On the Integration of Web Modeling Languages: Preliminary Results and Future Challenges

Manuel Wimmer[1,‡], Andrea Schauerhuber[2,*], Wieland Schwinger[3,‡], Horst Kargl[1,‡]

[1]Business Informatics Group
Vienna University of Technology
{wimmer, kargl}@big.tuwien.ac.at
[2]Women's Postgraduate College for Internet Technologies
Vienna University of Technology
schauerhuber@wit.tuwien.ac.at
[3]Department of Telecooperation
Johannes Kepler University Linz
wieland.schwinger@jku.ac.at

Abstract. The Unified Modeling Language (UML) is considered as the lingua franca in software engineering. Despite various web modeling languages having emerged in the past decade, in the field of web engineering a pendant to UML cannot be found yet. In the light of this "method war" the question arises if a unification of the existing web modeling languages can be successfully applied in the style of UML's development and thus promote Model-driven Web Engineering (MDWE). In such a unification effort we defer the task of designing a "Unified Web Modeling Language". Instead, we first aim at integrating three prominent representatives of the web modeling field, namely WebML, UWE, and OO-H, in order to gain a detailed understanding of their commonalities and differences as well as to identify the common concepts used in web modeling. This integration is based on specifying transformation rules allowing the transformation of WebML, UWE, and OO-H models into any other of the three languages, respectively. To this end, a major contribution of this work is the languages' definitions made explicit in terms of metamodels, a prerequisite for model-driven web engineering for each approach. Furthermore, the transformation rules defined between these metamodels - besides representing a step towards unification - also enable model exchange.

Keywords: Web Modeling, Model Integration, Common Metamodel for Web Modeling, Model-Driven Web Engineering

‡ This work has been partly funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) and FFG under grant FIT-IT-810806.

* This work has been partly funded by the Austrian Federal Ministry for Education, Science, and Culture, and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002.

1 Introduction

In the past decade various modeling approaches have emerged in the research field of web engineering including WebML [7], UWE [13], W2000 [1], OOHDM [26], OO-H [10], WSDM [8], and OOWS [25]. Each of those approaches follows the similar goal of counteracting a technology-driven and ad hoc development of web applications. Beyond this, we notice similar and simultaneous extensions to the individual web modeling approaches, e.g., for supporting context-aware web applications [2, 6, 9], business process modeling [4, 14], and lately model-driven web engineering [15, 18]. The current situation somewhat resembles the object-oriented modeling "method war" of the 90ies. A situation from which after a unification process the UML [24] eventually has become the *lingua franca* in software engineering. In the light of the current "method war" in the research field of web engineering (cf. Figure 1) the question arises if a unification of the existing web modeling approaches can be successfully applied as it was achieved for the UML and thus promote MDWE in academia as well as in practice.

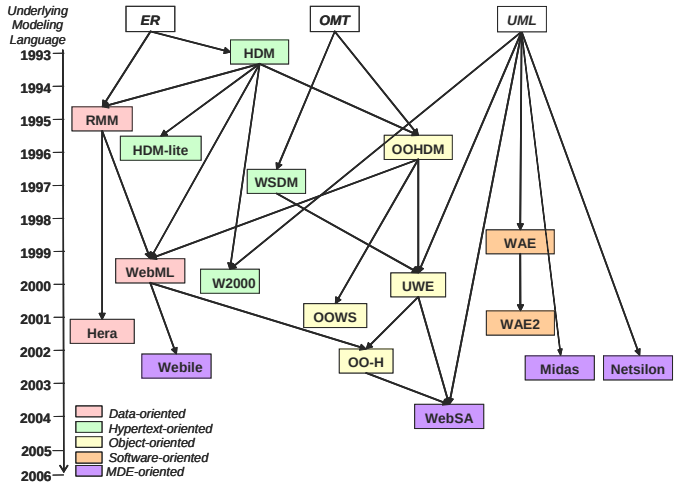


Figure 1:Web Modeling Languages History, based on [28]

As a prerequisite for unification a common agreement on the most important web modeling concepts is essential. This agreement can only be gained when investigating the concepts used in existing web modeling languages and fully understanding the languages' commonalities and differences. In the MDWEnet initiative [16], we therefore defer the task of designing a "Unified Web Modeling Language". Instead, we first aim at integrating three prominent representatives of the web modeling field, namely WebML, UWE, and OO-H, since they are well elaborated and documented as well as supported by modeling tools. This integration is based on specifying and implementing transformation rules allowing the transformation of WebML, UWE, and OO-H models into any other of the three languages, respectively. This way a detailed understanding of the common concepts used in web modeling can be obtained as well as their different realizations in the three selected languages. On the

basis of this integration task the definition of a common metamodel for web modeling can be achieved in the future.

Consequently, the major contribution of this work is a *step towards identifying the common concepts in web modeling* by first defining transformations between different modeling languages. We present the general integration approach as well as first results on the integration of WebML and OO-H.

