**XVII International Scientific Conference on Industrial Systems (IS'17)**
**Novi Sad, Serbia, October 4. – 6. 2017.**
University of Novi Sad, Faculty of Technical Sciences,
Department for Industrial Engineering and Management
Available online at http://www.iim.ftn.uns.ac.rs/is17

# From the Decomposition Algorithm to the Model Driven Transformations in Database Design

**Angelina Vujanović**
(Teaching assistant, University of Novi Sad, Faculty of Technical Sciences, Trg D. Obradovića 6, 21000 Novi Sad, Serbia, avujanovic@uns.ac.rs)
**Slavica Kordić**
(Assistant professor, University of Novi Sad, Faculty of Technical Sciences, Trg D. Obradovića 6, 21000 Novi Sad, Serbia, slavica@uns.ac.rs)
**Ivan Luković**
(Full professor, University of Novi Sad, Faculty of Technical Sciences, Trg D. Obradovića 6, 21000 Novi Sad, Serbia, ivan@uns.ac.rs)
**Milan Čeliković**
(Teaching assistant, University of Novi Sad, Faculty of Technical Sciences Trg D. Obradovića 6, 21000 Novi Sad, Serbia, milancel@uns.ac.rs)
**Jovana Vidaković**
(Assistant professor, University of Novi Sad, Faculty of Sciences, Trg D. Obradovića 3, 21000 Novi Sad, Serbia, jovana@uns.ac.rs)

**Abstract**

*There are a lot of different approaches to information system (IS) development which are based on different data models. Which data model will be chosen depends on the problem domain, the knowledge and also on the personal preferences of an IS developer. In our previous research we have developed a tool that provides an incremental approach to IS development which is based on the FT, the Extended Entity-Relationship (EER) and the class data models, a Multi-Paradigm Information System Modeling Tool (MIST). MIST provides transformations of the FT and the EER database models to the relational data model and also transformations of the EER to the class data model. The goal of this paper is to use the decomposition algorithm in a database design project and to apply it in the scope of the Model Driven Software Development process. In this way it can be possible to overcome the problems of a database schema design process based on the EER data model and transformations into the relational data model. Therefore, we have decided to extend our tool functionalities in order to support the decomposition algorithm. Besides, its purpose to overcome the problems that the EER approach introduces, the implemented decomposition algorithm can be used in education as well.*

**Key words:** *Decomposition Algorithm. Model Driven Software Development. Relational Database Design*. *Multi-Paradigm Information System Modeling Tool.*

## 1. INTRODUCTION

During the last few decades, a number of approaches to the information system (IS) development and design has grown. With the emergence of a large and complex IS that are interoperable in highly changeable environment, the choice of the appropriate approach becomes one of the key tasks in an IS development. One of the broadly accepted approaches is a model-driven approach. Model Driven System Engineering (MDSE) and Model Driven Software Development (MDSD) use the power of modelling to address a set of systems development problems. The power of MDSD and MDSE lies in the power of abstractions [1]. Models in MDSD are the main building concepts that faithfully reflect the reality, such as data structures in programming. Model-driven approach increases the importance of models and their power and allows the developers to work at a higher level of abstraction, using concepts and structures which are closer to the end users.

In our previous research [2-5] we have developed a tool that provides a model-driven approach to IS development and evolution, the Multi-Paradigm Information System Modeling Tool (MIST). The MIST is a software tool aimed to provide an incremental approach to IS development that is based on the form type (FT) data model and the Extended Entity-Relationship (EER) data model. The MIST provides transformations of the FT and the EER database models to the relational data model and also transformations of the EER to the class data model [6].The latest extension of the MIST is the EER2XML component, which provides transformations of the EER

database model to a generic XML schema specification [7].

Besides the choice of appropriate approach in IS development, the one also important task is a database schema design. The further IS development heavily depends on a correct database schema design-a badly designed database schema can have seriously consequences on the further development. In the last decades, many data models and paradigms in a database design have appeared. Some of them are characterized as implementation data models (relational data model), the other ones are characterized as conceptual data models (Entity-Relationship (ER) data model and its extensions (EER)). Conceptual data models are mostly used in the database design process to create conceptual database schema specifications.

