A DSL for EER Data Model Specification

Milan Čeliković
University of Novi Sad, Faculty of Technical Sciences
Novi Sad, Serbia
milancel@uns.ac.rs

Vladimir Dimitrieski
University of Novi Sad, Faculty of Technical Sciences
Novi Sad, Serbia
dimitrieski@uns.ac.rs

Slavica Aleksić
University of Novi Sad, Faculty of Technical Sciences
Novi Sad, Serbia
slavica@uns.ac.rs

Sonja Ristić
University of Novi Sad, Faculty of Technical Sciences
Novi Sad, Serbia
sdristic@uns.ac.rs

Ivan Luković
University of Novi Sad, Faculty of Technical Sciences
Novi Sad, Serbia
ivan@uns.ac.rs

Abstract

In this paper we present a domain specific language (DSL) for Extended Entity-Relationship (EER) data model approach, named EERDSL. EERDSL is a part of our Multi-Paradigm Information System Modeling Tool (MIST) that provides EER database schema specification at the conceptual level and its transformation into a relational data model, or a class model. EERDSL modeling concepts are specified by Ecore, one of the commonly used approaches to create meta-models. In the paper we present both textual and graphical notations of EERDSL. Since only few modeling constraints may be described at the level of abstract syntax, we use Object Constraint Language (OCL) to specify complex validation rules for EER models.

Keywords: domain specific language, Extended Entity-Relationship data model, databases, information systems, Ecore, object constraint language

1. Introduction

As the information technology constantly grows and evolves, new possibilities for its incorporation in information systems (ISs) continue to appear. There are many ways and approaches that may be deployed to improve the process of IS development. Some of them deploy domain specific languages (DSLs) [17], computer-aided software engineering (CASE) tools or Model Driven Software Development (MDSD) tools. Our focus is on the usage of DSLs based on model driven approaches. In our previous research we developed Integrated Information Systems Studio Tool (IIS*Studio) [1,13,14,15]. IIS*Studio is a software tool that provides a model driven approach to IS design. IIS*Studio, as a software tool assisting in IS design and generating executable application prototypes, currently provides: (i) Conceptual modeling of database schemas, transaction programs, and business applications of an IS; (ii) Automated design of relational database subschemas in the 3rd normal form (3NF); (iii) Automated integration of subschemas into a unified database schema in the 3NF; (iv) Automated generation of SQL/DDL code for various database management systems (DBMSs) [1]; (v) Conceptual design of common user-interface (UI) models; (vi) Automated generation of executable prototypes of business applications; (vii) Modeling check constraints and untypical
functionalities of business applications [14]; and (viii) Reverse engineering of relational databases to form type (FT) models [2]. Our focus in research is on the design and generation process of a database and application software of an IS.

One of the main motives for developing IIS*Studio is in the following. For many years, the most favorable conceptual data model is widely-used Entity-Relationship (ER) data model with its extensions. Majority of existing CASE tools provide as the first step creation of an ER database schema. The next step in the design process is the ER database schema transformation into the relational database schema. Such scenario has many advantages, but also there are serious disadvantages [12].

