Proceedings of the Federated Conference on Computer Science and Information Systems pp. 977–984
DOI: 10.15439/2015F224
ACSIS, Vol. 5

A Model-driven Approach to Data Structure Conceptualization

Sonja Ristić
University of Novi Sad,
Faculty of Technical Sciences,
Trg D. Obradovića 6,
21000 Novi Sad, Serbia
Email: sdristic@uns.ac.rs

Slavica Kordić, Milan Čeliković
University of Novi Sad,
Faculty of Technical Sciences,
Trg D. Obradovića 6,
21000 Novi Sad, Serbia
Email: {slavica, milancel}@uns.ac.rs

Vladimir Dimitrieski, Ivan Luković
University of Novi Sad,
Faculty of Technical Sciences,
Trg D. Obradovića 6,
21000 Novi Sad, Serbia
Email: {dimitrieski, ivan@uns.ac.rs}

Abstract— **Reengineering of an existing information system can be carried out: to improve its maintainability, to migrate to a new technology, to improve quality or to prepare for functional enhancement. An important phase of a data-oriented software system reengineering is a database reengineering process and, in particular, its subprocess – a database reverse engineering process. The reverse engineering process contains two main phases: data structure extraction and data structure conceptualization. In the paper we present a blueprint of a model-driven approach to database reengineering process that is one of the results of our research project on model-driven intelligent systems for software system development, maintenance and evolution. Within that process hereinafter we focus on the data structure conceptualization process and propose a model-driven approach to data structure conceptualization. Proposed process is based on model-to-model transformations implemented by means of Atlas Transformation Language.**

I. Introduction

An information system (IS) implemented to fulfill organizational information requirements would adapt to emerging business models and technology changes and innovations. Organizations are facing the problem of maintenance, evolution or even replacing of a part or whole legacy information/software system. A new system can be redeveloped from scratch, but in that case the knowledge captured in the legacy system is lost. Legacy system replacement or reengineering can be done with significantly reduced amount of effort and cost if the conceptual models are reconstructed from them. Summarizing the definitions of reengineering process presented in [1]–[5] it can be concluded that reengineering entails: (i) the reverse engineering activities aimed at creation of a more abstract view of the system; (ii) the restructuring of the abstract view; and (iii) the implementation of the system in a new form by means of forward engineering activities. A majority of software systems may be characterized as data-oriented systems. They are centered around a persistent data structure, like set of files or a database. The reengineering of a data-oriented software system makes up the code reengineering and the database reengineering processes that could be carried out almost independently one of another. In the paper we deal with a phase of database reengineering process.

The persistent structures, primarily databases, are at the core of most company information systems. The knowledge captured in them can serve as an important resource in a legacy information system modernization project and they are a common source of reverse engineering processes. The database reverse engineering (DBRE) is, according to Hainaut *et al*. [6], the process of recovering the conceptual schema of a database.

The emergence of Model-driven Software Engineering (MDSE) enables the building of more effective reverse engineering solutions. Model-driven Reverse Engineering (MDRE) is the application of MDSE principles, methods and tools to the reverse engineering process, relying on: meta-models, models and model transformations. The Model-driven Architecture (MDA) specified by the Object Management Group (OMG) [7] currently is the most mature formulation of MDSE paradigm.

The OMG has noticed a necessity to understand and evolve existing software assets and launched the Architecture-driven Modernization (ADM) initiative [8]. Within our research project on model-driven intelligent systems for software system development, maintenance and evolution, spanning throughout last past years we have developed a model-driven approach to database reengineering process, adhering ADM framework. In this paper we present just a blueprint of that approach using ADM horseshoe as a reference model. After that we focus on the database reverse engineering (DBRE) process within the database reengineering process.

Hainaut *et al*. in [9] have presented the general architecture of the DBRE process divided in two main phases: data structure extraction and data structure conceptualization. During the data structure extraction phase it is hard to fully apply model-driven principles. The main reason is the fact that some activities need user interaction and the process cannot be fully automated. Therefore, in the

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.

paper we propose a model-driven approach to data structure conceptualization phase of database reverse engineering process. It is implemented within the IIS*Studio development environment aimed at evolutive and incremental IS development and reengineering [10]–[14]. Our approach is mainly based on MDSE and Domain Specific Language (DSL [15], [16]) paradigms. The approach is purely platform independent and strictly differentiates between the specification of a system and its implementation on a particular platform. IIS*Case tool, as a part of IIS*Studio is aimed at IS forward engineering, while IIS*REE tool is aimed at IS reverse engineering. Model transformations implemented in these tools are based on several database meta-models and their classification will be presented at the very beginning of the paper.

Apart from Introduction and Conclusion the paper has 5 sections. In Section 2 the classification of IIS*Studio meta-models is presented. A sketch of a model-driven approach to database reverse engineering is proposed in Section 3. Detailed description of a data conceptualization process is given in Section 4. A case study is presented in Section 5, and related work is treated in Section 6.

II. IIS*Studio Meta-Models Classification

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 the generic name **database models**. Numerous data models are proposed and they can be classified 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**. In the context of MDSE data models may be seen as meta-models. A database schema is expressed by means of the concepts of a selected meta-model. According to the MDSE terminology, such database schema is called database model and it should conform to the appropriate meta-model. An EER database model conforms to the EER meta-model, while a relational database model conforms to the relational meta-model.

