Proceedings of the Federated Conference on Computer Science and Information Systems pp. 1577–1580
DOI: 10.15439/2016F408
ACSIS, Vol. 8. ISSN 2300-5963

A Model-to-Model Transformation of a Generic Relational Database Schema into a Form Type Data Model

Sonja Ristić, Slavica Kordić, Milan Čeliković, Vladimir Dimitrieski, Ivan Luković
University of Novi Sad,
Faculty of Technical Sciences,
Trg D. Obradovića 6, 21000 Novi Sad, Serbia
Email: {sdristic, slavica, milancel, dimitrieski, ivan}@uns.ac.rs

Abstract—**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 this paper we present one of the model-to-model transformations from a chain of transformations aimed at transformation of a generic relational database schema into a form type data model. The transformation is a step of the data structure conceptualization phase of a model-driven database reverse engineering process that is implemented in IIS*Studio development environment.**

I. Introduction

A MODEL-DRIVEN (MD) approach to information system (IS) and software (re)engineering addresses complexity through abstraction. A complex system consists of several interrelated models organized through different levels of abstraction and platform specificity. Through a forward engineering process models need to be refined and integrated and used to produce code and therefore they would undergo a series of transformations. Each transformation adds levels of specificity and detail. A chain of model-to-model (M2M) transformations is 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). Conversely, in a reverse engineering process, the abstraction level of models and degree of platform independency are increasing throughout the chain of transformations.

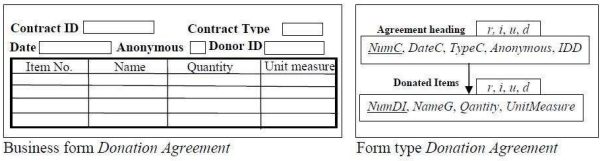
Through a number of research projects on MD intelligent systems for IS development, maintenance and evolution, we have developed the IIS*Studio development environment. It is aimed to provide the IS design, generating executable application prototypes and IS reverse engineering. Our approach is mainly based on the MD information system and software engineering [1] and domain specific language (DSL) paradigms ([2], [3]). In [4] we discuss the importance of meta-modeling in the context of database reverse engineering and review different database meta-models (MM) that are used in the database reengineering process applied in IIS*Studio. In [5] we propose an MD approach to data structure conceptualization phase of database reverse engineering process that is conducted through a chain of M2M transformations. In this paper we present the final step of the conceptualization phase—the M2M transformation of a generic relational database schema into a form type model.

The form type concept and the IIS*Studio architecture are given in Section 2. Classifications of form types and relation schemes are described in Section 3. The transformation of a generic relational database schema into a form type data model is presented in Section 4 and related work is discussed in Section 5.

II. Form Type Concept

A form type is central IIS*Studio PIM concept, used to model the structure and constraints of various business forms that are broadly used in organizations to conduct daily operations and to communicate with their affiliated entities. They are a source for eliciting user information requirements and for designing and developing user-oriented information systems. Initially, each form type (FT) is an abstraction of a business form. However, it may be enriched by additional specifications like specifications of: key and unique constraints; check constraints; allowed database CRUD (Create, Retrieve, Update and Delete) operations applied by means of screen computerized forms to manipulate data of an IS; functionalities concerning relationships between generated screen forms, i.e. transaction programs. The business form *Donation Agreement* (*DA-bf*) is presented on the left-hand side of Fig. 1. It is business form used in the Safe House Center (SHC) that provides support for those children impacted by domestic violence. The SHC is in a great extent based on donations and SHC IS has to support donation process. One of the activities is keeping track about the donation agreements. The business form *Donation Agreement* may be modeled by the form type *Donation Agreement* (*DA-ft*). The simplified representation of the structure of the *DA-ft*, which generalizes the *DA-bf*, is presented on the right-hand side of Fig. 1. A form type is a hierarchical structure of form type components. The form type *Donation Agreement* (Fig. 1) has two component types: *Agreement Heading* and *Donated Items*.

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.