In order to provide design of various platform independent models (PIMs) by IIS*Studio, we created a number of modeling, meta-level concepts and formal rules that are used in the design process. We have also developed and embedded into IIS*Studio, tools that apply such concepts and rules. They assist designers in creating formally valid models and their storing as repository definitions. The main features of IIS*Studio may be found in [6]. There is a strong need to have PIM concepts specified formally in a platform independent way. PIM concepts need to be fully independent of repository based specifications that typically may include some implementation details. Our research is based on the Meta Object Facility (MOF) 2.0 [18] to formally describe IIS*Studio PIM concepts. As we could not find standardized implementation of MOF, we decided to use Ecore meta-meta-model, which is provided by Eclipse Modeling Project (EMP) [8]. Besides, created meta-model can be used for the software tool verification in EMP environment and providing software documentation in a formal way. It also represents an abstract syntax for IIS*CDesLang – a textual domain specific language presented in [6], [16]. EMP is the Eclipse environment that provides software development process based on model driven approach (MDA). With the improvement of EMP, we decided to implement some of the existing IIS*Studio functionalities, as the next logical step in the evolution of IS development methods, using this technology. Besides the IIS*Studio functionalities, we have decided to support database schema design based on the Extended Entity-Relationship (EER) data model, as a commonly used, traditional approach. Therefore, we have developed a DSL for the specification of EER database schema specifications, named EERDSL. Although there is a plethora of EER modeling tools, we have decided to develop our own DSL, as we need to integrate EER and FT approaches. The FT approach is supported by a DSL developed using Ecore in the Eclipse environment. The highest level of integration is only possible if EERDSL is also developed using Ecore, allowing the usage of the same repository used by the FT approach. As EER approach is present in almost every book on databases, we believe that our DSL may also be used for educational purposes, such as learning about: (i) EER concepts and developing a database specification at the conceptual level; and (ii) MDSD approach by means of the EER approach the students are familiar with, since it is extensively taught in the previous database courses. FT and EER approaches of our new Eclipse-based tool both provide conceptual database schema modeling. The tool is named Multi-paradigm Information System modeling Tool (MIST). In MIST, both approaches may be used simultaneously. We developed transformations that provide EER models to be transformed into FT models. For both FT and EER models, we provide in MIST a transformation into a relational data model. In this paper we present MIST tool with a focus on EERDSL. EERDSL provides database schema modeling at the conceptual level using graphical and textual notation. EERDSL modeling concepts are represented by the graphical notation proposed by Thalheim in [25]. We have specified textual notation in our own way as there is no standard for it.

Apart from Introduction and Conclusion, the paper is organized in four sections. Related work is elaborated in Section 2. In Section 3 we present the architecture of MIST. In Section 4 we give a presentation of EER meta-model implemented in EMP environment. In Section 5, we give a concrete syntax specification of EER concepts. We present textual and graphical notation of EERDSL, with an example of EERDSL usage.
2. Related Work

EER is one of the commonly used approaches for the database schema design at the conceptual level. PowerDesigner [23], ERWin [4], SmartDraw [22], Oracle Designer [21], or Cameo Data Modeler [5] for MagicDraw are commercial CASE tools that support EER approach. These tools support only graphical notation according to one of the well-known notations, such as Chen’s [7], Elmasri-Navathe’s [9], Barker’s [3], or Merise [24]. Also, all of the aforementioned CASE tools are built as the stand-alone tools. The EERCASE tool presented in [11] is integrated with the Eclipse environment. It provides all of the EER concepts represented with Elmasri-Navathe’s graphical notation.

Our tool is integrated with the Eclipse environment. EERDSL component of our tool provides database schema specification using concepts from the EER approach. The concepts are represented by graphical notation represented by Thalheim. Apart from graphical notation, provided by all of the aforementioned tools, our tool also provides EER database schema modeling with a textual notation. As there is no standard for EER textual notation, we have specified textual notation in our own way.

3. The Architecture of MIST

In Fig. 1, we present the architecture of the MIST tool. The tool contains following components: FTDSL, Synthesis, Business Application Generator, EERDSL, EER2Rel, EER2Class, SQL Generator, Java Generator, and R2FT.

![Fig. 1. MIST architecture](image)

FTDSL component comprises a textual domain specific language having the abstract syntax specified by Ecore meta-meta-model. FTDSL allows a designer to model an IS at a conceptual level using FT concepts. IS model based on the FT concept includes not only the database schema of an IS, but also business applications and their graphical user interfaces (GUIs). After the process of the conceptual IS modeling, an IS designer may generate a model of a relational database schema, using the synthesis algorithm. The Synthesis component is an improved synthesis algorithm presented in [12], [19]. The improved synthesis algorithm takes a Universal Relation Schema (URS) expressed by FT concepts as an input and produces a relational database schema as an output. The MIST tool contains the Business Application Generator component, as the FTDSL component may be used to specify business applications of an IS. This component takes an FT model as an input and produces Java code of a modeled business application as an output. As the FT specification is enriched with GUI details, the generated application prototype may be executed and used to perform basic operations over the database.

