TOWARDS MODEL-DRIVEN EVOLUTION OF DATA WAREHOUSES

Christian Kurze, Marcus Hofmann, Frieder Jacobi, André Müller and Peter Gluchowski
Chemnitz University of Technology, Faculty of Economics and Business Administration, 09107 Chemnitz, Germany

Keywords: Data Warehouse, Model-driven Architecture, Architecture-driven Modernization, Model Management, Re-engineering, Data Warehouse Evolution.

Abstract: Data warehouse systems represent centralized data collections used for the purposes of data analysis and decision support. Development and maintenance of extensive data warehouse systems require appropriate support through methods and tools. Therefore, we introduce the project Computer-Aided Warehouse Engineering (CAWE). It is a model-driven approach to the engineering of data warehouse systems. Especially the process of data warehouse evolution, i.e. the maintenance of the core data storage component, the data warehouse in narrow sense, is a tedious task since the change of data structures implies the challenge of ensuring integrity and history of corporate data. The paper at hand provides research in progress and suggests the adoption of a multidimensional algebraic formalism to the model-driven development paradigm.

1 INTRODUCTION

Nowadays, data warehouse systems form the backbone of many companies. They contain a subject oriented, integrated, time variant and consistent collection of data and therefore build the central component of modern decision support systems (Inmon et al., 2008).

An increased company size leads to a considerable complexity of these systems. At the same time, the applied methods and tools for system development and management cannot deal with this grade of complexity. Subsequently, extensive data warehouse systems are hard to handle. A loss of efficiency, faulty analyses, the absence of necessary adaptations of the system and also high costs in maintenance and further development form the negative consequences.

We adopt the promising approach of model-driven architecture (MDA) (Object Management Group, 2003; Czarnecki and Helsen, 2006) in order to ease development and evolution of complex data warehouse systems. Our focus is put on the data warehouse in a narrow sense, i.e. the core data storage component.

The paper is structured as follows: after introducing the CAWE approach, we describe the multidimensional algebraic formalism of (Blaschka, 2000) and show a possible adoption in the CAWE prototype.

2 THE CAWE APPROACH

Aspects like data safety, data security and compliance put higher demands on development and documentation of data warehouses as well as metadata management. Furthermore, there is a shift in emphasis from the IT perspective towards the business perspective. Because of that, there is a need for adequate, continuous methods and tools to support the lifecycle management of data warehouses from the business layer towards the implementation. The following tasks are practically relevant:

- Documentation and assessment of existing systems.
- Implementation of new systems on the basis of the requirements of the business layer.
- Migration of legacy systems into modern architectures.

Data warehouse systems consist of different layers: operational systems, ETL processes, multidimensional data storage, and applications (Chaudhuri and Dayal, 1997). Each layer has to be taken into account during the development process in order to get a fully fledged data warehouse system.

Kurze C., Hofmann M., Jacobi F., Müller A. and Gluchowski P..
TOWARDS MODEL-DRIVEN EVOLUTION OF DATA WAREHOUSES.
DOI: 10.5220/0003444303560360
In *Proceedings of the 13th International Conference on Enterprise Information Systems* (ICEIS-2011), pages 356-360
ISBN: 978-989-8425-53-9

Combining the typical layers of data warehouse systems and the viewpoints of MDA, (Mazón and Trujillo, 2008) presented a promising framework which has been extended and modified by (Kurze and Gluchowski, 2010), as shown in figure 1. The intended advantages are: definition of a systematic and well-structured way for the development of heterogeneous data warehousing layers; increased productivity by generating large parts of the implementation; better portability and adaptability by using the concept of separation of concerns; an integrated modeling framework which supports comprehensive requirements definition support, as proposed, for example, by (Guo et al., 2006); support for system evolution based on an architecture-driven modernization (ADM) (Ulrich and Newcomb, 2010) and the explicit integration of the business domain.