Business form *Donation Agreement* Form type *Donation Agreement*

Fig. 1 The business form *Donation Agreement* and its form type

A form type in IS design by means of IIS*Studio has a dual role. On the one hand it provides an important input data for database design, and on the other hand it is a source for the generation of a sole transaction program and its screen or report form. IIS*Studio introduces *FT data model* based on FT concept [6] aimed at conceptual database design.

IIS*Studio comprises: IIS*Case, IIS*UIModeler, and IIS*Ree tools that communicate by means of shared repository aimed at storing project specifications. The IIS*Case tool supports IS forward engineering process. The IIS*UIModeler is aimed at modeling of graphic user interface (GUI) static aspects via UI templates. The IIS*Ree tool enable reverse engineering (RE) of relational databases to conceptual data models. The RE process is implemented by means of a series of M2M transformations between database models (*database model transformations*) based on meta-models that are conformed by the source and target database models. A blueprint of IIS*Studio database RE process is presented in [5]. Here we present one step of that process aimed at transformation of a generic relational database schema into a form type data model.

III. Classifications of Form Types and Relation Schemes

A *form type* $\mathcal{F}$ is a named tree structure, whose nodes are called *component types* (CTs). Let $C(\mathcal{F})$ denotes a set of CTs making up the form type $\mathcal{F}$. Each CT is identified by its name within the scope of a FT, and has nonempty sets of attributes and keys, and a possibly empty set of unique constraints. Formally, a CT is a named pair $N(Q, O)$, where N denotes name of the CT, Q is the set of CT attributes $Q=\{A_1, .., A_n\}$ and O is a set of CT constraints. O is a union of: a set of key constraints, a set of unique constraints and a singleton containing a tuple constraint. The tuple constraint of a CT refers to a set of attribute-based constraints (attribute data type specification and not-null constraint) paired with a tuple-based constraint (constraint on tuple value).

Let $C(\mathcal{F}) = \{N_i(Q_i, O_i) \mid i = 1, ... m\}$. $W(\mathcal{F})$ denotes a set of the form type attributes that satisfies (1) and (2).

$$\bigcup_{i=1}^{m} Q_i = W(\mathcal{F}) \text{ and} \quad (1)$$

$$(\forall N_i, N_j \in C(\mathcal{F}))(i \neq j \Leftrightarrow Q_i \cap Q_j = \varnothing). \quad (2)$$

Three categories of FTs can be identified: $\mathcal{F}\_Basic$—an elementary form type containing only one root component type (Fig. 2); $\mathcal{F}\_Tree^2$—a form type containing a root component type with only one child component type (Fig. 3); and $\mathcal{F}\_Tree^n$—a form type that apart from a root component type contains an arbitrary number of child component types (Fig. 4).

