MDA based approach towards Design of Database for Banking System

Harsh Dev
Phd,Professor, Department of CSE
Pranveer Singh Institute of Technology,
Kanpur U.P., India

Amit Seth
Research Scholar, Department of CSE
Kanpur Institute of Technology,
Kanpur U.P., India

ABSTRACT
The Indian Banks collects huge amount of data which the banks are unable to turn into fast transaction and also unable to maintain and utilize it in a proper manner. Other problems are also occurring w.r.t. the reusability of the system in a different environment designed by the developer. Nowadays we are focusing upon interoperability of database and system hardware. In this paper we have proposed a new design of banking Database system of a bank using the modern MDA approach of software engineering to improve the maintainability, portability and flexibility. We have shown that MDA approach also provides interoperability, easy maintenance of highly computational business oriented system by taking the study of banking database. In this paper we have shown how database modeling is done with MDA for modelling at different levels like CIM, PIM, and PSM, which gives a new dimension to the database based application development.

Keywords
Model-driven development (MDD), Model Driven architecture (MDA), Entity Relationship (ER) Diagram, Platform independent models (PIM), Computation Independent Models (CIM), Platform specific models (PSM).

1. INTRODUCTION
While it is feasible to build much more complex and larger database-based application systems, we are struggling with two major problems: development speed and costs. Systems are never built using only one technology and systems always need to communicate with other systems. With each new technology, much work needs to be done again and again. Furthermore, there is the problem of continuously changing requirements [1].

Several recent papers have already addressed some similarities between database technology and MDA concepts ([1, 2, 3, and 4]). However it is far from being clear how should database engineering concepts be reinterpreted in terms of MDA.

The requirements of a database do not remain constant during its life time and therefore the database has to evolve in order to fulfill the new requirements. Since database evolution activities consume a large amount of resources [5], they are considered of great practical importance and, as a consequence, much research has been focused on analyzing ways of facilitating this task [6, 7]. In particular, among the several problems that are related to evolution activities [8], one of the most important is that of 'forward database maintenance problem' (or 'redesign problem', according to [7]). This problem faces how to reflect in the logical and extensional schemas 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 [7,5]), authors of the [9] have presented an architecture for managing database evolution.

Another problem in Database System is productivity, w.r.t. transforming the similar database design in different type of database. Developers are mainly focused on high-level CIM and PIM models. Maintenance of database, portability and different database transformation are also the main issues. In our work we have focused on the Interoperability in real world problems in which most system need to communicate with the other. Reusability of database design is also the main issue for other similar type of database.

Each software product has to satisfy two kinds of requirements: functional, and nonfunctional. The same relates to the database which is a part of a software system. As database is a typical part of many software systems, many questions arise, for example: Can an MDA be used effectively for database design? What kind of (non-) functional requirements can be delegated to database management system? How to formalize nonfunctional requirements? How they influence the physical model of the database? [2].

In this paper we have shown how to implemented database designing process based on MDA approach.

Model Driven Architecture (MDA) is a new methodology proposed by OMG. MDA provides an open, vendor-neutral approach to the challenge of business and technology change. Based firmly upon OMG's established standards, MDA aims to separate business or application logic from underlying platform technology. Platform-independent applications built using MDA and associated standards can be realized on a range of open and proprietary platforms, including CORBA, J2EE, .NET, and Web Service or other Web based platforms. Fully-specified platform-independent models can enable intellectual property to move away from technology-specific code, helping to insulate business applications from technology evolution, and further enable interoperability [10].

Organizations can benefit from the application of the model-driven approach to the existing activities of the software development life cycle, such as requirements gathering, business analysis, process modeling, systems design, service definition, integration, solutions design, code generation, automatic transformations, etc. [11]

The Model Driven Architecture (MDA) is meant to facilitate system development by using models for representing both the "problem" and its "solution". In its ideal form, software development based on MDA would follow a development process that begins by producing models of the problem domain at a high level of abstraction, and then proceeds by gradually and automatically transforming them into executable code with the help of tools [12].