Since the development of data warehouse systems is strongly aligned to software development, the research framework is based on the general terms forward, reverse and re-engineering. They are well discussed within the context of software-engineering (Chikofsky and Cross, 1990). Furthermore, their application has been transferred to data engineering problems by (Aiken, 1996).

In order to support a triple-driven design of data warehouses as a combination of data provided by operational systems, user requirements against the data warehouse as well as goals defined independently of available data and user requirements (Guo et al., 2006), the framework defines forward and reverse engineering for data warehouses as a whole spreading over the given layers of a data warehouse system (Kurze and Gluchowski, 2010):

- *Forward engineering* of data warehouses requires a reverse engineering of operational systems in order to support data-driven requirements engineering as well as forward engineering of multidimensional data storage, applications and ETL processes.
- *Reverse engineering* of data warehouses is also based on a reverse engineering of operational systems as a requirement for reverse engineering of ETL processes. Furthermore, it consists of reverse engineering of multidimensional data storage and applications.
- *Re-engineering* of data warehouses as a combination of forward and reverse engineering is based on a reverse engineering of operational systems, and a re-engineering of ETL processes, multidimensional data storage, and applications.

Figure 1 summarizes the framework and shows the necessary metamodels of each data warehousing layer as well as their corresponding MDA viewpoint. The whole framework is encompassed by a frame that offers the theoretical concepts of model management as well as facilities to generate technical and end-user documentation. It ensures the seamless integration into metadata management systems.

The connection of requirements of end-users, corporate strategy and data sources supports a triple-driven requirements engineering and helps to increase the quality of reverse engineering initiatives by aligning and improving automatically generated artifacts with the implicit knowledge of users.

Furthermore, the framework includes a metamodel for each layer of data warehouses according to the MDA viewpoint. Requirements can be captured in a metamodel suitable for end-users, e.g. ADAPT (Bulos and Forsman, 2006; Gluchowski et al., 2009). They form the basis for model-to-model (M2M) transformations into *Platform Independent Models* (PIMs) for ETL, multidimensional data storage (MD), and applications. Within each layer there are further M2M and model-to-text (M2T) transformations that generate source code. In the reverse direction there are text-to-model (T2M) and M2M transformation to gather abstract models from actual implementations.

Due to different implementation alternatives, the multidimensional data layer requires the distinction between a relational and a pure multidimensional implementation on *Platform Specific Model* (PSM) and *Code* viewpoints.

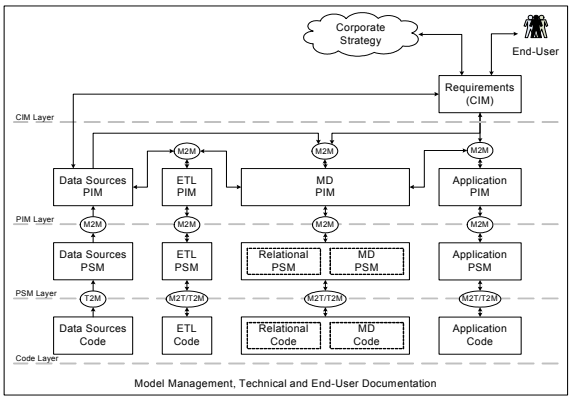

Figure 1: Framework for model-driven forward, reverse, and re-engineering of data warehouse systems (Kurze and Gluchowski, 2010).

Focusing on the multidimensional data layer, figure 2 shows the metamodels which are necessary for its evolution in the ADM horseshoe model. Within the first step, an *MD PIM* and a *Requirements* model are reverse engineered from the current implementation (depicted as *Relational PSM* and

Relational Code as well as *MD PSM* and *MD Code*). The second evolution step is the enrichment of the requirements. The result is a second requirements model: *Requirements'* which is transformed into a new *MD PIM'* as well as into target PSMs and corresponding code (*Relational PSM'*, *MD PSM'*, *Relational Code'*, and *MD Code'*).

