MD3 – Integrated Model-Driven Data Design for Objects, XML, and Relational Databases

Darius Šilingas

UAB „Baltijos programinė įranga“ mokymų skyriaus vadovas
No Magic Europe, Training Department Manager, PhD
Savanoriu av. 363, LT-49425 Kaunas, Lithuania
Tel.: +370 37 705899
E-mail: darius.silingas@nomagic.com

The Model-Driven Architecture (MDA) paradigm promotes raising the level of abstraction and development efficiency by leveraging visual modeling instead of textual programming as the main means for producing software artifacts. In this paper, we focus on applying the MDA approach to data design. We introduce an integrated model-driven data design (MD3) framework, which consists of the data design workflow definition, a small subset of UML for conceptual data modeling, UML profiles for representing XML and relational database schemas, verification rules for checking model completeness and correctness, transformations between data design abstraction layers, and a customized integrated modeling environment. Application of the MD3 framework is illustrated by a small representative data design sample from the library domain. The MD3 framework is a conceptual starting point for developing more specialized and formalized data design methods and tools.

Introduction

Model-driven development is a new software development paradigm, which was introduced at the beginning of the XXI century. Probably the best-known version of this paradigm is Model-Driven Architecture (MDA) (OMG, 2003) that has been proposed and supported by the Object Management Group (OMG). MDA promotes the development of software based on models specified in the Unified Modeling Language (UML) (OMG, 2009) or other visual modeling languages. This approach aims to raise the level of abstraction compared to the usual programming in textual languages like Java, C++, and others. MDA raised a lot of hype in the software industry and was considered to be a potential silver bullet (Brooks, 1987), which should allow increase of the software development efficiency by an order of magnitude. However, now the hype is over, and although it is possible to say that the MDA approach has been reported as a success in a number of projects, in general it is facing a lot of challenges and is far from achieving its early promise (Thomas, 2004; France, Rumpe, 2007). The basic problem of MDA is that its scope is very wide – the modeling can be used in all the major software engineering activities, such as requirement analysis, design, programming, and testing (Silingas, Vitiutinas, 2007). However, the MDA solutions leading to the executable software are typically centered on a specialized well-defined solution domain. One of the areas where MDA is very natural to apply is data design, which has long traditions of separating conceptual, logical, and physical data views as well as using graphical entity relationship diagrams in relational data theory (Chen, 1976). However, recently the object-oriented and XML-oriented data design has been the major alternative techniques that are heavily used in practice. It is typical to integrate systems using objects, relational databases, and XML for data persistence. A certain research has been pursued on implementing the model-driven development

of relational databases (Silingas, Kaukenas, 2004) and XML schemas (Bernhauer et al., 2007). However, the optimal solution for data modelers would be an integrated environment for the model-driven development of object, relational and XML data structures. This could help the practitioners to survive in a multi-platform data design problem, which is sometimes called a *Bermuda triangle of data design*. In this paper we propose a framework MD3 (it stands for Model Driven Data Design and emphasizes data design application to 3 solution domains: objects, XML, and relational databases), which enables integrated data modeling and transformations between abstraction levels. A prototype of an integrated modeling environment supporting the MD3 framework was implemented as a plug-in to MagicDraw UML, which provides a rich set of features for creating a domain-specific modeling environment based on UML profiles (Silingas et al., 2009).

To illustrate the proposed schema, we will present examples for each step based on a fictitious library information management system *MagicLibrary*, which is used as a case study for MagicDraw UML R&D and professional trainings and consultations. The first artifact – a domain ER model – can be represented in UML as a very simple class diagram providing a visual dictionary for domain concepts (see Figure 2 for an example).

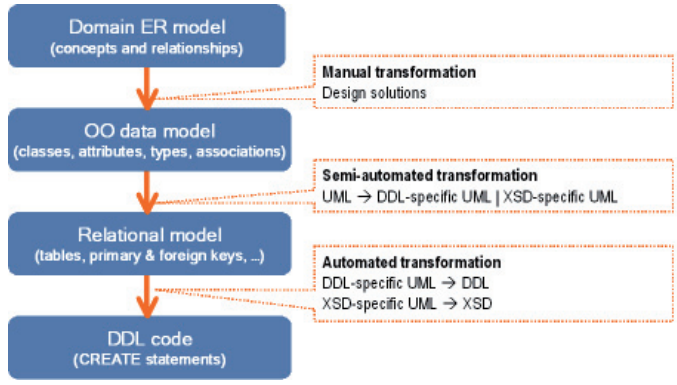

Figure 1. ***Travelling through abstraction levels for data design***

MD3 Framework: Principles, Samples, and Integrated Modeling Environment

MDA introduces different abstraction levels for models: computation independent model (CIM), platform independent model (PIM), and platform specific model (PSM). The main idea is to use automated transformations to get from PIM to PSM (see Figure 1) and from PSM to the code. In the data design world, we proposed to use a specialized version of MDA abstraction levels framework, which is presented in Figure 1.

