# An MDA–Based Approach to Managing Database Evolution (position paper) ⋆

Eladio Domínguez [a] Jorge Lloret [a] Ángel L. Rubio [b]
María A. Zapata [a]

[a] *Dpto. de Informática e Ingeniería de Sistemas.*
*Facultad de Ciencias. Edificio de Matemáticas.*
*Universidad de Zaragoza. 50009 Zaragoza. Spain.*

[b] *Dpto. de Matemáticas y Computación. Edificio Vives.*
*Universidad de La Rioja. 26004 Logroño. Spain*

## 1 Introduction

In this paper we are going to present the status of an ongoing research about the relationships between a particular architecture [5] that tackles with a concrete problem related to database technology –the problem of database evolution- and the Model-Driven Architecture (MDA [14]).

The requirements of a database do not remain constant during its life time and therefore the database has to evolve in order to fulfil the new requirements. Since database evolution activities consume a large amount of resources [13] they are considered of great practical importance and, as a consequence, much research has been focused on analyzing ways of facilitating this task [2,16]. In particular, among the several problems that are related to evolution activities (see [11]), one of the most important is that of 'forward database maintenance problem' (or 'redesign problem', according to [16]). This problem faces how to reflect in the logical and extensional schemata the changes that have occurred in the conceptual schema of a database. As a contribution towards achieving a satisfactory solution to this problem (that has not been found yet, despite a lot of efforts by different researchers [16,13]), some of the authors of the present paper have presented in [5] an architecture for managing database evolution.

On the other hand, the MDA is an initiative led by the Object Management Group (OMG) that embodies "the expanded vision necessary to support interoperability with specifications that address integration through the entire systems life cycle: from business modeling to system design [...] and evolution"[14]. Therefore, although MDA deals mainly

⋆ This work has been partially supported by DGES, projects TIC2000-1368-C03-01, TIC2002-01626, and by Ibercaja-University of Zaragoza, project IB 2002-TEC-03

*Email addresses:* noesis@posta.unizar.es (Eladio Domínguez), jlloret@posta.unizar.es (Jorge Lloret), arubio@dmc.unirioja.es (Ángel L. Rubio), mazapata@posta.unizar.es (María A. Zapata).

with software development and less with data modeling issues, it is implied explicitly in evolution tasks. We have found out that several key features of MDA are shared by our above-mentioned architecture for managing database evolution. Firstly, the notion of mapping, used in MDA in order to transform Platform Independent Models (PIMs) into Platform Specific Models (PSMs) –and backwards-, is also a central point when dealing with database modeling and evolution. Secondly, both approaches made an intensive use of metamodels in order to represent 'modeling knowledge'.

In this paper we make a proposal of how to get both worlds (database evolution and MDA) closer. The main aim of the work is to investigate how to take advantage of the advances in the MDA field (in particular the development of MDA-based tools) in order to use them in a databases evolution context. The remainder of the paper is organized as follows. Section 2 explains our view of the relationships between database engineering and MDA, presenting in Section 3 the specific relationships between MDA and our architecture for managing database evolution. Finally, some conclusions are outlined in Section 4.

## 2 Database Engineering and MDA

Several recent papers have already addressed some similarities between database technology and MDA concepts (see [15,10]). However it is far from being clear how should database engineering concepts be reinterpreted in terms of MDA.

Traditionally, in the databases field, the term 'conceptual schema' refers to a model that captures the user's information requirements, for example by means of an Entity/ Relationship (ER) Model. Following the ANSI-SPARC Architecture [3], a conceptual schema is independent of any physical implementation (and of any Database Management System, DBMS). Moreover, a conceptual schema is independent of any computational aspects. Because of that, we think that the 'conceptual schema' idea corresponds with the MDA 'Business Model' concept. Unfortunately, the MDA Specification Document leaves out Business Models quickly, and focuses its attention on PIMs and PSMs, so much so that Business Models are omitted from the MDA Metamodel Description (page 12 of [14]). We advocate for making use of the footnote on page 7 of this Document, where it is said that "while [Business Model] need not be explicitly present in a particular usage of the MDA Scheme, MDA accommodates it consistently in the same overall architecture".