While this approach would be suitable for pure software artifacts, it is not suitable for data engineering problems: it does not take into account the instance layer of data. When generating new code, especially for the relational data storage, instance data might be modified or even deleted. The particular feature of data warehouses is the offering of historical data, i.e. data must not be deleted: after an evolution, analyses with historical data must still be possible. Therefore, it is essential not to modify any instance data in an uncontrolled way. The approaches of *schema and data evolution* are promising to overcome this issue, cf., for example, (Favre, 2009; Blaschka, 2000; Kaas et al., 2004).

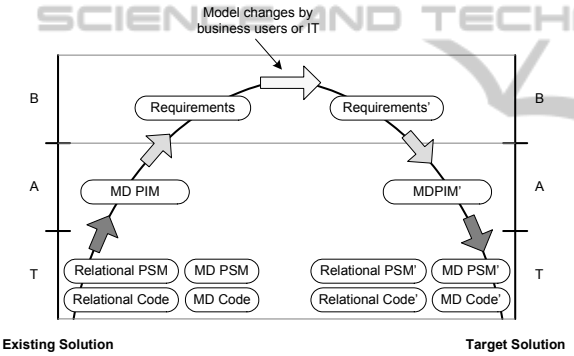

Figure 2: Re-engineering of multidimensional data in the ADM Horseshoe Model.

For data-intensive applications like data warehouse systems the following research issue results: Which methods support the evolution of data warehouse systems within the scope of the model-driven engineering considering the instance level?

Basically, the above remarks lead to two use cases for the re-engineering:

1) The transfer of all the structures and instance data from one existing platform to another.
2) Evolution of the data warehouse system based on new functional requirements, in the meaning of an adjustment of existing structures persistent over all levels of abstraction.

Particularly because of missing real-world implementations, the latter is in the focus of further treatment in the paper at hand.

3 DATA WAREHOUSE EVOLUTION

3.1 Theoretical Approach

(Blaschka, 2000) and (Kaas et al., 2004) take into account the challenges of schema evolution with respect to instance data. (Blaschka, 2000) developed an algebraic formalism for the treatment of multidimensional schemata. He provides two models for multidimensional data: multidimensional schema and multidimensional instance. Based on these models, it has been possible to deduce a set of 14 elementary operators, which can represent all changes within the given multidimensional model. The schema change operations by a user are logged, so that a sequence of operators for schema evolution is derived. Well-defined priority rules for each operator ensure a correct transformation. Based on the 14 conceptual operators, Blaschka derived a mapping of logical operators, which are executable on the relational star schema. They form the basis to generate SQL code which performs the actual evolution. Kaas et al. further examined the logical operators and show particular implementation challenges in star and snowflake schemata.

The existence of two models, before and after the change, can lead to a theoretically infinite sequence of operators to represent the changes. Nevertheless, the sequence of operators can be determined using planning algorithms developed in the area of artificial intelligence (Barr and Feigenbaum, 1990). The approach of model management (Bernstein and Melnik, 2007) proposes several abstract operators. Particularly interesting for the evolution of data warehouses is the *Diff* operator which describes the changes done on a model. Subsequently, this operator allows deriving a difference model representing the changes a user made on requirements artifacts.

Our evolution approach is grounded on Blaschka's concept. We make use of the abstract *Diff* operator to determine changes on a model. These changes are the basis to determine the necessary operations on platform-specific models which represent the actual implementation of evolution. Figure 3 illustrates the concept described.

The framework assumes a platform-independent model as well as the associated platform-specific models and their implementation. They can either be generated by a forward engineering or be the result of a reverse engineering. Furthermore, there exists a manipulated platform-independent model (PIM'). Basically, this model could be transformed into plat-

form-specific models and code which would override existing schemata and instance data.

