Bridging WebML to Model-Driven Engineering: From DTDs to MOF

Andrea Schauerhuber
Women's Postgraduate College of Internet Technologies
Vienna University of Technology
Favoritenstrasse 9-11
1040 Vienna, Austria
+43 1 58801 18894
schauerhuber@wit.tuwien.ac.at

Manuel Wimmer
Business Informatics Group
Vienna University of Technology,
Favoritenstrasse 9-11
1040 Vienna, Austria
+43 1 58801 18829
wimmer@big.tuwien.ac.at

Elisabeth Kapsammer
Information Systems Group
Johannes Kepler University Linz,
Altenberger Strasse 69
4040 Linz, Austria
+43 70 2468 9852
ek@ifs.uni-linz.ac.at

Wieland Schwinger
Department of Telecooperation,
Johannes Kepler University Linz
Altenberger Strasse 69
4040 Linz, Austria
+43 70 2468 9260
wieland.schwinger@jku.ac.at

Werner Retschitzegger
Information Systems Group
Johannes Kepler University Linz,
Altenberger Strasse 69
4040 Linz, Austria
+43 70 2468 8883
werner@ifs.uni-linz.ac.at

ABSTRACT

Meta-models are a prerequisite for model-driven engineering (MDE) in general and consequently for model-driven web engineering in particular. Various modelling languages, just as in the web engineering field, however, are not based on meta-models and standards, like OMG's prominent Meta Object Facility (MOF). Instead they define proprietary languages rather focused on notational aspects. Thus, MDE techniques and tools can not be deployed for such languages preventing to exploit the full potential of MDE in terms of standardized storage, exchange, and transformation of models. The WebML web modelling language is one prominent example that does not yet rely on an explicit meta-model in the sense of MDE. Instead, it is defined in terms of a document type definition (DTD), i.e., a grammar-like textual definition for specifying the structure of XML documents, and implicitly within the accompanying tool. Code generation then has to rely on XSLT-based model-to-code transformations.

In this paper, we propose a meta-model for WebML which is based on the Meta Object Facility (MOF) to bridge WebML to MDE. To establish such a meta-model, instead of re-modelling WebML's meta-model from scratch, a semi-automatic approach is provided that allows generating MOF-based meta-models on the basis of DTDs. The meta-model for WebML accomplishes the following aims: First, it represents an initial step towards a transition to employing MDE techniques (e.g., model transformations or language extensions through profiles) within the WebML design methodology. Second, the provision of a MOF-based meta-model ensures interoperability with other MDE tools. Third, it represents an important step towards a common meta-model for Web modeling in the furture.

Keywords

Web Modelling Language, Meta-model, DTD, Model-Driven Web Engineering

1. INTRODUCTION

Model-driven engineering (MDE) [1] has received considerable attention during the last years and is well on its way to becoming a promising paradigm in software engineering. In MDE, models replace code as the primary artefacts in software development processes. MDE forces developers to focus on modelling the problem domain and not on programming one possible platform-specific solution. Thus, the abstraction from specific programming platforms and the definition of model transformations allow generating several platform-specific implementations. In this respect, the key prerequisite for MDE is the employment of a modelling language definition standard such as the Object Management Group's (OMG) Meta Object Facility (MOF) [2], allowing for standardized storage (e.g., Eclipse Modelling Framework – EMF [3]), exchange (e.g., XML Metadata Interchange Format - XMI [4]), and transformation (e.g., Query/View/Transformation - QVT [5]) of models.

Considering MDE in the area of Web application development, various modelling approaches have been proposed in the past 10 years, such as WebML [6], UWE [7], OO-H [8], W2000 [9], WSDM [10], OOHDM [11], and OOWS [12] aiming at counteracting a technology-driven and ad hoc development of Web applications. While some of these approaches already provide tools and techniques for modelling Web applications in a platform-independent way, their code generation facilities, if existent, mostly support only one specific platform, yielding transformations from a model directly to code and do not profit from the above mentioned benefits of MDE. For these reasons, although first proposals for a transition to the model-driven paradigm in Web engineering have already been made, e.g., W2000 [13], UWE [7], Muller et al. [14], the majority of existing Web development methodologies are not yet defined using a language definition standard in the sense of MDE.