1.1 Model-driven development (MDD)
Model-driven development (MDD) represents an approach to system engineering where models are used in the understanding, design, construction, deployment, operation,

maintenance and modification of software systems. Model transformation tools and services are used to align the different models, ensuring that they are consistent across e.g. different refinement levels. Model-driven development in represents a business-driven approach to software systems development that starts with a computation independent model (CIM) describing the business context and business requirements. The CIM is refined to a platform independent model (PIM) which specifies services and interfaces that the software systems must provide to the business, independent of software technology platforms. The PIM is further refined to a platform specific model (PSM) which describes the realization of the software systems with respect to the chosen software technology platforms.[13]

1.2 Model-driven Architecture (MDA)

We propose here a collective lifecycle for MDA-based software development that can be used as a basis for constructing MDA-based methodologies.

The phases and activities of the proposed lifecycle are described here in different levels of abstraction. In the following Fig 1 we have shown the process of specifying a system independently of the software execution platform to that of transforming the system specification into one for a particular software execution platform.

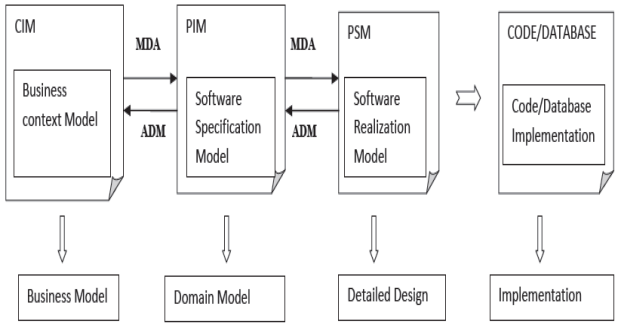

Fig 1: MDA process used in database banking

Banking database design is difficult to understand and also difficult to handle technical complexity found in the system due to which problems of maintainability and reusability persists for database system. We have designed database using MDA approach to enable the specific model. In Fig 1 we have shown the database design process using MDA. The MDA separates the bank database in the level of abstraction**.** The first level CIM describes requirements of the system which defines Business model. The second level PIM describe the software specifications which defines the domain model of the system. The third level PSM describe the software realization model which defines detailed design of the system. The resulted database design is implemented for a particular required platform.

In addition to the business-driven approach, a model-driven framework should also address how to integrate and modernize existing legacy systems according to new business needs. This approach is known as architecture-driven modernization (ADM) in the OMG.[13]

1.3 Existing MDA Technologies and Standards

MOF (Meta Object Facility): Meta-modelling language, repository interface (JMI), interchange (XMI)

UML (XML Metadata Interchange): Standard modelling language .Instance of the MOF model. For developers and "meta-developers"

CWM (Common Warehouse Metamodel): Modelling languages for data warehousing applications. (e.g. Relational DBs)

QVT (Queries/View/Transformations): Transformations definition language. Also for Queries and Views of models.

SPEM (Software Process Engineering Metamodel): metamodel and a UML profile used to describe a concrete software development process.

XMI (XML Metadata Interchange): XMI 2.1 is a format to represent models in a structured text from. In this way UML models and MOF metamodels may be interchanged between different modelling tools.

2. MDA IMPLEMENTED IN DATABASE DESIGN

The process of software development usually involves a database design. Traditionally, during database design three different models of database are built: conceptual, logical and physical. Conceptual model represents a modeled domain with additional constraints that expressed static and dynamic integrity demands. Logical model also represents the modeled domain but in terms of a chosen data model. In the paper the relational data model is assumed. [2] says that physical model is a data model tailored to a given database system and all the models may be expressed in terms of UML language.

MDA involves a collection of models, where every subsequent model is developed on the basis on its preceding model. There are many similarities between traditional process of database design and MDA, so it means that MDA may be applicable for that purpose [2].

MDA introduces three types of models representing different abstraction levels, i.e.: Computation Independent Model (CIM), Platform Independent Model (PIM), and Platform Specific Model (PSM) [15]. So that it can be easily maintained and adapted in different environment. Functional requirements (FRs) capture the intended behavior of the system or specify what the system will do. This behavior may be modeled as services, tasks or functions the system is required to perform. Non-functional requirements (NFRs) define required system properties such as performance, security, maintainability, etc[2]. External quality characteristics include: functionality, reliability, usability, efficiency, maintainability and portability. Each of them is subdivided in several sub-characteristics. These sub-characteristics can be measured by well-defined metrics [26].

Our proposed design separates the levels of abstraction which is easier to create, develop, reuse and maintain. Since we can test the validity of our models at every level of abstraction in MDA therefore the validation and verification of models become easier.