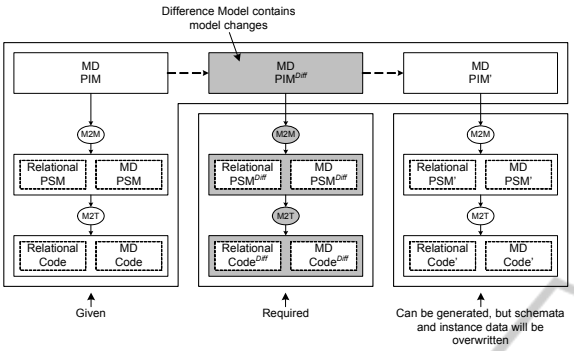

Figure 3: Framework for re-engineering of multidimensional data models.

Therefore, we assume a difference model *MD PIM^{Diff}* which contains all changes made by a user on the *MD PIM*. This model is used to deduce a sequence of operators which, applied to the source model *MD PIM*, represents a path from *MD PIM* to the target model *MD PIM'*. These operators conform to the operators by Blaschka on his multidimensional schema model. The mapping of these abstract operators to specific systems (as done by Blaschka and Kaas et al. for the relational implementation) is the basis to generate various difference models on a platform-specific viewpoint.

Based on this sequence of operations, code, which adjusts the actual implementation, is generated through M2T transformations. While changes to the structure of the data are executed mostly automatically, changes to the data pool itself may require manual assistance under certain circumstances (e.g. the insertion of a new dimension level requires the specification of hierarchical classifications).

3.2 Adoption in CAWE

The next step consists of transferring the theoretical problem description to the prototype developed within the CAWE project. An adequate starting point is the schema evolution algebra introduced by Blaschka. Since its capabilities are constrained by his underlying mathematical models, some peculiarities of multidimensional modeling cannot be described properly. To expand Blaschka's formalism to the metamodels developed for CAWE, an implementation of this formalism via a computer algebra system is planned. This enables the extended set of operators to be tested for correctness. Furthermore, there has to be a seamless integration, i.e. a formal mapping, of abstract algebra and the CAWE metamodels. This ensures the fit between metamodels and algebra as well as the determination of model transformations representing the operators.

Another task is the determination of the *MD PIM^{Diff}* model and the underlying operators. (Blaschka, 2000) supposes to log the change actions done by a user on the conceptual model. However, a sequence determined in this way does not necessarily need to be optimal. The Eclipse Modeling Framework (EMF) technology underlying the CAWE prototype has an appropriate built-in tool for this purpose: EMF Model Transaction. It has to be shown in which way the generic operators of the EMF can be mapped on those of the extended algebra. A second way to determine the difference model is the usage of EMF model compare. Its result is a formal difference model. Following this approach, it is necessary to deduce the operations by the given difference between the source and the target model.

Once the needed operations are determined, the next task is to distinguish their correct sequence while considering that pre- and post-conditions hold. Basically, this issue can be described best as a planning problem. A solution using methods from the area of artificial intelligence research seems to be obvious (Barr and Feigenbaum, 1990). Further analysis whether existing algorithms can be used or utilized as basis for a solution of this problem is essential. This process is supported through the implementation of the computer algebra system, too.

For the creation of transformations from multidimensional operators into their relational implementation, the following step is necessary: It has to be determined algorithmically, which multidimensional operators have to be realized as a *composite operation* in the relational database (e.g. *insert attribute* with a following *connect attribute to...)*. If these operators have been defined on a relational level, the data definition and data manipulation scripts being necessary for the schema evolution can be derived. Blaschka already derived scripts for the star schema using his algorithm. Also (Kaas et al., 2004) followed up this topic and described the implementation of eight operators identified by them for star and snowflake schemata. The CAWE approach takes up this state of research, adjusts it to the developed metamodels and integrates it into the prototype.

4 CONCLUSIONS AND FUTURE WORK

The problem of the data warehouse evolution is of significant practical relevance and has been analyzed multiple times before. On an abstract, theoretical level this problem has been formalized by the approach of model management (Bernstein and Melnik, 2003). Particular research focusing on the evolution of data warehouses has been done, for example, by (Blaschka, 2000) and (Kaas et al., 2004). Their approaches are very promising. Therefore, the CAWE approach adopts the aforementioned ideas to support the evolution of data warehouse schemata as well as instance data.