Besides representing an important step towards unification, the transformation rules also enable *model exchange* between the three different languages. For defining the transformation rules, the languages' definitions had to be made explicit in terms of *metamodels*, which in turn represent a prerequisite for enabling model-driven web engineering for each individual approach. On the basis of tool adapters the models' representation within the approaches' tools could be translated into instances of these metamodels and vice versa thus also insuring interoperability. Furthermore, it will be possible to *exploit the different strengths of each web modeling approach*, e.g., code generation facilities for different platforms such as J2EE in WebML's WebRatio[1] tool and PHP in OO-H's tool VisualWade[2].

In the remainder of the paper, we discuss our methodology for integrating existing web modeling languages in Section 2. We elaborate on preliminary results of the integration task with respect to WebML and OO-H in Section 3 and provide our lessons learned in Section 4. Finally the paper is concluded with a discussion on future challenges in integrating as well as unifying web modeling languages.

2 Integration Methodology used in MDWEnet

In this section we discuss the general methodology used for the integration of WebML, OO-H, and UWE. We first explain why integration on the basis of already existing language artifacts is not possible. Second, we outline a model-based integration framework, and third, we discuss how to obtain the most important prerequisite for integration – the metamodels for the three web modeling languages.

Why is the integration on the basis of existing language artifacts not possible?

In Table 1 we present an overview of the formalisms used for defining WebML, OO-H, and UWE as well as the approaches' model storage formats. When looking at the languages' definitions, one can easily identify that each language is specified in a different formalism, even in different technological spaces [17], which a-priori prevents the comparability of the languages as well as model exchange.

For the integration of modeling languages in general and for web modeling languages in particular, the first requirement is that the languages are defined with the same meta-language. This enables to overcome syntactical heterogeneities and to compare the language concepts in the same formalism. Furthermore, defining languages with the same formalism also allows expressing their model instances in the same formalism which further fosters comparability of the languages' concepts

[1] www.webratio.com
[2] www.visualwade.com

and beyond allows the uniform processing of the models, e.g., their visualization or transformation.

Table 1. Differences concerning Language Definition and Model Storage.

	Language Definition	Model Storage
WebML	WebRatio, DTD[3]	XML documents
OO-H	VisualWade, Rational Rose Model	Proprietary format
UWE	ArgoUWE, UML Profile	XMI

Consequently, it seems necessary to split up the integration process in order to tackle two distinct integration concerns, namely *syntactical* integration and *semantical* integration: In the first step, i.e. the syntactic integration, the different formats used by WebML, UWE and OO-H are aligned towards one common integration format. For example, a WebML model represented in terms of an XML document has to be translated into this common integration format. The second step, i.e. the semantical integration step, covers the transformation of a model from one language into another one, e.g. from WebML to OO-H, while preserving the semantics of the input model within the output model. This transformation is based on transformation rules which require the input models to be available in the common integration format.

How to use model-based techniques for integration purposes?

We decided to apply a model-based approach and use techniques and technologies which have emerged with the rise of Model Driven Engineering (MDE) [3]. MDE mainly propagates two techniques which are relevant for integration purposes: (1) metamodels for defining the concepts of modeling languages, and (2) model transformations. Model transformations in the context of MDE can be divided into *vertical* model transformations and *horizontal* model transformations [19]. While the first kind concerns transformations between different abstraction levels, e.g., transforming platform-independent models into platform-specific models, the latter is used for transformations at the same level of abstraction, e.g., for model refactoring. Consequently, in this work we rely on horizontal model transformations.

In Figure 2, we present our model-based integration framework, which is based on the tool integration pattern of Karsai et al. [12]. The framework is built upon open-source technologies for MDE, which have been developed under the hood of the Eclipse project. In particular, we are using the Eclipse Modeling Framework (EMF) [5], as a model repository for a common syntactic integration format and EMF's Ecore, i.e. an implementation of the Meta Object Facility (MOF) standard [21], as the meta-language for defining the metamodels for WebML, OO-H, and UWE. Furthermore, we employ ATL [11] as model transformation language to implement the transformation rules and finally, the ATL engine for actually executing the transformation. In Figure 2, we also sketch the model-based integration process of

[3] Recently, two different proposals for a WebML metamodel have been published in parallel [20, 27].

WebML (WebRatio), OO-H (VisualWade), and UWE (ArgoUWE), which is described more detailed in the following:

1) **Syntactic Integration**. On the basis of *tool adapters* for bridging the native model storage formats of the approaches' tools towards the EMF models can be integrated syntactically. Thus, realizing import functionality the tool adapters have to parse the models in their native format and generate an XMI [22] version for the EMF. In addition, the tool adapters also must be capable of exporting the models by transforming them into the tools' native format.
2) **Semantic Integration**. After the syntactic integration, the user can focus on the correspondences between modeling concepts of different languages. This is done by relating the metamodel elements and implementing the integration knowledge in terms of ATL model transformation rules.
3) **Execution of the Transformations**. The model transformation rules then can be executed in a model transformation engine. More specifically, the ATL engine reads an input model, e.g. a WebML model, and generates an output model, e.g. an OO-H model, according to the transformation rules. Subsequently, the generated models can be exported via the specific tool adapter to the proprietary tool.
4) **Definition of a Common Metamodel for Web Modeling**. The top of Figure 2, illustrates the goal of MDWEnet, i.e., a unification of existing web modeling languages in terms of a common metamodel for web modeling. By defining the metamodels for WebML, OO-H, and UWE, as well as working out the integration knowledge in a first step, we hope that the creation of such a common metamodel is easier to achieve afterwards. For the future, the common metamodel for web modeling can serve as a pivot model and thus lowering the integration effort drastically.