The one typical scenario of a database schema design process consists of the following steps: creating an EER database schema, transforming it into the relational database schema and its implementation under different database management systems (DBMSs). Despite that this scenario has many advantages it also has some serious disadvantages. The transformation process of an EER database schema to relational database schema is based on applying well-known mapping rules. The common belief is that these rules will guarantee the satisfaction of the third normal form condition (3NF) per se. However, a reality is different and these mapping rules should be taken only as advisable because transformation process does not only depend on the formal mapping rules, but also on the problem domain semantics. There are many examples in which the same structure of EER database schema should not be transformed into the same relational database schema structure, due to the different semantics assigned to the EER structure.

Besides the EER data model, the relational data model has also a considerable popularity among the database practitioners and researchers for its conceptual simplicity and mathematical background. Many principles, logical problems and their solutions that concerning database design are defined at the level of relational data model. Only some of these problems, solutions and principles are exploited in database design practices. One of the most important algorithms for database schema design are normalization algorithms-decomposition and synthesis algorithms. The focus in this paper is on the decomposition algorithm and its practical usage.

The goal of this paper is to use the decomposition algorithm in database design projects and to apply it in the scope of MDSD process. Using decomposition algorithm it is possible to overcome the problems of database schema design based on the EER data model and its transformation into the relational data model. In order to support the decomposition algorithm, we have decided to extend the MIST functionalities with the decomposition component. Our decomposition component uses as an input a Universal Relation Schema (URS) specification and produces a finite set of relation schemes as an output.

Apart from Introduction and Conclusion, the rest of the paper is organized as follows. Related work is discussed in Section 2. A brief overview of the decomposition algorithm is given in Section 3. Transformations that are used in order to implement decomposition algorithm and meta-models that are used as input and output in transformation process are presented in Section 4. A case study that illustrates a usage of implemented decomposition algorithm is given in Section 5.

## 2. RELATED WORK

This section gives a brief overview of the similar tools and approaches to the IS development.

As it is already mentioned, we have developed a tool that provides a model-driven approach to IS development and evolution. MIST provides an IS development that is based on the FT and the EER data model, also transformations of the FT and the EER database models to the relational data model, transformations of the EER to the class data model and the EER to the generic XML schema specification [2-7]. Because of the mentioned problems of a database schema design based on the EER data model and its transformation to the relational, we have extended functionalities of our tool with the Synthesis component [8]. The synthesis component implements the improved synthesis algorithm, taking the FT model and transforming it into the URS specification. The synthesis algorithm then takes the URS specification and produces a relational database model as an output. Till now, approach based on the synthesis algorithm has been successfully applied in many projects. As the synthesis algorithm proved to be successful, we decided to extend our tool with another normalization algorithm [12], the decomposition algorithm. For the purpose of this paper we implemented the original decomposition algorithm.

Besides self-references that are given in this paper, there are many other references presenting the results of this research effort. With the increased use of relational database, many automated tools are being developed. In [13] authors have reported on a tool that helps users to specify functional dependencies (fds). There is also a Prolog-based system for normalization through Boyce-Codd Normal Form (BCNF) [14], where is also incorporated a new algorithm for projecting fds on the sub relations. In [15] authors developed a method that maps an EER schema into a relational schema and normalizes it latter into inclusion normal form (IN-NF). Unlike classical normalization, IN-NF takes interrelational redundancies into account.

To the best of our knowledge, neither of previously mentioned papers use the decomposition algorithm in the scope of MDSD process as we do.

## 3. DECOMPOSITION ALGORITHM

The decomposition algorithm is a way of ensuring that a database structure is suitable for general-purpose querying and free of certain undesirable characteristics,

such as insertion, update and deletion anomalies. The decomposition algorithm represents a method of systematic decoupling relation scheme into two smaller relation schemes, starting with the URS which can be assumed as a universal relation, containing: (i) a universal set of attributes $\mathcal{U}$ and (ii) a set of functional dependencies $\mathcal{F}$ defined over $\mathcal{U}$. The algorithm will repeatedly decompose schemes based on the functional dependence (fd) till all relation schemes S = $\{N_i(R_i, \mathcal{K}_i) \mid i \in \{1,..., n\}\}$, where $N_i$ is the scheme name, $R_i$ is the set of attributes of relational scheme and $\mathcal{K}_i$ is the set of keys of a relation scheme $N_i$, are in a desired normal form [10]. In this paper as desired normal form we consider BCNF.

