Cent. Eur. J. Comp. Sci. • 4(3) • 2014 • 150-159
DOI: 10.2478/s13537-014-0218-1

VERSITA

Central European Journal of **Computer Science**

Database reverse engineering based on meta-models

Research Article

Sonja Ristić*, Slavica Aleksić†, Milan Čeliković‡, Vladimir Dimitrieski§, Ivan Luković¶

University of Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia

Received 28 February 2014; accepted 03 July 2014

Abstract: Reengineering is one of the key concepts in software maintenance and evolution. It generally includes some form of reverse engineering followed by some form of forward engineering or restructuring. In the paper we focus on database reverse engineering. Model-driven software engineering promotes the idea of abstracting implementation details by focusing on: models as first class entities and automated generation of models or code from other models. In the approach meta-models are used to define the modeling languages. A database reverse engineering process can benefit of integrating meta-modeling and meta-models in the process. The plethora of models related to databases points out to the need and importance of model-to-model transformations between these models at different abstraction levels. These transformations are based on meta-models that are conformed by the source and target models of the transformations. A database reverse engineering process can be performed through a chain of model-to-model transformations based on a set of meta-models. In the paper we discuss the importance of meta-modeling in the context of database reverse engineering and present a case study illustrating an approach to database reverse engineering.

Keywords: database reverse engineering • model-driven software engineering • meta-models • intensional database meta-models.

1. Introduction

Reengineering is one of the key concepts in software maintenance and evolution. It generally includes some form of reverse engineering followed by some form of forward engineering or restructuring. Chikofsky and Cross define reverse engineering as the process of analyzing a subject system aimed to: i) identify the system's components and their interrelationships and ii) create representations of the system in another form or at a higher level of abstraction [1]. Previous researches on information system (IS) reengineering have made great achievements concerning: the forward engineering, the identifying of system's components and their interrelationships and the creation of representations of

* *E-mail: sdristic@uns.ac.rs (Corresponding author)*
† *E-mail: slavica@uns.ac.rs*
‡ *E-mail: milancel@uns.ac.rs*
§ *E-mail: dimitrieski@uns.ac.rs*
¶ *E-mail: ivan@uns.ac.rs*

a system in another form. But, they have not yet been successful enough in creating representations of the system at a higher level of abstraction.

Model-driven approaches to software development increase the importance and power of models. Model is no longer just a bare image of a system taken at the end of design process used mostly for the communication and documentation purposes. Model-driven software engineering (MDSE) promotes the idea of abstracting implementation details by focusing on: models as first class entities and automated generation of models or code from other models.

Each model is expressed by the concepts of a modeling language. Meta-models are used to define the modeling languages. In that way, a meta-model of a modeling language defines a set of valid models expressed by the concepts of that language. These models are said to be conformant to the meta-model. OMG's Model-Driven Architecture (MDA) [2] currently is the most mature formulation of MDSE paradigm. The OMG's Meta Object Facility (MOF) defines the metadata architecture that lies at the heart of MDA[1]. MOF is used to define semantics and structure of generic meta-models or domain specific ones.

A complex system may consist of many interrelated models organized through different levels of abstraction. Each of them is conformant to a meta-model. A chain of model-to-model (M2M) transformations should be completed starting from an initial model at the highest level of abstraction (Platform Independent Model, PIM), through the less abstract models, with different levels of platform specificity (Platform Specific Models, PSMs), and resulting in an executable program code that represents a model at the lowest level of abstraction (fully PSM). These M2M transformations transform a model conformant to a meta-model into another one conformant to a different meta-model. An MDSE process should cover: modeling that produces description models from existing systems; forward engineering that produces specification and prescription models; and reverse engineering that produces description models from engineered software systems. Favre [3] emphasizes the importance of reverse engineering and its integration with forward engineering in MDSE process to support a smooth evolution of software.

A reverse engineering process (and consequently a whole reengineering process, too) can benefit of integrating the meta-modeling and meta-models in the process.

In the paper we focus on database reverse engineering. Relational or object-relational (OR) databases are a common source of reverse engineering. Starting from physical database schema, recorded into relational database schema (RDBMS) data repository, the conceptual database schema (entity-relational (ER) or extended entity-relational (EER) database schema e.g.) or logical database schema (based on the relational data model e.g.) could be extracted. Both of them are at a higher level of abstraction then the baseline physical database schema. The plethora of models related to databases points out to the need and importance of M2M and model-to-text (M2T) transformations between these models at different abstraction levels. These transformations are based on meta-models that are conformed by the source and target models of the transformations.