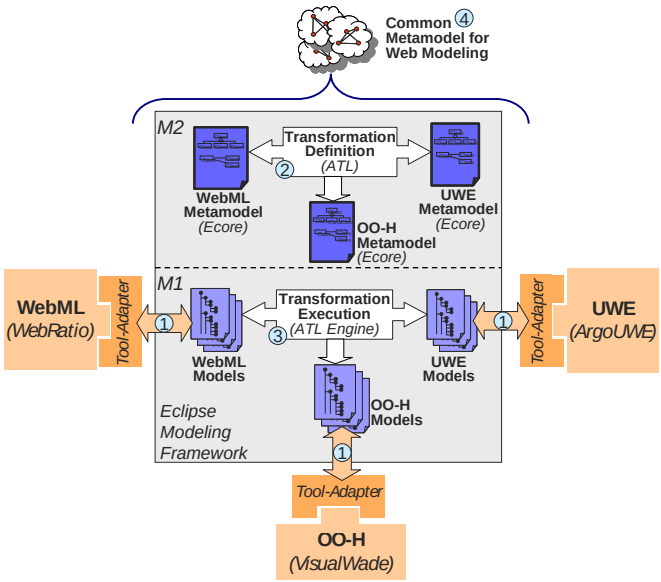


Figure 2: Model-based Integration Framework

What's missing for a model-based integration and how to close the gap?

As a key-prerequisite for a model-based integration, the metamodels for WebML, OO-H, and UWE must be available, which currently, however, is not the case. Within the MDWEnet initiative, we have decided to use a top-down approach for building the individual metamodels by starting with a focused set of requirements which are specific to the web modeling domain [30]. This approach has the advantage that we can concentrate on the core modeling constructs of the web modeling domain supported by the addressed approaches instead of focusing on a huge amount of concepts available in the individual approaches and implemented in their tools. Following this top-down approach, a set of modeling requirements for the core of web modeling were defined each focusing on a specific modeling problem. In the following, these requirements are briefly explained and categorized into requirements for *content modeling*, *hypertext modeling*, and *content management modeling*.

Layer 0 – Content Modeling. This layer is required to express domain objects and their properties on which the web application is built upon.
Example: Class *Student* with attribute *name*, *age*, and a relationship to the class *Professor*.

Layer 1 – Hypertext Modeling. This layer covers the requirements for web applications that allow navigation among the hypertext nodes and publish within a node the content extracted from domain objects (cf. Layer 0), possibly based on input provided by the user. The following four cases are subsumed by Layer 1:

- *Global Navigation:* This case requires a starting point in the web application, i.e. a home page, and subsequently, a navigation mechanism for moving to another page of the hypertext.
- *Content Publication:* This case requires a page, which publishes a list of domain objects and displays for each object a set of attribute values.
- *Parametric Content Publication:* This case requires a page, which publishes a list of domain objects each having attached a navigation mechanism, e.g., a button, an anchor. This mechanism shall allow the user to navigate to the details of the object.
- *Parametric Content Publication with Explicit Parameter Mapping:* This case requires one page, which contains an input form with various input fields. The user inputs are used for computing a set of domain objects. Thereby, the attribute values of the objects need to satisfy a logical condition including as terms the input provided by the user.

Layer 2 – Content Management Modeling. This layer covers the requirements for web applications that allow the user to trigger operations for updating the domain objects and their relationships (cf. Layer 0).
Example: Create a new instance of type *Student*. Update the *age* value of the instance of type *Student* where *name*='Michael Smith'.

The definition of metamodels is of course an art on its on and can be approached in different ways. For the purpose of this work it was decided to employ an example-based approach by a process of obtaining a metamodel from the aforementioned requirements as follows [16]: One or more concrete modeling examples were derived from the requirements specification and modeled in the respective modeling language within each approach's accompanying tool. The code generation facilities of each tool were then used to find out if the examples modeled were semantically identical, i.e.,

the generated applications should work in the same way. From these models the language concepts which have been used were identified, as well as how these concepts were related to each other. Consequently, this information is then defined in a corresponding metamodel. These metamodels should allow expressing the same models as within the approaches' tools, meaning the same information must be expressible in the models.

3 Preliminary Results

In this section we present our preliminary results. First, we briefly discuss the modeling examples realizing the MDWEnet's modeling requirements for web modeling and provide the resulting metamodels in Section 3.1. In order to illustrate how, on basis of those metamodels, the integration is realized with ATL in Section 3.2 we then present excerpts of the set of ATL transformation rules that have been defined for the metamodels.