This paper explores that developers can develop database system quickly because they have a reliable high-level graphical view of the system and can achieve lower maintenance costs because of different levels of abstraction of the database.

In [2] the authors have done the Feasibility Analysis of MDA-based Database Design. In [1] the authors only explained the MDA approach improves the development quality of traditional database based application development. In this paper we have shown how the MDA approach is implemented in database using the study of banking database system and also shown that the proposed design have number of advantages.

2.1 Computation Independent Models (CIM)

In MDA, system requirements are modeled using a Computation Independent Model (CIM). This model is called business model and it uses a vocabulary that is familiar to the domain experts. A CIM does not show details of the systems structure, but the environment in which the system will operate, being useful to understand the problem [15]. In [14] the authors have defined the scope of the software system through problem domain analysis and also the unambiguous black-box definition of the system, its objectives, and its scope have been produced [14].

In the following Fig 2 we have represented the banking system of in the form of CIM model. Here we have shown all entities related to the customer and bank. Entities are shown in grey box with their attributes in white box. This also shows relation between them(one entity in relation with another entity). The CIM hides structural details of the system. The presented model can do the computation of the business logic like addition, subtraction or any complex calculation from different type of problem of different bank. So in this way we can easily understand and handle the problem requirement models. This clearly supports reusability and maintainability.

In Fig 2 below we have shown the UserLogins(entity) with their attribute are UserloginID, UserLogin and UserPassword(attributes) related to another entity TransactionLog(entity) with their attributes are TransactionID, TransactionDate,TransactionType, TransactionAmount, NewBalance, AccountID, CustomerID, EmpolyeeID and UserLoginID both entity are shown with their relationship. UserLogins may have zero or one to many TransactionLog. CIM hides the details of data type of the attributes. Even it does not show the primary and foreign keys.