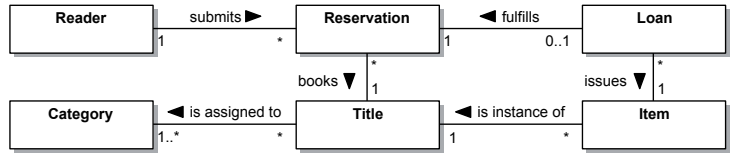

Figure 2. ***A fragment of CIM for MagicLibrary – domain concept relations***

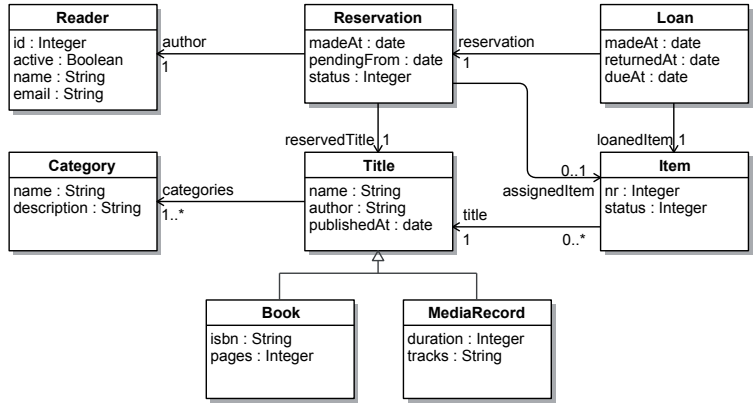

Figure 3. ***A fragment of PIM for MagicLibrary – object-oriented data model***

To go over to the next abstraction level – PIM, we propose to use the object-oriented paradigm with a more detailed class modeling in UML (see Figure 3). The first version of this abstraction level can be derived automatically from CIM, but it is very typical to introduce the class properties, add specialized or intemediate concepts to address design issues and ensure completeness of data model. Further on, this PIM model can be automatically transformed into a DDL-specific model (see Figure 4), and an XML schema-specific model (see Figure 5). The major difference between those two PSM models is their metamodels that in the case of MD3 are implemented as UML profiles providing different sets of stereotypes applicable to UML concepts used for data modeling. The PSM models can be transformed into an XML schema or DDL code using a completely automated model to code transformation. Small fragments of the generated DDL and XML schema code are presented in Table 1.

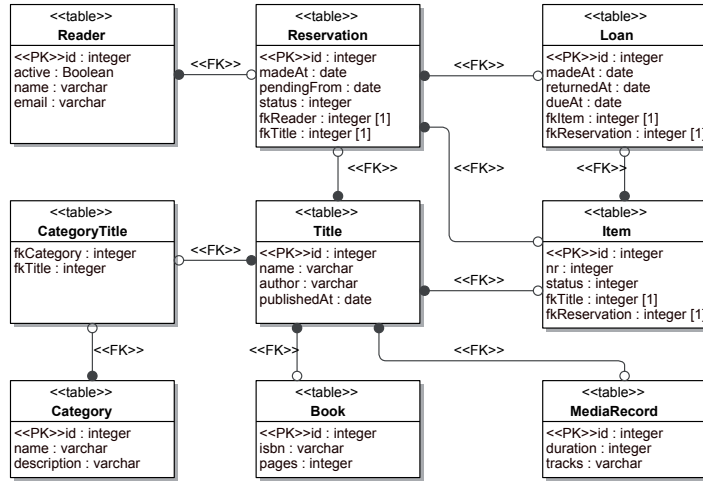

Figure 4. ***A fragment of PSM for MagicLibrary – relational database schema model***

