Using DSM to Generate Database Schema and Data Management

Jaroslav Zacek¹, Zdenek Melis², Frantisek Hunka², Bogdan Walek¹

¹ Centre of Excellence IT4Innovations, Faculty of Science, University of Ostrava
Dvorakova 138/7, 701 03, Ostrava, Czech Republic
jaroslav.zacek@osu.cz, bogdan.walek@osu.cz

² Department of Computer Science, Faculty of Science, University of Ostrava
Dvorakova 138/7, 701 03, Ostrava, Czech Republic
zdenek.melis@osu.cz, frantisek.hunka@osu.cz

ABSTRACT

Current technologies based on DSM allow creating modeling tools on any domain. In the field of economics and business management these tools are invaluable part especially in the area of business process modeling. Created models, however, are not able to work with enterprise data due to the variable structure and they cannot perform analysis or simulations. This paper focuses on the use of DSM generator to create a database based on the created model. Along the database the environment enabling easy handling of corporate data is generated.

KEYWORDS

Modeling; Simulation; Domain-specific modeling; DSL; REA ontology; REA model level; source code generation; database

1 INTRODUCTION

Domain-specific modeling (DSM) is a software engineering methodology narrowly focused on one particular specific domain. Narrow focus allows working with domain terms including their meaning. Based on these terms the domain-specific language containing the basic syntax and semantics of the language is created. Another part of the DSM is a generator and domain framework. The generator converts the created model to a specific syntax. Domain framework forms a layer between the generated code and the existing code in the target environment, and reduces the complexity of the code [1].

The aim of modeling tool is the creation of valid models. These models illustrate a reality, but for their wider use for example in simulation, analysis or report generation areas it is necessary to create a model able to work with business data. DSM is used primarily to create modeling tools of different domains. The variability of these domains does not allow generalizing the interconnection of created models with the database.

This paper describes the way to connect created model with the database by using the source code generator. In order to change the basic model created by domain-specific modeling tool into interactive and able to work with enterprise data, it is necessary to generate not only a database schema but also scripts allowing operations with data and the user interface capable of applying these scripts.

2 THE REA ONTOLOGY DOMAIN-SPECIFIC MODEL

An example of domain that requires working with enterprise data is the REA enterprise ontology. REA Ontology (Resource, Event, Agent) is a concept for creating business infrastructure design based on ownership and exchange [8]. It is based on the concept of economic exchanges, increasing the enterprise value [5].

According to the level of abstraction the REA ontology can be divided into 4 levels. The most important level is the REA model level. It describes the various business processes. At this
level the most important information about a company are modeled, because it answers the question why the process is carried out. The REA model level shows different changes in the value of resources that are based on concepts of economic exchanges and conversions increasing the enterprise value [9]. Concepts of this level are divided by functionality into two groups - the operational level and the policy level. The operational level is the basis of the model. It describes events that have already happened. The basis of the operational level is formed by concepts (resource, event, agent) and semantic abstractions (exchange and conversion), which increase the value of the company. The policy level represents what could, should, or should not happen. It contains concepts and semantic abstractions defining rules, schedules, contracts, and other possible extensions of the model such as grouping, typification, or commitments [3].

Fig. 1 shows a sample model of the purchase process, including basic attributes. Among agents (seller and buyer) the contract was concluded. This contract generates commitments for both parties. These commitments represent a promise or an obligation to perform an economic event in a specific time. Domain-specific modeling allows creating modeling tools over narrowly specified domain. Combining REA ontology and DSM the modeling tool was created. This tool allows the creation of valid models of business processes and their subsequent use [10]. The domain language is the most visible part of the modeling tool. Other parts of the DSM are the generator and domain framework. The generator allows generating outputs from created models. The template defines outputs and it can be almost anything - reports, tests, analysis, simulations, source code, etc. Domain framework includes support methods for the generator such as integration with existing code, removing duplication in the generated code, hiding the target environment, etc.