The algorithm guarantees: (a) preserving the input set of attributes, (b) the lossless join condition over the whole finite set of relational schemes, by embedding a relation scheme into S whose key is a key of universal relation scheme ($\mathcal{U}$, $\mathcal{F}$), if necessary. Preservation of the input set $\mathcal{F}$ is not guaranteed, as some of the fds can be lost in the decomposition process. If any fd is lost during the decomposition process, after all relation schemes are in the desired normal form, it is possible to merge relation schemes with the equivalent keys. This leads to the degradation of achieved normal forms, but results in compensation of the lost fd.

The choice of fd ($Y \rightarrow A \in \mathcal{F}$) on which the decoupling of relation scheme is based, is one of the most important tasks in the decomposition algorithm. There are three criteria (names of criteria are written in bolds) for selecting the appropriate fd:

- **the first one (C1)** – introduces the conditions that guarantee preservation of the input set $\mathcal{F}$ (the third condition in (1)), but do not guarantee that all relation schemes are in BCNF. The selected fd is a non-trivial fd that is not a result of a key dependencies (the first and second condition in (1)).

$$Y \rightarrow A \text{ so as } A \notin Y \wedge \mathcal{U} \notin Y^+ \wedge (\mathcal{F}^+ = (\mathcal{F}_{|Y(\mathcal{U}\setminus A)} \cup \mathcal{F}_{|YA})^+ \quad (1)$$

- **the second one (C2)** – these conditions allow fd to have a key on the left hand side but it is also important that the union of the left and right hand sides of that fd does not contain all attributes from the set $\mathcal{U}$ (the second condition in (2)). The input set $\mathcal{F}$ is also preserved (the third condition in (2)).

$$Y \rightarrow A \text{ so as } A \notin Y \wedge AY \subset \mathcal{U} \wedge (\mathcal{F}^+ = (\mathcal{F}_{|Y(\mathcal{U}\setminus A)} \cup \mathcal{F}_{|YA})^+ \quad (2)$$

- **the third one (C3)** – introduces a requirement of selecting non-trivial fd that is not result of key dependencies (both conditions in (3)). This requirement guarantees that all relation schemes are in BCNF.

$$Y \rightarrow A \text{ so as } (A \notin Y) \wedge \mathcal{U} \notin Y^+ \quad (3)$$

After finding the appropriate fd ($Y \rightarrow A$), relation scheme is decoupled as follows: one scheme contains all attributes without ones on the right side of the selected fd and the set of fds $\mathcal{F}$ defined over that set of attributes $((R_1, \mathcal{F}_1) = ((\mathcal{U}\setminus A)Y, F_{|(\mathcal{U}\setminus A)Y})$ and the another scheme contains union of attributes from the left and right side of the selected fd and the set of fds $\mathcal{F}$ defined over that set of attributes $((R_2, \mathcal{F}_2) = (YA, F_{|YA})$.

In the next section we present implementation of the previously described decomposition algorithm.

## 4. DECOMPOSITION COMPONENT OF THE MIST

In order to support the decomposition algorithm, we have extended the MIST functionalities with the decomposition component. In this section we present part of our decomposition component that is responsible for transforming the URS into a finite set of relation schemes. The decomposition component of the MIST provides model-to-model (M2M) transformations (written in bolds):

- **the first one (T1)** – covers the basic decomposition algorithm, without merging relational schemes with the equivalent keys and
- **the second one (T2)** – responsible for merging relational schemes with the equivalent keys, if it is necessary.

The model that is transformed in **T1** conforms to the meta-model of the URS which is transformed into the model of a finite set of relation schemes. The model that is transformed in **T2** is the output from **T1**, or to be more precise, the model of a finite set of relation schemes. The same model is also output from **T2**, but with the merged relational schemes.

The meta-model is presented in the following subsection. The output of the decomposition component, a finite set of relation schemes, may be further used in the process of code generation.

### 4.1 META-MODEL

In this subsection, we present a meta-model of the URS in more details. The components of our meta-model are given in Fig. 1. In the rest of this subsection we describe each of the components with the corresponding meta-model class (written in italics).

