Proceedings of the 8th International Conference on Applied Informatics
Eger, Hungary, January 27–30, 2010. Vol. 2. pp. 295–303.

Interoperability of Model-driven Web Engineering approaches

Attila Adamkó, Lajos Kollár

Department of Information Technology
Faculty of Informatics, University of Debrecen
e-mail:{adamkoa|kollarl}@inf.unideb.hu

Abstract

Model-driven Web Engineering (MDWE) approaches provide methodologies and tools for both the design and the development of most kinds of Web applications. They address different concerns of Web applications by using separate models (e.g., structural, navigational, presentational) which are transformed into Web page skeletons by model compilers. These methods are more or less applicable for data-driven Web applications but with the growing complexity of Web systems (Web workflow systems, e-commerce, e-government, etc.), they need to address new requirements, as well. To answer these new challenges, new concerns are needed (for example, process models, etc.). However, this requires a lightweight, extensible, loosely coupled set of models for designing applications because not all kinds of Web applications need every concern.

In this paper, an approach for the interoperability of (some) existing methodologies based on metamodeling, model transformations and model weaving will be introduced. This approach allows the MDWE methodologies to be extended in a consistent manner where new model kinds are separated and weaved together with the classical models that each approach supports.

Keywords: MDWE, Web Applications, Metamodeling, Model transformations, Model weaving

1. Introduction

Due to the evolution of Web technologies experienced in the past 10–15 years, the Web has become a primary platform for developing applications. However, as these technologies evolve very fast, they might become obsolete soon. Developers of Web applications need sophisticated solutions that support the whole product lifetime of an application that is able to cope with the skyrocketing changes of the underlying technologies.

Model-driven Web Engineering is a still emerging field aiming at providing sound model-based solutions for building Web applications that try to separate the abstract design (PIM) from the concrete technological platforms (PSMs).

2. Related research

2.1. Web Engineering methodologies

Existing model-based Web Engineering approaches provide different methods and tools for both the design and the development of various kinds of Web applications. In order to reduce complexity, most of the methodologies propose the separation of different views (i.e., models) of the application into 3 levels: structural (or content), navigational (or hypertext) and presentational models. For more information see [15]. Figure 1 shows the most common design dimensions of the currently existing methodologies.

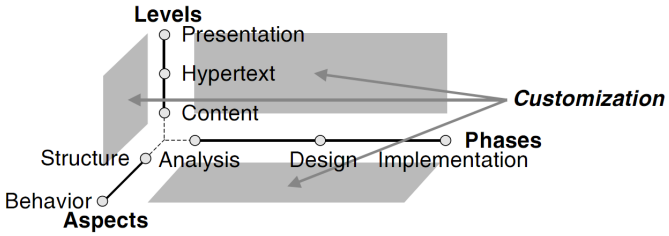

Figure 1: Design dimensions of Web applications [15].

In addition, some methodologies add some new models (or refine existing ones) to obtain a more fine-grained solution when modeling the application. Despite the separation, the levels should be interconnected in order to be able to capture the semantics behind the elements of the different models, e.g., the navigational objects are based on certain elements of the content model.

Beyond the creation of the models for the corresponding levels, Web application designers need to be aware of the various aspects of the systems to be modeled. Some applications are providing access to more or less static information hence they require much less behaviour modeling compared to systems that need to perform several complex business processes like e-commerce applications. Both structure and behaviour need to be modeled using a uniform notation that has to cope with the specific characteristic of each of the levels.

There is another approach worth mentioning when talking about Web application design. Unfortunately, there is no consensus in the literature about the general phases of the development which means that the order of steps involved in modeling the levels is up to the modeler. Moreover,

Many design methods can be found in the literature: OOHDM [14], OO-H [4], UWE [7], W2000, WSDM and WebML [2] are among the most popular ones. From

a modeler's perspective, each of them offer some possibilities for modeling the levels and aspects mentioned above, and they all come with a guideline for the development process. On the other hand, today's situation is somehow similar to the well-known "object-oriented method war" of the 1990ies (see Figure 2). That "method war" has ended with the unification of the different modelling notations which resulted in the UML so the real question is that can this strategy also work for the existing web engineering approaches or not. In Section 4 we elaborate our viewpoint on this topic.