Common examples of conceptual data models are (Extended) Entity-Relational (ER, EER) data model and Object-oriented (OO) (also called Class) data model. Relational and Object-relational (OR) data models are used predominantly for logical (implementation) database schema design and database implementation.

IIS*Studio uses form-type (FT) data model for conceptual IS and database design. Like EER and OO data models, it is conceptual data model. A *form type* is the main modeling concept in FT (data) meta-model. It generalizes document types, i.e. screen forms that users utilize to communicate with an information system. Using the form type concept, a designer specifies screen or report forms of transaction programs and, indirectly, specifies (i) an initial set of attributes, constraints and future database schema, (ii) basic functionalities of future transaction programs and (iii) components of their user interface. A form type concept, as well as related concepts of a domain and attribute, is platform independent. A form type is a named tree structure, whose nodes are called component types. Each *component type* is identified by its name in the scope of the form type, and has nonempty sets of attributes and keys, and a set of unique constraints that may be empty. Besides, to each component type must be associated a set of allowed database operations. A part of FT meta-model can be seen in the third row of the table presented in Fig. 4. A detail description of FT data model and FT meta-model may be found in [17].

In this paper, we use the MDSE terminology based on the four-layered architecture proposed by OMG common meta-meta-model Meta-Object Facility (MOF) standard [18]. System under study (SUS) is at the M0 level. The concept of a model is specialized depending on the level, in which a model is located: a model at M1 level, a meta-model at M2 level and a meta-meta-model at M3 level. An SUS is represented by a model at M1 level, which conforms to a meta-model at M2 level that is conformant with a meta-meta-model at M3 level. In [19] we have proposed a classification of database meta-models and distribution of database meta-models and database models across the MOF level stack. Adapted version of the classification is presented in Fig. 1 where can be seen different kinds of database meta-models that describe database models at certain abstraction level:

1. generic database schema meta-models:
   - generic conceptual database schema meta-models,
   - generic logical database schema meta-models;
2. standard physical database schema meta-models; and
3. vendor-specific physical database schema meta-models.

<table>
<tr><td></td><td colspan="4">MOF Architecture</td></tr>
<tr><td>M3</td><td colspan="4">EMOF/CMOF/Ecore</td></tr>
<tr><td rowspan="3">M2</td><td rowspan="2">Conceptual database schema meta-model (MM)</td><td colspan="2">Implementation database schema (dbS) MM</td><td>Physical dbS MM</td></tr>
<tr><td>Logical database schema MM</td><td>Standard dbS MM</td><td>Vendor-specific Physical dbS MM</td></tr>
<tr><td colspan="4">Generic db schema MM</td></tr>
<tr><td></td><td>(1)</td><td>(2)</td><td>(3)</td><td>(4)</td></tr>
<tr><td></td><td>EER dbS MM, Class dbS MM, IIS*Case PIM concepts MM</td><td>Relational db schema MM, OR db schema MM …</td><td>SQL:2008 dbS MM, SQL:2011 dbS MM</td><td>Oracle 10g dbS MM, Oracle 11g dbS MM, MySQL dbS</td></tr>
<tr><td>M1</td><td>EER dbS 1, Class dbS 2, IIS*Case Form Type Model of UniversityDb IS</td><td>Relational db schema of UniversityDb IS, OR dbS 1 …</td><td>SQL:2011 dbS 1, SQL:2008 dbS 1 …</td><td>Oracle 10g dbS UniversityDb IS, MySQL dbS 1, MySQL dbS 2</td></tr>
<tr><td>M0</td><td colspan="3">Logical data structure of a database</td><td>Database instance</td></tr>
</table>

Fig. 1 Classification of database meta-models

In each field of the table in Fig. 1, that is on the intersection of column (**i**), where **i** ∈ {1, 2, 3, 4}, and row M**j**, where **j** ∈ {0, 1, 2, 3}, a list of exemplar models is given. In the column (1) at the M2 level there are several conceptual database schema meta-models: EER, OO (Class) and FT meta-model (so-called IIS*Case PIM concepts MM, where PIM stands for Platform Independent Model). Throughout the paper the names FT MM and IIS*Case PIM

concepts MM will be used interchangeably. In the column (1) at the M1 level there are several conceptual database schemas (database models) that are conformant with an appropriate meta-model from the field in column (1) and level M2. For instance: IIS*Case Form Type Model of *UniversityDB* IS is a conceptual database schema of *UniversityDB* information system and it conforms to the IIS*Case PIM concepts MM (FT MM).

It could be noticed that models (meta-models) placed in the same field of the table are at the same abstraction level. In a way, EER database schema meta-model is at the same abstraction level as class database schema meta-model and IIS*Case PIM concepts meta-model. The abstraction level of the models (meta-models) throughout the columns of the same row is decreasing with increasing of the column number **i.** The models (meta-models) placed in column 1 are at the highest abstraction level, while the models (meta-models) placed in column 4 are at the lowest abstraction level. A relational database schema of *UniversityDb* IS is at higher abstraction level than its implementation under the Oracle10g database management system, represented as vendor-specific Oracle 10g database schema of *UniversityDb* IS.