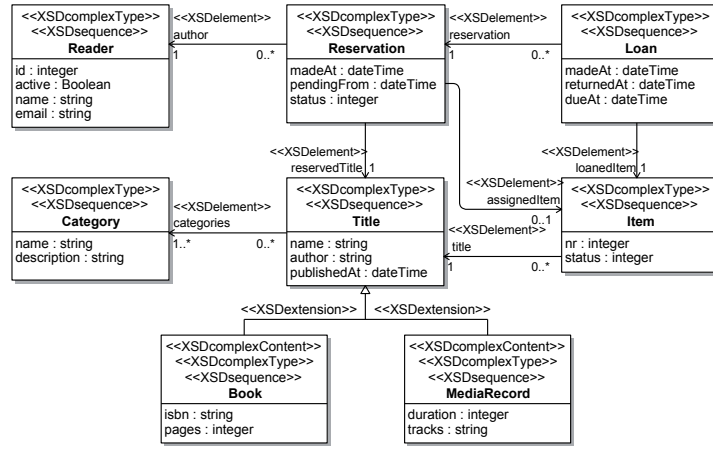

Figure 5. ***A fragment for PSM for MagicLibrary – XML schema model***

Table 1. ***Samples of the database and XML schema code, generated from PSM models***

Database Schema Code Fragment	**XML Schema Code Fragment**
CREATE TABLE Reservation **(** id integer **PRIMARY KEY**, madeAt *date*, pendingFrom *date*, status *integer*, fkReader *integer* **NOT NULL**, fkTitle *integer* **NOT NULL**, **FOREIGN KEY**(fkTitle) **REFERENCES** Title(id), **FOREIGN KEY**(fkReader) **REFERENCES** Reader(id));	<xs:complexType name=”Reservation”> <xs:sequence> <xs:element maxOccurs=”1” name=”author” type=”Reader”/> <xs:element maxOccurs=”1” name=”reservedTitle” type=”Title”/> <xs:element maxOccurs=”1” minOccurs=”0” name=”assignedItem” type=”Item”/> </xs:sequence> <xs:attribute name=”madeAt” type=”xs:dateTime”/> <xs:attribute name=”pendingFrom” type=”xs:dateTime”/> <xs:attribute name=”status” type=”xs:integer”/> </xs:complexType>

In order to support the proposed MD3 framework, a customized version of MagicDraw UML environment has been developed. It contains the following features:

- UML profiles for the XML schema and DDL;
- Customization suites for virtually transforming data design stereotypes into the first-class modeling concepts;
- Custom diagrams for each data design abstraction layer – concepts, data structure, database schema, and XML schema diagrams;
- Validation rule suites capable to verify a data design model for completeness;
- Model-to-model transformations automating PIM to PSM conversion;
- Code generators that allow generating the DDL and XML Schema code based on platform-specific data design models and plain text report templates using Velocity Template Language scripts.

Due to the limitation of the paper size, the MD3 integrated development environment is not presented in detail here. However, it is rather a typical example of building a domain-specific modeling environment, which is discussed in detail in (Silingas at al., 2009).

Summary

The paper introduced a model-driven data design framework *MD3*, which promotes using UML for computation-independent (CIM) and platform-independent (PIM), data modeling, and application of semi-automated transformations of platform-independent models into platform-specific data models (PIM) by using UML profiles for specifying XML and relational database schemas. It also supports fully automated generation of XSD and DDL scripts from platform-specific models. The paper includes a number of concise illustrations for applying this framework to a case study of a fictitious library information management system *MagicLibrary*, which illustrate the validity of the proposed approach.

The MD3 framework provides the principles and workflow for practitioners to cope with the multi-platform data structure design problem by focusing on platform-independent object-oriented data modeling and automated transformation to platform-specific representations, such as the XML schema and the relational database schema. In order to support pragmatic concerns of the MD3 framework, a prototype of MD3 integrated modeling environment has been implemented as a plug-in to the MagicDraw UML tool. It allows data designers to prepare and manage data design models in a simple and intuitive manner. However, the user acceptance of such an integrated modeling environment still needs to be proved by running the user experience survey.

While the MD3 framework lays out the foundations for a data structure design, there are still many details of applying this method that need to be researched and improved. Also, it is possible to improve the integrated modeling environment by adding or customizing various features. Therefore the MD3 framework should be considered as a source for developing more specialized and formalized data design methods and tools.