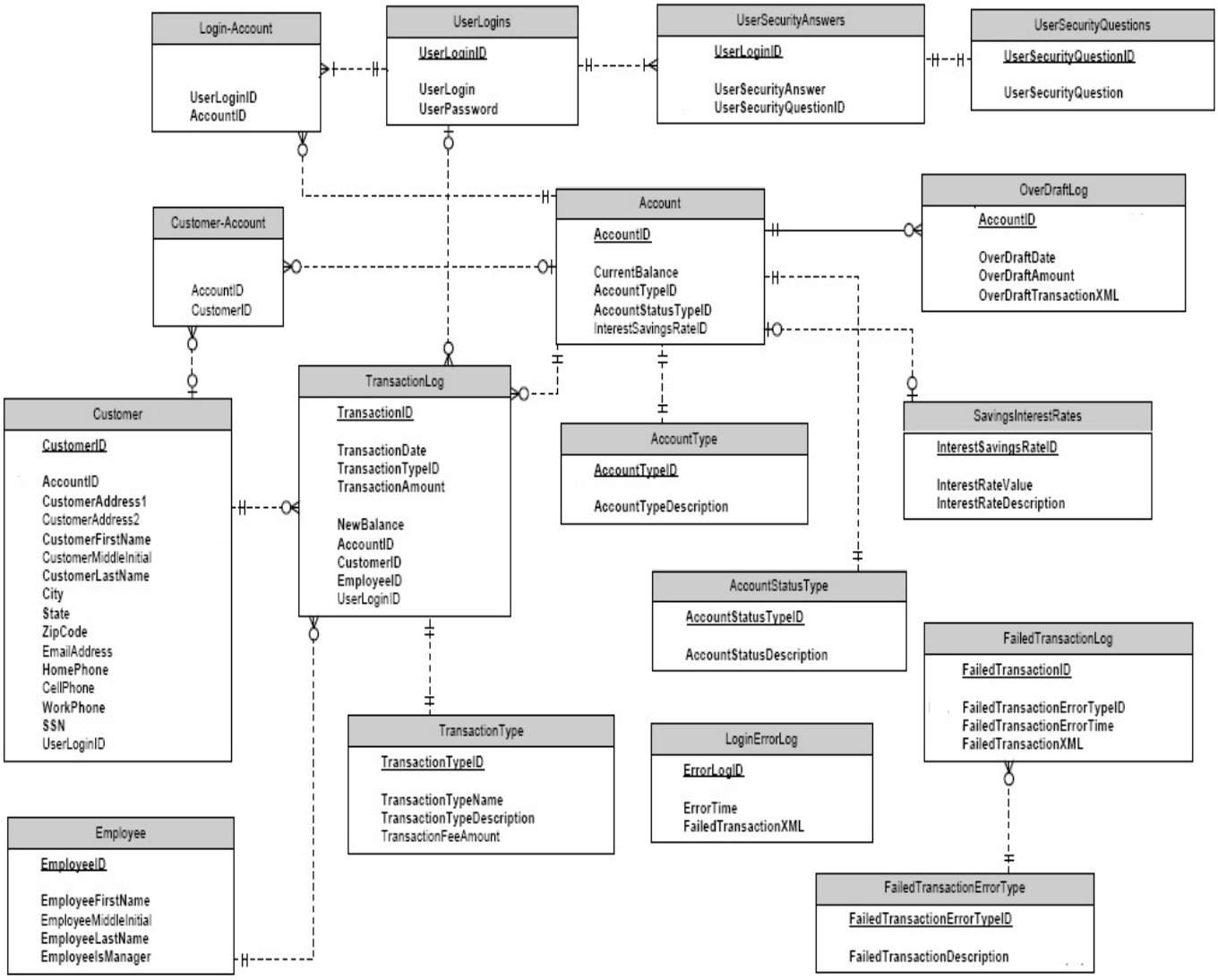

Fig 2: ER Digram Presented in CIM

2.2 Platform independent models (PIM)

In the second level of abstraction authors [15] found the Platform Independent Model (PIM), which is a model with a relatively high abstraction level, which is independent from any implementation technology.

A platform independent analysis model is defined through analyzing the requirements model. System functionalities are described in the analysis PIM while maintaining traceability to the requirements model. Developers may use appropriate model elements stored in a model repository to produce some parts of this PIM. This model is not the final PIM, but forms the foundation for producing the final version. Conventional OO analysis techniques can be used for this activity, which is typically executed in an iterative and incremental fashion [14].

In Fig 3 PIM captures database information implementation-independent system model and business process model.

A platform independent model of database is a view of the system from a viewpoint of platform independence. It hides detail design of database system. It is also representing platform independent information. It carries more detail than that of CIM. It also represents the primary key and foreign key of the entity. Like Customer entity have Primary key (PK) UserLoginID. PIM mdoel presented here also defines the datatype of attributes. Following Fig 3 also shows that PIM CustomerID and Foreign keys (FK) AcountID and gathers descriptive behavior of the system which shows all the information needed. This shows the platform independent way of describing the domain requirement of the system.

2.3 Platform specific models (PSM)

Developing formal and automatic transformations between models (e.g. PIM-PSM) is the main advantage of MDA [26]. In [16] authors have given transformations following the declarative approach of QVT, thus relations between elements of the metamodels are used for constructing the transformation between models (i.e. PIM and PSM).

Two models PIM and PSM describe the same system in much common way in MDA. To get a PSM from a PIM, different artifacts of the system are mapped from one model to another. Hence, it is necessary to formulate a set of transformation criteria that allows converting a source model e.g. PIM representing one view of the system into the target model e.g. PSM representing another view of the system. If model transformation is described in some formal language and there exists an algorithm for its automatic execution, then it is