3.1 Derived Metamodels

Our first task after the MDWEnet's modeling requirements for web modeling have been agreed on has been the derivation of concrete modeling examples realizing these requirements specifications. Inspired by previous examples in the web modeling domain, we are using excerpts of the often referred to album store running example [6], which covers all the aforementioned requirements. After defining the modeling examples, each of them was modeled within the approaches' tools, i.e., WebRatio, VisualWade, and ArgoUWE, respectively. Furthermore, we used the code generation facilities to compare the behavior of the models by executing the generated web applications.

On the basis of the modeling examples, each expressed in WebML, OO-H, and UWE, we identified the language concepts used in the individual examples and obtained first versions of the metamodels for WebML as well as for OO-H. The metamodel for UWE is currently under preparation. Beyond, we have grouped the metamodels' elements into packages which directly correspond to the layers of the modeling requirements presented in Section 2. In the following, the class structures of the metamodels for WebML and OO-H are presented and briefly explained. For more detailed versions of the metamodels the reader is referred to [30].

WebML Metamodel. In Figure 3, we present the resulting WebML metamodel, i.e., its packages, classes and their interrelationships. While the *Structure* package and *ContentManagement* package correspond to the *Layer 0* and *Layer 2* of the modeling requirements, respectively, for *Layer 1* two packages have been defined, namely *Hypertext* and *HypertextOrganization*.

The *Content* package contains modeling concepts that allow modeling the content layer of a web application. Since WebML's content model is based on the ER-model, it supports ER modeling concepts: An Entity represents a description of common features, i.e., Attributes, of a set of objects. Entities that are associated with each other

are connected by Relationships. Unlike UML class diagrams, ER diagrams model structural features, only.

The *ContentManagement* package contains modeling concepts that allow the modification of data from the content layer. The specific ContentManagementUnits are able to create, modify, and delete Entities (cf. EntityManagementUnit) as well as establish or delete Relationships between Entities from the content layer (cf. RelationshipManagementUnit).

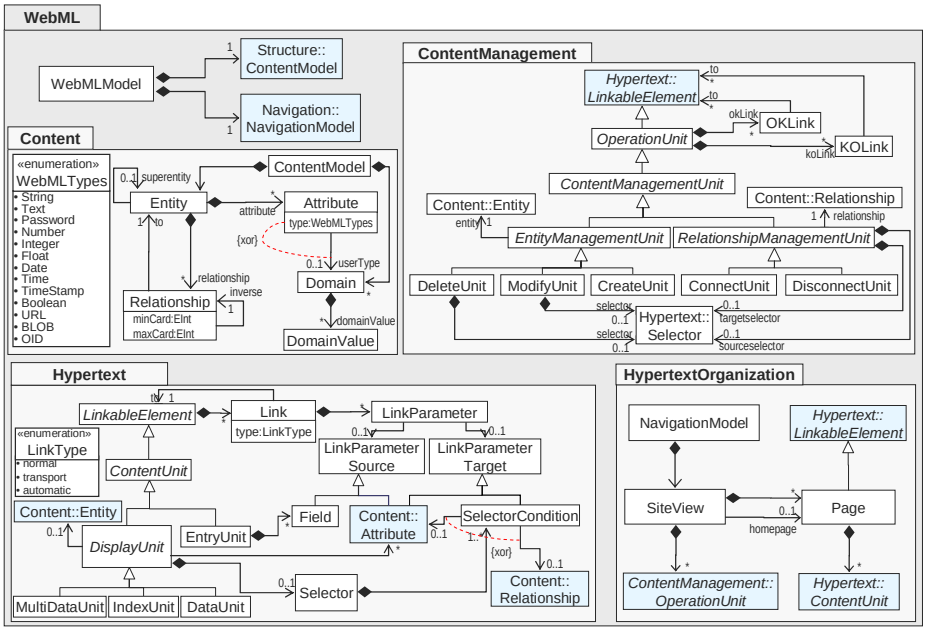


Figure 3: The WebML Metamodel

In contrast, the hypertext layer represents a view on the content layer of a web application, only. The *Hypertext* package summarizes ContentUnits, used, for example, to display information from the content layer which may be connected by Links in a certain way.

The *HypertextOrganization* package defines the Page modeling concept which is used to organize and structure information from the content layer, e.g., ContentUnits from the *Hypertext* package, SiteViews group Pages as well as operations on data from the content layer, e.g., OperationUnits from the *ContentManagement* package. More specifically, SiteViews represent groups of pages devoted to fulfilling the requirements of one or more user groups.

OO-H Metamodel. The class structure of the resulting OO-H metamodel is presented in Figure 4. Similar to the WebML metamodel, the *Layer 0* and *Layer 2* modeling requirements are realized by corresponding packages in the OO-H metamodel, i.e., the *Content* package and *Service* package, respectively. Concerning *Layer 1*, two packages have been defined, however, namely the *Navigation* and *Presentation* packages.