In order to create a meta-model it is necessary to perceive the concepts specific to a particular domain [9]. As it is already mentioned, the decomposition algorithm is based on the URS and by the equivalent transformations of $\mathcal{F}$ the algorithm produces a finite set of relation schemes S. For purpose of creating meta-model to which the input/output model of transformation process conforms, the following concepts are necessary: attributes, keys, relation schemes and fds. Based on these concepts we have created our meta-model. In this paper we present only parts of the URS meta-model which are important for the decomposition method, the rest can be found in [8].

The set of relation schemes (*SetOfRelationSchemes*) is the root element of our meta-model. Each root element has at least one or more relation schemes that are modelled by the *RelationScheme* class. The class

*RelationScheme* contains only name as attribute. Each relation scheme has one or more attributes that are modelled by the *Attribute* class, zero or more functional dependencies modelled by the *FDependencies* class and zero or more keys modelled by the *Key* class. Each attribute has a unique name (universal relation assumption). For that purpose restriction that two attributes with the same name cannot be entered is implemented. Each key comprises one or more attributes of the relation scheme. A functional dependence has one left side and one right side, which are modelled by the *LeftSide* and the *RightSide* class, respectively. Each left and right side of fd has zero or more attributes. There is also a restriction that the left or the right side of fd cannot contain attributes that are not in the set of attributes of that relation scheme. The class *ClosureOfFD* is created for the purpose of transformation algorithm and comprises zero or more fds.

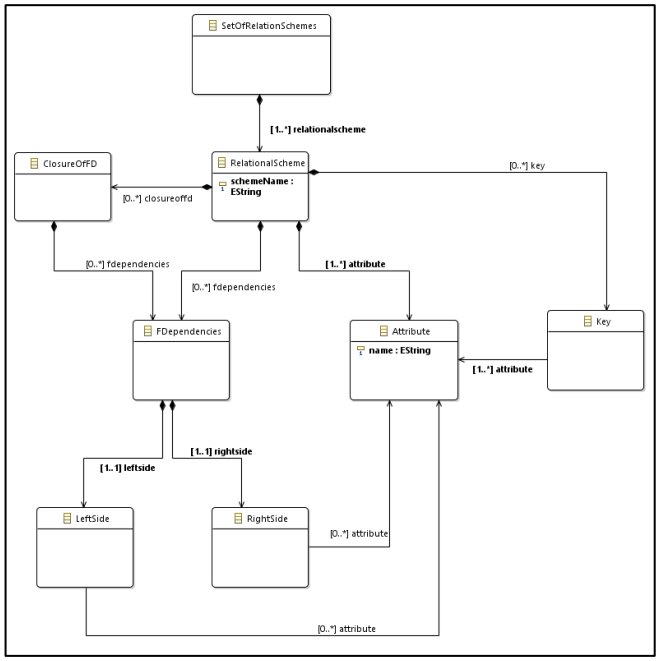


**Figure 1.** The meta-model of a URS

## 4.2 ATL TRANSFORMATIONS

The transformations as an input and output use a model that conforms to the meta-model mentioned in the previous subsection. The transformations are specified in the Atlas transformation language (ATL) [11]. In this paper we present steps of the decomposition algorithm without the ATL implementation details.

As it is said in the introduction part of this section, **T1** is used for the decomposition algorithm (without merging relational schemes) and **T2** for merging relational schemes after decomposition, if it is necessary. Before the implementation of the decomposition algorithm it is necessary to find the candidate keys for the URS. Therefore we have implemented the algorithm for finding candidate keys [12]. It is also important to implement the algorithm for attributes closure and fds closure [12], which will be used in further steps of the decomposition algorithm. After implementation of the previous algorithms, the next step of our transformation process is finding an appropriate fd from the $\mathcal{F}$ on which the decoupling of relation schemes will be based.

First, we try to find that fd using **C1**. In this step of transformation we are passing through the $\mathcal{F}$ and for every fd check whether it satisfies **C1**. The algorithms for checking the appropriate criteria satisfying are implemented based on the (1), (2) and (3) given in the previous section. For that purpose we used the previous implemented algorithms for finding keys, attributes closure and fds closure. We used the first fd that satisfies **C1** to decouple the URS as it is described in the previous section. In this process we assign an unique name ($N_1$, $N_2$,..., $N_n$) to every relation scheme. If there is no fd that satisfies **C1** then we try to find a fd that satisfies **C2**. The method of finding appropriate fd is similar to the previous one. If there is no fd that satisfies **C2** then we try to find a fd that satisfies **C3**.