The next step in a database development process is to obtain, starting from the conceptual schema, a 'logical schema' that is described using the (Object-) Relational Model. The logical schema is closer to computational aspects, but it is not dependent on any particular DBMS. Therefore, a logical schema can be seen as a Platform Independent Model from the point of view of MDA. This idea is strengthened by analyzing one of the core specifications of the MDA, the Common Warehouse Metamodel (CWM [4]). This specification includes a metamodel for Relational data resources, that is based in the SQL Standard [12]. However, as it has been proven in the literature [18,17], each particular DBMS implements its specific version of SQL, in such a way that a (object-) relational schema that it is assumed to be in accordance with the standard, can be not valid for a particular DBMS. This is the main reason because we think that a description of a 'logical schema' based on the (object-) relational approach and/or the standard SQL should be considered as a 'Platform Independent Model', and that a description of that schema using the particular version

of SQL of a particular DBMS should be considered as a 'Platform Specific Model'. These ideas are strengthened by the functionality of different software tools that are used to automatize the data modeling process, such as DB-MAIN [11]. A user of this tool can draw a conceptual (ER) schema, that is automatically translated to a logical (relational) schema. Then, the tool allows the user to guide the translation process of this logical schema towards SQL code adapted to a particular platform (DBMS), such as Oracle.

In order to illustrate graphically these ideas, in Figure 1 we show a modification of the MDA Metamodel Description from the one presented in [14]. There are three basic differences between the original description and our proposal. Firstly, Business Models are explicitly included as a new (meta)class, which leads to include two associations between 'Business Model' and 'PIM' classes. Secondly, an association class 'BM-PIM Mapping Techniques' is included, in order to represent the translation modeling knowledge from Business Models (conceptual schemas in the database context) to PIMs (logical schemas). It seems surprising to us that in the original MDA Metamodel Description there are explicit association classes to represent 'PSM Mapping Techniques' (from PSM to PSM) and 'PIM Mapping Techniques' (from PIM to PIM), but there not exists analogous association classes to represent 'PIM to PSM Mapping Techniques' or 'PSM to PIM Refactoring Techniques'. Our proposed inclusion of the class 'BM-PIM Mapping Techniques' (that in our opinion it is essential, at least in the database context), makes us think about the necessity of other classes (like, for instance, 'PIM-BM Refactoring Techniques', useful for database reverse engineering). Lastly, our metamodel description includes a 'BM Mapping Techniques' association class, which is a key point in a database evolution setting, since, as we have said before, in the forward database maintenance problem the changes in the conceptual schema (Business Model) are taken into account to determine changes in the logical and extensional schemata.

We recognize that this is not the only possible interpretation. There exists some controversy in the literature about the use of ER schemata as Business Models [9]. As another example, in [10] a different approach is considered, since ER schemata are identified as PIMs, and relational schemata are identified as 'prototypical' PSMs. Moreover, relational schemata are used somehow as Business Models in [1]. However we think that all these approaches are not mutually exclusive, since it is a matter of 'level of detail'. Different kinds of ER schemata can be used at different levels of abstraction and with different levels of detail. In fact, this is likely quite near of the MDA vision, since it defines concepts such as 'abstraction', 'refinement' and 'viewpoint'.

## 3 Database evolution and MDA

In this section we are going to outline the basic characteristics of our proposed architecture for managing database evolution (for details, see [5]). Afterwards, we are going to show which are the relationships between this architecture and our interpretation (that has been described in the previous section) of the database concepts in terms of MDA.

Our proposed architecture for managing database evolution makes use of a structural artifact that consists of three components: an information schema, an information base and an information processor. The information schema defines all the knowledge relevant to the system (and therefore it plays a 'metamodel' role), the information base describes the

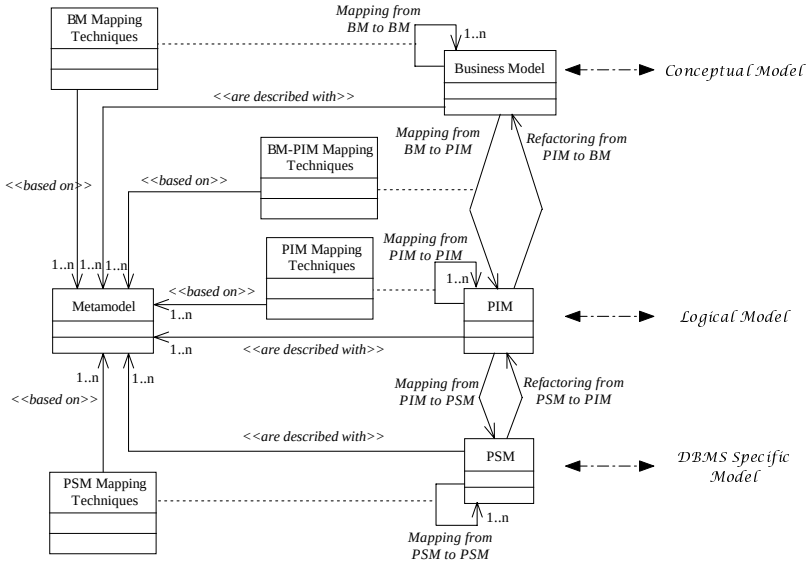


Fig. 1. MDA Metamodel Description (modified from [14]) and its relation with database concepts