3 DATABASE GENERATION PROCESS

In order to work with the enterprise data in the model, it is necessary to ensure their persistence. It is not possible to use the static database due to model variability. Therefore it is necessary to
create new database with the corresponding structure for each model [7]. The process of creating such a database can be fully automated using the template generator.

The general procedure of the REA model transformation into a relational database schema was described in the paper [6], and the transformation of selected ontology concepts in [4]. The mapping of the model into a relational database schema is performed in 6 steps [2]:

1. Create a table for each table – Each entity in the model creates a specific table in the database independently of a type (resources, events, agents).
2. Create a table for each many-to-many relationship – Based on the cardinality binding declared in the REA model, a connecting table in the database must be created.
3. Examine tables with one-to-one relationship – In this phase it must be decided which tables are useful to keep and which can be merged with others.
4. Identify the attributes and assign the primary key for each table – In this phase the structure of individual tables is created based on model elements. Choosing a useful attribute is important for capability to provide relevant data from the database. The primary key is a unique table identifier.
5. Implement relationships using foreign keys – As soon as the primary keys are identified, foreign keys are defined to a mutual table connection between individual tables.
6. Normalization – The last step of mapping of the relation database model is a database structure transformation to the third normal form.

Multiplicity of links is statically defined in the metamodel of the ontology. Metamodel is the basis of the modeling tool and forms important part of domain-specific language (DSL). It contains structure of elements, links and rules.

REA ontology does not use directional links. This can cause complications in the determination of foreign keys. The notional direction of links should therefore correspond to semantic meaning of elements of the ontology. The way to store a reference to information must correspond to the logical design and to the way of using the information system. Table 1 presents the list of foreign keys in each element of the ontology.

<table>
<thead>
<tr>
<th>Element</th>
<th>Foreign keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>-</td>
</tr>
<tr>
<td>Event</td>
<td>Resource, Agent, Event</td>
</tr>
<tr>
<td>Agent</td>
<td>-</td>
</tr>
<tr>
<td>Resource type</td>
<td>Resource</td>
</tr>
<tr>
<td>Event type</td>
<td>Event</td>
</tr>
<tr>
<td>Agent type</td>
<td>Agent</td>
</tr>
<tr>
<td>Commitment</td>
<td>Resource type, Agent type, Event type, Resource, Agent, Event, Commitment</td>
</tr>
<tr>
<td>Contract</td>
<td>Commitment, Agent, Agent type</td>
</tr>
<tr>
<td>Schedule</td>
<td>Commitment, Agent, Agent type</td>
</tr>
</tbody>
</table>

Template engine T4 performs the source code generation procedure (Text Template Transformation Toolkit), which is part of Visual Studio. T4 ensures operation of all DSM generators. This engine transforms the created template into C# class containing created methods and code fragments. This class is compiled and executed. The output is generated code.

Generator's script reads the structure of the model (business process) and extracts necessary information about structure, relationships and parameters. Based on this information SQL statements are successively generated. Generator creates all required statements for creating tables in database and for data manipulations (SELECT, INSERT, DELETE and UPDATE) and their variations to match needs of information system.

Fig. 2 shows an example of the script to create database tables. Methods called by this script are implemented in the support class. It makes these methods useable for all parts of generator. This way of implementation allows saving of extracted and processed data in memory, which greatly speeds up the overall code generation (for example searching foreign keys for the specific element is performed only once, but results can be
used repeatedly by different parts of the generator). As mentioned before, the script of generator consists of several files (sub-generators), where each file is responsible for generating the one part of the output. This division allows easy management of running generator and its possible extension and modification. Currently the created script contains 15 files for generating various parts of the output. The most important parts are described on table 2. The other files generate additional variations and extensions of SQL scripts.

