KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2011
Cluj-Napoca (Romania), July 4–6, 2011, pp. 25–35

INTEROPERABILITY ISSUES OF MDWE METHODOLOGIES

ATTILA ADAMKÓ AND LAJOS KOLLÁR

Abstract. 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 (MDWE) 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). However, current MDWE approaches cannot provide solutions for all kinds of the requirements against a software system therefore a lightweight, extensible, loosely coupled set of models for designing applications are needed.

This paper introduces an approach for the interoperability of (some) existing methodologies based on metamodeling, model transformations and model weaving which 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.

1. Introduction

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

Received by the editors: April 25, 2011.

Key words and phrases. MDWE, Web Applications, Metamodeling, Model transformations, Model weaving.

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.

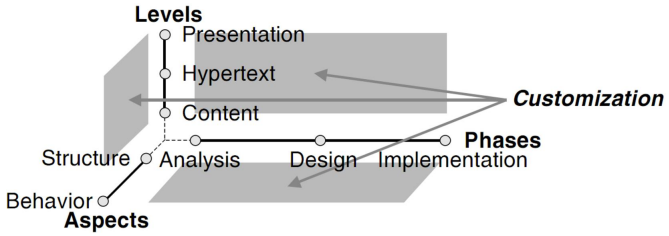

FIGURE 1. Design dimensions of Web applications [12].

models. For more information see [12]. Figure 1 shows the most common design dimensions of the currently existing methodologies.

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.

Current design methods offer some possibilities for modeling the levels and aspects mentioned above but they all has a unique approach (e.g., offering some model kinds that the others not) so this field is not standardized.

There is an other approach worth mentioning when talking about Web application design. Let this be either a fortune or an unfortune, 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. Therefore, reuse of models can hardly be achieved, especially when they are described according to the requirements of two different methodologies.

In this paper, we provide a solution based on metamodeling to allow models to be reused in different contexts. This is achieved by describing the semantics of the link between the corresponding models using model weaving.

The next section (Section 2) describes some related work, introducing the basic concepts that we used in our research. Section 3 describes the problems to solve while our proposal is described in Section 4 where the pros and cons of the approach are also discussed. Besides conclusions, a possible way of improvement is briefly mentioned in Section 5.