In the paper we discuss the importance of meta-modeling in the context of database reverse engineering. Apart from the Introduction and Conclusion, the paper has four sections. In Section 2 a review of different database models important in database reengineering process is given. A database reverse engineering scenario is presented in Section 3. An example of a M2M transformation may be found in Section 4. Related work is elaborated in Section 5.

2. Database models

Software development process produces numerous models of complex application artifacts. In the paper we focus on models relating to databases. For these models we use a generic name **database models**.

A database is a collection of related data stored on some storage medium controlled by the database management system (DBMS). DBMS supports data abstraction so that different users can use data at preferred level of detail. The description of database that is specified during database design is called **database schema**. A **data model** provides the means to achieve data abstraction and to express database schema. According to Date and Darwen definition [4] revised by Eessaar [5]: "A data model is an abstract, self-contained, implementation-independent definition of elements of a 4-tuple of sets (T, S, O, C) that together make up the abstract machine with which database users interact, where T is a

[1] *Meta-Object Facility, [Online] Available: http://www.omg.org/mof/. (retrieved February, 2014)*

set of data types; S is a set of data structure types; O is a set of data operation types; C is a set of integrity constraint types." Numerous data models are proposed. Elmasri and Navathe classify data models according to the types of concepts they use to describe the database structure, as follows: i) high-level or **conceptual data models**; ii) representational or **implementation data models**, and iii) low-level or **physical data models** [6]. ER, EER and object-oriented (OO) data models are conceptual data models and they are used for conceptual database schema design. Relational and OR data models are used predominantly for logical (implementation) database schema design and database implementation. A database schema is a model of a database and has to conform to a data model. A DBMS is based on a data model, too. Hence, there are relational DBMS and OR DBMS, e.g.
Relational data model is a superior logical data model and the acceptance of RDBMSs is widespread, too. Structured Query Language (SQL) is currently available in most commercial and open-source RDBMSs. It is also the focus of a continuous standardization process, resulting in SQL standards (the current revision is: SQL:2011, ISO/IEC 9075:2011). However, issues of SQL code portability between major RDBMS products still exist due to a lack of full compliance with the standard and proprietary vendor extensions. Therefore, even the mapping between SQL database schemas extracted from RDBMS data repository of different vendors may be a serious problem. Forasmuch variety of relational database schema models we classify them as: generic, standard physical and vendor specific physical database schemas. Generic database schemas conform to the theoretical relational data model. Standard physical database schemas conform to a relational data model specified in an SQL standard. A vendor specific physical database schema conforms to a vendor specific data model.
One possible scenario of a database reengineering process could consist of following activities.

1. Reverse engineering activities:
   (a) a physical database schema extraction from a legacy database,
   (b) a generic relational database schema generation from the physical database schema, and
   (c) a platform independent database schema generation from the generic relational database schema.
2. Restructuring and improving of the platform independent database schema.
3. Forward engineering activities:
   (a) new generic relational database schema generation from the restructured and improved platform independent database schema,
   (b) a vendor specific SQL script generation from the new generic relational database schema, and
   (c) execution of generated SQL code under the vendor specific RBMS.

Several M2M and M2T transformations have to be specified to automate the aforementioned scenario. For example, the transformation in step 1.b will be based on meta-model of a generic relational database schema and on meta-model of a vendor specific physical database schema. The transformation in step 1.c will be based on meta-model of a platform independent database schema and on the meta-model of a generic relational database schema.
In our paper [7] we have proposed a classification of database meta-models and distribution of database meta-models and database models across the MOF level stack. An adapted version of the classification, from [8], is presented in Fig. 1.

3. A database reverse engineering scenario

Through a number of research projects lasting for several years, we developed the IIS*Studio development environment aimed at forward IS engineering process. Our approach is mainly based on MDSE and DSL (Domain Specific Language) paradigms. The main idea is to provide the necessary PIM meta-level concepts to IS designers, so that they can easily model semantics in an application domain. Afterward, a number of formal methods and complex algorithms may be utilized to produce database schema specifications and IS executable code, without any considerable expert knowledge. The main purpose of IIS*Studio is to provide a chain of necessary M2M and M2T transformations, so as to generate fully functional transaction programs and applications that are executed over a database. Modeling of a system by IIS*Studio is performed at a high level of abstraction, but with a usage of the concepts that are domain