specific objects perceived in the Universe of Discourse (a 'model' role), and the information processor receives messages reporting the occurrence of events in the environment. In order to respond to the events received, the information processor can send structural events towards the information base and/or towards the information schema and can generate internal events that inform other processors of the changes performed in it. This structural artifact is used within our architecture giving rise to four structures which are used to store, respectively, the conceptual modeling knowledge, the translation process, the logical modeling knowledge and the extension. The corresponding components of each one of these structures as well as the way in which they are related appear in Figure 2. The name of each one of these components has been modified in an attempt to capture the type of knowledge that they store.

There are two main characteristics of this architecture that make it different of other proposals. On the one hand it includes an explicit translation component that stores information about the way in which a concrete conceptual database schema is translated into a logical schema. This component plays an important role in enabling the automatic propagation of evolution from the conceptual to the extensional schemata. On the other hand, a meta–modeling approach [6] has been followed for the definition of the architecture. Within this architecture, three meta–models are considered which capture, respectively, the conceptual, logical and translation modeling knowledge.

There are several relationships between this architecture and MDA concepts. First of all, both make a explicit use of metamodels. For example, metamodels are used within the architecture in order to check the validity of events issued from the environment to carry out evolution tasks. Second, the translation information is stored explicitly in our architecture, so that it embeds a certain notion of 'mapping'. This component is essential in our database evolution architecture, because when a modification of the conceptual schema is carried out, a new set of elementary translations is determined without it being necessary to apply once again the translation algorithm from scratch. Last, the consideration of MDA concepts have made us to think about the relationship between the logical and extensional information systems of our architecture. In [5] we said that "the logical database schema can be seen as the information base of the logical information system or as the information

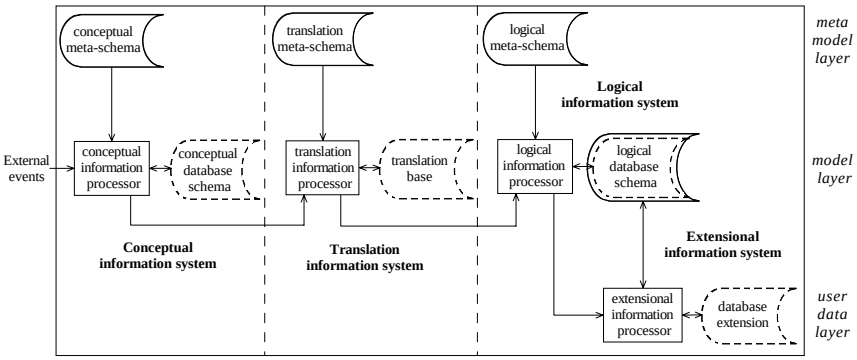


Fig. 2. Architecture for Database Evolution

schema of the extensional one. For this reason two different components of our architecture store the same information". However, this situation led us to define in our architecture "some rules, called *correspondence rules* (in the same sense as in [13]). These rules govern the correspondence between the elements of each one of the two components". The MDA notions suggest that these 'correspondence rules' must play a more relevant role, since they are acting as 'PIM to PSM Mapping'. We think that this point needs of further research.

Finally, it is necessary to point out that we have developed an implementation of the architecture that has allowed us to test its functionality. This implementation is based on the RDBMS Oracle 8i and the Programming Language PL/SQL. In this implementation we use the DBMS as a kind of 'MOF Repository', in a very similar way as described in Chapter 9 of [8]. In particular, Frankel says in that Chapter that "to a MOF Repository, transformations rules are just another kind of metadata that it manages according to the general pattern", and that "one strategy for producing the code to execute the transformations is to create a generator that reads a set of M1 [Model Layer] transformation rules and generates the transformation code that executes the rules on M0 [User-Data Layer] Data". This description corresponds quite accurately with the behavior of our current implementation.