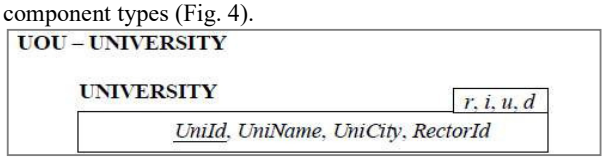


Fig. 2 An example of $\mathcal{F}\_Basic$ form type

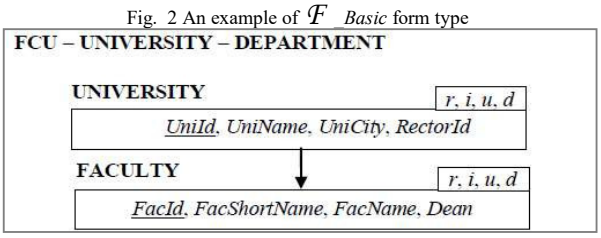


Fig. 3 An example of $\mathcal{F}\_Tree^2$ form type

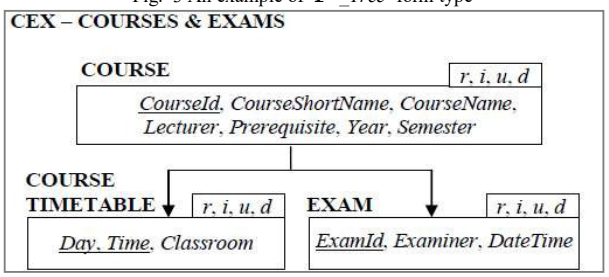


Fig. 4 An example of $\mathcal{F}\_Tree^n$ form type

Hammer et al. [7] have proposed a classification of relation schemes in the context of the transformation of a relational database schema into Entity-Relationship (ER) database schema. Here we present a classification that is adapted according to the target FT data model. There are three kinds of relation schemes: basic (BR); weak (WR); and all keys (AK) relation scheme. A BR relation scheme is a relation scheme whose PK does not properly contain a key attribute of any other relation. A WR relation scheme N satisfies the following three conditions: i) a proper subset of its PK contains key attributes of other basic or weak relation schemas; ii) the remaining attributes of its PK do not contain key attributes of any other relation scheme; and iii) it has an identifying parent relation scheme and properly contains the PK of its parent relation scheme. An AK relation scheme contains only key attributes of other relation schemes, and does not contain any other self-inherent attributes.

A graphic representation of a relational database schema is presented in Fig. 5. Underlined attributes belong to a key of a relation scheme. If a relation scheme has two candidate keys their attributes are underlined with different lines. The relation schemes *University* and *Project* are BR relation schemes. *Faculty*, *Department* (first version), *Employee*, the second version of *Department* relation schema (below the first version) and relation scheme *Lecturer* are WR relation schemes. The relation scheme *WorkOn* is an example of AK relation scheme.

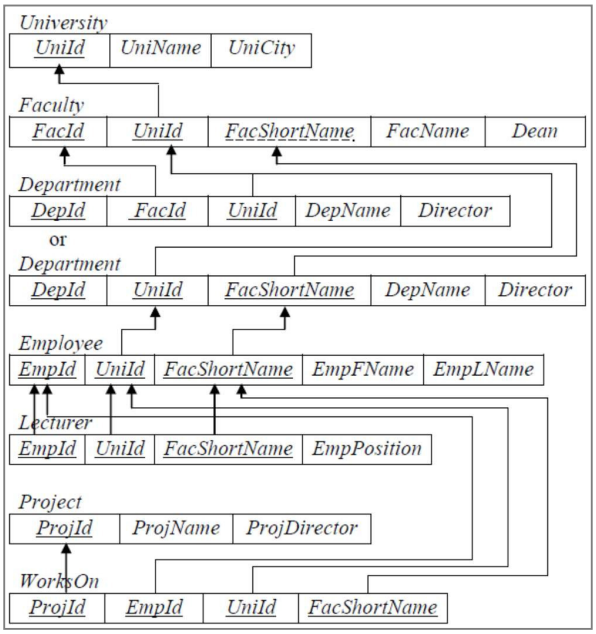


Fig. 5 An example of a relational database schema

IV. Transformation of a Generic Database Schema into a Form Type Data Model

The transformation of a model conformant with generic relational database meta-model into a model conformant with FT meta-model is PSM2PIM transformation. It generates all relevant combinations of form types. It is a user who chooses form types to be introduced in the form type data model. The remaining form types are deleted. The proposed transformation is carried out in three steps. In the first step, a $\mathcal{F}$ _Basic form type is created for each relation scheme from a relational database schema. The input parameters for this transformation are:

$$S = \{N_i(R_i, O^{RS}_i) \mid i = 1,.., n\}, \text{ and} \quad (3)$$

$$O^{RS} = K^{RS} \cup UQ^{RS} \cup CH^{RS}, \quad (4)$$

where S is a set of the relation schemes $N_i(R_i, O^{RS}_i)$. N_i is relation scheme name, R_i is nonempty set of its attributes and O^{RS}_i (4) is a union of three sets containing: key constraints, unique constraints, and tuple constraints.