<table>
<tr><th>MOF level</th><th colspan="5">MOF Architecture</th></tr>
<tr><td>M3</td><td colspan="5">EMOF/CMOF/Ecore</td></tr>
<tr><td rowspan="3">M2</td><td rowspan="2">Data model meta-model (MM)</td><td rowspan="2">Conceptual database schema MM</td><td colspan="2">Implementation database schema MM</td><td>Physical dbS MM</td></tr>
<tr><td>Logical database schema MM</td><td rowspan="2">Standard database schema (dbS) MM</td><td rowspan="2">Vendor-specific Physical dbS MM</td></tr>
<tr><td colspan="2">Generic db schema MM</td></tr>
<tr><td>(1)</td><td>(2)</td><td>(3)</td><td>(4)</td><td>(5)</td></tr>
<tr><td>Relational dm MM,
OR dm MM,
ER dm MM,
OO dm MM</td><td>ER db schema MM,
IIS*Studio db schema MM,
…</td><td>Relational db schema MM,
OR db schema MM
…</td><td>SQL:1999 db schema MM,
SQL:2003 db schema MM …</td><td>Oracle 10g db schema MM,
MySQL db schema,
dBase III+ db schema</td></tr>
<tr><td>M1</td><td>ER db schema MM,
Relational db schema MM, …
SQL:2003 db schema MM …
Oracle 10g db schema MM,
MySQL db schema</td><td>ER db schema 1,
ER db schema 2,
…
IIS*Studio db schema of UniversityDb IS,
IIS*Studio db schema 2 …</td><td>Relational db schema of UniversityDb IS,
Relational db schema 2, …
OR db schema 1 …</td><td>SQL:2003 db schema 1,
SQL:2003 db schema 2,
SQL:1999 db schema 1, …</td><td>Oracle 10g db schema of UniversityDb IS,
MySQL db schema 1,
MySQL db schema 2,
…</td></tr>
<tr><td>M0</td><td>ER db schema 2,
Relational db schema of UniversityDb IS,
SQL:2003 db schema 2,
SQL:1999 db schema 1,
Oracle10g db schema 1,
MySQL db schema 1</td><td colspan="3">Logical data structure of a database</td><td>Database instance</td></tr>
</table>

Figure 1. A classification of database meta-models [8].

specific for IS development. Such software specifications can be used as a sound basis for the generation of DDL (Data Definition Language) database schema specifications and fully executable application prototypes. A detailed description of IIS*Studio modeling concepts can be found in [9] and a MOF based meta-model of IIS*Studio PIM concepts is presented in [10]. The IIS*Studio database meta-model is positioned at M2 MOF level in column (2) of the classification presented in Fig. 1. Here we sketch a database reverse engineering scenario that can be integrated with our IIS*Studio, so as to extend its functionality with support for database reverse engineering process.

Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction. Currently, relational or object-relational databases are at the core of most company's information systems, hosting critical information for the day to day operation of the company. The knowledge captured in them can serve as an important resource in a legacy information system modernization projects and they are a common source of reverse engineering process. Starting from physical database schema, recorded into relational database schema data repository, the conceptual database schema or logical database schema could be extracted. In our approach we start from a legacy relational database and through a process of model extraction, model semantic enrichment and M2M transformations we generate PIM conformant with IIS*Studio meta-model. Once we get an IIS*Studio database schema conformant with IIS*Studio meta-model, IIS*Studio development environment may be used to perform forward engineering process. In the process, an improved database schema and fully functional transaction programs and applications are generated. They are to be executed over a database that is an instance of reengineered database schema.

In Fig. 2 an activity diagram of suggested database reverse engineering process is presented. The first step is model extraction from legacy database (lane 1 in Fig. 2). Let us assume that it is a database instance of an Oracle 10g database schema of University information system. The database instance is positioned at M0 level in column (5) and the Oracle 10g database schema of University information system is at M1 level in column (5) of the database model classification