Amongst the approaches not yet in line with MDE, WebML is one of the most elaborated Web modelling languages stemming from academia and is supported already over several years by the commercial tool WebRatio[1]. WebML's language concepts are partly defined in terms of XML document type definitions (DTDs) [15], i.e., a grammar-like textual definition for specifying a structure for XML documents and partly hard-coded within the corresponding modelling tool. In contrast to MOF, however, DTDs represent a rather restricted mechanism for describing

[1] www.webratio.com

modelling languages, e.g., with respect to expressiveness, extensibility as well as readability and understandability for humans. Furthermore, WebRatio internally represents models in XML [15], and second, uses XSLT [16] for code generation. Since XSLT, however, is not intended for heavy structural transformations, writing XSLT programs for code generation is difficult and error-prone. Concerning these problems, a meta-model-based approach allows expressing transformation rules in a more compact and readable way by using existing MDE conform model transformation languages such as QVT [5] and ATL [17].

In order to define WebML's language concepts in an MDE-suitable way and thus to bridge WebML to MDE, a MOF-based meta-model for WebML is a prerequisite. Considering the language's size, however, we refrain from re-modelling WebML from scratch, since this would be a cumbersome and error prone process. Instead, we reuse the existing DTD-based language specification and concepts hard-coded within WebRatio and propose a semi-automatic process for MOF-based meta-model generation from DTDs.

The contributions of a meta-model for WebML are as follows: (1) Such a meta-model represents an important prerequisite and thus, an initial step towards a transition to employ model-driven engineering techniques (e.g., model transformations or language extensions through profiles) within the WebML design methodology. (2) The provision of a MOF-based meta-model ensures interoperability with other MDE tools. Moreover, our transformation approach enables the visualization of any DTD-based language in terms of MOF-based meta-models and thus, enhances the understandability of those languages. (3) Additionally, it is also an important step towards a common reference meta-model for Web modeling languages [7] in the future.

The remainder of this paper is organized as follows. As a prerequisite to establishing our semi-automatic approach to generating MOF-based meta-models from DTDs, Section 2 explains concepts from DTDs and meta-models as well as certain deficiencies of DTDs when used as a mechanism for defining modelling languages. Section 3 then describes our transformation process, including a set of transformation rules, and heuristics giving indication for a manual refactoring, as well as a presentation of the implementation of the semi-automatic transformation approach in the form of our MetaModelGenerator (MMG). In Section 4, we apply our transformation framework to the WebML DTD and discuss the resulting WebML meta-model. An evaluation of the meta-model's completeness and quality is given in Section 5. While Section 6 gives an overview of related work, we finally outline our conclusions and future work in Section 7.

2. DTDs AND MOF AT A GLANCE

A natural prerequisite for bridging WebML to MDE is to elaborate on the expressiveness of DTDs, i.e., the concepts used to describe the WebML language, with respect to MOF. In the context of OMG's meta-level architecture [18], this means that a WebML model, which is represented by an XML document, relates to the instance level (M1). Such a model has to conform to the WebML DTD describing the WebML language concepts at the meta-level (M2). The WebML DTD in turn is based on the DTD-grammar [15] defined at the meta-meta-level (M3)[2]. Analogously, MOF concepts defined at M3 are used to describe meta-models in the sense of MDE at M2. In our case, this is the targeted WebML meta-model of which instances in terms of WebML models can be formulated at M1 (cf. Figure 1).