In Fig. 6 parts of generic relational schema MM and FT meta-model are presented in the first row. In the next raw ATL rules are presented that specify a mapping between the concepts of these MMs alongside with five lazy rules aimed at mapping: optional and mandatory relation scheme attributes to optional and mandatory CT attributes; and relation scheme key, unique and tuple constraints to CT key, unique and tuple constraints, respectively.

In the next step of the transformation a $\mathcal{F}$ _$Tree^2$ form type is generated for each referential integrity constraint, having WR relation scheme on the left-hand side. The set of input parameters in addition to (3) and (4) contains a set of referential integrity constraints of a relational database schema (5):

$$RIC = \{ric_i: N_l[LHS] \subseteq N_r[RHS] \mid i = 1,.., m\}, \quad (5)$$

where N_l and N_r are relation schemes, *LHS* and *RHS* are subsets of attribute sets R_l and R_r of relation schemes N_l and N_r, respectively. For each ric_i from *RIC*, with N_l that is WR relation scheme, a $\mathcal{F}$ _$Tree^2$ FT is created.

In the last step of the transformation an $\mathcal{F}$ _$Tree^n$ form type is created for each relation scheme that is referenced by at least two WR relation schemes. Besides, an $\mathcal{F}$ _$Tree^n$ form type is created for each relation scheme that is referenced by at list one WR relation scheme that is referenced by some WR relation scheme, too.

V. Related Work

Hainaut et al. in [8] describe main steps of database RE. Vendor-specific physical or standard relational meta-model mainly are found on the source side of RE transformations. On the other side, ER [9], object-oriented [9]–[11], standard/vendor-specific relational [12] or object-relational [13] MMs occur on the target side. There are various research works about the use of forms in different contexts: Tsichritzis [14] introduces the concepts of form type, Shu [15] proposed using forms to specify system requirements, in [16] is presented a usage of business forms as input data for the process of database schema design. A form-based approach for reverse engineering of relational databases is proposed in [17].