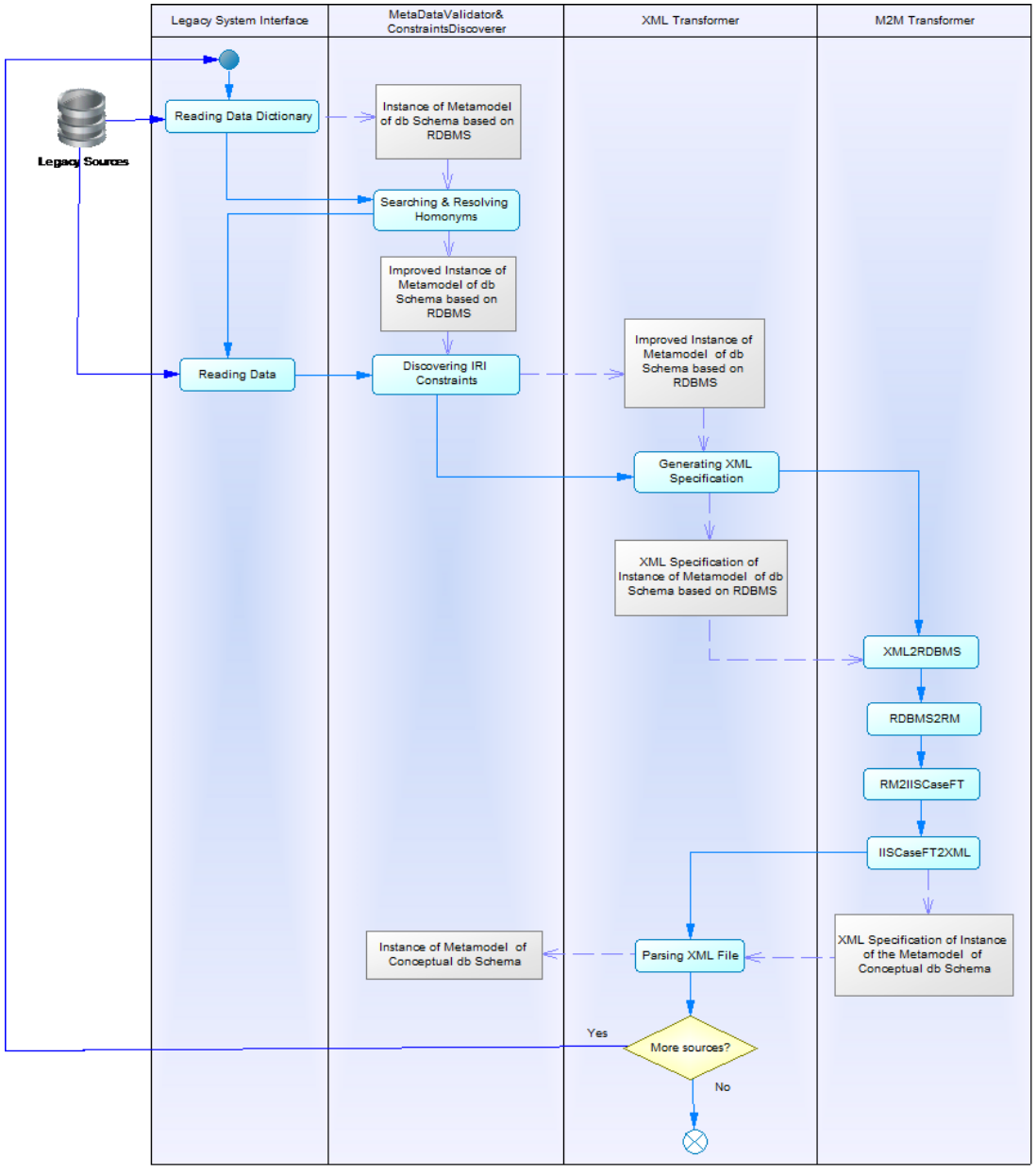


Figure 2. An activity diagram of a database reverse engineering process.

presented in Fig. 1. A physical database schema, recorded into relational database schema data repository is captured into a model conformant with RDBMS meta-model.

At this stage it is possible to improve extracted model through the process of searching and resolving problems with homonyms and synonyms. Additional constraints, which are not specified in the data dictionary, may be discovered in the database instance and specified using concepts of RDBMS meta-model, too. These activities are presented in lane 2 in Fig. 2.

Generally, lane 3 in Fig. 2 is optional. If the activities from lanes 1, 2 and 4 are performed in the same technological space then there is no need for the activities in lane 3 to be executed. In the case of our IIS*Studio the activities from lanes 1 and 2 are implemented in OracleJ Developer environment. On the other side, in the purpose of specifying and managing meta-models we have used the Eclipse Modeling Framework (EMF), Eclipse Juno 4.2.1. and OCL 3.2.1.