After we finish with decoupling and every scheme is in BCNF we check if the input $\mathcal{F}$ is preserved. If it is not, we merge relation schemes with the equivalent keys.

## 5. CASE STUDY

In this section we introduce a small case study in order to illustrate concepts, data models and transformations described in previous sections. The idea is to enter URS syntax correctly written as input, as it is shown on the left side of the Fig. 2 and to get a finite set of merged relational schemes at the output, as it is shown on the right side of the Fig. 5.

As URS in this case study we use:

$$\mathcal{U} = \{a, c, e, g\}, \mathcal{F} = \{e \rightarrow c,\ ac \rightarrow g,\ ac \rightarrow e\}$$

The input in **T1** is model that conforms to the meta-model of URS, as it is shown on the left side of the Fig. 2 and output is a finite set of unmerged relational schemes, as it is shown on the right side of the Fig. 2.

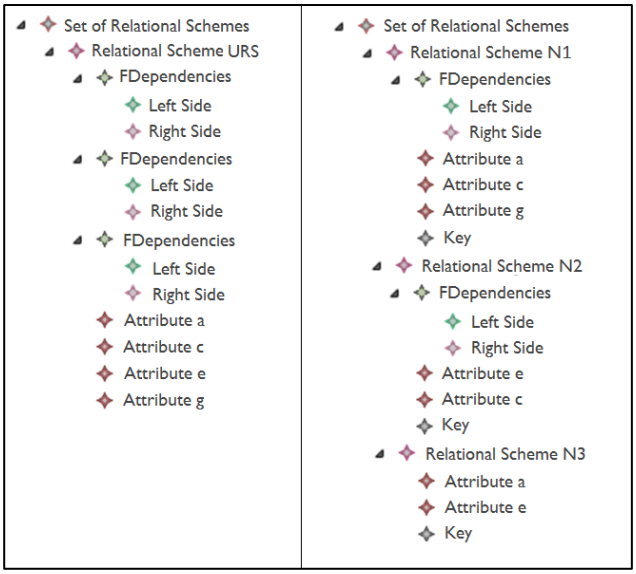


**Figure 2.** The source and target models of **T1**

The first step in our transformation process is to find candidate keys for the relational scheme. For that purpose we implemented algorithm for finding candidate keys [12], so based on the input sets $\mathcal{U}$ and $\mathcal{F}$ we found following candidates for the URS:

$\mathcal{K}$ = {ae, ac}

After finding the candidate keys, the next step is to check if the starting relational scheme is in the BCNF. If it is not, then we examine all fds from the $\mathcal{F}$ to find appropriate one for the decomposition. Based on the previously mentioned three criteria, we find that neither one of the fds from the set $\mathcal{F}$ satisfies **C1**. So we continue with examination of the set $\mathcal{F}$ and find that fd ac → g satisfies **C2**. We use it to decouple the URS. The process of decoupling based on the selected fd is given in Fig. 3.

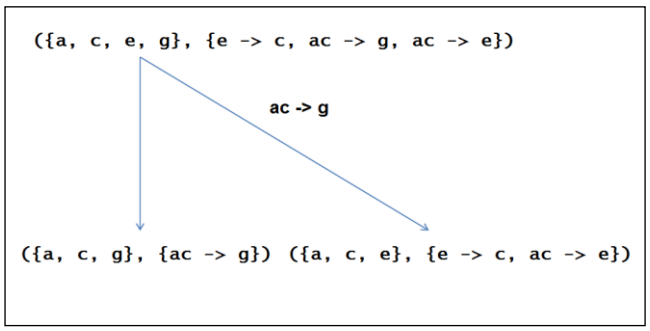


**Figure 3.** Decomposition based on the fd ac->g

After decoupling, we have relational scheme **$N_1$ ({a, c, g}, {ac → g})** which is in the BCNF (there is also checking in which normal form is specific relational scheme) and the following one

$\mathcal{U}_1$= {a, c, e}, $\mathcal{F}_1$= {e → c, ac → e}

which is not in the BCNF. In the further steps, we examine only the relational scheme which is not in the BCNF. Based on the three criteria, we find that neither one of the fds in the set $\mathcal{F}_1$ satisfies **C1** nor **C2**. So we continue with examination of **C3** and find that fd e→ c satisfies it. We use it for decoupling. Fig. 4 shows the decomposition of the relational scheme based on the selected fd.

