DIFFERENT APPROACHES TO MDWE: BRIDGING THE GAP

ATTILA ADAMKÓ AND LAJOS KOLLÁR

ABSTRACT. This paper will give a short overview and comparison of Model-driven Web Engineering (MDWE) practices applied in both academy and industry with the aim of bridging the gap between those approaches. While Domain Specific Languages (DSLs) are used to express a high level outline of the imagined systems, the industrial approaches are focusing on a lower level and the distance of the two fields are mostly too wide to apply both in one development process. The goal of this paper is to propose a way to merge the two sides. DSLs can help to build prototypes in a very rapid manner. Based on those prototypes, common models can be derived that can serve as a basis for model-driven generation of Web applications using well-known production frameworks.

1. INTRODUCTION

Model-driven engineering has become more than a promising way of creating applications that are based on abstractions: MDE brings software development much closer to domain experts. It is the way how the MDA and MDD field could find its way to real life scenarios. However, these prominent ideas without fully realized key technology features cannot lead the way. Without executable modelling, meta-modelling, language engineering and proper tool support they are no more than interesting and idealistic research directions.

In the beginning of the 21st century there were several promising projects to support MDD and MDE. Nowadays, after a decade of the born of MDA and supporting technologies the list of possible tools with the mentioned key features are much more limited, only a few of them remain alive. We can

observe a landscape shift from general purpose languages to domain-specific ones. It opens the way for bridging the gap between stakeholders, domain experts and software developers.

2. Domain-specific languages and modeling

In software engineering, a domain-specific language (DSL) is a *"a computer programming language of limited expressiveness focused on a particular domain"* [6]. A DSL can be either a visual language, like the languages used by the Eclipse Modeling Framework, or textual languages.

A sound language description contains an abstract syntax, one or more concrete syntax descriptions, mappings between abstract and concrete syntaxes, and a description of the semantics. The abstract syntax of a language is often defined using a metamodel. The semantics can also be defined using a metamodel, but in most cases in practice the semantics are not explicitly defined but they have to be derived from the runtime behavior.

In model-driven engineering, many examples of domain-specific languages may be found, like OCL, a language for decorating models with assertions and constraints, or QVT, a domain-specific model transformation language. However, languages like UML are typically general purpose modeling languages.

Domain-specific languages have important design goals that contrast with those of general-purpose languages:

- domain-specific languages are less comprehensive;
- domain-specific languages are much more expressive in their domain;
- domain-specific languages should exhibit minimum redundancy.

3. Academical practices to MDWE

Modeling and systematic design methods are important and emerging fields in the Academic sector. Several new methodologies had born from this direction and resulted better software development practices. However, application of those methodologies are very time-consuming therefore it is mostly unacceptable for the industry. Several years are required before an approach becomes a well-functioning method that is ready for industrial use. The following sections describe some of the most interesting model-driven languages and methodologies (without attempting to be comprehensive). (The first one is an odd one out because it has both academic and industrial aspects.)

3.1. **WebML, WebRatio.** WebML [4] is a visual notation for specifying the content, composition, and navigation features of hypertext applications, building on ER and UML. Its not only for visualizing the models rather than a

methodology for designing complex data-intensive Web applications. It provides graphical, yet formal, specifications, embodied in a complete design process. Why it is an odd one out in this list that it has a commertial tool which can assist by visual design.

WebML enables designers to express the core features of a site at a high level, without committing to detailed architectural details. WebML concepts are associated with an intuitive graphic representation, which can be easily supported by CASE tools and effectively communicated to the non-technical members of the site development team. The specification of a site in WebML consists of four orthogonal perspectives:

- Structural Model
- Hypertext Model
  - Composition Model
  - Navigation Model
- Presentation Model
- Personalization Model

At first sight the business process modell is missing from the WebML methodology but its lying inside the composition model and navigation model, jointly named as hypertext model. Native BPMN is not supported by the tool, however there is a transformator which can transform a BPMN diagram into the WebML domain enriching the Application model.

Furthermore, WebML has been extended to cover a wider spectrum of front-end interfaces, thus resulting in the Interaction Flow Modeling Language (IFML), adopted as a standard by the Object Management Group (OMG) in 2013.

The WebML language started as a research project at Politecnico di Milano and later tool support has also been added. This software is called WebRatio that has become an industrial product so it can be considered as a success story for bridging the gap between academics and industry. In an ideal world, more ideas originating from the academic sector should reach that stage.