Transformations are implemented in ATL IDE (ATLAS Transformation Language Integrated Development Environment), version 3.3.1[2]. Therefore, the activities (M2M transformations) in lane 4 are implemented in a different technological space. The interface between these technological spaces is realized via XML specifications that are created in lane 3. M2M transformations presented in lane 4 are aimed at transformations of a vendor specific physical database schema (an Oracle 10g database schema of *UniversityDb* IS – UNIORA, bolded model in column (5) in Fig. 1, e.g.), via a generic, logical relational database schema (a generic relational database schema of *UniversityDb* IS – UNIRDBS, bolded model in column (3) in Fig. 1, e.g.), into a conceptual database schema (an IIS*Studio database schema of *UniversityDb* IS – UNIIIS, bolded model in column (2) in Fig. 1, e.g.). In the following section an example of an M2M transformation is presented.

4. Case study

In this section an example of UNIORA2UNIRDBS transformation based on database meta-models will be presented. *UnivesityDb* database instance is system under study (SUS) consisting of two tables: *University* with columns *UniId, UniName* and *UniCity*; and *Faculty* with columns *UniId, FacId, FacShortName, FacName* and *Dean*. UNIORA2UNIRDBS transformation transform vendor specific physical database schema into generic, relational database schema. Parts of Oracle 10g database schema meta-model and generic relational database schema meta-model are presented in Fig. 3.

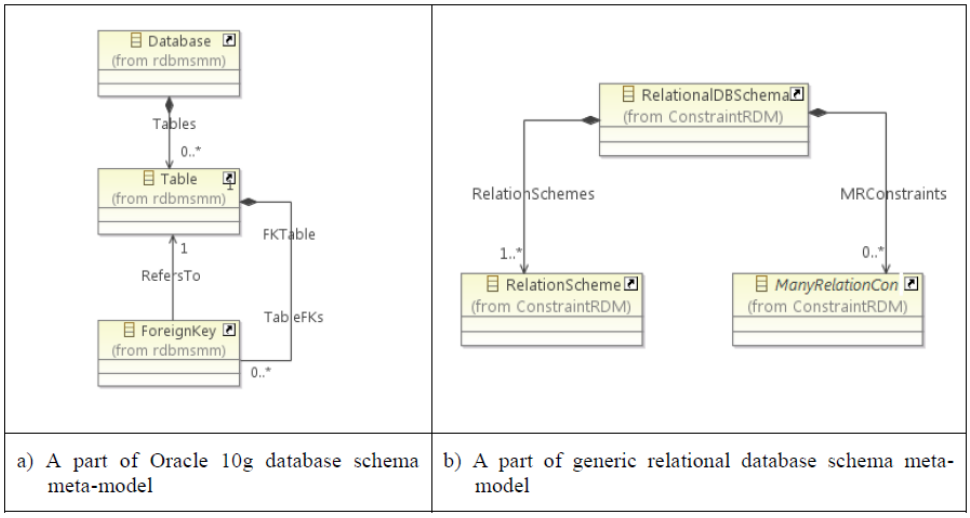


Figure 3. Database schema meta-models.

In Fig. 4 ATL rule for mapping Database onto *RelationalDBSchema* is presented. It uses two ATL helper methods to find foreign key constraints and to find inverse referential constraints. These helper methods are presented in Fig. 5 and Fig. 6, respectively. Fig. 7 contains the source and target models of UNIORA2UNIRDBS transformation.

[2] *Eclipse Foundation, "ATL (ATLAS Transformation Language) Project, ATL/User Guide", available at: http://wiki.eclipse.org/ATL/User_Guide, 2010 (retrieved February, 2014)*

```
rule Database2RDBSchema {
from
   db : RDBMS!Database
to
   out : RM!RelationalDBSchema (
      Name <- db.Name,
      RelationSchemes<-db.Tables,
      MRConstraints <-
      db.getAllFKsForDB().union(thisModule.getCandidateForIRIC()->
                                                collect(e | thisModule.FK2IRIC(e)))
      )}
```