Presented classification is very important for our DBRE approach and for understanding model transformations that are based on classified meta-models. Detailed description of meta-models that we use in this paper may be found in our papers [14], [17] and [19]–[20].

III. A Model-Driven Approach to Database Reengineering Process

Reengineering is one of the key concepts in software maintenance and evolution. It generally includes some form of reverse engineering followed by restructuring (optional) and some form of forward engineering. 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 the system in another form. But, they have not yet been successful enough in creating representations of the system at a higher level of abstraction. 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 this section we present a blueprint of a model-driven approach to database reengineering implemented in IIS*Studio. In Fig. 2 we use ADM horseshoe model to illustrate main steps of proposed approach.

Database reengineering process begins with a legacy database. Relational databases are at the core of most company information systems and that is why, here in Fig. 2, we assume that legacy database is a relational database. Starting from a physical database schema, that is recorded in the relational database schema data repository, the conceptual database schema or logical database schema may be extracted. All of these database schemas represent models at different levels of abstraction.

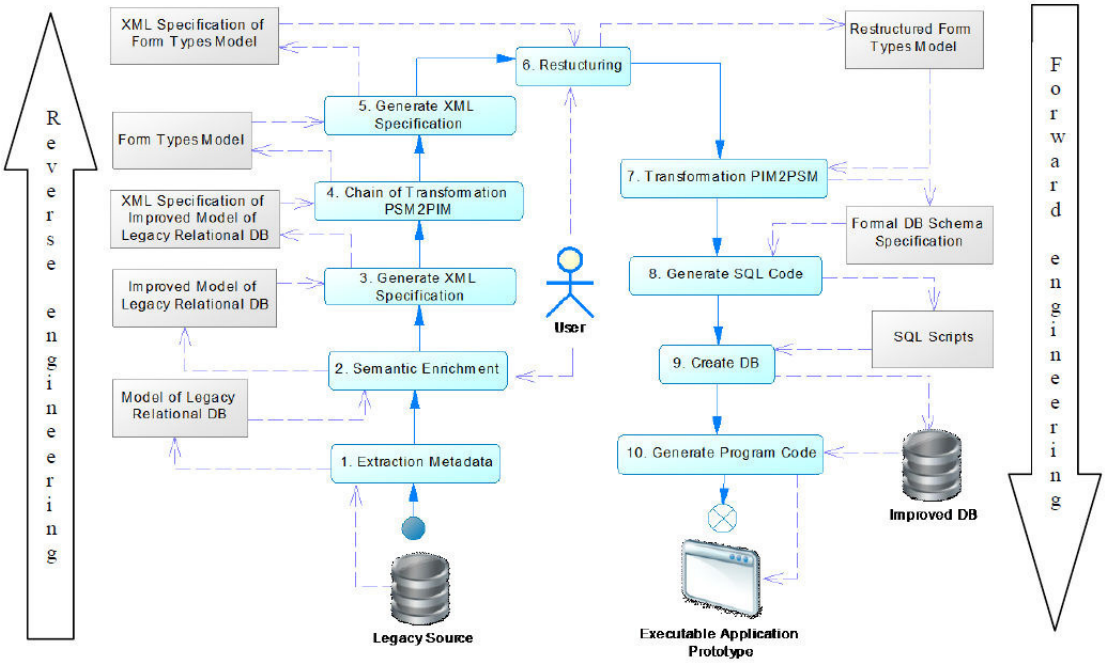

Fig. 2 Horseshoe model of model-driven approach to database reengineering implemented in IIS*Studio

Reverse engineering phase covers steps 1–5, where steps 1–3 belong to the data extraction phase, and steps 4 and 5 make the data conceptualization phase. Step 6 is optional and represents restructuring phase. Forward engineering phase covers steps 7–9. Step 10 is not the step of database reengineering process and it is an activity of code reengineering process.

In the first step information about supported data types, tables, columns, check constraints, primary key and unique key constraints accompanied with foreign key constraints are captured from a legacy database repository and are placed in IIS*REE repository.

Additional information about the inverse referential constraints and homonym inconsistencies are discovered in the step 2 (Semantic Enrichment step). User interaction is required to control data extraction process at this stage. This interaction prevents full automation of data extraction phase and that is the reason why this phase is realized by means of traditional Java programming in OracleJ Developer environment. The similar problem arises again in step 6 (Restructuring step) of the reengineering process because this step requires user interaction, too.

Once it is captured, the physical database schema (model) may be transformed into desired conceptual database model. A designer may choose target conceptual meta-model among EER, Class and FT meta-models. In Fig. 2 FT meta-model is selected. Depending on selected target meta-model a model transformation or a chain of model transformations will be executed. These transformations are based on appropriate meta-models that can be at different levels of abstraction. The distribution of the model transformations and supporting meta-models across MOF and abstraction levels is given and explained in Section IV.