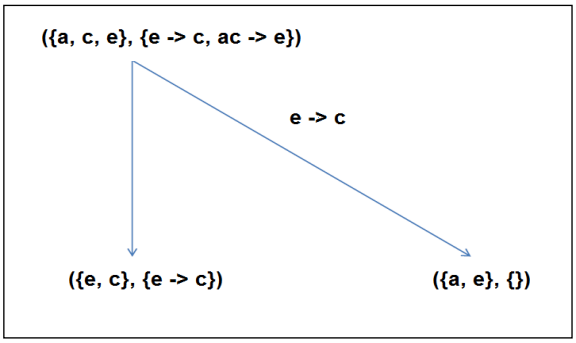


**Figure 4.** Decomposition based on the fd e->c

Now, we have set of relational schemes:

- **$N_1$({a, c, g}, {ac → g}) $K_1$={ac},**
- **$N_2$({e, c}, {e → c}) $K_2$={e}** and
- **$N_3$({a, e}, {}) $K_3$={ae}**

which all are in the BCNF. This set of relational schemes represents the result of the **T1**.

After we finish with decoupling and every scheme is in the BCNF, in **T2** we check if the input set $\mathcal{F}$ is preserved. The input model in **T2** is the output model from **T1**, a set of unmerged relational schemes. In this case study the set $\mathcal{F}$ is not preserved, so we merge relational schemes with the equivalent keys (**$N_1$** and **$N_3$**).

Finally, we have set of relational schemes:

- **$N_1$({e, c}, {e → c}) $K_1$={e},**
- **$N_2$({a, c, e, g}, {ac → g, ac → e}) $K_2$={ac, ae}**

where **$N_1$** is in the BCNF and **$N_2$** is in the 3NF. This set represents the result of **T2**. On the left side of the Fig. 5 the output from the previous transformation **T1** is shown, which is also an input in **T2**. On the right side there is the result of **T2**, a finite set of relational schemes.

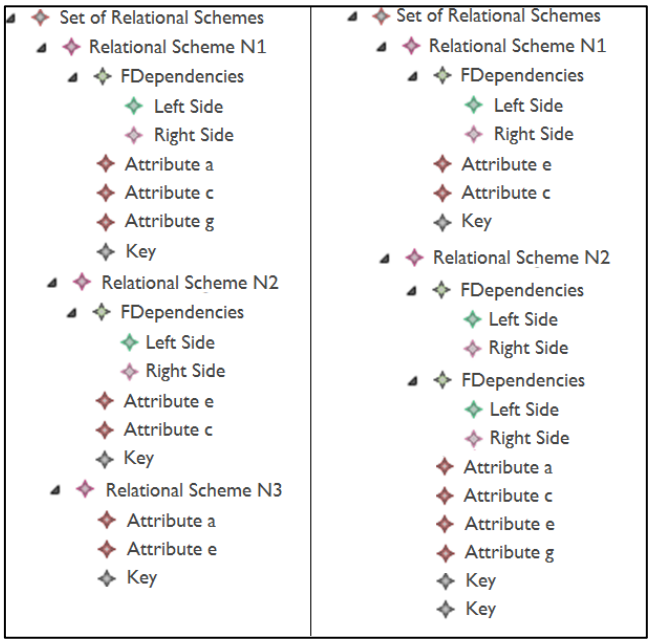


**Figure 5.** The source and target models of **T2**

## 6. CONCLUSION

During our previous researches we have developed a tool that provides a model-driven approach to IS development and evolution, named MIST. The MIST is a software tool aimed to provide an IS development that is based on the FT data model and the EER data model and also provides transformations of the FT and the EER database models to the relational data model, the transformations of the EER to the class data model and the EER to the XML schema specification.