3.2. **UML-based Web Engineering.** UWE [8] applies the MDA pattern to the Web application domain from the from analysis to the generated implementation. Model transformations play an important role at every stage of the development process. The main reasons for using the extension mechanisms of the UML instead of a proprietary modelling technique are the acceptance of the UML in the field of software development and its extensibility with profiles. The UWE design approach for Web business processes consists of introducing specific process classes that are part of a separate process model with a defined interface to the navigation model.

Transformations at the platform independent level support the systematic development of models, like deriving a default presentation model from the navigation model. Then transformation rules that depend on a specific platform are used to translate the platform independent models describing the structural aspects of the Web application into models for the specific platform. Finally, these platform specific models are transformed to code by model-to-text transformations. Computer aided design using the UWE method is possible using MagicUWE, a plugin for MagicDraw.

3.3. **WebDSL.** WebDSL [11] is a domain-specific language for developing dynamic Web applications with a rich data model. The goal of WebDSL is to get rid of the boilerplate code you would have to write when building a Java application and raise the level of abstraction with simple, domain-specific sub-languages that allow a programmer to specify a certain aspect of the application and the WebDSL compiler would generate all the implementation code for that aspect. The main features of WebDSL are: Domain modeling, Presentation, Page-flow, Access control, Data validation, Workflow, Styling, Email.

Initially there were three sub-languages: a data modeling language, a user interface language and a simple action language to specify logic. WebDSL applications are translated to Java Web applications, and the code generator is implemented using Stratego/XT and SDF.

Although WebDSL is mainly a research project, a case study for domain-specific languages in the Web field [7], it can be usable by anybody with some programming experience.

4. Industrial practices

OMG provides a key foundation for Model-Driven Architecture, which unifies every step of development and integration from business modeling, through architectural and application modeling, to development, deployment, maintenance, and evolution. These concepts and directives have served as a basis for several solutions built by various companies. The only problem with these artifacts is the complexity. Complexity requires time and deep knowledge but in real life projects this factor is the most limited one. Companies are focusing on fast development time and choose frameworks supporting productivity rather than clear but time-consuming analysis phases. However, the following products could be used in an industrial environment because they have been proven to be working solutions while utilizing the OMG foundations.

4.1. **AndroMDA.** In short, AndroMDA is an open source MDA framework which works on UML models and utilizing plugins and components to generate

source code for a given programming language. Models are stored in XMI format produced from different CASE-tools. AndroMDA reads models into memory, making these object models available to its plugins. These plugins define exactly what AndroMDA will and will not generate. Each plugin is completely customizable to a project's specific needs.

It is mostly used by developers working with J2EE technologies and generate code for Hibernate, EJB, Spring and Web Services. AndroMDA allows customization of the templates used for code generation, therefore you can generate any additional code you want. We have seen AndroMDA in action and found it very promising in 2010 [1].

Currently, the only problem is its lost update cycle. The home page was last updated in 2011 and also the sourceforge repository's main branch seems to be stopped. However, there is a small sign of life because timestamps on several files in the SNAPSHOT branch show fresh (summer 2013) modification date. Because it was successfully applied on one of our industrial projects, it has the potential and possibility to became an alternative way for bridging the two world.

4.2. **openArchitectureWare (moved to Eclipse Modeling).** OpenArchitectureWare (oAW) was the second way for MDA around 2009. It was a modular MDA/MDD generator framework supporting arbitrary models and providing a language family to check and transform models as well as generate code based on them. OAW had strong support for EMF (Eclipse Modelling Framework) based models but could work with other models (e.g., UML2, XML or simple JavaBeans) too. The main power of oAW was the workflow engine which allowed to define generator/transformation workflows.

OAW was inherited by the Eclipse Modeling Framework and became a basis for it forming one big integrated family—Xpand (and Xtend), MWE (ant-like definition of transformations chains) and Xtext (textual DSL). It provides model-to-model and model-to-text transformation languages but they are not based on standards.

4.3. **Acceleo.** Acceleo is a pragmatic implementation of the Object Management Group (OMG) MOF Model to Text Language (MTL) standard. The creator was a French company, named Obeo. Nowadays, Acceleo is an Eclipse project mostly developed in Java and available under the Eclipse Public Licence (EPL) provided by the Eclipse Foundation. During the transition, the language used by Acceleo to define a code generator has been changed to use the new standard from the OMG for model-to-text transformation, MOFM2T. Acceleo is built on top of several key Eclipse technologies like EMF and, since the release of Acceleo 3, the Eclipse implementation of OCL (OMG's standard

language to navigate in models and to define constraints on the elements of a model). It has a very good tool support for productive coding.