In order to provide a conceptual specification of an IS database model, MIST comprises EERDSL component. EERDSL includes both graphical and textual domain specific languages. EERDSL abstract syntax is specified by Ecore meta-meta-model. The most IS designers favor one of the well-known EER graphical notations for the database modeling process. The other group of IS designers are more familiar to modeling based on the modeling concepts represented in a textual way. In order to support both groups...
of the designers, we developed the DSL that provides database modeling in both ways. The IS designer is able to use both model representations at the same time. By this, two different viewpoints over the same model are provided in MIST. EER module provides the generation of the relational database model and the class model from EER database model.

EER2Rel component of MIST provides transformation of EER model to the relational data model. Models being transformed conform to the EER meta-model and relational meta-model, respectively. The relational data model may be further used in the process of SQL code generation. For this purpose, the SQL Generator component is developed.

EER2Class component of MIST provides a transformation of an EER model to a class model. The class model may be used to generate code in some of the object-oriented programming languages. Our Java Generator component is used to generate Java code from the provided class model.

In order to provide reverse engineering of a relational database model to the FT model, R2FT component has been developed. The component comprises a transformation specification from the relational data model to the FT model. Also, this component may be used to transform an EER database model to the FT model through the relational data model.

4. Extended Entity-Relationship Meta-Model

In this section we present concepts of our EER PIM meta-model specified by Ecore meta-meta-model. Modeling concepts in the EER data model are: entity, n-ary relationship, IS-A relationship, identification dependency relationship, categorization relationship, gerund, attribute, key, and domain. In the rest of this section, we present the names of meta-model and model concepts in brackets and italic. The root concept in our meta-model presented in Fig. 2 is the model (EERModel). Each EER model comprises zero or more entities (Entity). Entity concept is used to model a class of real observed entities in the specified IS. In some approaches, the Entity concept is named Entity Type concept. Each entity may contain zero or more attributes (Attribute). The Attribute concept is used to model a property of selected real entities in the designed IS.

A domain (Domain) represents a set of allowed values that an attribute may have. Each domain is based upon a primitive domain (PrimitiveDomain), such as integer, string, real, boolean, date, and time. Each attribute is associated with exactly one domain.

---

**Fig. 2.** The EER meta-model
(AttributeDomain). For each attribute, length and default value may be specified. Therefore, in order to provide domain reusability, domains may be specified once at the level of EER model, and further reused and restricted at the level of attributes. An entity specification requires a specification of the set of keys (Key). Each key is specified as a set of attributes of the entity. Only one key is declared as the primary key.

The EER data model provides modeling different relationship types. Relationships between entities are classified as: n-ary relationship (Relationship), identification dependency relationship (IdentificationDependency), IS-A relationship (ISA), and categorization relationship (Categorisation). N-ary relationship represents an association between two or more entities. For each entity in the n-ary relationship (RegularEntity) its role, minimum (MinCardinality) and maximum (MaxCardinality) cardinalities need to be specified. A minimum cardinality may have one of two values: one or zero, while a maximum cardinality may be provided with the values of one or more. For each n-ary relationship, a designer may specify attributes. Gerunds are used to specify the rules, by means of just selected entity instance combinations of some entities may be used in relationships with other entity's instances. The main features of different relationship types may be found in [25].