## 4 Conclusions

In this paper we have shown some ideas of an ongoing research about relating MDA and a particular architecture for managing database evolution. We think that the database evolution context can be an interesting and valid application area for MDA, and we have seen several points where both approaches possibly can benefit from each other:

- The experience accumulated along the years by database researchers and practitioners in the fields of schemata translation and evolution can be useful to make advances in the MDA field. For example, the interpretation of our database evolution architecture in MDA terms has led us to propose an enhancement of the MDA metamodel.
- The other way round, taking the MDA ideas into account in the database context can give place to introduce new viewpoints on current database approaches. In our case, the use of the MDA approach has suggested us the need of further research about the notion of 'correspondence rules' within our evolution architecture.
- The database evolution context can take advantage of the advances in the MDA field. In particular, we see the foreseeable development of MDA-based tools as an opportunity for database researchers and practitioners.

## References


[1] Serge Abiteboul, Victor Vianu, Brad Fordham, Yelena Yesha, Relational Transducers for Electronic Commerce, *Journal of Computer and System Sciences*, Vol. 61, N. 2, 2000, 236–269.

[2] L. Al-Jadir, M. Léonard, Multiobjects to Ease Schema Evolution in an OODBMS, in T. W. Ling, S. Ram, M. L. Lee (eds.), *Conceptual modeling, ER-98*, LNCS 1507, Springer, 1998, 316–333.

[3] ANSI/SPARC Report, *ACM SIGMOD Newsletter*, Vol. 7, N. 2, 1975.

[4] Common Warehouse Metamodel Specification Version 1.1, OMG Document formal/03-03-02.

[5] E. Domínguez, J. Lloret, M. A. Zapata, An architecture for Managing Database Evolution, *Proceedings of ER 2002 Workshop on Evolution and Change in Data Management*, To appear in LNCS, 2002, 64–75.

[6] E. Domínguez, M. A. Zapata, J. J. Rubio, A Conceptual Approach to Meta–Modelling, in A. Olivé, J. A. Pastor (Eds.), *Advanced Information Systems Engineering, CAISE'97*, LNCS 1250, Springer, 1997, 319–332.

[7] R. A. Elmasri, S. B. Navathe, *Fundamentals of Database Systems (3rd ed.)*, Addison-Wesley, 2000.

[8] David Frankel, *Model Driven Architecture – Aplying MDA to Enterprise Computing*, Wiley Publishing, 2003.

[9] Michalis Glykas, George Valiris, Formal methods in object oriented business modeling, *Journal of Systems and Software*, Vol. 48, N. 1, 1999, 27–41.

[10] M. Gogolla, A. Lindow, M. Richters, P. Ziemann, Metamodel Transformation of Data Models, *Workshop in Software Model Engineering, Dresden, Germany, http://www.metamodel.com/wisme-2002/*, 2002.

[11] J. L. Hainaut, V. Englebert, J. Henrard, J. M. Hick, D. Roland, Database Evolution: the DB-MAIN approach, in P. Loucopoulos (ed.), *Entity-Relationship approach- ER'94*, Springer Verlag, LNCS 881, 1994, 112–131.

[12] ISO/IEC 9075-2: 1999, *Information Technology - Database languages - SQL - Part 2: Foundation (SQL/Foundation)*, 1999.

[13] J. R. López, A. Olivé, A Framework for the Evolution of Temporal Conceptual Schemas of Information Systems, in B. Wangler, L. Bergman (eds.), *Advanced Information Systems Eng., CAiSE 2000*, Springer, LNCS 1789, 2000, 369–386.

[14] J. Miller, J. Mukerji (eds.), Model Driven Architecture (MDA), Object Management Group, Document number ormsc/2001-07-01, July 9, 2001.

[15] Bernard Morand, Models transformations: from mapping to mediation, *Workshop in Software Model Engineering, Dresden, Germany, http://www.metamodel.com/wisme-2002/*, 2002.

[16] A. S. da Silva, A. H. F. Laender, M. A. Casanova, An Approach to Maintaining Optimized Relational Representations of Entity-Relationship Schemas, in B. Thalheim (ed.), *Conceptual Modeling- ER'96*, Springer Verlag, LNCS 1157, 1996, 292–308.

[17] C. Turker, Schema Evolution in SQL-99 and Commercial (Object-) Relational DBMS. In: H. Balsters, B. De Brock, S. Conrad (eds.), *Database Schema Evolution and Meta-Modeling, Post-Proceedings of the 9th Int. Workshop on Foundations of Models and Languages for Data and Objects*, LNCS 2065, 2001, 1–32.

[18] C. Turker, M. Gertz, Semantic Integrity Support in SQL:1999 and Commercial (Object-) Relational Database Management Systems, *The VLDB Journal* Vol. 10, No. 4, 2001, 241–269.