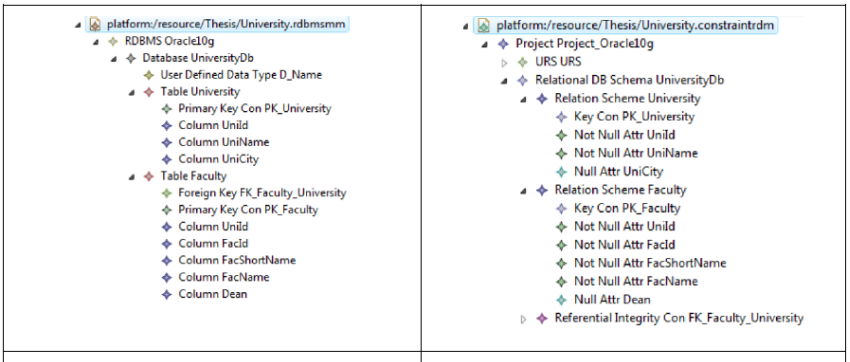
Figure 4. ATL rule for mapping *Database* into *RelationalDBSchema* concept.

```
helper context RDBMS!Database def: getAllFKsForDB() : Set(RDBMS!ForeignKey) =
   self.Tables -> collect(e | e.TableFKs);
```

Figure 5. ATL helper for foreign key constraints finding.

```
helper def: getCandidateForIRIC(): Set(RDBMS!ForeignKey) =
   RDBMS!ForeignKey.allInstances()->select(e| e.InverseReferentialIntegrityCon);
```

Figure 6. ATL helper for inverse referential constraints finding.



a) Model of a part of Oracle 10g database schema

b) Model of a part of generic relational database schema

Figure 7. The source and target models of UNIORA2UNIRDBS transformation.

It can be seen that Database *UniversityDb* concept is mapped into the RelationalDBSchema *UniversityDb* concept. Tables *University* and *Faculty* are mapped into the Relation Schemes *University* and *Faculty*, respectively. Columns are mapped into *Not Null* or *Null Attr* concepts with corresponding names, and *Primary Key Con* concepts into *Key Con* concepts with corresponding names. New concept *Referential Integrity Con* of target meta-model corresponds to *Foreign Key* concept of source meta-model.

5. Related work

Database reverse engineering is widely spread area of research and there is a huge number of references covering it. Hainaut et al. describe main steps of database reverse engineering [11]. According to them extraction of database structure is a process inverse to physical database schema design of forward process, while conceptualization is in a great

extent inverse to logical database schema design. They emphasize that most system forward and reverse engineering activities are transformational in nature, i.e. that they can be modeled as chains of schema transformations. Several authors based their research on reverse engineering descriptions in [11]. Perez et al. [13] and Boronat et al. [12] create OO conceptual database schema from the relational data dictionary.

Atzeni, Cappellari and Gianforme in [14] propose a framework focused on schema mappings. The proposal is based on a relational database formal basis, but the usage of a new meta-meta-model (known-as Supermodel), different from MOF, makes it hard to develop bridges towards the universe of MOF-compliant proposals.

Gogolla et al. [15] have sketched how syntax and semantics of the ER and relational data model and their transformation can be understood as platform independent and platform specific models. Presented ER and relational meta-models are very simple and cannot be classified according meta-model classification presented in our work.

Fidalgo et al. in [16] present meta-model of enhanced ER data model. It can be classified as conceptual database schema meta-model.

Polo, Garcia-Rodriguez and Piattini [17] present the technical and functional descriptions of a tool designed specifically for database reengineering. In the case study they propose simplified relational and object-oriented meta-model. Vara et al. [18] have implemented an ATL model transformation that generates an OR database model from a conceptual data model and a MOFScript model-to-text transformation that generates the SQL code for the modeled database schema. As part of the proposal they have defined a MOF-based Domain Specific Language (DSL) for OR database modeling as well as a graphical editor for such DSL. They presented Oracle 10g meta-model that can be classified as vendor-specific physical database schema meta-model according to the classification presented in our paper.

Lano and Kolahdouz-Rahimi [19, 20] presented case study of UML to relational database model transformation. In the context of relational database schema meta-model presented in our paper the relational database meta-model presented in [19] and [20] is rather simplified and does not differentiate between standard and vendor specific constructs.

In [21] a process is proposed to automatically generate Web Services from relational databases. SQL-92 meta-model has been used to represent the database model, that can be classified as standard physical database schema meta-model according to the classification presented in our paper.

Calero et al. [22] have introduced SQL:2003 meta-model that can be seen as a standard database schema meta-model.

Cabot et. al. present a new method for the analysis of declarative M2M transformations based on the automatic extraction of OCL invariants implicit in the transformation definition [23]. In a case study, they used simplified UML class meta-model, that can be classified as a generic database schema meta-model according to the classification presented in our paper.