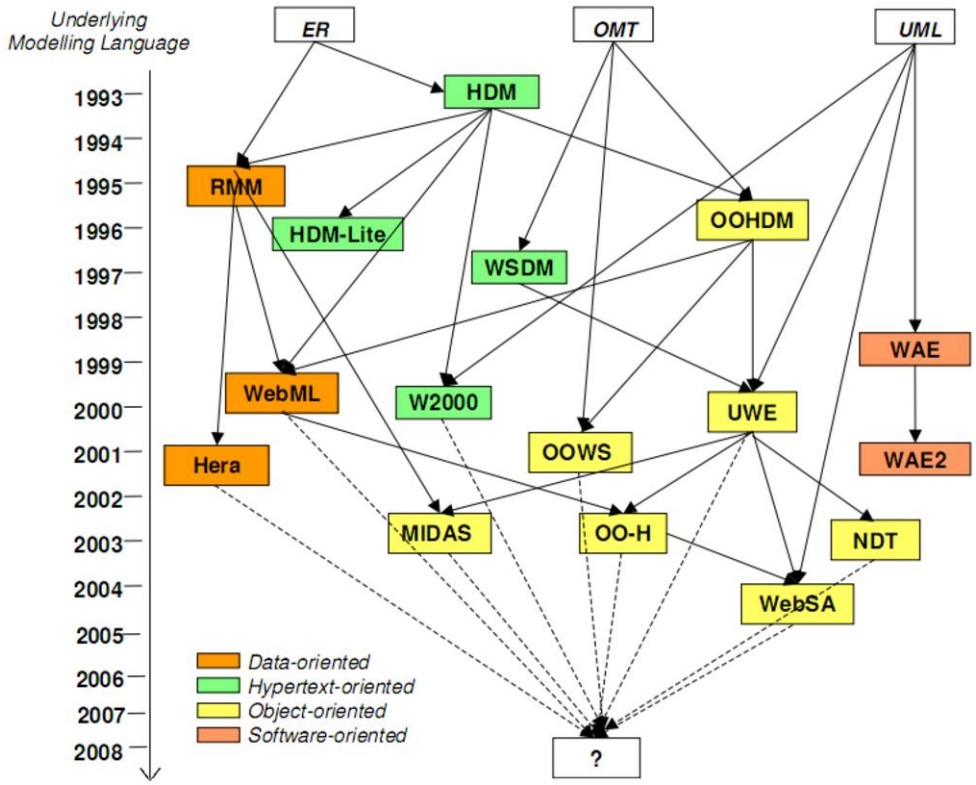

FIGURE 2. Evolution of WE languages and/or methodologies [12].

2. Related research

2.1. **Web Engineering methodologies.** In the past decade many design methods have been created: OOHDM [11], OO-H [4], UWE [5], W2000, WSDM and WebML [1] 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.

As the field of Model-Driven Web Engineering (MDWE) approaches follow the well-known Model-Driven Engineering principles, some methods create

Computational Independent Models (CIMs, e.g., in the form of requirement models), almost all of them allows the creation of Platform Independent Models (PIMs) for structure, navigation, presentation or business processes and most of them provides a means of obtaining Platform Specific Models (PSMs) for various platforms (e.g., J2EE, .NET, Spring, Struts, etc.) that can further be transformed into code.

2.2. **Domain-specific modeling.** 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.

2.3. **Metamodeling.** Some of the existing Web application design methods (e.g., UWE, WebML) offer a metamodel, as well [6, 9]. 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.

2.4. **Model transformation.** 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).

2.5. **Model weaving.** 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 [2].

2.6. **Web Engineering Interoperability common reference metamodel.** Based on the joint work of the creators of several Web Engineering approaches, a Web Engineering Interoperability (WEI) common reference metamodel has been developed. According to [8], 'WEI is a model-driven Web engineering architectural framework for organizing the models that address the different concerns of Web application development'. WEI captures three kinds of viewpoints of a Web application (i.e., data, business logic and user interface, see Figure 3) comprising 13 model types (which of course does not mean that all of them needs to be used in a single application). These model types reflect the various concerns that are covered by the existing Web Engineering methodologies.

The WEI reference model allows the Web application developers not to start the design from scratch (unlike many other methodologies) because some models and tools can be reused. The main problems are arising when two (or more) models representing different views of the same system should be integrated.

3. Problem statement

Most of the methods mentioned in Section 1 are using different notations for describing 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.

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. The main reason behind it is that 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