Acknowledgements

I would like to thank Tomas Juknevicius and other members of MagicDraw R&D team who worked on implementing DDL and XML schema code engineering in MagicDraw UML – the presented approach utilizes some of the tool features that are already available. Also, I would like to thank Erik Meijer from Microsoft Research, who introduced *the data modeler's Bermuda triangle problem* and inspired me to write this paper.

REFERENCES

BERNAUER, Martin; KAPPEL, Gerti; KRAMLER, Gerhard. (2004). Representing XML schema in UML – A Comparison of Approaches. *Web Engineering*, *LNSC 3140*, p. 767–769. ISBN 978-3-540-22511-9.

BROOKS, Frederick P. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. *Computer*, vol. 20 (4), p. 10–19.

FRANCE, Robert; RUMPE, Bernhard. (2007). Model-driven Development of Complex Software: A Research Roadmap. *FOSE '07: 2007 Future of Software Engineering*. Washington, DC: IEEE Computer Society, p. 37–54. ISBN 0769528295.

OMG (2003). MDA Guide Version 1.0.1. Access via Internet: <http://www.omg.org/docs/omg/03-06-01.pdf>.

OMG (2009). OMG Unified Modeling Language (OMG UML), Superstructure, v2.2. Access via Internet: <http://www.socialresearchmethods.net/tutorial/tutorial.htm>http://www.omg.org/spec/UML/2.2/Superstructure/PDF>.

SILINGAS, Darius; VITIUTINAS, Ruslanas (2007). Towards UML-Intensive Framework for Model-Driven Development. In: *Balancing Agility and Formalism in Software Engineering: Second IFIP TC 2 Central and East European Conference on Software Engineering Techniques, CEE-SET 2007, LNCS 5082*, p. 116-128. ISBN 978-3-540-85278-0.

CHEN, Peter Pin-Shan (1976). The Entity-Relationship Model – Toward a Unified View of Data. *ACM Transactions on Database Systems (TODS)*, vol. 1, no. 1, p. 9–36, ISSN 0362-5915.

SILINGAS, Darius; KAUKENAS, Saulius (2004). Applying UML for Relational Data Modeling. Iš: *Proceedings of the Thirteenth International Conference on Information Systems Development: Advances in Theory, Practice and Education (ISD 2004)*. Vilnius, 2004. ISBN 9986-05-762-0.

SILINGAS, Darius; VITIUTINAS, Ruslanas; ARMONAS, Andrius; NEMURAITE, Lina (2009). Domain-Specific Modeling Environment Based on UML Profiles. In: *Proceedings of Information Technologies '2009*. Kaunas: Technologija, p. 167–177.

THOMAS, Dave (2004). MDA: Revenge of Modelers or UML Utopia? *IEEE Software*, Los Alamitos, CA: IEEE Computer Society, vol. 21, no. 3, p. 15–17. ISSN 0740-7459.

MD3 – INTEGRUOTAS MODELIAVIMU PAGRĮSTAS DUOMENŲ PROJEKTAVIMAS OBJEKTINĖMS, XML IR SĄRYŠINIŲ DUOMENŲ BAZIŲ TECHNOLOGIJOMS

Darius Šilingas

S a n t r a u k a

Modeliais pagrįstos architektūros (MDA) paradigma siūlo didinti programinės įrangos kūrimo efektyvumą keliant abstrakcijos lygį. Tuo tikslu siūloma kaip pagrindinę programų konstravimo priemonę naudoti ne tekstines programavimo kalbas, o vizualias modeliavimo kalbas, tokias kaip UML. Šiame straipsnyje nagrinėjamas MDA taikymas duomenų struktūrų projektavimui. Pristatomas MD3 metodikos karkasas, kuris apibrėžia duomenų projektavimo principus, veiksmų seką, UML poaibį, skirtą duomenims modeliuoti, UML plėtinius XML schemoms ir DDL kodui reprezentuoti, transformacijas tarp skirtingų duomenų abstrakcijos lygių bei duomenų struktūrų projektavimui specializuotą modeliavimo aplinką. MD3 taikymas iliustruojamas nedideliais, bet reprezentatyviais pavyzdžiais iš bibliotekos dalykinės srities. Pristatomas MD3 metodikos karkasas suvokiamas kaip pradinis atspirties taškas labiau specializuotiems ar formalizuotiemsų duomenų projektavimo metodams ir įrankiams kurti.