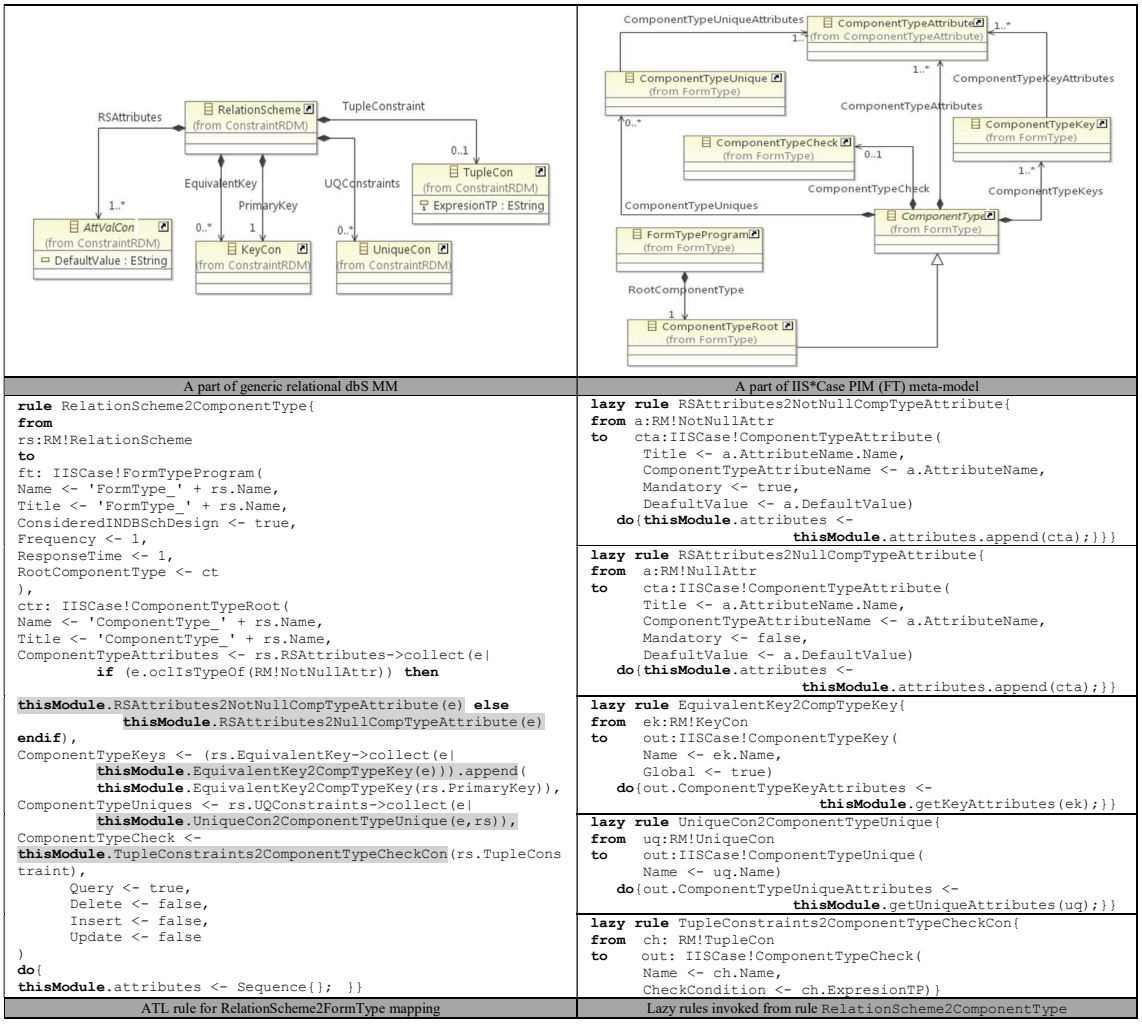
VI. Conclusion

The main reason to develop our IIS*Ree reverse engineering tool was to take advantage of our approach to database schema generation during: the integration of independently designed ISs, legacy database schema restructuring and improvement of empirically designed database schemas. FT specification 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. The MMs and models that we use in our approach are intensional data models. System evolution can be supported by automatic MD data migration and extensional database MMs could play important role in its implementation. Our future research has to consider extensional database MMs and possible usage of category theory [18] for PIM specification of model transformations in order to automate the process of database model transformations generation.

References

[1] J.M Favre, " Foundations of Model (Driven) (Reverse) Engineering: Models.", *Dagstahl Seminar Proceedings*, 2005.

[2] 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



A part of generic relational dbS MM

A part of IIS*Case PIM (FT) meta-model

```
rule RelationScheme2ComponentType{
from
rs:RM!RelationScheme
to
ft: IISCase!FormTypeProgram(
Name <- 'FormType_' + rs.Name,
Title <- 'FormType_' + rs.Name,
ConsideredINDBSchDesign <- true,
Frequency <- 1,
ResponseTime <- 1,
RootComponentType <- ct
),
ctr: IISCase!ComponentTypeRoot(
Name <- 'ComponentType_' + rs.Name,
Title <- 'ComponentType_' + rs.Name,
ComponentTypeAttributes <- rs.RSAttributes->collect(e|
        if (e.oclIsTypeOf(RM!NotNullAttr)) then

thisModule.RSAttributes2NotNullCompTypeAttribute(e) else
          thisModule.RSAttributes2NullCompTypeAttribute(e)
endif),
ComponentTypeKeys <- (rs.EquivalentKey->collect(e|
        thisModule.EquivalentKey2CompTypeKey(e))).append(
        thisModule.EquivalentKey2CompTypeKey(rs.PrimaryKey)),
ComponentTypeUniques <- rs.UQConstraints->collect(e|
        thisModule.UniqueCon2ComponentTypeUnique(e,rs)),
ComponentTypeCheck <-
thisModule.TupleConstraints2ComponentTypeCheckCon(rs.TupleCons
traint),
      Query <- true,
      Delete <- false,
      Insert <- false,
      Update <- false
)
do{
thisModule.attributes <- Sequence{}; }}
```