4.4. **Spring.** One of the most powerful industrial solutions for Java EE application development is the Spring Framework. It is an open source application framework and Inversion of Control (IoC) container for the Java platform. The core features of the Spring Framework can be used by any Java applications, but there are extensions for building web applications on top of the Java EE platform. Although the Spring Framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBeans (EJB) model. The Spring Framework comprises several modules that provide a range of services including Inversion of Control container, Aspect-oriented programming, Data access, Transaction management, Model–View–Controller pattern, Remote access framework, Authentication and authorization, Messaging and Testing. These capabilities make it a strong candidate for industrial projects.

5. Closing the gap

A number of issues why the high majority of the industry is not committed to model-driven development has been identified in [5]. The authors' findings include that technical innovation does not go hand in hand with making profit; model-driven approaches are still believed to novel which means that they cannot be trusted enough for adoption; and MDD is considered to be heavy-weight, complex and not mature enough to be used in a real-world development project. They also emphasize that *"simplicity is a key feature that helps to sell a technology"*, however, model-driven approaches can hardly be called simple.

Their additional observations are quite similar to those of [3] and [2]: successful application of model-driven techniques require a different approach to work (and a basic understanding and commitment) from project participants, revolutionary technologies require new forms of organization, tool support still does not reach the required level, etc. A very important observation states that *"the tools, training, and expectations of professionals under MDE are not as well developed and established as those under more traditional software development dynamics"* [2].

Despite of having some model-driven solutions that has become an industrial product (e.g., WebRatio), their range of application cannot be compared with those of the well-known and widely used frameworks like Spring.

Therefore, the existing gap between academical and industrial approaches can be filled, on the one hand, by developing better and better tools and teaching more and more people to think in models, or, on the other hand,
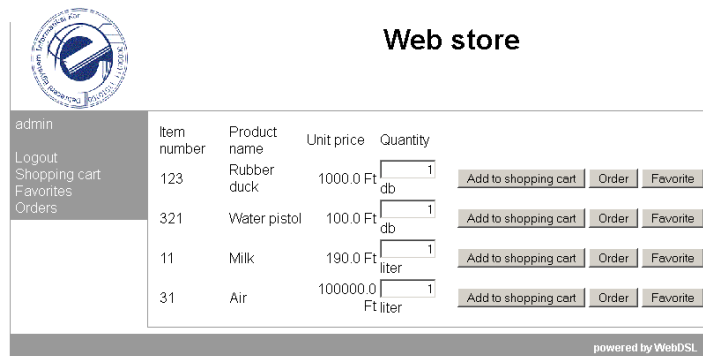
Figure 1. Web store implementation in WebDSL.

it might be filled by establishing model-based solutions which use the widespread production frameworks as a target platform. Our work is focused on that latter field: we propose a new model-driven generator that is able to generate source code and the necessary configuration files to the Spring framework having Hibernate as a persistence framework.

Since domain-specific languages (especially WebDSL) has been proved to be very effective in rapid prototyping, our starting point in development was to use it for easily and quickly define the concepts of the domain. This initial implementation might serve as a basis for building UML models that can later be transformed (using a generator developed for that purpose) to Spring. The reason for having UML models as intermediate artifacts is twofold: this way the development can be started by creating the appropriate UML models while it allows other DSL-to-UML mappings to be added later, extending the capabilities of the system this way.

5.1. **Demo application—Web store.** In order to demonstrate our ideas, a basic implementation for a Web store application has been created [10]. The project has been developed primarily for demonstrational purposes so it lacks much functionalities that a real web shop should implement.

This demo emphasizes the rapid prototyping capabilities of WebDSL: it is quite straightforward to create a base (but still useable) application in no more than 1,000 lines of code.

Figure 1 shows the developed application that has been transformed to Java code by the built-in generator included in the WebDSL Eclipse plugin.