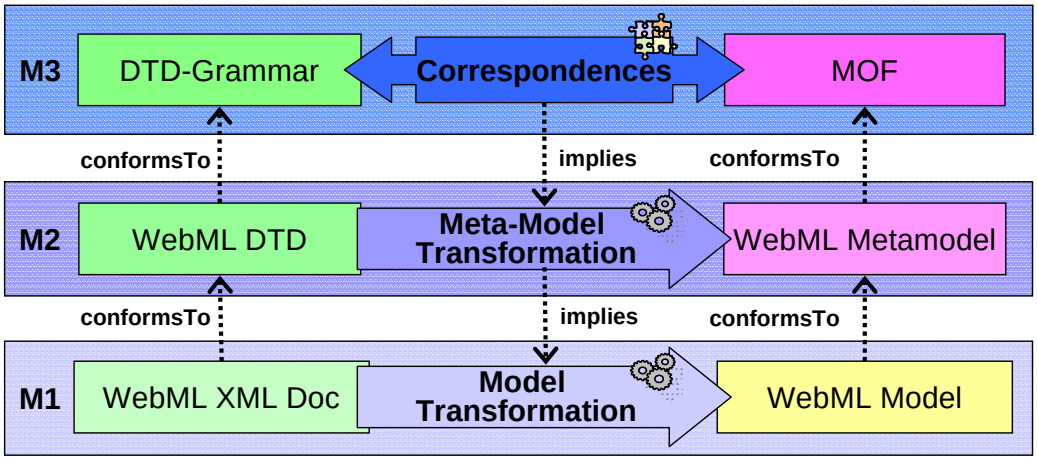

Figure 1: Interrelationships Between the Language Layers of DTD and MOF

This discussion shows that the two M3 level formalisms, in terms of the DTD-grammar and MOF respectively, represent the concepts on which to identify correspondences. In turn, these correspondences serve as a basis for M2-level and consecutively M1-level transformations done by our framework presented in Section 3.

In the following, UML class diagrams [18] are used as a common formalism to explain and to illustrate the major concepts of the DTD-grammar (cf. Section 2.1) and MOF (cf. Section 2.2).. This explanation serves as the basis for identifying differences in the expressiveness of the two meta-meta-languages (cf. Section 2.3).

[2] Please note that, while in case of WebML a DTD is used to define a modelling language and therefore can be assigned to the M2 level, DTDs typically are used at M1 in order to describe the structure of data stored in XML documents.

2.1 Document Type Definition (DTD) Concepts

The UML class diagram given in Figure 2 is based on our previous work [19] and depicts those DTD concepts present in WebML DTD. These concepts need then to be considered for finding correspondences to MOF concepts and consequently are reviewed briefly in the following.

In general, DTD's contain markup declarations comprising element types (`XMLElemType`) and attributes (`XMLAttributes`) for defining the logical structure and primarily entity declarations (e.g., `ParameterEntityDec`), as a reuse mechanism for certain reoccurring markup declarations, for defining the physical structure.

Element types, being first-class citizens in DTDs, have a name and are specialized into `XMLAtomicET` (contains no other element types but character data), `XMLEmptyET` (no content is allowed), `XMLAnyET` (the content is not constrained), `XMLCompositeETMixedContent` (a mix of character data and child element types), and `XMLCompositeETElemContent` (consists of an `XMLContentParticle`). An `XMLContentParticle` either is an `XMLSequence`, an `XMLChoice`, or an `XMLElemType`. An `XMLChoice` or an `XMLSequence` can be enclosed in parentheses for grouping purposes and suffixed with a '?' (zero or one occurrences), '*' (zero or more occurrences), or '+' (one or more occurrences), whereas the absence of a particular symbol denotes a cardinality of exactly one.