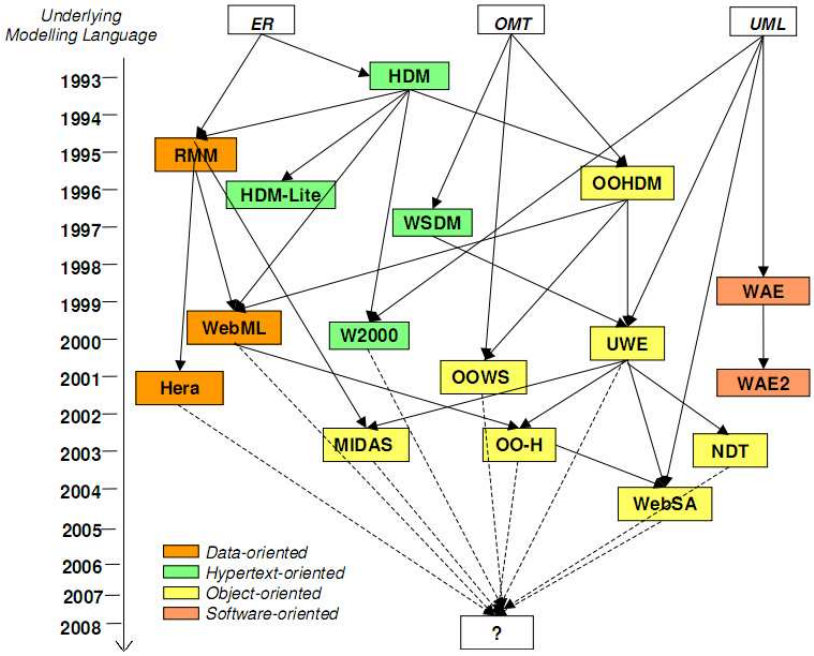

Figure 2: Evolution of Web modeling languages [15].

2.2. Domain-specific modeling, Metamodeling

The main goal of domain-specific modeling is to raise the level of abstraction by specifying the solution directly using domain concepts. The final product (and maybe several intermediate artifacts, as well) are generated based upon these high-level specifications. It also allows the stakeholders and domain experts to concentrate to the domain only. Domain-specific languages (DSLs) are built in order to capture domain semantics. A very common (but not the only) way of defining DSLs is metamodeling. The previously mentioned Web application design methods contain notations that can be used for describing a model of a Web application so they can be considered as DSLs for Web applications hence.

Some of the existing Web application design methods (e.g., UWE, WebML) offer a metamodel, as well [8, 13]. This allows model-based development since one need to build models conforming to the appropriate metamodel in order to capture the structural, navigational or presentational structure of the application to be developed. However, in the most of the cases, these models mix the different levels of Web applications that results in a solution that might be appropriate for the given application domain but makes the reuse of models or model parts almost impossible.

2.3. Model transformation, Model weaving

Model transformations are the most important operations in model engineering, describing how elements in the source model are converted into elements in the target model. This is achieved by relating the corresponding metamodel elements in the source and the target metamodels. Transformations can be classified into two categories: vertical transformations (a.k.a. refinements) are defined between models of different abstraction levels (e.g., PIM—PSM mappings), while horizontal transformations are mappings between models of the same level of abstraction (e.g., for improving or correcting a model). Examples of model transformation languages are QVT [12] , ATL [5] or MOLA [6] .

Weaving models are used to explicitly describe fine-grained relationships between models and metamodels (that are models themselves, as well) and execute operations based on them. With the help of applying weaving models, large metamodels that capture all aspects of a system can be avoided and a lattice of metamodels can be constructed instead where each metamodel that focuses on its own domain is maintained independently from the others. The links defined by the weaving model have some associated semantics about the linked elements.

Since a weaving model is a model itself, it can be a subject of applying a model transformation that results in a new model transformation. This should be applied to the left woven model in order to produce an instance of the right woven model that captures the semantics defined by the weaving link. For more information on model weaving and the differences between weavings and transformations, see [3].

3. Problem statement

Most of the methods mentioned in Section 2.1 are using different notations for these models, hence the interoperability between them is very hard to achieve. This also deceases reuse as one cannot import, for example, a conceptual model or a part of it when developing an application for a similar domain.

When adopting a model-based solution for application design, the main artifact is a model. The design process of an application will result in a set of models that is a starting point of a model-driven code generation process. This is a well-known fact in Model-driven Engineering and therefore is common in the various existing Model-Driven Web Engineering approaches, as well.

The idea of complete integration of the existing languages and methodologies, i.e., developing a common metamodel and unified phases of development that everyone will use in the future is utopian and (in our opinion) it must not be the goal of any integration or interoperability efforts. Its reasons are twofold. First of all, there are several proposals in the literature that address the creation of a common metamodel but the different approaches presented in [9, 11, 16] are good examples for demonstrating why it cannot be considered as "common".

Secondly, different domains and various flavours of Web applications may require different styles of modeling and it is almost impossible to achieve such a common modeling notation which is easy to understand and work with while being flexible enough to solve the uprising issues. Therefore we should work on bridging the different models together that allows (or promises, at least) the interchangeability of models and/or model pieces instead.