In the *Content* package, OO-H's content model is based on the UML class diagram: A Class represents a description of common structural and behavioral features, e.g., Attributes and Operations, respectively. Classes can be connected with each other via Associations.

The *Service* package contains the modeling concept ServiceNode that allows the execution of arbitrary operations defined at the content layer. The modeling concept ServiceLink is needed to connect NavigationalNodes with ServiceNodes, and in addition, to transport information in terms of arguments from NavigationalNodes to Operations.

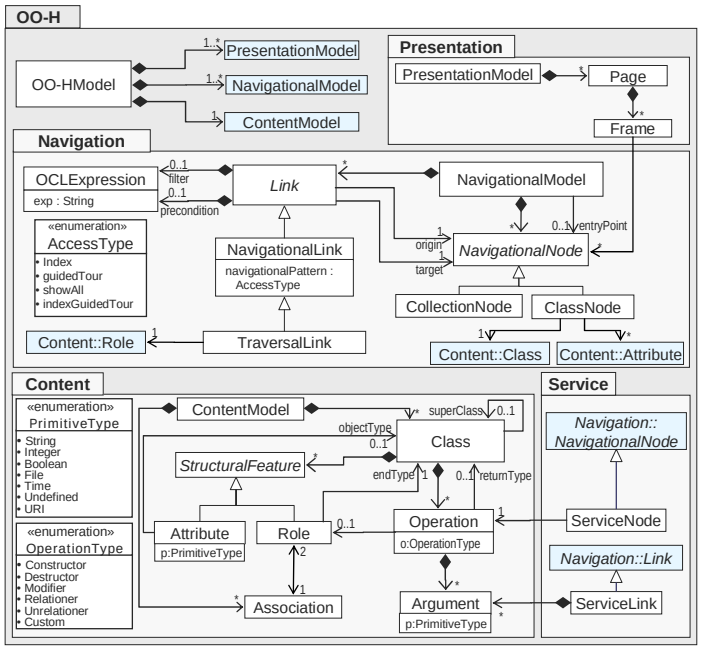


Figure 4: The OO-H Metamodel

The *Navigation* package represents a view on the content layer of a web application. In the *Navigation* package two types of NavigationalNodes can be distinguished, namely ClassNodes displaying information from the content layer, and Collections providing additional information such as navigation menus. Both types have in common that they may be connected by Links. OCLExpressions attached to Links either filter certain objects which should be displayed at the target NavigationalNode or are used as preconditions that must be assured to access the target NavigationalNode.

The *Presentation* package defines the Page modeling concept which is used to organize and structure the NavigationalNodes of the navigation layer.

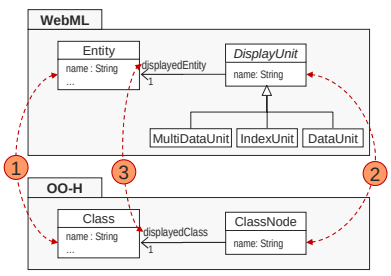
3.2 Model Transformations Using ATL

Following we discuss one representative example for the commonalities between WebML and OO-H, in order to exemplify how integration is achieved on the basis of metamodels and model transformation.

As already mentioned, ATL was used as model transformation language, which is a uni-directional, rule-based transformation language. For a full integration, consequently, transformation rules have to be specified for both directions, e.g. from WebML to OO-H and vice versa. An ATL rule consists of a query part (*from* keyword), which collects the relevant source model elements, and a generation part (*to* keyword) creating the target model elements.

In Figure 5 (a) we illustrate the semantic correspondences between WebML and OO-H metamodel elements and present two ATL rules implementing the transformation from WebML to OO-H in Figure 5 (b).

1. Rule *Entity_2_Class* is responsible for transforming each *Entity* of the WebML model into a *Class* in OO-H.
2. Rule *DisplayUnit_2_ClassNode* is responsible for transforming each instance of the concrete subclasses of *DisplayUnit* into *ClassNodes*.
3. This minimal example already shows some advantages of using ATL in contrast to using a general-purpose programming language. When executing ATL rules, a "trace model" is created transparently, which saves how instances are transformed. In our example the ATL engine traces which Class instance is generated for an Entity instance. Therefore, it is possible to retrieve the Class instance for the referenced Entity, which allows for the simple statement *cN.displayedClass <- dU.displayedEntity*.



(a) Metamodel Correspondences

```
1. rule Entity_2_Class{ 1
2.   from e: WebML!Entity
3.   to c: OOH!Class{
4.      c.name <- e.name,
5.      …
6.   }
7. }
8.
9. rule DisplayUnit_2_ClassNode{ 2
10.  from dU: WebML!DisplayUnit
11.  to cN: OOH!ClassNode{
12.     cN.name <- dU.name,
13.     cN.displayedClass <-
14.        dU.displayedEntity 3
15.  }
16. }
```