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. Model transformations used in step 4 are implemented in ATL IDE (ATLAS Transformation Language Integrated Development Environment), version 3.3.1 [21]. Hereinbefore is mentioned that steps 1 and 2 are implemented in one technological space (OracleJ Developer), and step 4 in another one (EMF). Step 3 is used to bridge the technological gap and to export captured data by means of XML document conformant with XML meta-model. Step 6 is implemented in the same technological space as steps 1 and 2, due to the fact that it requires user interaction, too. Step 5, like the step 3, is used to bridge the technological gap, but in the opposite direction. Therefore, strictly speaking, it is not part of the data conceptualization phase.

In a forward engineering process, designers start with a high abstraction level model, abstracting from all kinds of platform issues. Through a chain of model-to-model (M2M) transformations, ending up with a model-to-text (M2T) transformation, the initial platform independent model transforms iteratively to a series of models with less degree of platform independency, introducing more and more platform specific extensions. Details of forward engineering process with activities in steps 7–10 are presented in some of ours already published papers, like [10]–[12], and we omit these details here.

In the next section we focus on the data structure conceptualization phase.

IV. Data Structure Conceptualization Phase

A complex system may consist of many interrelated models organized through different levels of abstraction. Each of them is conformant to a meta-model. In a forward engineering process a chain of 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. Conversely, in a reverse engineering process, the abstraction level of models and degree of platform independency are increasing throughout the chain of transformations.

In this section we present the activities of data structure conceptualization phase. Model transformations implemented in the conceptualization phase are presented in a rectangular coordinate system in Fig. 3. The abscissa represents MOF level of a model, and the ordinate represents abstraction level of a model in the context of its platform specificity. The input of the process is XML specification of captured physical database model. This XML specification conforms to XML meta-model. The data conceptualization phase is realized as a chain of three M2M transformations: 1. XML2RDBMS, 2. RDBMS2RM, and 3. RM2IISCase. The fourth M2M transformation presented in Fig. 3 is IISCase2XML transformation and it corresponds to the step 5 in Fig. 2, and, as it was explained in Section III, it is not part of data conceptualization phase.

The first transformation transforms a model conformant with XML meta-model into a model conformant with an SQL standard meta-model. It could be SQL: 2008 dbS meta-model, e.g., listed in bold style in column (3) and row M2 in Fig. 1.

The transformation RDBMS2RM transforms a model conformant with an SQL standard meta-model into a model conformant with generic relational db meta-model (bolded text in column (2), row M2 in Fig. 1). It is not possible to transform a model conformant with an SQL standard directly into a model conformant with FT (IIS*Case PIM concepts) MM. The reason lies in the fact that FT approach to database design is based on the Universal relation schema assumption (URS assumption). Physical database meta-models and database meta-models based on SQL standard do not support URS, while generic relational db meta-model does.

Both of aforementioned transformations are PSM2PSM transformations. The third one, RM2IISCase is a PSM2PIM transformation. It transforms a model conformant with generic relational db meta-model into a model conformant with FT meta-model (so-called IIS*Case PIM concepts MM). It is placed in column (1) and row M2 in Fig. 1. In a way the data conceptualization phase is finished and

conceptual database model is generated from a legacy database. The obtained model is transformed in an XML document by means of the forth transformation IISCase2XML.

It is important to emphasize that model transformations in Fig. 3 are modeled as conformant with ATL meta-model. Each of them is based on two meta-models: source and target meta-model. The source meta-model of a model transformation is at lower abstraction level than the target meta-model of that transformation in a reverse engineering process. At M1 MOF level each transformation execution transforms an input database schema (model) into an output database schema (model). Although having the same name in Fig. 3, relational database schemas that are input models for transformations 1, 2 and 3, respectively, are not the same models and they are conformant with different meta-models. Their platform independency is ascending throughout the chain of model transformations.