Guerra et al. have presented transML, a family of languages to help building transformations using well-founded engineering principles [24]. They presented platform meta-model, meta-model of the specification languages and mapping meta-model. In the paper [5] Eessaar explained why it is advantageous to create meta-model of a data model. He demonstrated that a meta-model could be used in order to find similarities and differences with other data models.

The importance of generic models is also emphasized by Atzeni, Gianforme and Cappellari in [25]. They have shown how a meta-model approach can be the basis for numerous model-generic and model-aware techniques. They presented a classification of data model constructs and their distribution beyond six data models.

Beggar, Bousetta and Gadi [26] propose a reverse engineering process based on MDSE that presents a solution to provide a normalized relational database which includes the integrity constraints extracted from legacy data. They extract entirely the description of legacy data from only source code and physical files. COBOL file section meta-model is proposed, that can be classified as generic file schema meta-model.

The application of categorical structures as expressing tool for differential calculus and its possible application for formal descriptions of models at varying levels of abstraction is discussed in [27]. Alongside with the formal methods presented in [28] and [29] these results may be used to enable formal specification of M2M transformations based on meta-models.

6. Conclusion

The abstract syntax of a modeling language is formalized by means of a meta-model, which also serves as a basis to interchange and transform models. Meta-models are crucial for both model designers and tool developers. In the paper we discuss the importance of meta-models in database reverse engineering process. We proposed a database reverse engineering scenario based on database schema meta-models and presented an M2M transformation between physical

database schema and generic relational database schema. Our classification and distribution of database models across the MOF level stack enables systematic approach for mapping specification between different models/meta-models and development of appropriate M2M or M2T transformations.
In order to do that, corresponding meta-models have to be specified. Some of these meta-models are already designed [7, 30]. We use these meta-models to develop a chain of M2M transformations starting with a legacy relational database schema and to integrate them with our IIS*Studio development environment. The chain of these transformations will enable reverse engineering, while IIS*Studio tool will be used for forward engineering and generating executable application prototypes.
Meta-models that we presented in this paper are intensional meta-models. Our future research has to consider extensional database meta-models, too. Besides, we plan to upraise the level of abstraction and to specify meta-models of different data models as the basis for PIM of M2M transformations between database meta-models conformant with different data models meta-models.

Acknowledgements

Research presented in this paper was supported by Ministry of Education, Science and Technological Development of Republic of Serbia, Grant III-44010, Title: Intelligent Systems for Software Product Development and Business Support based on Models.

References

[1] E.J. Chikofsky, J.H. Cross, Reverse engineering and design recovery: A taxonomy. IEEE Softw. 7(1), 13–17, 1990
[2] J. Mukerji, J. Miller, MDA Guide Version 1.0.1, document omg/03-06-01 (MDA Guide V1.0.1), 2003. http://www.omg.org/, (retrieved February 2014)
[3] J.M. Favre, Foundations of Model (Driven) (Reverse) Engineering: Models, Dagstahl Seminar Proceedings 4101, 2005
[4] C.J. Date, H. Darwen, Types and the Relational Model. The Third Manifesto, 3rd ed (Addison Wesley, USA, 2006)
[5] E. Eessaar, Using Meta-modeling in order to Evaluate Data Models, In Proceedings of: The 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Corfu Island, Greece, February (WSEAS Stevens Point, Wisconsin, USA, 2007) 16–19
[6] R. Elmasri, B.S. Navathe, Database Systems: Models, Languages, Design and Application Programming, Sixth Edition (Pearson Global Edition, USA, 2011)
[7] S. Ristic, S. Aleksic, M. Celikovic, I. Lukovic, An EMF Ecore based relational dB schema meta-model, In: Proceedings of: The 6th International Conference on Information Technology ICIT 2013, Amman, Jordan
[8] S. Ristić, S. Aleksić, M. Čeliković, I. Luković, Meta-modeling in the Context of Database Reengineering, In: Proceedings of: The 12th Conference Informatics'2013 (International Scientific Conference on Informatics), Technical University of Kosice, Slovakia, 2013, 162–167
[9] S. Ristic, S. Aleksic, I. Lukovic, J. Banovic, Form-Driven Application Generation: A Case Study, In Proceedings of: The Eleventh International Conference on Informatics, INFORMATICS'2011, Roznava, Slovakia, November 16 – 18, Technical University of Kosice, Slovakia, 2011, 115–120
[10] M. Čeliković, T. Luković, S. Aleksić, V. Ivančević, A MOF based Meta-Model and a Concrete DSL Syntax of IIS*Case PIM Concepts, Comp. Sci. Inform. Sys. 9(3), 1075–1104, 2012
[11] J.L. Hainaut, J. Henrard, J.M. Hick, D. Roland, V. Englebert, Database design recovery, Eighth Conferences on Advance Information Systems Engineering (Springer, Berlin Heidelberg, 1996) 463–480
[12] A. Boronat, J. Perez, J.A. Cars, I. Ramos, Two Experiences in Software Dynamics, JUCS 10(4), 428–453, 2004
[13] J. Perez, I. Ramos, V. Anaya et al., Data reverse engineering of legacy databases to object oriented conceptual schemas, Electron. Notes Theor. Comput. Sci. 74(4), 1–13, 2002
[14] P. Atzeni, P. Cappellari, G. Gianforme, MIDST: model independent schema and data translation, In Proceedings of: The 2007 ACM SIGMOD international Conference on Management of Data (Beijing, China, June 11 - 14, 2007),