(b) Transformation Rules

Figure 5: Metamodel Correspondences and Transformation Rules Excerpt

4 Lessons Learned

Following, we summarize our lessons learned concerning the integration of WebML and OO-H. In general, the integration of WebML and OO-H has turned out to be straight-forward for the most part. At least for the core concepts of web modeling, which have been the focus of the MDWEnet initiative, there exist many commonalities between the two languages. Since the chosen modeling examples

could be realized in each language the languages can be considered to have "equal" expressivity with respect to the defined core requirements. Nevertheless, we also faced differences between the languages, which aggravated the integration. When integrating languages based on their metamodels, further information, which often are not covered by the metamodels, must be incorporated into the transformation rules. This kind of information is on the one hand incorporated into the code generator and on the other hand defined by the frameworks for which code is generated. Some of these differences and the complexity they introduced during integration are explained following the structure of the modeling requirements layers. Nevertheless, from our current experiences we are able to conclude that the differences can be eliminated within the transformations rules. Due to space restrictions and readability reasons we explain the transformation rules textually and refer the reader to [30] for detailed information on the ATL code.

4.1 Content Modeling (Layer 0)

As can be seen in Figure 1, WebML and OO-H have different origins. WebML is based on the ER-model, which is typically used in the context of modeling database schemas. In contrast, OO-H has emerged from an object-oriented background. Consequently, in WebML each Entity has a set of operations which are "implicitly" available and need not be defined by the modeler, i.e., WebML's ContentManagementUnits actually represent a data manipulation language (DML). These operations include typical create, update, and delete operations as well as operations for linking Entities (cf. Table 2). In contrast, in OO-H there are some predefined operation types available, which have to be explicitly defined for each Class by the modeler (cf. Table 2). Thus, when transforming WebML Entities in OO-H Classes, the default operations must be created for each corresponding Class, in order to ensure that OO-H's ServiceNodes can execute them.

Table 2:.Comparison of Object Operations between WebML and OO-H.

WebML Content Management Units	OO-H Operations
CreateUnit	Constructor()
ModifyUnit	Modifier()
DeleteUnit	Destructor()
ConnectUnit	Relationer()
DisconnectUnit	Unrelationer()

Example: Figure 6 (a) shows an excerpt of the content model of the album store example. In Figure 6 (b) we depict the corresponding OO-H content model that needs to be generated by the transformation rules. For each Entity in the content model of WebML an OO-H Class is generated. Besides transforming the Entities' Attributes, in OO-H the Constructor(), Destructor(), and Modifier() operations must be defined for the Class as well. Likewise for each Relationship of an Entity the Relationer() and Unrelationer() operations have to be generated for the corresponding Class in the OO-H content model.

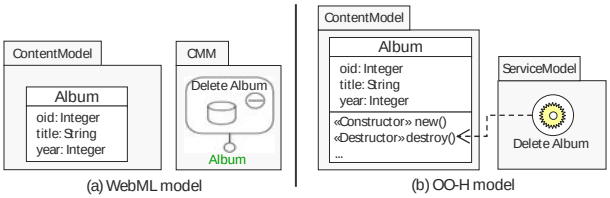


Figure 6: Content Modeling and Hypertext Modeling in WebML and OO-H

4.2 Hypertext Modeling (Layer 1)

Generally speaking, one could say that WebML is the more explicit language compared to OO-H, i.e., in the way that there are much more language concepts used. In particular, this is the case for hypertext modeling where OO-H uses a minimal set of concepts, which are refined with OCLExpressions, i.e., preconditions and filters, for Links. In contrast to WebML, where various types of ContentUnits are available, OO-H uses the concepts ClassNode and Collection, only. The actual content and behavior of ClassNodes is defined by their incoming Links. Furthermore, for parameter passing WebML offers LinkParameters with explicit source and target parameter bindings, while in OO-H this is again expressed by OCLExpressions.

Besides the difference in the number of explicit concepts, WebML and OO-H both use their own selector language for computing the content to be displayed. While in OO-H the Object Constraint Language (OCL) [23] has been reused and extended, WebML's selector language is defined within the metamodel as well as based on the concepts of Selector and SelectorCondition [30]. However, in the current version of the OO-H metamodel, the modified grammar for the OCL is not yet covered as it is done for the WebML selector language in the WebML metamodel. Thus, currently the OCL statements are hard-coded in the transformations rules as ordinary Strings. Incorporating the OCL grammar into the OO-H metamodel and the refinement of the model transformations in order to define the OCL statements as model elements is subject to future work. In the following, an example illustrating these differences between WebML and OO-H is given.

Example: A search scenario is given, where in the first page the user provides input, i.e., a certain year, for searching the set of albums. Figure 7 (a) shows the example modeled with WebML[4], where the EntryUnit *AlbumSearch* with a Field named '*from*' represents the input form. The Link to the IndexUnit *AlbumResults* carries the user input in terms of a parameter. Therefore, a LinkParameter is assigned to the Link, which has as LinkParameterSource the input Field and as LinkParameterTarget the SelectorCondition of the *AlbumResults* IndexUnit. This SelectorCondition computes the subset of all albums where the input value of the user equals the value of the year attribute. In Figure 7 (b), the same information is modeled with OO-H, where a separate concept for the information that is transported via Links is not available. More specifically, the search scenario can be modeled with a Collection *AlbumSearch* and a Link to the ClassNode *AlbumResults*. The Link

[4] Please note that ellipse-shaped legends are not part of WebML's notation.

contains a filter OCLExpression `dst.year = ?,` with the question mark standing for the user's input value and `dst.year` meaning the '*year*' Attribute of the *Album* Class.