In order to express more complex constraint rules in EER models, we use Object Constraint Language (OCL) [20]. By OCL we specify constraints that cannot be expressed by means of Ecore modeling concepts. First we have to specify OCL invariant that enables checking if the primary key is defined at the level of an Entity. Only when an entity is a subtype in an IS-A relationship, the primary key may be unspecified. At the level of the ModellingConcept class, an OCL invariant supports checking the uniqueness of all its instances. In an IS-A relationship, the classification attribute is selected from the set of supertype entity attributes. In the Categorisation class, OCL invariant supports checking the difference between category and categorization entities. At the level of identification dependency relationship, an OCL invariant checks if weak and regular entities are different. The following OCL invariant enables checking if the sequence of attributes in the key belongs to the set of the attributes of that entity:

```java
invariant invalidAttributeInKey:
  if(not keySet.keyAttributes->isEmpty()) then
    if(entityAttributes->notEmpty()) then
      entityAttributes->includesAll(keySet.keyAttributes)
    else
      entityAttributes->notEmpty()
    endif
  else
    keySet.keyAttributes->notEmpty()
  endif
```

5. EERDSL Textual and Graphical Notations

A specification of the concrete syntax is one of the important steps in development of a DSL. Our goal is to develop a DSL in support of the IS design process. Therefore, we have specified a meta-model that defines the structure and semantics of necessary concepts of our EERDSL language. Such meta-model specification represents, at the same time, the abstract syntax of EERDSL, aimed at conceptual modeling of database schemas.

There are numerous tools for the DSL development. We have used our meta-model specified by Ecore in EMF as an abstract syntax specification. By means of Eclipse plug-ins named Xtext [26] and Eugenia [10], we have generated the concrete syntax of EERDSL. In this section we present only a part of the concrete syntax grammar rules.

EERDSL textual notation is generated in the XText plug-in. It is expressed by Extended Backus–Naur Form (EBNF). In this section, we present textual notation rules for the selected EER PIM concepts only.