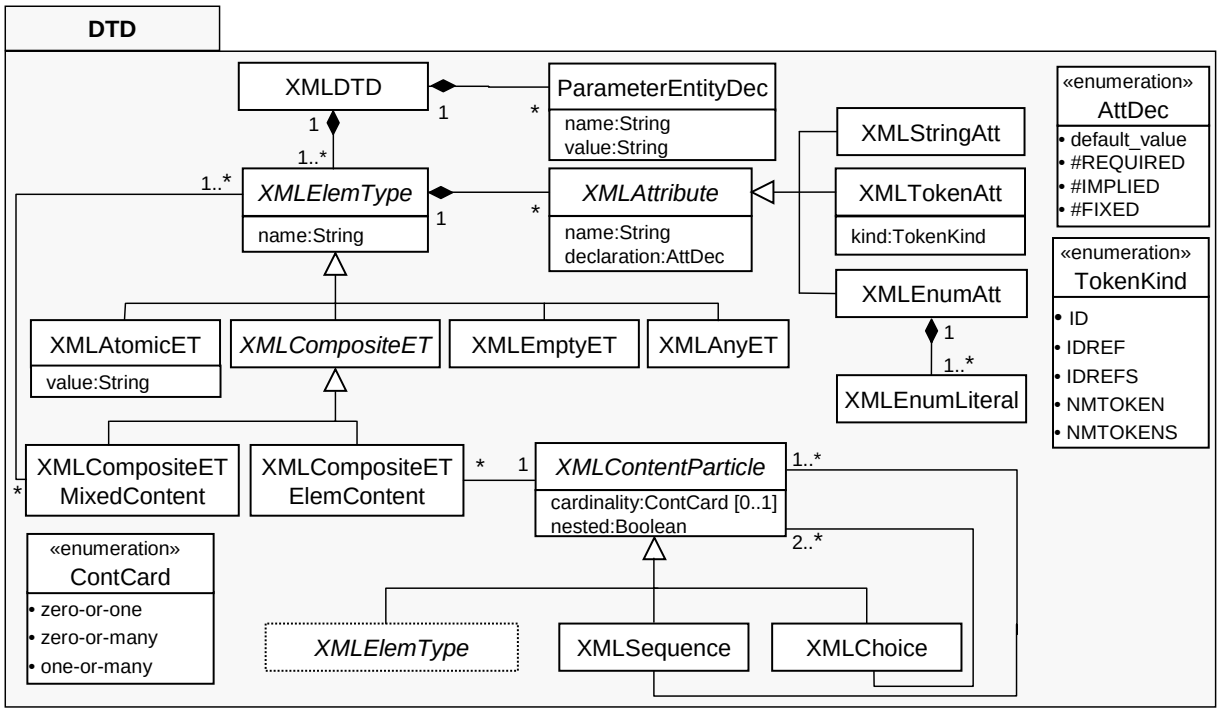

Figure 2: Overview of Relevant DTD Language Concepts

Attribute declarations declare one or multiple `XMLAttributes` (i.e., name-value pairs) for a single element type. Each `XMLAttribute` has a name, a data type, and a default declaration. The most commonly used data types for attributes are: `CDATA` (`XMLStringAtt`), `ID`, `IDREF` (refers to one ID-typed element), `IDREFS` (refers to multiple ID-typed elements), and Enumeration (`XMLEnumAtt`). For default declarations there are four possibilities: `#IMPLIED` (zero or one), `#REQUIRED` (exactly one), `#FIXED` (the attribute value is constant and immutable), and Literal (the default value is a quoted string).

Please note that, we however ignore the order constraints imposed by DTDs and the majority of physical structures of the DTD-grammar (i.e., general entity declarations, notation declarations as well as `XMLAttributes` of type ENTITY, ENTITIES, and NOTATION), since they are relevant to XML documents rather than DTDs and the purpose of finding correspondences to MOF concepts is rather questionable [20].

2.2 MOF Concepts in Terms of Ecore

In the following we briefly present the most important concepts of MOF with respect to finding correspondences to DTD concepts. Note that, by the time of writing there is no standardized implementation of MOF 2.0 available. Therefore, we use Ecore [3] which is a slightly modified EMOF implementation in Java, provided by the widespread Eclipse Modelling Framework (EMF). It is interesting to note that MOF consists of two parts, namely *Essential MOF (EMOF)* and *Complete MOF (CMOF)*. While the former is a small language based on the principles of object-orientation for defining modeling languages, the latter is a more complex language which provides also concepts for the specification of implementation details of model repositories.

In Figure 3, we summarize the most important concepts of Ecore with respect to finding correspondences to DTD concepts.