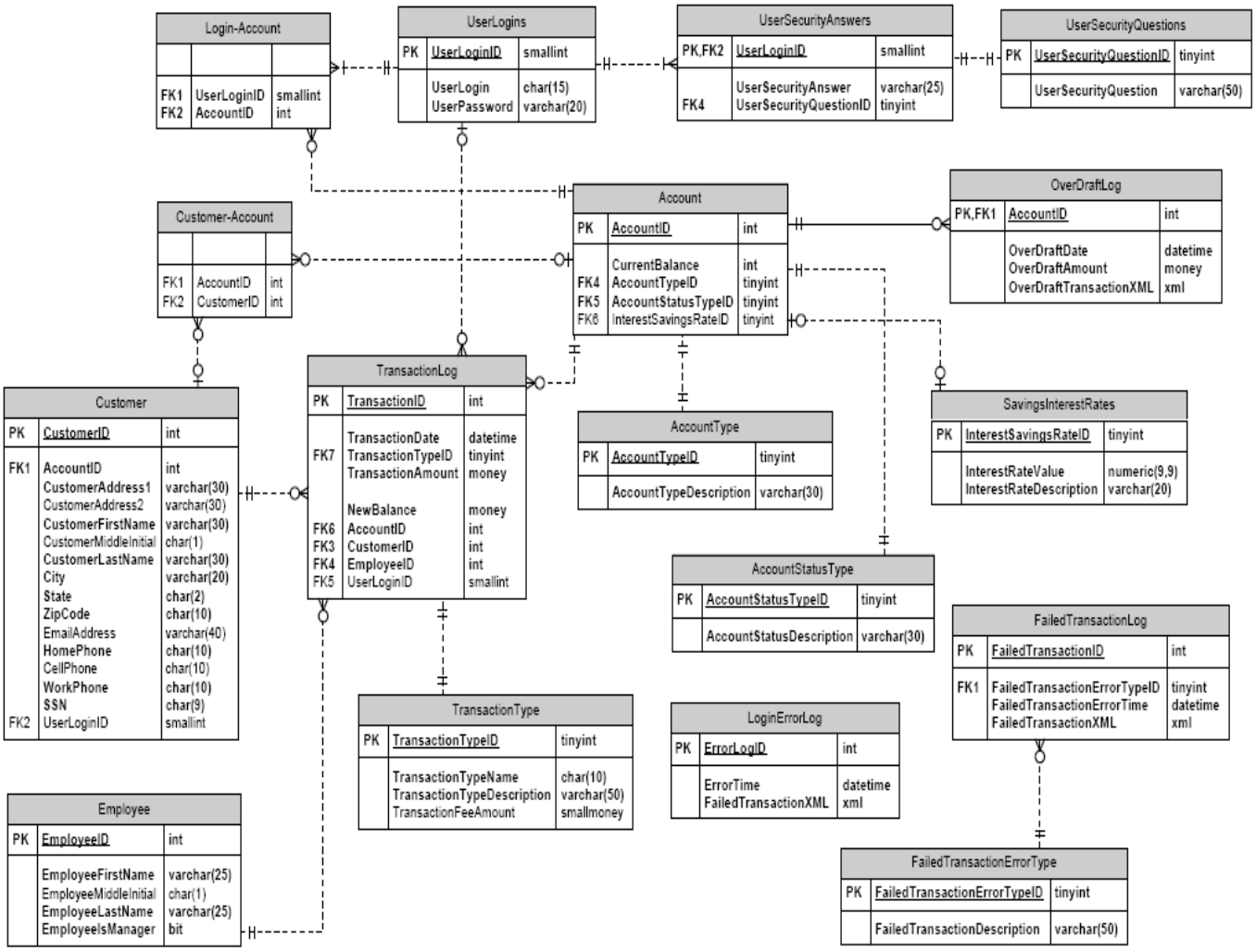

Fig 3: ER Diagram Design Presented in PIM

A platform specific model is a view of the system from the platform specific viewpoint. A PSM combines the specifications in the PIM with the details that specify how the system uses a particular type of platform. The PSM represents the PIM taking into account the specific platform characteristics [13].

In the following Fig 4 we have shown PIM transformation into PSM using transformation specification. There are some transformation tools, such as Mia-Transformation, which performs model-to-model transformation and Mia-Generation [20], which performs model to code transformation. Here we have shown that the specific language can be used for transformation. PSM can represent all quires, procedures and triggers which is implemented in platform specific system.

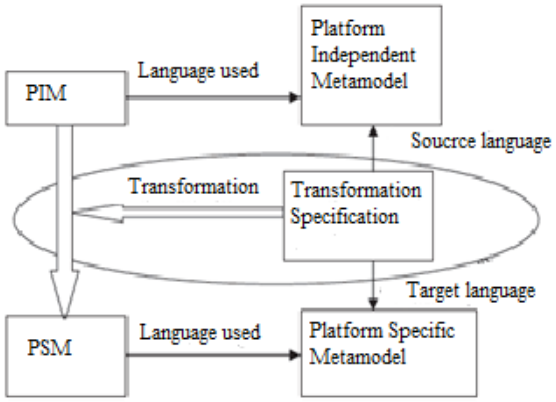

Fig 4: Transformation PIM to PSM

Some transformations are applicable to some special application purposes while others may serve multiple domains. Hence domain of application also forms the basis for classification of transformation approaches. As is the case with FUJABA (From Uml to Java And Back Again), it aims at and strives to achieve the conversion from UML diagrams to standard Java source codes and vice versa, the process in turn referred to as round-trip engineering [18]. Graphical and textual languages have been used in PROGRESS (PROgrammed Graph REwriting System) to bring about the ascribed graph structures and graph transformations [19].