SIGMOD '07 (ACM, New York, NY, 2007) 1134–1136.
[15] M. Gogolla, A. Lindow, M. Richters, P. Ziemann, Meta-model transformation of data models, Position paper. WISME at the UML 2002
[16] R.N. Fidalgo, E. Alves, S. España et al., Metamodeling the Enhanced Entity-Relationship Model, JIDM 4(3), 406–420, 2013
[17] M. Polo, I. Garcia-Rodriguez, M. Piattini, An MDA-based approach for database re-engineering, J. Softw. Maint. Evol. 19(6), 383–417, 2007
[18] J. Vara, B. Vela, V.A. Bollati, E. Marcos, Supporting model-driven development of object-relational database schemas: a case study, In: R. Paige (Ed.), Theory and Practice of Model Transformations (Springer, Heidelberg, Berlin, 2009) 181–196
[19] K. Lano, S. Kolahdouz-Rahimi, Model-driven development of model transformations, Theory and Practice of Model Transformations (Springer, Berlin Heidelberg 2011) 47–61
[20] K. Lano, S. Kolahdouz-Rahimi, Constraint-based specification of model transformations, J. Syst. Software 86(2), 412–436, 2013
[21] R.P. del Castillo, I. García-Rodríguez, I. Caballero, PRECISO: a reengineering process and a tool for database modernization through web services, In: M.J. Jacobson Jr., V. Rijmen, R. Safavi-Naini (Eds.) SAC 2009, LNCS, 5867 (Springer, Heidelberg, 2009) 2126–2133
[22] C. Calero, F. Ruiz, A. Baroni, Brito e Abreu F., Piattini M., An ontological approach to describe the SQL:2003 object-relational features, Com. Stand. Interf. 28(6), 695–713, 2006
[23] J. Cabot, R. Clarisó, E. Guerra, J. De Lara, Verification and validation of declarative model-to-model transformations through invariants, J. Sys. Soft. 83(2), 283–302, 2010
[24] E. Guerra, J. de Lara, D. Kolovos, R. Paige, O. dos Santos, Engineering model transformations with transML, Software and Systems Modeling (Springer-Verlag, Berlin, Heidelberg, 2011)
[25] P. Atzeni, G. Gianforme, P. Cappellari P., A universal meta-model and its dictionary, Large-Scale Data and Knowledge-Centered SystemsÂă1, 38–62, 2009
[26] O.E. Beggar, B. Bousetta, T. Gadi, Getting Relational Database from Legacy Data-MDRE Approach, Computer Engineering and Intelligent Systems 4(4), 10–32, 2013
[27] W. Steingartner, D. Radaković, Categorical structures as expressing tool for differential calculus, In: Proceedings of: The 12th Conference Informatics'2013 (International Scientific Conference on Informatics) Technical University of Kosice, Slovakia, 2013, 77–82
[28] V. Slodičák, Some useful structures for categorical approach for program behavior, Journal of Information and Organizational Sciences 35(1), 99–109, 2011
[29] V. Slodičák, P. Macko, Some New Approaches in Functional Programming Using Algebras and Coalgebras, Electron. Notes Theor. Comput. Sci. 279(3), 41–62, 2011
[30] S. Ristić., S. Aleksić, M. Čeliković, I. Luković, Meta-modeling of inclusion dependency constraints, In Proceedings of: The 6th Balkan Conference in Informatics (BCI '13) (ACM, New York, NY, USA, 2013) 114–121