In this paper, we present an alternative approach to the database schema design process that is not based on a typical scenario of creating an EER database schema and then its transformation into the relational database

schema. Instead of that, our approach uses a decomposition algorithm, as well as many other algorithms created for that purpose. In order to support decomposition algorithm in our tool, we have extended functionalities of the MIST by developing a decomposition component. The decomposition component uses the URS specification as an input and produces a finite set of relation schemes as an output. One of the advantages of our approach is that it overcomes some problems that may arise in the process of transforming the EER database schema into the relational database schema because transformation process does not only depend on the formal mapping rules, but also on the problem domain semantics. There are many examples in which the same structure of the EER database schema should not be transformed into the same relational database schema structure, due to the different semantics assigned to the EER structure. In such cases, the quality of designed database schemas is dependant of designer's theoretical knowledge and previous experience.

Till now, we have successfully applied the tool and approach in many projects. We also use it for educational purposes in the course of domain specific languages and model driven software development.

Further development considers the use of our tool and approach in a large scale IS projects. One future development of our decomposition component can be implementation of checking the preservation of the lossless connectivity. The checking consists of the determining the key of the URS and verifying whether the final set of relation schemes includes a relation scheme with the key of URS. One improvement can also be a support for the set of constraints, such as referential and inverse referential integrity, not null and check constraints.

## 7. ACKNOWLEDGMENT

The research presented in this paper was supported by Ministry of Education, Science and Technological Development of Republic of Serbia, GrantIII-44010.

## 8. REFERENCES

[1] Nolan, B., Brown B., Balmell L., Bohn T. and Wahli, U. (2008), *Model Driven Systems Development with Rational Products*, IBM Redbooks.

[2] Luković, I., Mogin, P., Pavičević, J. and Ristić, S. (2007), "*An approach to developing complex database schemas using form types*", Software: Practice and Experience 37 (15), pp. 1621–1656.

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

[4] Luković, I., Popović, A., Mostić, J. and Ristić, S. (2010),"*A tool for modeling form type check constraints and complex functionalities of business applications*", Computer Science and Information Systems, Vol. 7, Issue 2, pg. 359–385.

[5] Luković, I., Ristić, S., Mogin, P. and Pavičević, J. (2006), "*Database schema integration process–a methodology and aspects of its applying*", Journal of Mathematics, Vol. 36, Issue 1, pg. 115–140.

[6] Dimitrieski, V., Čeliković, M., Aleksić, S., Ristić, S., Alargt, A. and Luković, I. (2015), "*Concepts and Evaluation of the Extended Entity-Relationship Approach to Database Design in a Multi-Paradigm Information System Modeling Tool*", Computer Languages, Systems and Structures, Vol. 44, pg 299-318.

[7] Poznanović, M., Kordić, S., Vidaković, J., Ristić, S. and Čeliković, M. (2017), "*An Approach to Generating Specifications of a Database Schema for DBMSs based on XML Data Model"*. In: Zdravković, M., Konjović, Z., Trajanović, M. (Eds.) ICIST 2017 Proceedings Vol. 1, pp.195-200.

[8] Luković, I. (2009), "*From the synthesis algorithm to the model driven transformations in database design*". In: Proceedings of 10th International Scientific Conference on Informatics, Herlany, Slovakia, pp. 978-988.

[9] Brambilla, M., Cabor, J. and Wimmer, M. (2012) *Model-Driven Sofware Engineering in Practice*, Morgan & Claypool Publishers.

[10] Mogin, P., Luković, I. and Govedarica M. (2004), *The Principles of Database Design*, University of Novi Sad, Faculty of Technical Sciences.

[11] ATL [Online]. Available at: https://eclipse.org/atl/

[12] Mogin, P. and Luković, I. (1996), *The Database Principles*, University of Novi Sad, Faculty of Technical Sciences.

[13] Hasan, W. and Andyork, B.W. (1985), "*A database design tool*", AITG Tech. Rep. 003, Digital Equipment, Corp,. Hudson, Mass.

[14] Ceri, S. and Gottlob, G. (1986), "*Normalization of relations and Prolog*", Commun. ACM, pg. 524-545.

[15] Kolp, M. and Zimanyi, E. (1998), *"Prolog-Based Algorithms for Database Desing*", Proceedings of the 6th International Conference on Practical Applications of Prolog, PAP'98, London, UK.