3. CONCLUSION

In this paper we have applied MDA in traditional database based application developments. MDA gives database development a new thought. We conclude that MDA approach provide following advantages in designing the database system.

- Using MDA we can represent system specifications, which separates the specification of functionality from specification of the implementation therefore it reacts swiftly to the changing functional and technological requirements.
- In this paper we have represented database design for software development using MDA based approach, which will expedite the development time.
- We have presented hare that database can be applied to the same PIM to produce different type of PSM. Thereby increasing the reusability of design in terms of implementation in different database application.
- Proposed model separates the system into highly cohesive components which are easier to create, develop and maintain. Thereby it increases the developer's productivity in terms of generation of design of different databases from same type of metamodels.
- MDA approach efficiently represent the three type of model representing different abstraction level (CIM, PIM and PSM), which supports the portability of designed system.
- Our proposed model will also enhance the validation & verification process at different level of abstraction.

4. FUTURE SCOPE

The above work can be extended for the design of multi-dimensional databases for handling complex databases such as multimedia database. It can also be used to model the mobile based data mining system.

5. ACKNOWLEDGMENTS

I especially extend my sincere and grateful thanks to **Associate Prof. Mrs. Shubha Jain**, Head of the Department of Computer Science Department and **Assistant Prof. Mr. Akhilesh Kumar Yadav** Kanpur Institute of Technology, GBTU University, Kanpur, Uttar Pradesh, India.

6. REFERENCES

[1] Li.B., Liu.S, Yu.Z. 2005. The 9th International Conference on Computer Supported Cooperative Work in Design Proceedings, "Applying MDA in Traditional Database-based Application Development".

[2] Dubielewicz, I., Hnatkowska, B., Huzar, Z., Tuzinkiewicz, L .2006. Proceedings of the International Conference on Dependability of Computer Systems , IEEE" Feasibility Analysis of MDA-based Database Design"2006.

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

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

[5] L´opez, J. R., Oliv´e A. 2000. A Framework for the Evolution of Temporal Conceptual Schemas of Information Systems, in B. Wangler, L. Bergman (eds.), Advanced Information Systems Eng., CAiSE, Springer, LNCS 1789, 369–386.

[6] Al-Jadir, L., L´eonard. M. 1998. 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, 316–333.

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

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

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

[10] OMG. 2004. "How systems will be built" http://www.omg.org/mda/,2004.

[11] Guttman, M. and Parodi, J. 2007. Real-Life MDA: Solving Business Problems with Model Driven Architecture. San Francisco, CA: Morgan Kaufmann Publishers.

[12] Gholami, M.F, Ramsin,R.2010. International Conference on Intelligent Systems, Modelling and Simulation, Strategies for Improving MDA-Based Development Processes.

[13] OMG, "OMG Model Driven Architecture", Object Management Group (OMG). http://www.omg.org/mda .

[14] Asadi, M., Ravakhah, M., Ramsin,R.,Second.2008. Asia International Conference on Modelling & Simulation "An MDA-based System Development Lifecycle", IEEE.

[15] Miller, J., Mukerji, J.2003. MDA Guide Version 1.0.1., Object Management Group.

[16] OMG .2002.2nd Revised Submission: MOF 2.0 Query/Views/Transformations. http://www.omg.org/cgi-bin/doc?ad/05-03-02

[17] Singh, Y., Sood, M.2009. "Models and Transformations in MDA," cicsyn, pp.253-258, 2009 First International Conference on Computational Intelligence, Communication Systems and Networks, IEEE computer society.

[18] "Public Domain case tool for UML" http://www.fujaba.de, Sep15, 2008.

[19] Schurr, A. 1990 "PROGRESS: A VHL-Language based on Graph Grammars," Proc. of the 4th International Workshop on Graph Grammars and their application to Computer Science, pp.641-659.

[20] ISO/IEC 9126-2:2002(E), Software engineering – Product quality – Part 2: External metrics.