Table 2. Parts of the generator

<table>
<thead>
<tr>
<th>Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ControlGenerator</td>
<td>It provides flow control of the generator. It runs sub-generators, provides data transfer and file management.</td>
</tr>
<tr>
<td>Extractor</td>
<td>It includes methods for extracting data from the model and their further processing.</td>
</tr>
<tr>
<td>SupportMethods</td>
<td>It includes support methods for other sub-generators (such as modification of table names).</td>
</tr>
<tr>
<td>CreateTable</td>
<td>Generates SQL statements to create and remove the database structure.</td>
</tr>
<tr>
<td>InsertData</td>
<td>Generates SQL commands to insert data into the database.</td>
</tr>
<tr>
<td>DeleteData</td>
<td>Generates SQL commands to delete a record from the table.</td>
</tr>
<tr>
<td>UpdateData</td>
<td>Generates SQL commands for updating of data in tables.</td>
</tr>
<tr>
<td>SelectAll</td>
<td>Creates SQL statements for selecting data from different tables.</td>
</tr>
<tr>
<td>SectionWhere</td>
<td>Generates different variations for additional specification of data selection.</td>
</tr>
</tbody>
</table>

When all parts of SQL statements are generated, domain framework removes previous database (the previous database is replaced by the new one every time the model changes) and runs the script to create a new one. Then it will start the user interface generator.

4 ENVIRONMENT GENERATION PROCESS

In order to use the generated SQL scripts, it is necessary to create a basic user interface. It is the task for the domain framework and generator. This interface allows the user to interact with the data over the created model. Basic functions of this
interface include adding, editing and viewing data, its analysis, visual representation (graphs), simulations and print option.

The most of the applications is static - completely independent of the model. Namely the internal logic of the application, the structure and appearance of user interface, mathematical-economic core for analytical, statistical and simulation operations, and reports and print functions. The generator generates only parts of the application working with the data model (using generated scripts) and parts depended on the interconnection and data structure. The generator does not have to create the entire application, but only a small portion and the rest creates the domain framework. It will also ensure the integration of the generated code with static code and prepare the application for the usage.

Every time when the original model is changed the entire application must be newly generated. Therefore the domain framework must remove previous version of the application.

![Figure 3. Application generated by automated process](image)

Fig. 3 illustrates an example of the generated application that is able to work with data of the model. To facilitate the implementation each element of the model has assigned a unique ID. With this ID, elements are clearly recognizable even in the case they have same name. After selecting the element some data from the database are displayed. This process uses generated scripts. Any change of these data is automatically reflected in the database. The application includes a simple data analyser that clearly displays the data in graph form - there was created 7 modules for data analysis (XY graph, pie chart, bar chart, linear regression graph, basic statistics, financial balance and dualities overview). Other modules can be easily added and their creation is identical to the creation of similar modules for desktop applications.

5 FUTURE WORK

The disadvantage of this approach is that we need to create a new database every time when the model is changed. The user will lose all data inserted into the database. In the case of minor changes (such as the change of an attribute, adding / removing an element, etc.), when a substantial portion of the data is unchanged, it would be good to edit a specific table in the database (the ALTER TABLE statement) instead of generating the new database.

The solution would be as follows:

The user chooses the generator (creating the new DB / editing the original one) by the new element inserted into the toolbox. If the user inserts it into the model, the generator uses a template for editing database (due to comparison of the old model with the new one, the SQL statement to change the database is generated). If the model does not contain this element, the standard process of generating the database is performed. Modifying of an existing database structure is possible only in case of minor changes; in the case of major changes (e.g. data type change) the modification is not possible.

The other issue that must be resolved is considering the object data type. The current version of application replaces any object by a String object because it is not able to define its internal structure. The solution would be to create separate tables for each object (in the case of the relational database) or create an empty object (if the object-relational or the object database is used).

5 CONCLUSION

This paper introduced a simple way how to automatically transform models into simple but usable information systems. This procedure uses the DSM generator to create the database and
applications that can work with it. The first prototype of the application brings the environment to work with data. It allows viewing and editing data, moreover the software also use graph views, analysis, simple statistics and simulations. The first part of the paper introduced the DSM technology and the REA Enterprise ontology. Then it describes the way to generate a database, SQL scripts for working with data and the individual sub-generators. The second part of the paper described how to handle database data by automatically generated application. This application can be extended and adjusted to the needs of the business.

Acknowledgement: This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

6 REFERENCES