41] Thomas, O. and Scheer, A.-W., "Tool Support for the Collaborative Design of Reference Models – A Business Engineering Perspective," in: Sprague, R. H. (ed.), *Proceedings of the 39th Annual Hawaii International Conference on System Sciences : 4–7 January 2006, Kauai, Hawaii*, Los Alamitos, CA: IEEE Computer Society Press, 2006
[42] van Belle, J.-P., *A Framework for the Analysis and Evaluation of Enterprise Models*, Cape Town, South AfricA, University of Cape Town, Dept of Information Systems, PhD Thesis, 2003
[43] van Belle, J.-P., "Evaluation of Selected Enterprise Reference Models," in: Fettke, P. and Loos, P. (eds.), *Reference Modeling for Business Systems Analysis*, London: Idea Group Publishing, 2007, pp. 266–286
[44] van der Aalst, W. M. P., Dreiling, A., Gottschalk, F., Rosemann, M. and Jansen-Vullers, M. H., "Configurable Process Models as a Basis for Reference Modeling," in: Kindler, E. and Nüttgens, M. (eds.), *Business Process Reference Models : Proceedings of the Workshop on Business Process Reference Models (BPRM 2005), Nancy, France, September 5, 2005*, Nancy, 2005, pp. 76–82
[45] vom Brocke, J., *Referenzmodellierung : Gestaltung und Verteilung von Konstruktionsprozessen*, Berlin: Logos, 2003 (in German)
[46] vom Brocke, J., "Design Principles for Reference Modeling: Reusing Information Models by Means of Aggregation, Specialisation, Instantiation and Analogy," in: Fettke, P. and Loos, P. (eds.), *Reference Modeling for Business Systems Analysis*, London: Idea Group Publishing, 2007, pp. 47–75
[47] vom Brocke, J. and Buddendick, C., "Reusable Conceptual Models – Requirements Based on the Design Science Research Paradigm," in: Hevner, A. R. (ed.), *First International Conference on Design Science Research in Information Systems and Technology : February 24–25, 2006, Claremont, CA ; Proceedings*, 2006
[48] vom Brocke, J. and Thomas, O., "Designing Infrastructures for Reusing Conceptional Models – A General Framework and its Application for Collaborative Reference Modelling," in: Abramowicz, W. and Mayr, H. C. (eds.), *Business Information Systems : 9th International Conference on Business Information Systems (BIS 2006) ; May 31-June 2, 2006, Klagenfurt, Austria*, Bonn: Köllen, 2006, pp. 501–514
[49] Wand, Y. and Weber, R., "Research Commentary: Information Systems and Conceptual Modeling – A Research Agenda," *Information Systems Research* 13 (2002), no. 4, pp. 363–376
[50] Warnecke, G., Stammwitz, G., Hallfell, F., and Förster, H., "Evolutionskonzept für Referenzmodelle," *Industrie Management* 14 (1998), no. 2, pp. 60–64 (in German)
[51] Weber, R., *Ontological foundations of information systems*, Melbourne : Coopers & Lybrand, 1997
[52] Wolff, F. and Frank, U., "A Multi-Perspective Framework for Evaluating Conceptual Models in Organisational Change," in: *Proceedings of the 13th European Conference on Information Systems, Information Systems in a Rapidly Changing Economy, ECIS 2005, Regensburg, Germany, May 26–28, 2005*, pp. 1283–1294
[53] Zeller, A., *Configuration Management with Version Sets : A Unified Software Versioning Model and its Applications*, Braunschweig, Techn. Univ., PhD Thesis, 1997

Dr. Oliver Thomas was a research assistant at the Institute for Information Systems at the Saarland University from 1999 to 2002. In 2002, the institute was integrated into the German Research Center for Artificial Intelligence (DFKI) as a department. Dr. Thomas has been the deputy head of the research department and a senior researcher in this department since 2003. He also heads the Business Process Management research cluster at the Institute for Information Systems. His main research fields are 'Integrated information systems for manufacturing, service industries and administration', 'Methods and tools for business process management', 'Information modeling and reference modeling' and 'Product-service systems'. He also lectures at the Saarland University and holds the courses 'Information Modeling' and 'Reference Modeling' for the study paths 'business administration' and 'bachelor/master of information systems'. In addition, Dr. Thomas holds university teaching positions at University of Hamburg (Germany), Martin Luther University Halle-Wittenberg (Germany), and at the Aoyama Gakuin University, in Tokyo (Japan). Since 2004 Dr. Oliver Thomas is visiting associate professor at Aoyama Gakuin University.