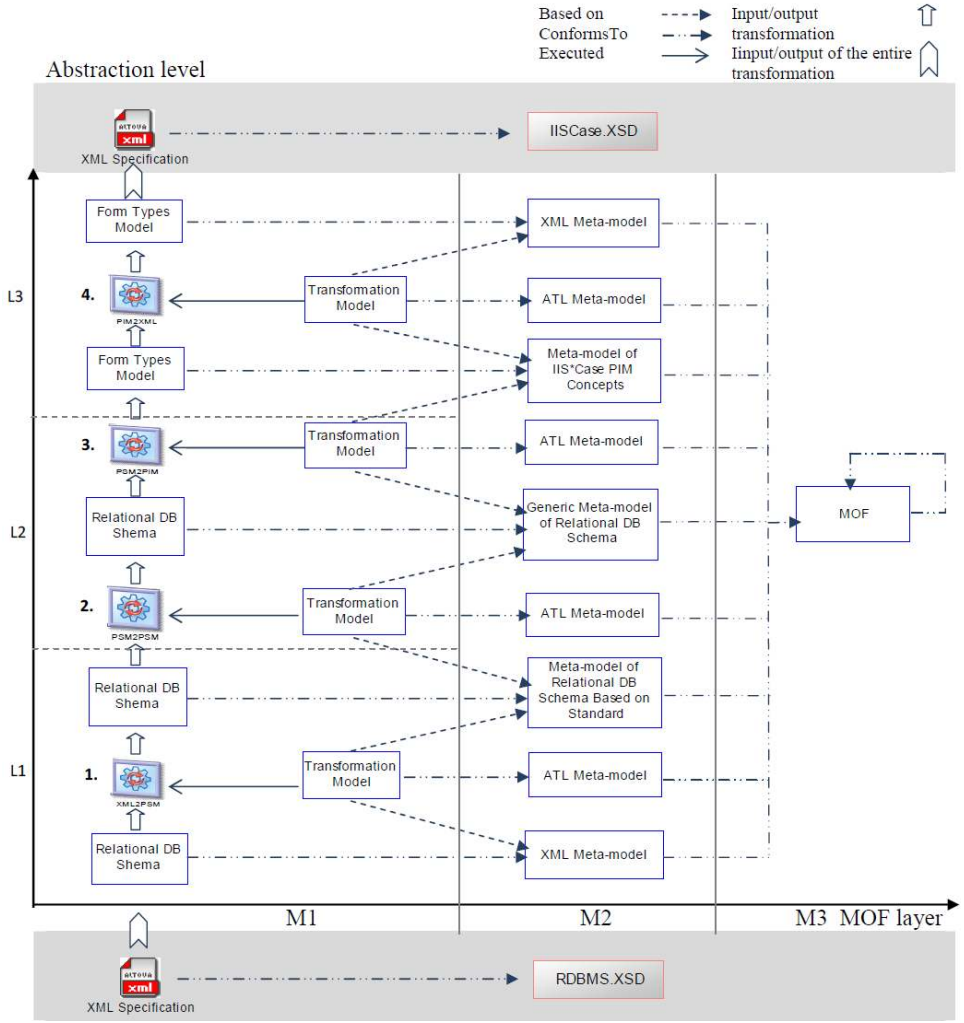

Fig. 3 Data structure conceptualization through a chain of M2M transformations

V. Case Study

In this section an example of a chain of model transformations that implements data conceptualization phase of database reverse engineering process is presented. *UnivesityDb* database instance is SUS consisting of two tables: *University* with columns *UniId*, *UniName* and *UniCity*; and *Faculty* with columns *UniId*, *FacId*, *FacShortName*, *FacName* and *Dean*. It is implemented under the Oracle database management system. In the data extraction phase vendor-specific physical database schema is extracted from legacy database repository and afterwards semantically enriched with the information collected from legacy database and through interaction with designers. The specification of vendor-specific physical database schema is written in an XML document that conforms to XML meta-model. At the very beginning of the process the transformation XML2RDBMS is executed and a SQL standard database schema is generated, conformant, in the particular case, with the SQL: 2008 dbS meta-model. By means of RDBMS2RM transformation the SQL standard database schema is transformed into a generic relational database schema. These two database schemas are not the same and they conform to different meta-models. In Fig. 4 a transformation RM2IISCase, from a generic relational database schema into a form type data model is presented in more details.