New models, processes and transformations should be included into the existing design methods when new aspects arise. However, these changes to a methodology are very risky and can cause several problems. In [10], three categories of concerns were identified:

- dependent concern, that depend on some other (earlier defined) concern(s), e.g., navigation (which depends on the conceptual model);
- replacement concern, that fully replaces a previously defined concern, e.g., presentation;
- orthogonal concern, that is a brand new concern which is completely independent of all the others, e.g. business process models.

However, we are not against the creation of subsequent metamodels and/or methodologies as they can result in better description of system parts or improved development processes. We only claim that a common metamodel is not the Holy Grail of MDWE as each and every "common" one will most probably fail as being a universal solution because the diversity of Web applications will require new answers for such questions that probably had not been asked by the time of developing the common metamodel.

4. Proposed solution

Our goal is to establish an extensible model-based framework which can provide interoperability among the existing Web modeling languages. This task has to be achieved by separating the different concerns (i.e., levels, phases and aspects) of Web applications in order to be able to either reuse relevant model parts or "transfer" a model into another notation (e.g., after a structural model is created conforming the metamodel of language A we decide to build the navigational model in language B since it might be more appropriate for our goals).

Hence, it is extremely important that the metamodels defining the languages for describing the various aspects of a Web application need to be separated from

each other as much as possible. So we suggest of decomposing the various methods into a combination of models, each of which conforms to a well-defined part of the whole application domain regardless of the language used for the notation. For example, that allows of describing the structural model either in relational model, Entity Relationship (ER), UML or by using any custom DSL but it requires the separation of the structural model from any other models (e.g., navigational or requirements model). Besides, we suppose that no method uses a notation that does not conform to the MOF metapyramid (in fact, this is not a heavy constraint).

In our proposed solution, model weaving should appear on two levels:

1. On *intra-method* level, the relationships existed before the decomposition of the concerns need to be defined in a weaving model in order to be able to produce the same level of expressiveness. Let us consider the well-known conference management system as an example! In UWE, for instance, we would have a UML class called *Paper* in the structural model while its derived (and stereotyped) versions would appear in the navigational and presentational model, as well. Instead of the given method's built-in notation for this derivation, weaving links should be established in a weaving model that comprises statements about the relationship between the models in question. This weaving model can also be used later on when the starting point of the design is the building of the structural model as it captures the semantics that structural model elements also become (stereotyped) elements of the navigational model under given circumstances so a transformation might be applied to the structural model in order to create an initial version of the navigational one.

2. On *inter-method* level, when the relationships described by the weaving model define which model elements of a given model M_a conforms to which model elements in M_b. M_a and M_b here typically have the same level of abstraction (e.g., they both are structural models described by different methodologies) and the weaving model is defined between their corresponding MM_a and MM_b metamodels. For example, if one of the methods uses ER for describing the structural model while the other one applies UML for the same purpose, then the weaving model should contain that the strong entity type of the ER corresponds to a class in a UML class diagram, etc. This approach allows not only the generation of such a model transformation based on the weaving model that can transform a model in a notation into another model of another notation but model traceability is also supported.

The work is currently in progress: we are working on creating the weaving links and transformations starting from PIMs defined with some of the well-known MDWE methodologies and resulting in a generated Spring Web Flow-based implementation with the help of the AMMA platform. This work is a continuation of our previous work on creating model-based Web applications that is discussed in [1]. In the future we plan to establish an ontology for the Web application design process that defines the semantics of the specific Web application models in general.

4.1. Advantages and disadvantages

Besides the separation of concerns, another advantage of this approach is the ability of creating various model kinds in addition to the "classicals" (i.e., those conforming to the levels of the Web applications): for example, if an (either existing or brand new) methodology formalizes UML2 use case diagrams during the requirements elicitation phase this forms a separate concern of the application that can be utilized either in intra-method or inter-method weaving or both. (This is, of course, the responsibility of the creator of the weaving(s).) This is true for creating a high-level business process model for the application, as well. However, not all Web applications are supposed to have models of all model types as the nature of the application might not require some kind of models (e.g., a Web Service as a Web application does not need a presentational model at all).

This approach can easily be applied to both PIMs and PSMs: let us suppose that we have created an abstract presentational model which needs to be mapped onto some concrete presentational technology (e.g., JSF, XHTML with XForms, JSP, etc.). All what we need before deploying the UI is to create a model transformation that maps an abstract presentational model onto a PSM that conforms to the chosen technology's metamodel. This enables the extensibility of the framework not only with subsequent PIM model types but with platform-specific technologies, as well. The same applies also to other PIM—PSM transformations, of course. If someone, for example, wants to use Spring Web Flow (SWF) and JSF in order to capture the semantics of Web navigaton with the help of finite state machines, only a weaving model that defines the relationship among navigational and presentational PIMs and SWF's and JSF's PSMs are needed.