To be able to adopt this idea to the model-driven data warehouse engineering in general and to its integration in CAWE in detail, we intend the following further steps:

- Implementing Blaschka's algebra in a computer algebra system like Mathematica.
- Determination of operators from a given difference model.
- Selection and implementation of an appropriate planning algorithm from the area of artificial intelligence to determine the correct sequence of operators.
- Development of metamodels for difference models to represent the operators which are necessary on the different MDA viewpoints.
- Design and implementation of transformations between the developed difference metamodels considering the results of Blaschka and Kaas et al.
- Extension of Blaschka's formalism according to the CAWE metamodels in order to support advanced multidimensional modeling concepts.

ACKNOWLEDGEMENTS

This research has been partly funded by the European Social Fund and the Free State of Saxony.

REFERENCES

Aiken, P., 1996. *Data Reverse Engineering: Slaying the Legacy Dragon*, McGraw-Hill. New York et al.

Barr, A., Feigenbaum, E. A., 1990. The Handbook of Artificial Intelligence, Volume 1, Addison-Wesley. Reading et al, 2nd edition.

Bernstein, P. A., Melnik, S., 2007. Model management 2.0: manipulating richer mappings. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, pp. 1–12.

Blaschka, M., 2000. *FIESTA: A Framework for Schema Evolution in Multidimensional Databases*. Dissertation Thesis at Technical University Munich, Germany.

Bulos, D., Forsman, S., 2006. Getting started with ADAPT. Online available: http://www.symcorp.com/downloads/ADAPT_white_paper.pdf. Last access: 2011-01-11.

Chaudhuri, S., Dayal, U., 1997. An overview of data ware housing and OLAP technology. *ACM SIGMOD Record*, 26(1), pp. 65-74.

Chikofsky, E. J., Cross, J. H., 1990. Reverse engineering and design recovery: a taxonomy. *IEEE Software*, 7(1), pp. 13–17.

Czarnecki, K., Helsen, S., 2006. Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3), pp.621–645.

Favre, C., Bentayeb, F., Boussaid, O., 2009. A Survey of Data Warehouse Model Evolution. In *Handbook of Research on Innovations in Database Technologies and Applications: Current and Future Trends*, Hershey. New York.

Gluchowski, P., Kurze, C., Schieder, C., 2009. A Modeling Tool for Multidimensional Data using the ADAPT Notation. In *Proceedings of the 42nd Hawaii International Conference on System Sciences*. IEEE.

Guo, Y., Tang, S., Tong, Y., Yang, D., 2006. Triple-driven data modeling methodology in data warehousing: a case study. In *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP*. ACM, p. 59–66.

Inmon, W. H., Strauss, D., Neushloss, G., 2008. *DW 2.0: the architecture for the next generation of data warehousing*, Morgan Kaufmann. Amsterdam et al.

Kaas, C., Pedersen, T. B., Rasmussen, B., 2004. Schema evolution for stars and snowflakes. In *Proceedings of the 6th International Conference on Enterprise Information Systems*. INSTICC, pp. 425–433.

Kurze, C., Gluchowski, P., 2010. Computer-Aided Warehouse Engineering (CAWE): Leveraging MDA and ADM for the development of Data Warehouses. In *Proceedings of the 16th Americas Conference on Information Systems*. AIS, paper 282.

Mazón, J. N., Trujillo, J., 2008. An MDA approach for the development of data warehouses. *Decision Support Systems*, 45(1), pp. 41–58.

Object Management Group, 2003. *MDA Guide Version 1.0.1*. Online available: http://www.omg.org/cgi-bin/doc?omg/03-06-01. Last access: 2011-01-11.

Ulrich, W., Newcomb, P. H., 2010. Information Systems Transformation: Architecture-Driven Modernization Case Studies, Morgan Kaufmann. Amsterdam et al.