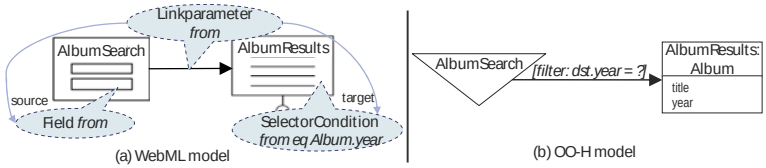


Figure 7: WebML Unit Types vs. OO-H Filter Conditions

This example illustrates the need to integrate the various WebML ContentUnits with OO-H Collections and ClassNodes as well as WebML LinkParameter and SelectorConditions with OO-H filter OCLExpressions.

4.3 Content Management Modeling (Layer 2)

Due to the differences at the content modeling layer, the modeling concepts for content management modeling are also differently defined in WebML and OO-H. For each operation on Entities of the content modeling layer WebML offers an explicit modeling concept, e.g., CreateUnit, DeleteUnit, and ConnectUnit. In contrast, OO-H's *Service* package encompasses two concepts only, namely ServiceNode and ServiceLink. This means that OO-H does not differentiate between the typical create, update, and delete operations by defining sub-concepts of ServiceNode. Instead a ServiceNode has a reference to the Operation which should be executed when the ServiceNodes is entered.

Example: The given scenario describes the deletion of a specific album by an authorized user. In Figure 6 (a) a DeleteUnit *DeleteAlbum* is shown which might be accessed, e.g., through an IndexUnit *AlbumSearch* (cf. Figure 7 (a)). Likewise, concerning OO-H a ServiceNode *DeleteAlbum* might be accessed, e.g., through a ClassNode (cf. Figure 7 (b)). For the given scenario we assume that the Selectors and SelectorConditions are translated according to the transformation rules defined for the hypertext modeling layer. Beyond, each OperationUnit from the WebML model needs to be translated into a ServiceNode in the OO-H model. Thereby, the reference identifying the corresponding operation type (cf. Table 2) must be set for the ServiceNode.

5 Conclusions and Future Challenges

In this paper we have presented our methodology of integrating three of the most prominent web modeling approaches, namely WebML, OO-H, and UWE, on the basis of a set of core web modeling requirements. As a proof of concept, we have defined the core languages in Ecore-based metamodels and subsequently, have implemented the integration in ATL model transformations rules. From our preliminary results and lessons learned from the integration of WebML and OO-H sofar, we conclude that the

core of the three languages can be integrated without loosing information. Nevertheless, the presented results are only a first step in the direction of a full integration of the languages and to the definition of a common metamodel for web modeling.

Future challenges concerning the integration of WebML and OO-H include the finalization of the integration for their core modeling concepts which requires the OCL version used in OO-H to be incorporated in the metamodel. Therefore, we plan to employ the EBNF_2_Ecore transformer [29], which is capable of generating the corresponding metamodel elements from a textual EBNF grammar. On the basis of this we intend to finalize the transformation rules from OO-H to WebML.

The UWE metamodel is currently under preparation. As soon as a first stable version is available, we plan to integrate UWE with the two other modeling languages as well. We expect that a third language would bring further insights for building the common metamodel for web modeling and on these results a first unification of the modeling concepts can be proposed for the core requirements.

Beyond the core requirements, the modeling requirements and modeling examples need to be extended to other web modeling concerns such as presentation, context-awareness, and business processes in the future to broaden the view on the unification of the modeling concepts. Furthermore, a refinement of possible variants of modeling requirements, in order to find further sub-concepts and alternative modeling styles would be of interest.

Acknowledgments. We would like to thank the members of the MDWEnet initiative that have contributed to this paper in terms of preliminary work, including Pierro Fraternali (Politecnico di Milano) for setting up the set of modeling requirements and Cristina Cachero, Jaime Gomez, Santiago Meliá, Irene Garrigós (Universidad de Alicante) as well as Nora Koch (LMU München) for their work on the UWE metamodel.