5.1.1. *WebDSL implementation and problems.* The WebDSL implementation contains cca. 1,000 lines of code. It includes the definition of business entities (Product, User, Order, OrderItem, Cart, etc.), pages and navigation plus
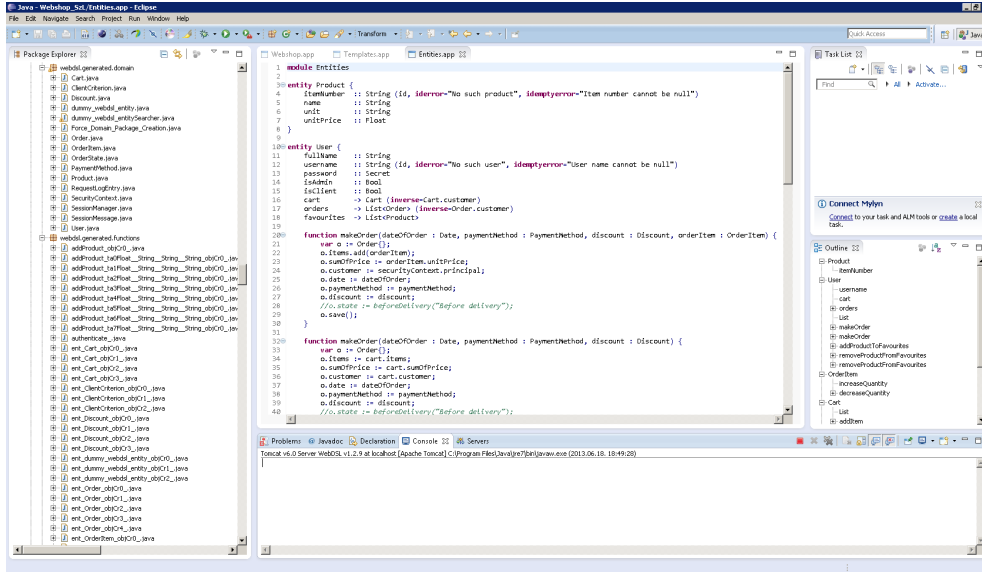
FIGURE 2. Excerpt of module Entities (middle) and the generated files (left).

some information on layout and access control. The WebDSL generator has generated 948 Java source files which is quite hard to understand and maintain. Figure 2 shows an excerpt of the module describing the business entities and Of course, a model-driven solution is not intended to modify the generated code (since you are required to modify the model and then re-generate), however, in practice, it is sometimes needed.

5.2. **WebDSL to UML transformation.** DSLs are backed with metamodels that capture the abstract syntax (i.e., the knowledge of the domain the DSL is aimed at). Based on that metamodel it is straightforward enough to transform the representation onto a model conforming the UML metamodel so due to lack of space we omit the details of the transformation.

5.3. **Generation of Spring implementation from UML model.** A generator that is able to generate Spring and Hibernate based implementation of a Web application from a stereotyped UML class diagram, has been developed [9]. The generator itself is built on top of Acceleo and is able to provide implementation for three layers: data layer, data access layer and business logic layer. For the data layer, it generates Hibernate entity classes, the data access layer will contain data access objects (DAOs), while the business logic layer contains POJIs (Plain Old Java Interfaces) for describing the services.
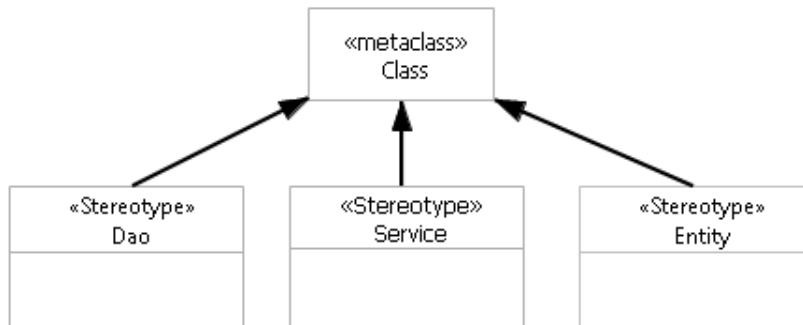
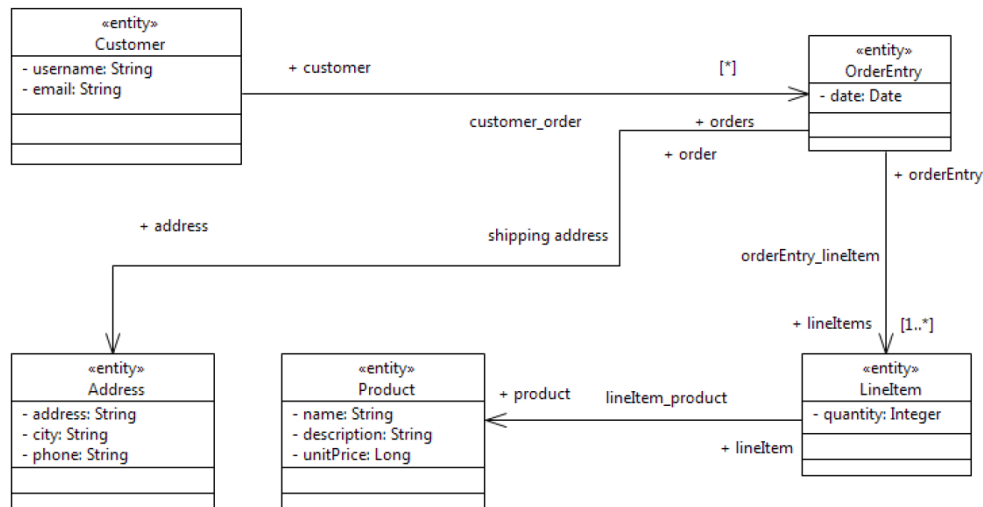FIGURE 3. Stereotypes the generator understands.