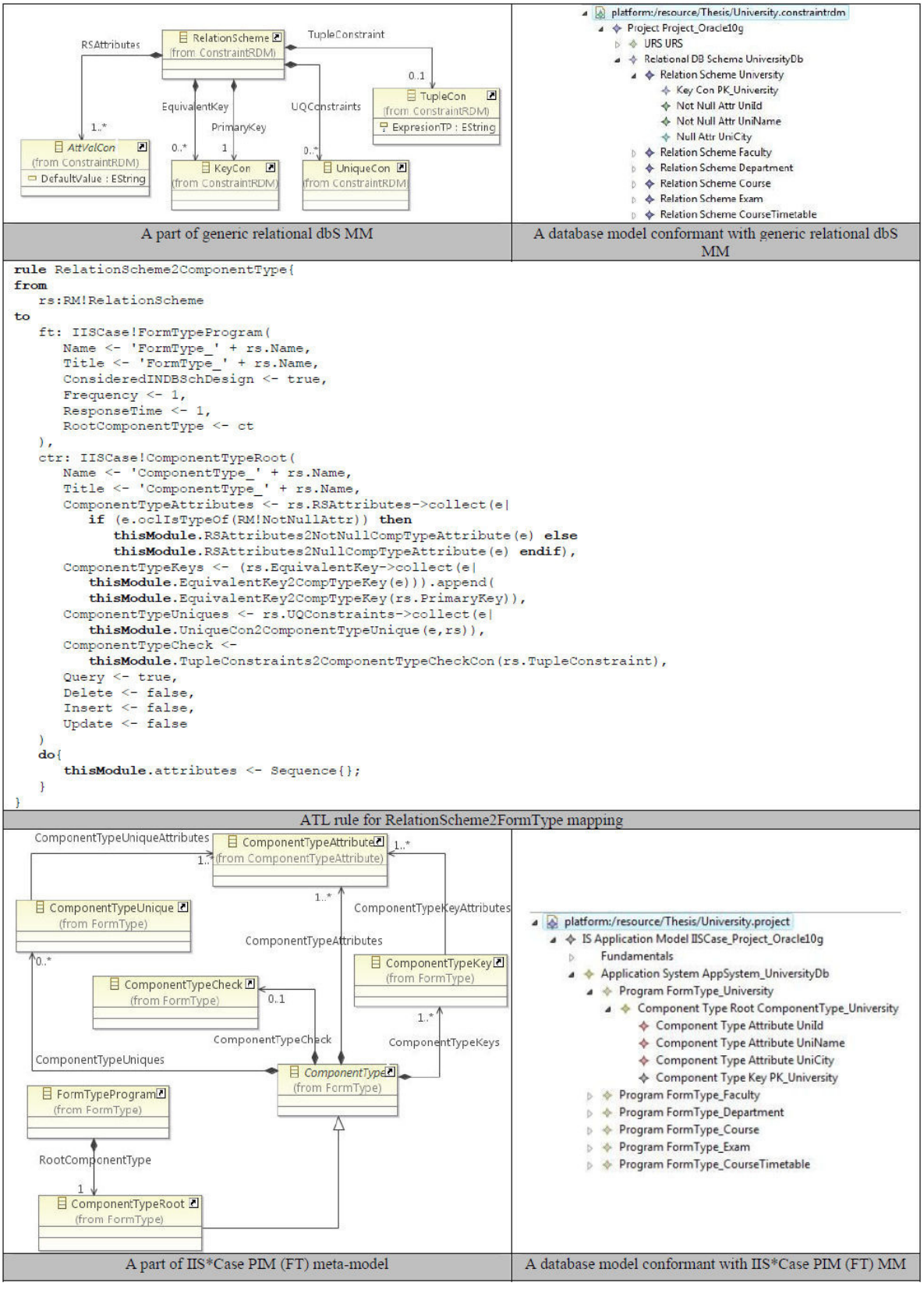

Fig. 4 An example of RM2IISCase transformation

In the first row a part of the generic relational dbS meta-model and a database model conformant with it are presented. In the third row the target IIS*Case PIM concepts meta-model and a conformant form-type database model are presented. In the middle of the table, the second row contains an ATL rule for mapping concepts from the generic

relational dbS meta-model to the concepts of IIS*Case PIM concepts meta-model. The model transformation gets the relational database model from the first row as input and transforms it into the form-type database model presented in the third row.

ATL rules presented in Fig. 4 are just the small excerpt from all ATL rules used to specify model transformations in IIS*REE. Presented rules are aimed at mapping concepts of source and target meta-models at the highest level of details. Mappings at the lower level of details alongside with belonging helpers are omitted here.

VI. Related work

Hainaut *et al.* [6], [9], [22] describe main steps of database reverse engineering and several authors based their research on these proposals. Perez *et al*. [23] and Boronat *et al*. [24] create object-oriented (class) conceptual database schema from the relational data dictionary. Gogolla *et al*. [25] 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. Beggar *et al*. [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. In [27] a process is proposed to automatically generate Web Services from relational databases. SQL-92 meta-model has been used to represent the conceptual database model. Polo *et al.* [28] present the technical and functional descriptions of a tool specifically designed for database reengineering based on simplified relational and object-oriented meta-model.

Aforementioned approaches to database conceptualization are mostly based just on two database meta-models. Vendor-specific physical or standard relational meta-model mainly are found on the source side of M2M transformation. On the other side, EER, class or standard/vendor-specific relational meta-models occur on the target side of M2M transformation, depending on selected conceptual schema. Our approach supports transformations between vendor-specific relational, standard relational, generic relational, FT, EER, and class meta-models. Our classification and distribution of database models across the MOF level stack (Fig. 1) enables systematic approach for mapping specification between different models/meta-models and development of appropriate M2M transformations. To the best of our knowledge, there is no similar systematical overview of database meta-models.

Some authors obtain relational database schema as the final result of data structure conceptualization process. According to Hainaut *et al.* [22] relational database schema cannot be seen as a pure conceptual database schema. It is not fully platform independent due to the fact that it is compliant with the data model of a family of database management systems. In our approach FT database schema is obtained as the result of the data structure conceptualization process. FT specification is based on business forms, users are familiar with, and in that manner it models system as-is in a platform independent way. At the same time, the specification is platform independent prescription model of future screen and report forms and input for series of M2M transformations that ends up with M2T transformation generating application prototype.

Research work on database model transformations can be broadly classified as unidirectional model transformation algorithms and multi-model (multi-language) transformation algorithms. The examples of unidirectional model transformation algorithms, proposed by Lano and Kolahdouz-Rahimi and Beggar *et al.* can be found in [29] and [26], respectively. Multi-model transformation algorithms can be ranged from linking each model to a graph [30], or description logic language (like those presented in [31]–[33]) to transformations mediated by a dictionary of common terms [34]. These results we consider as important in the context of our future research on unifying of different data model meta-models.

VII. Conclusion

When a legacy system become too costly to maintain, or when new technologies need to be incorporated, system need to be replaced or somehow reengineered. An important phase of a data-oriented software system reengineering is a database reengineering process and, in particular, its subprocess – a database reverse engineering process. In the paper we focus on the data structure conceptualization process and propose a model-driven approach to data structure conceptualization that is based on M2M transformations implemented by means of ATL.