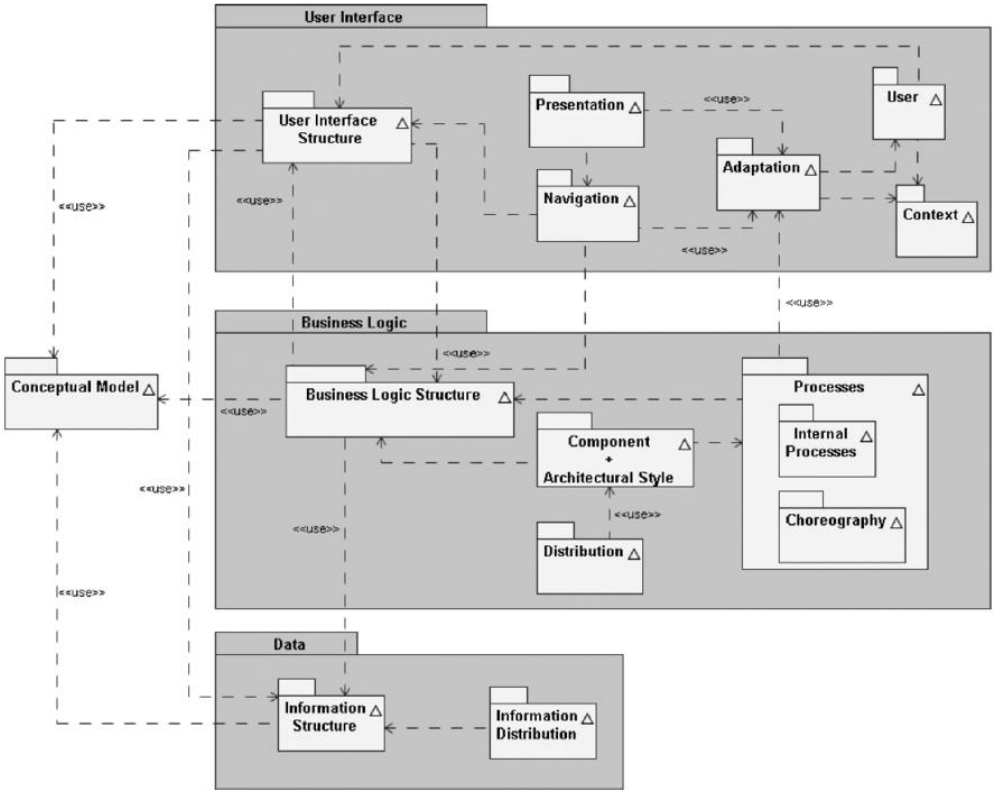

FIGURE 3. WEI common reference metamodel [8].

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 [7], 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. How to achieve interoperability?

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 [10] 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.

On intra-method level, it is extremely important that weaving links are created to also achieve the traceability for model elements to keep track about each element that is used in the description of another concern.

On inter-method level, WEI can be very useful (however, not required) because if (for two methodologies X and Y) X-to-WEI and WEI-to-Y weavings are described then there are no manual interactions needed to transform from X to Y.

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 'classical' ones (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).) The same holds 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).

The approach can easily be used with both PIMs and PSMs: let us suppose that we have created an abstract presentational model which needs to

mapped onto some concrete presentational technology (e.g., JSF, XHTML with XForms, JSP, etc.). All what we need before deploying the UI is 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 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 wants to use Spring Web Flow, for example, in order to catch the semantics of Web navigaton with the help of finite state machines,

Despite of the fact that we disagree with the existence of a common metamodel for Web Engineering that can be widely and exclusively used, a metamodel that is common regarding numerous methods can do us a good turn as it can serve as a reference (or pivot) model for transformations (which is WEI, in our case). However, this metamodel does not deserve to be called 'common': from the framework's point of view, it is a 'regular' one that can be used the same way than any other metamodels (i.e., they are subject to intra- and inter-method weaving).

The whole idea allows of some sort of customizing 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) it 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). If we happen to have a common metamodel, the task becomes much easier. Let us suppose 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. **Implementation.** Since the basic idea involves a lot of model weavings, it was very straightforward to choose Eclipse as our implementation platform. The ATLAS Model Management Architecture (AMMA, [3]) platform (beside others) contains a transformation language (ATL) and a model weaver (AMW), both based on the well-known Eclipse Modeling Framework (EMF).

5. 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 approach provides more and more precise links (using weavings) between the modeling elements which 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.

References

[1] 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.

[2] M. D. D. Fabro, J. Bézivin, F. Jouault, E. Breton, and G. Gueltas. AMW: a generic model weaver. In *Proc. of the 1re Journe sur l'Ingnierie Dirige par les Modles (IDM05)*, 2005.

[3] M. D. D. Fabro and F. Jouault. Model Transformation and Weaving in the AMMA Platform. In *Pre-proc. of the Generative and Transformational Techniques in Software Engineering Workshop*, pages 71–77, 2005.

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

[5] 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.

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

[7] 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.

[8] N. Moreno and A. Vallecillo. Towards interoperable web engineering methods. *J. Am. Soc. Inf. Sci. Technol.*, 59:1073–1092, May 2008.

[9] 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.

[10] D. Schwabe. A conference review system. `www.dsic.upv.es/~west/iwwost01/files/ConferenceReviewSystem.doc`, 2001. [Online; accessed 24-April-2011].

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

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

Department of Information Technology, Faculty of Informatics, University of Debrecen, H–4032 Debrecen, Egyetem tér 1., Hungary
E-mail address: adamkoa@inf.unideb.hu

Department of Information Technology, Faculty of Informatics, University of Debrecen, H–4032 Debrecen, Egyetem tér 1., Hungary
E-mail address: kollarl@inf.unideb.hu