FIGURE 4. The domain model of the simplified Web store.

XML-based configuration files for both Spring and Hibernate are also generated.

The initial implementation of the generator deals only with class diagrams. Stereotypes are used to determine which layer the class belongs to. Figure 3 shows the stereotypes that are processed by the generator.

The domain model that has been created is shown in Figure 4. Its classes are stereotyped with <<entity>>.

Figure 5 demonstrates the Shopping Cart page of the generated Spring application.

FIGURE 5. The Shopping Cart page of the generated Spring application.

6. CONCLUSION

For the MDE community, it is very important that new ideas and methods developed in the academia appear in industrial practice. Without it, model-driven practices will never reach their potential. This is the reason why we need increasing number of success stories like WebRatio. However, only a few of the proposed academic approaches receive broader attention from the industry.

In this paper, we gave a short (and, of course, highly incomplete) overview of the current state of model-driven engineering in academics and industry. In order to close the gap, we proposed a solution for generating applications for the Spring platform which is widely used across the industry. However, this work is not finished: by the time of this paper, our system is able to generate an application based on either a WebDSL descripiton or a UML model but generation of business logic is still an open question.

REFERENCES

[1] A. Adamkó and C. Bornemissza. Developing Web-Based Applications Using Model Driven Architecture and Domain Specific Languages. In *Proceedings of the 8th International Conference on Applied Informatics*, pages 287–293, 2010.

[2] J. Aranda, D. Damian, and A. Borici. Transition to model-driven engineering: what is revolutionary, what remains the same? In *Proceedings of the 15th international conference on Model Driven Engineering Languages and Systems*, MODELS'12, pages 692–708, Berlin, Heidelberg, 2012. Springer-Verlag.

[3] P. Baker, S. Loh, and F. Weil. Model-driven engineering in a large industrial context — motorola case study. In *Proceedings of the 8th international conference on Model Driven Engineering Languages and Systems*, MoDELS'05, pages 476–491, Berlin, Heidelberg, 2005. Springer-Verlag.

[4] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[5] T. Clark and P.-A. Muller. Exploiting model driven technology: a tale of two startups. *Softw. Syst. Model.*, 11(4):481–493, Oct. 2012.

[6] M. Fowler. *Domain Specific Languages*. Addison-Wesley Professional, 1st edition, 2010.

[7] D. M. Groenewegen, Z. Hemel, and E. Visser. Separation of Concerns and Linguistic Integration in WebDSL. *IEEE Software*, 27(5):31–37, 2010.

[8] N. Koch and A. Kraus. Towards a common metamodel for the development of web applications. Cueva Lovelle, Juan Manuel (ed.) et al., Web engineering. International conference, ICWE 2003, Oviedo, Spain, July 14-18, 2003. Proceedings. Berlin: Springer. Lect. Notes Comput. Sci. 2722, 497-506 (2003)., 2003.

[9] A. Lőcsei. Webalkalmazások modell alapú készítése, 2013. Bachelor's thesis, University of Debrecen, Hungary, In Hungarian. Thesis supervisor: Lajos Kollár.

[10] L. Szarka. Modell-Vezérelt Webfejlesztési Megoldások. Master's thesis, University of Debrecen, Hungary, 2013. In Hungarian. Thesis supervisor: Attila Adamkó.

[11] E. Visser. WebDSL: A Case Study in Domain-Specific Language Engineering. In R. Lämmel, J. Visser, and J. a. Saraiva, editors, *Generative and Transformational Techniques in Software Engineering II*, pages 291–373. Springer-Verlag, Berlin, Heidelberg, 2008.

Department of Information Technology, Faculty of Informatics, University of Debrecen, H–4028 Debrecen, Kassai út 26., Hungary
  E-mail address: adamkoa@inf.unideb.hu

Department of Information Technology, Faculty of Informatics, University of Debrecen, H–4028 Debrecen, Kassai út 26., Hungary
  E-mail address: kollarl@inf.unideb.hu