Despite of the fact that we disagree with the existence of a common metamodel for Web Engineering that can be widely and exclusively used, (the lattice of) metamodels that are common regarding numerous methods can do us a good turn as they can serve as reference (or pivot) models for transformations. However, these metamodels do not deserve to be called "common": from the framework's point of view, they are "regulars" that can be used the same way than any other ("non-common") metamodels (i.e., they are subject to intra- and inter-method weavings).

The whole idea allows some sort of customizing in the design process: the designer can choose what artifacts need to be created to build the system and he/she can either select an existing representation or create an own DSL for describing an artifact.

However, there are drawbacks of the solution, as well. A lot of weaving models need to be created even when having a relatively small number of methodologies between which we would like to enable interoperability. This is especially true when dealing with inter-method weavings since the non-existence of a pivot element means that (supposing the worst case) they can only be defined pairwise (i.e., how to relate method A's concepts onto method B's or method C's and so on). This task would be much easier if we had a common metamodel. Let us suppose that we use UML2 class diagrams for describing structural information and we have two methods, one of which uses relational model while the other one uses ER: all

we need are the two-way mappings between relational model and UML2 and ER and UML2, the mapping between relational and ER models can be derived by a composition.

4.2. Conclusions and future work

In this paper we introduced our visions about the interoperability among the various existing model-driven Web engineering solutions. Our proposed solution is heavily based on both some existing metamodels for the different domains (aspects) of Web application design and the model transformations that provide mappings for introducing new concerns into a methodology. This is only a step behind of the creation of some ontologies for Web modeling which should result in more precise understanding of the underlying (meta)models. This would also allow the tools supporting MDWE methods to semantically understand and (re)use elements of the different methodologies. However, there is a lot of work to do by the Web Engineering community in order to define those ontologies.

Acknowledgements. The work is supported by TÁMOP 4.2.1./B-09/1/KONV-2010-0007/IK/IT project. The project is implemented through the New Hungary Development Plan co-financed by the European Social Fund, and the European Regional Development Fund.

References

[1] A. Adamkó and L. Kollár. MDA-Based Development of Data-Driven Web Applications. In J. Cordeiro et al., editors, *WEBIST (1)*, pages 252–255. INSTICC Press, 2008.

[2] 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.

[3] M. D. D. Fabro, J. Bézivin, F. Jouault, E. Breton, and G. Gueltas. AMW: a generic model weaver. In *Proc. of the 1čre Journée sur l'Ingénierie Dirigée par les Modčles (IDM05)*, 2005.

[4] J. Gómez and C. Cachero. OO-H method: extending UML to model web interfaces. pages 144–173, 2003.

[5] F. Jouault and I. Kurtev. Transforming Models with ATL. In J.-M. Bruel, editor, *MoDELS Satellite Events*, volume 3844 of *Lecture Notes in Computer Science*, pages 128–138. Springer, 2005.

[6] A. Kalnins, J. Barzdins, and K. Podnieks. MOLA - MOdel transformation LAnguage. http://mola.mii.lu.lv/, 2008.

[7] 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.

[8] A. Kraus and N. Koch. A Metamodel for UWE, 2003.

[9] F. Molina, J. Pardillo, C. Cachero, and A. Toval. Towards a Requirements-Aware Common Web Engineering Metamodel. In *LA-WEB'08: Proc. of the 2008 Latin American Web Conference*, pages 75–82. IEEE Computer Society, 2008.

[10] N. Moreno, S. Meliá, N. Koch, and A. Vallecillo. Addressing new concerns in model-driven web engineering approaches. In *Proceedings of the 9th international conference on Web Information Systems Engineering*, WISE '08, pages 426–442, Berlin, Heidelberg, 2008. Springer-Verlag.

[11] N. Moreno and A. Vallecillo. Towards interoperable Web engineering methods. *J. Am. Soc. Inf. Sci. Technol.*, 59(7):1073–1092, 2008.

[12] Object Management Group (OMG). Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Final Adopted Specification, 2007.

[13] A. Schauerhuber, M. Wimmer, and E. Kapsammer. Bridging existing Web modeling languages to model-driven engineering: a metamodel for WebML. In *ICWE'06: Workshop Proc. of the* 6^{th} *Int. Conf. on Web Engineering*. ACM, 2006.

[14] D. Schwabe and G. Rossi. An object oriented approach to web-based applications design. *Theor. Pract. Object Syst.*, 4(4):207–225, 1998.

[15] W. Schwinger and N. Koch. *Modeling Web Applications*, chapter 3, pages 39–64. John Wiley & Sons, 2006.

[16] M. Wimmer, A. Schauerhuber, W. Schwinger, and H. Kargl. On the Integration of Web Modeling Languages: Preliminary Results and Future Challenges. In 7^{th} *Int. Conf. on Web Engineering, Workshop Proc.*, pages 255–269, 2007.

Attila Adamkó, Lajos Kollár
H-4032 Debrecen, Egyetem tér 1.