Production rule for defining an EERModel is:

```plaintext
EERModel returns EERModel:
 'EERModel' name=EString '{'
 ('domains' '{' domains+=Domain (',' domains+=Domain)* '}' )? ```
This rule specifies a name of a model (name), possible domains (domains), different entities (entities), gerunds (gerunds), and relationships (relationships) created in the context of a model.

Eugenia plug-in provides generation of a graphical notation. The EERDSL graphical notation is expressed by graphical symbols that provide graphical language representations. In Table 1, we present the symbols of the EERDSL graphical notation only for the main EER PIM concepts. All concepts are represented with widely used graphical notation presented by Thalheim in [25].

<table>
<thead>
<tr>
<th></th>
<th>Entity</th>
<th>Attribute</th>
<th>Domain</th>
<th>Gerund</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-ary</td>
<td>entityName</td>
<td>attributeName</td>
<td>domainName</td>
<td>gerundName</td>
</tr>
<tr>
<td>relationship</td>
<td>relName</td>
<td>(a,b)</td>
<td>(a,b)</td>
<td>EX-OR</td>
</tr>
<tr>
<td>IS-A</td>
<td>IS-A</td>
<td>(b)</td>
<td>(a,b)</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>ID</td>
<td>(b)</td>
<td>IDName</td>
<td></td>
</tr>
<tr>
<td>relationship</td>
<td>Categorization</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In Fig. 3, we present a fragment of a model specified using a textual and graphical notation of EERDSL. It represents a part of faculty EER database schema, named FacultySystem. In the left part of the Fig. 3, a textual model is presented, while in the right part, a graphical notation is depicted. First, we have created an instance of the EERModel concept, named FacultySystem. In the example shown in Fig. 3, model FacultySystem comprises four domains: integer, varchar, date, and time. We have also specified entities: Student, Teacher, Subject, Chair and Faculty. Each student has attributes: studentID, studentsYear, studentName, and studentSurname. KeyStudent represents the primary key of Student entity, comprising attribute StudentID. Subject entity has attributes: subjectID and subjectName. KeySubject is the Subject primary key composed of the attribute subjectID. Teachers of a faculty are modeled by the Teacher entity, having the attributes teacherTitle, teacherID, teacherName, and teacherSurname. A relationship between teachers and subjects taught is modeled by TeachesClasses. Each teacher teaches one or more subjects, while a subject is taught by one or more teachers. The relation between students and subjects is modeled by the Takes relationship. Each student may attend one or more subjects, while a subject may be attended by zero or more students. Relationship Grades models students' grades given by teachers. As only a teacher that teaches a subject may grade students attend that subject, relationship Grades must relate relationships Takes and TeachesClasses. Therefore, relationships Takes and TeachesClasses are created as gerunds. Each student that attends a subject may be graded by exactly one teacher teaching the subject. A teacher teaching a subject may grade zero or many students on the subject. Each grading has examDate and grade attributes. Relationship between teachers and chairs is modeled by WorksAt relationship. Each teacher works at exactly one chair, while at one chair may work one or more teachers. Each chair is identified by its name and the faculty's facultyID attribute. Therefore, a relationship IsPartOf, between Chair and Faculty, is modeled by the identification dependency relationship.
EERModel FacultySystem {
  domains {
    Domain int primitiveDomain integer,
      ...domains varchar, date, time
  }
  entities {
    Entity Student {
      attributeSet {
        Attribute studentID domain int,
        Attribute studentYear domain int
      }
      keySet { keyStudent (studentID) }
      primaryKey keyStudent
    }, ...entities chair, faculty, subject, teacher
  }  //... entities in relationship
  gerunds {
    Gerund Relationship Takes {
      entitiesInRelationship {
        Student (one,many), Subject (zero,many)
      }
    }, ...gerund TeachesClasses
  }  //... relationships in relationship
  relationships {
    Relationship Grades {
      entitiesInRelationship {
        Takes (one, one), TeachesClasses (zero,many)
      }
      attributeSet {
        Attribute grade domain int, Attribute examDate domain Date
      }
    }, IdentificationDependency IsPartOf {
      weakEntity Chair, regularEntity Faculty (one,many)
    }, ...relationship WorksAt
  }
}

Fig. 3. Example of Faculty IS specification in EERDSL

6. Conclusion

In this paper we presented a DSL for EER data model specification, named EERDSL. Through our previous research we developed the MIST tool. It provides an IS design approach based upon the usage of FTDSL. Our intention was to provide database designers an opportunity to use EER data model, as a commonly used, traditional approach. First of all we needed to create the EER meta-model specified by Ecore that actually represents the abstract syntax of the language. At the level of meta-model, we specified OCL constraints in order to support constraints that cannot be expressed by means of Ecore modeling concepts. Then, we created both textual and graphical notations for EERDSL. In this way we support both group of designers, one preferring graphical modeling languages, and the other preferring textual programming languages. After the database schema specification at the conceptual level, MIST may be used for generation of the relational database model and a class model. As a final product, MIST provides a generation of SQL scripts and procedural code for inverse referential integrity constraints, from a relational model. The tool supports generation of Java classes from a class model.

In our further research, we plan to perform an empirical case study in order to compare FTDSL and EERDSL. To meet this goal, it is necessary to collect practical experience and lessons learned from IS designers. We need information about the amount of generated code, development process speed, and user satisfaction using both of these approaches. We also plan to introduce MIST to our database and MDSD university courses. Our intention is to validate the tool and the approach after gathering enough user feedback data. Another research direction is to extend EERDSL with new concepts allowing more detailed specifications of database models at the conceptual level. The concepts should provide constraint specifications. As we already have provided a conceptual specification of the check constraint at the level of FT models, we plan to create the appropriate formalisms for its specification at the level of EER.
model, too. Currently EERDSL provides graphical notation represented by Thalheim in [25]. We also plan to develop and embed into EERDSL some other widely used graphical notations, such as Merise or Elmasri-Navathe’s. In that way, we provide IS designers to become familiar with various EER graphical notations, which is particularly important in the education process.

Acknowledgment

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

References