References

1. Baresi, L., Colazzo, S., Mainetti, L., and Morasca, S.: W2000: A Modeling Notation for Complex Web Applications. In Mendes, E. and Mosley, N. (eds.) *Web Engineering: Theory and Practice of Metrics and Measurement for Web Development*. Springer, 2006.
2. Baumeister, H., Knapp, A., Koch, N., Zhang, G.: Modelling Adaptivity with Aspects. *Proc. 5th Int. Conf. on Web Engineering (ICWE05)*, Sidney, Australia, July 2005.
3. Bézivin, J.: On the Unification Power of Models, *SoSyM*, 4(2), 2005.
4. Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process modeling in Web applications. *ACM Trans. Softw. Eng. Methodol.* 15(4), 2006.
5. Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., and Grose, T.J.: *Eclipse Modeling Framework*, Addison-Wesely, 2004.
6. Ceri, S., Daniel, F., Matera, M., Facca, and F.: Model-driven Development of Context-Aware Web Applications, *ACM TOIT*, 7(2), 2007, to appear.
7. Ceri, S., Fraternali, P., Bangio, A., Brambilla, M., Comai, S., and Matera, M.: *Designing Data-Intensive Web Applications*, Morgan-Kaufmann, 2003.
8. De Troyer, O., Casteleyn, S., and Plessers, P.: Using ORM to Model Web Systems, *Proc. Int. Workshop on Object-Role Modeling*, Agia Napa, Cyprus, October 2005.

9. Garrigós, I., Casteleyn, S., Gómez, J.: A Structured Approach to Personalize Websites using the OO-H Personalization Framework. *Proc. of the 7th Asia-Pacific Web Conference (APWeb 2005)*, Shangai, China, March 2005.
10. Gómez, J., Cachero, C., Pastor, O.: Conceptual Modeling of Device-Independent Web Applications. *IEEE MultiMedia*, 8(2), 2001
11. Jouault, F., Kurtev, I.: Transforming Models with ATL: Proceedings of the Model Transformations. *Proc. of the Model Transformations in Practice Workshop at MoDELS*, Montego Bay, Jamaica, October 2005.
12. Karsai, G., Lang, A., Neema, S.: Tool Integration Patterns. Workshop on Tool Integration in System Developement, ESEC/FSE, Helsinki, Finland, September 2003.
13. Koch, N., Kraus, A.: Towards a Common Metamodel for the Development of Web Applications. *Proc. of the 3rd Int. Conf. on Web Engineering (ICWE 2003)*, July 2003.
14. Koch, N., Kraus, A., Cachero, C., Meliá, S.: Integration of Business Processes in Web Application Models. *J. Web Eng.,*. 3(1), 2004.
15. Koch, N., Zhang, G., Escalona, M.: Model transformations from requirements to web system design. *Proc. of the 6th Int. Conf. on Web Engineering (ICWE 2006)*, 2006.
16. Koch et al. MDWEnet: A Practical Approach to achieve Interoperability of Model-Driven Web Engineering Methods. In preparation, 2007.
17. Kurtev, I., Bézivin, J., and Aksit, M.: Technological spaces: An initial appraisal. *Proc. Of Int. Federated Conf. (DOA,ODBASE, CoopIS)*, Los Angeles, 2002.
18. Meliá, S., Gómez, J.: The WebSA Approach: Applying Model Driven Engineering to Web Applications. *J. Web Eng.*, 5(2), 2006.
19. Mens, T., Czarnecki, K., Van Gorp, P.: A Taxonomy of Model Transformations. *Language Engineering for Model-Driven Software Development - Dagstuhl Seminar Proceedings*, Dagstuhl, Germany, 2005.
20. Moreno, N., Fraternali, P., Vallecillo, A.: WebML modeling in UML. *IET Software Journal*, 2007, to appear.
21. Object Management Group (OMG). *Meta Object Facility (MOF) 2.0 Core Specification Version 2.0*. http://www.omg.org/docs/ptc/04-10-15.pdf, October 2004.
22. Object Management Group (OMG), *MOF 2.0/XMI Mapping Specification, v2.1*, http://www.omg.org/docs/formal/05-09-01.pdf, September 2005.
23. Object Management Group (OMG), *OCL Specification Version 2.0*, http://www.omg.org/docs/ptc/05-06-06.pdf, June 2005.
24. Object Management Group (OMG). *UML Specification: Superstructure Version 2.0*. http://www.omg.org/docs/formal/05-07-04.pdf, August 2005.
25. Pastor, O., Fons, J., Pelechano, V., Abrahao, S.: Conceptual Modelling of Web Applications: The OOWS Approach. In E. Mendes and N. Mosley (eds.) *Web Engineering: Theory and Practice of Metrics and Measurement for Web Development*. Springer, 2006.
26. Rossi, G., Schwabe, D.: Model-Based Web Application Development. In E. Mendes and N. Mosley (eds.) *Web Engineering: Theory and Practice of Metrics and Measurement for Web Development*. Springer, 2006.
27. Schauerhuber, A., Wimmer, M., Kapsammer, E., Schwinger, W., and Retschitzegger, W.: Bridging WebML to Model-Driven Engineering: From DTDs to MOF. *IET Software Journal*, 2007, to appear.
28. Schwinger, W., Koch, N.,: Modelling Web Applications. In Kappel, G., Pröll, B., Reich, S., Retschitzegger, W. (eds.) Web Engineering - Systematic Development of Web Applications, Wiley, June 2006.
29. Wimmer, M., Kramler, G.: Bridging Grammarware and Modelware. *Proc. of Satellite Events at the MoDELS 2005 Conference*, Montego Bay, Jamaica, October 2005.
30. http://www.big.tuwien.ac.at/projects/mdwenet/