ATL rule for RelationScheme2FormType mapping

```
lazy rule RSAttributes2NotNullCompTypeAttribute{
from a:RM!NotNullAttr
to   cta:IISCase!ComponentTypeAttribute(
       Title <- a.AttributeName.Name,
       ComponentTypeAttributeName <- a.AttributeName,
       Mandatory <- true,
       DeafultValue <- a.DefaultValue)
   do{thisModule.attributes <-
                    thisModule.attributes.append(cta);}}
```

```
lazy rule RSAttributes2NullCompTypeAttribute{
from  a:RM!NullAttr
to    cta:IISCase!ComponentTypeAttribute(
       Title <- a.AttributeName.Name,
       ComponentTypeAttributeName <- a.AttributeName,
       Mandatory <- false,
       DeafultValue <- a.DefaultValue)
   do{thisModule.attributes <-
                    thisModule.attributes.append(cta);}}
```

```
lazy rule EquivalentKey2CompTypeKey{
from  ek:RM!KeyCon
to    out:IISCase!ComponentTypeKey(
       Name <- ek.Name,
       Global <- true)
   do{out.ComponentTypeKeyAttributes <-
                    thisModule.getKeyAttributes(ek);}}
```

```
lazy rule UniqueCon2ComponentTypeUnique{
from  uq:RM!UniqueCon
to    out:IISCase!ComponentTypeUnique(
       Name <- uq.Name)
   do{out.ComponentTypeUniqueAttributes <-
                    thisModule.getUniqueAttributes(uq);}}
```

```
lazy rule TupleConstraints2ComponentTypeCheckCon{
from  ch: RM!TupleCon
to    out: IISCase!ComponentTypeCheck(
       Name <- ch.Name,
       CheckCondition <- ch.ExpresionTP)}
```

Lazy rules invoked from rule `RelationScheme2ComponentType`

Fig. 6 Relation scheme – to – $\mathcal{F}$ _Basic Form Type transformation

[3] I. Dejanović, G. Milosavljević, B. Perišić, M. Tumbas, "A Domain-Specific Language for Defining Static Structure of Database Applications", *Computer Science and Information Systems*, (ComSIS), ISSN:1820-0214, Vol. 7, No. 3, pp 409-440. 2010.

[4] 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

[5] S. Ristić, S. Kordić, M. Čeliković, V. Dimitrieski, I. Luković, "A Model-driven Approach to Data Structure Conceptualization" in *Proceedings of the 2015 FEDCSIS„* Vol. 5, DOI: http://dx.doi.org/10.15439/ 978-83-60810-66-8, pp. 977–984. 2015.

[6] 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

[7] M. Hammer, M. Schmalz, W. O'Brien, S. Shekar, N. Haldevnekar, *Knowledge Extraction in the SEEK Project Part I*, Tecnical Report TR-02-008, July 2002.

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

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

[10] 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.

[11] 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.

[12] Beggar O. E., Bousetta B., Gadi T., Getting Relational Database from Legacy Data-MDRE Approach, Computer Engineering and Intelligent Systems ISSN 2222-1719 Vol 4, No.4, 2013.

[13] J. Vara, B. Vela, V.A. Bollati E. Marcos, "Supporting model-driven development of object-relational database schemas: a case study", in: *Theory and Practice of Model Transformations*, R. Paige (Ed.), Heidelberg, Springer Berlin, 2009. pp. 181–196.

[14] D. Tsichritzis, "Form management",. *Communications of the ACM* 25 (5), pp. 453–478. 1982.

[15] N.C. Shu "FORMAL: a form-oriented, visual-directed application development system", *Computer*, pp 38– 49. 1985.

[16] J. Choobineh, S.S. Venkatraman, "A methodology and tools for derivation of functional dependencies from business form", *Information Systems* 17 (3), pp 269–282. 1992.

[17] 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.

[18] W. Steingartner, and D. Radaković, "Categorical structures as expressing tool for differential calculus", In: Proceedings of: The 12th Conference Informatics'2013 Technical University of Kosice, Slovakia, pp. 77–82, 2013.