The proposed framework offers a wide range of M2M transformations. IIS*Studio supports transformations between models conformant with EER, FT, relational and class meta-models. In the paper we present database conceptualization into a FT database schema. There are several reasons why we decide to support such kind of conceptualization. Firstly, the end users would participate in the restructuring of a conceptual schema during the database and IS reengineering process. The FT concept is closer to the end-users' perception of data, than the concepts of relational data model, that makes it easier to involve them in the restructuring process. Secondly, extracted FT specifications can be enriched with the specifications of the transaction programs and business applications and they can be used not only to support generation of a database schema of an IS, but to support the generation of an IS program code and appropriate graphical user interface (GUI). In that context, our future research can be directed towards improvement in the data extraction phase. Information captured from application GUI could contribute to the semantic richness of reverse engineered form-type model. Results presented in [35]–[38] support our idea. According to the future research in connection with data conceptualization phase we plan to specify meta-models of different data models and to unify different meta-models as the basis for platform independent

modeling of M2M transformations between database meta-models conformant with different data models meta-models, bearing in mind results presented in [39].

REFERENCES

[1] E.J. Chikofsky, and J.H. Cross, "Reverse engineering and design recovery: A taxonomy," *IEEE Software*, vol. 7(1), pp. 13–17, Jan. 1990.

[2] R. Kazman, S. Woods and J. Carrière, "Requirements for Integrating Software Architecture and Reengineering Models: Corum ii," in *Proc. of Working Conference on Reverse Engineering* (*WCRE*), Washington, 1998, pp. 154–163.

[3] R. S. Arnold, "A road map guide to software reengineering technology," in *Software Reengineering*, R. S. Arnold, Ed. Los Alamitos CA: IEEE Computer Society Press, 1993.

[4] S. Tilley and D. Smith, "Perspectives on Legacy System Reengineering", SEI White Paper, 1995.

[5] I. Jacobson and F. Lindström, "Re-engineering of Old Systems to an Object-oriented Architecture," in *Proc. of the ACM Conference on Object Oriented Programming Systems Languages and Applications*, New York, Oct. 1991, pp. 340–350.

[6] J. Hainaut, J. Henrard, D. Roland, V. Englebert, and J. Hick, "Structure Eliction in Database Reverse Engineering," in *Proc. of the 3th Working Conference on Reverse Engineering*, IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 131–139.

[7] J. Mukerji, and J. Miller, "MDA Guide Version 1.0.1, document omg/03-06-01 (MDA Guide V1.0.1)," <http://www.omg.org/>, (retrieved May, 2015)

[8] OMG Architecture Driven Modernization (ADM), <http://www.adm.omg.org >, (retrieved May, 2015)

[9] J. Hainaut, J. Henrard, D. Roland, V. Englebert, and J. Hick, "Knowledge Transfer in Database Reverse Engineering: A Supporting Case Study," in *Proc. of the 4th Working Conference on Reverse Engineering*, IEEE Computer Society Press, Los Alamitos, CA, 1997, pp. 194–203.

[10] I. Luković, P. Mogin, J. Pavićević, and S. Ristić, "An approach to developing complex database schemas using form types," *Software: Practice and Experience*, vol. 37 (15), pp. 1621–1656, 2007. doi: 10.1002/spe.820

[11] S. Aleksić, I. Luković, P. Mogin, and M. Govedarica, "A generator of SQL schema specifications," *Computer Science and Information Systems*, vol. 4(2), pp. 81–100, 2007.

[12] I. Luković, A. Popović, J. Mostić, and S. Ristić, "A tool for modeling form type check constraints and complex functionalities of business applications," *Computer Science and Information Systems*, vol: 7(2), pp. 359–385, 2010. DOI 10.2298/CSIS1002359L

[13] S. Aleksić, S. Ristić, I. Luković, and M. Čeliković, "A Design Specification and a Server Implementation of the Inverse Referential Integrity Constraints," *Computer Science and Information Systems*, vol. 10(1), pp. 283–320, 2013. DOI: 10.2298/CSIS111102003A

[14] S. Ristić, S. Aleksić, M. Čeliković, I. Luković, "Generic and Standard Database Constraint Meta-Models," *Computer Science and Information Systems*, vol. 11(2), pp: 679–696, 2014. DOI: 10.2298/CSIS140216037R

[15] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," *ACM computing surveys (CSUR)*, vol. 37(4), pp. 316–344, 2005.

[16] T. Kosar, N. Oliveira, M. Mernik, V. J. M. Pereira, M. Črepinšek, C. D. Da, and R. P. Henriques, "Comparing general-purpose and domain-specific languages: An empirical study," *Computer Science and Information Systems,* vol. 7 (2), pp. 247–264, 2010. DOI: 10.2298/CSIS1002247K

[17] M. Čeliković, I. Luković, S. Aleksić, V. Ivančević, "A MOF based meta-model and a concrete DSL syntax of IIS*case PIM concepts," *Computer Science and Information Systems*, vol. 9 (3) pp. 1075–1103, 2012. DOI: 10.2298/CSIS120203034C

[18] OMG Meta Object Facility (MOF), <http://www.omg.org/mof>, (retrieved May, 2015)

[19] S. Ristić, S. Aleksić, M. Čeliković, V. Dimitrieski, and I. Luković, "Database reverse engineering based on meta-models," *Central European Journal on Computer Science*, vol. 4(3), pp: 150–159, 2014. DOI: 10.2478/s13537-014-0218-1

[20] V. Dimitrieski, M. Čeliković, S. Aleksić, S. Ristić, I. Luković, "Extended entity-relationship approach in a multi-paradigm information system modeling tool, " in *Proceedings of the 2014 FEDCSIS*, Warsaw, Poland, ACSIS, Vol. 2, pp. 1611–1620, 2014. DOI: 10.15439/2014F239

[21] Eclipse Foundation, "ATL (ATLAS Transformation Language) Project, ATL/User Guide", available at: http://wiki.eclipse.org/ATL/User_Guide, 2010. (retrieved May, 2015)

[22] J-L. Hainaut, J. Henrard, V. Englebert, D. Roland, and J-M. Hick "Database Reverse Engineering," In *Encyclopedia of Database Systems*, L. Liu, and Özsu, T., Ed., Springer-Verlag, 2009.

[23] J. Perez, I. Ramos, and V. Anaya, "Data reverse engineering of legacy databases to object oriented conceptual schemas," *Electronic Notes in Theoretical Computer Science*, vol. 74(4), pp. 1–13, 2002.

[24] A. Boronat, J. Perez, J. A. Cars, and I. Ramos., "Two Experiences in Software Dynamics," *Journal of Universal Computer Science*, vol. 10(4), pp. 428–453, 2004.

[25] M. Gogolla, A. Lindow, M. Richters, and P. Ziemann, "Meta-model transformation of data models," Position paper. WISME at the UML 2002.

[26] Beggar O. E., Bousetta B., Gadi T., Getting Relational Database from Legacy Data-MDRE Approach, Computer Engineering and Intelligent Systems www.iiste.org ISSN 2222-1719 ISSN 2222-2863 (Online) Vol 4, No.4, 2013.

[27] R.P. del Castillo, I. García-Rodríguez, and I. Caballero, "PRECISO: a reengineering process and a tool for database modernization through web services", In: *Jacobson Jr.*, M.J., Rijmen, V., Safavi-Naini, R. Ed., SAC 2009. LNCS, vol. 5867, Springer, Heidelberg, pp. 2126–2133, 2009.

[28] M. Polo, I. Garcia-Rodriguez, and M. Piattini, "An MDA-based approach for database re-engineering," *J. Softw. Maint. Evol*., vol. 19 (6), pp. 383–417, 2007.

[29] K. Lano, S. and Kolahdouz-Rahimi, "Constraint-based specification of model transformations," *Journal of Systems and Software*, vol. 86(2), pp. 412–436, 2013. DOI: 10.1016/j.jss.2012.09.006

[30] M. Boyd, and P. McBrien, "Comparing and transforming between data models via an intermediate hypergraph data model," *Journal on Data Semantics*, vol. IV, pp. 69–109, 2005.

[31] R. P. Fillottrani, and C. M. Keet, "Conceptual Model Interoperability: A Metamodel-driven Approach," *Rules on the Web. From Theory to Applications, Lecture Notes in Computer Science*, pp 52–66, 2014. DOI: 10.1007/978-3-319-09870-8_4

[32] C. M. Keet,and P. R. Fillottrani, "Structural entities of an ontology-driven unifying metamodel for UML, EER, and ORM2," In: *Proc. of MEDI'13. LNCS*, vol. 8216, Amantea, Calabria, Italy, Springer, pp. 188–199, 2013. DOI: 10.1007/978-3-642-41366-7_16

[33] C. M. Keet, and R. P. Fillottrani, "Toward an ontology-driven unifying metamodel for UML Class Diagrams, EER, and ORM2," *Conceptual Modeling , Lecture Notes in Computer Science,* vol. 8217, pp 313–326, 2013. DOI: 10.1007/978-3-642-41924-9_26

[34] P. Atzeni, G. Gianforme, and P. Cappellari, "Data model descriptions and translation signatures in a multi-model framework," *AMAI Mathematics and Artficial Intelligence*, vol. 63, pp. 1–29, 2011. DOI: 10.1007/s10472-012-9277-y

[35] Kreutzová, Michaela, Jaroslav Porubän, and Peter Vaclavik. "First Step for GUI Domain Analysis: Formalization," *Journal of Computer Science and Control Systems*, vol. 4(1), pp. 65–69, 2011.

[36] M. Bačiková, and J. Porubän, "DSL-driven generation of GUI" *Central European Journal of Computer Science*, vol. 4(4), pp. 204–211, 2014. DOI 10.2478/s13537-014-0210-9

[37] M. Malki, A. Flory, and M. K. Rahmouni, "Extraction of Object-oriented Schemas from Existing Relational Databases: a Form-driven Approach," *INFORMATICA*, vol. 13(1), pp. 47–72, 2002.

[38] S. M. Benslimane, M. Malki, M. K. Rahmouni, and DJ. Beslimane, "Extracting Personalised Ontology from Data-Intensive Web Application: an HTML Forms-Based Reverse Engineering Approach," *INFORMATICA*, Vol. 18 (4), pp. 511–534, 2007.

[39] H. Kern, "Study of Interoperability between Meta-Modeling Tools" in *Proceedings of the 2014 FEDCSIS*, Warsaw, Poland, ACSIS, Vol. 2, pp. 1629–1637, 2014. DOI: 10.15439/2014F255