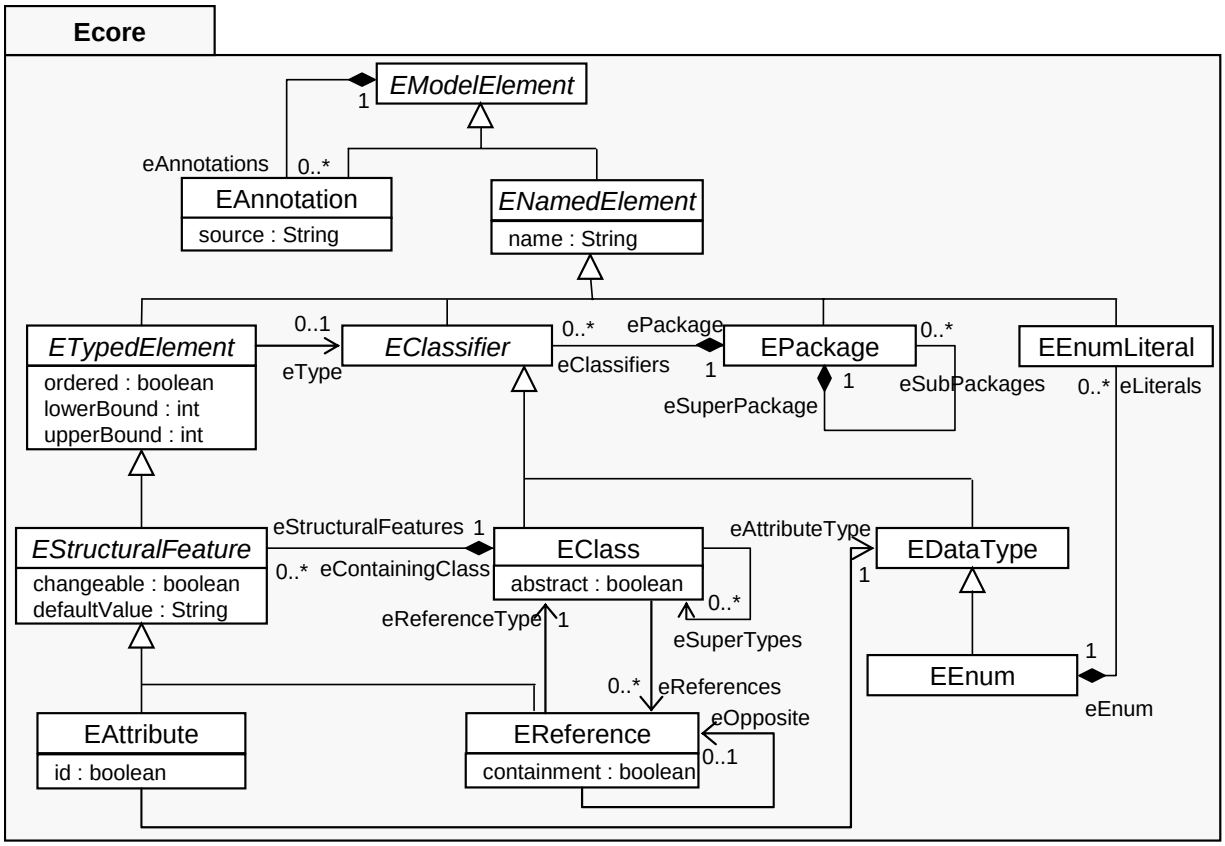

Figure 3: Overview of Relevant Ecore Language Concepts[3]

In Ecore there is a single root concept (`EModelElement`) being the base class for all modelling elements. Its sub-class `EAnnotation` is used for describing additional information which cannot be presented directly in Ecore-based meta-models. `ENamedElement` is the base for the remaining Ecore modelling elements, because it provides for a name attribute. An `EClassifier` represents a type in a model and as such, is the base class for `EClass` and `EDataType`, whereas an `ETypedElement` serves as the base for other modelling concepts having a type such as `EStructuralFeature`, which in turn represents a structural feature of an `EClass`. `EClasses` are the first-class citizens in Ecore-based meta-models. An `EClass` may have multiple `EReferences` and `EAttributes` for defining its structural features as well as multiple super `EClasses`. An `EAttribute` is part of a specific `EClass` and can have, as any `ETypedElement`, a lower and an upper bound multiplicity. Additionally, it can be specified as being able to uniquely identify the instance of its containing EClass (`id`) and as being `ordered`. The type of an attribute is either a simple

[3] Based on http://download.eclipse.org/tools/emf/2.2.0/javadoc/org/eclipse/emf/ecore/package-summary.html

data type or an enumeration. `EString`, `EBoolean`, `EInt`, and `EFloat` are part of Ecore's default data types set. `EEnum` allows to model enumerations defined by an explicit list of possible values, i.e., its literals (`EEnumLiterals`).

Analogous to `EAttribute`, an `EReference` is part of a specific `EClass` and can have a lower and an upper bound multiplicity. An `EReference` refers to an `EClass` and optionally to an opposite `EReference` for expressing bi-directional associations. Besides, an `EReference` can be declared as a being `ordered` and as a `containment` reference

`EPackages` group related `EClasses`, `EEnums`, as well as related `EPackages`. Each element is directly owned by an `EPackage` and each `EPackage` can contain multiple model elements.

2.3 DTD Deficiencies

When comparing meta-models specified in Ecore to DTDs it can be seen that DTDs considerably lack extensibility, readability, and understandability for humans, and above all expressiveness [20]. In the following, we describe the major deficiencies of DTDs if used as a mechanism for defining modelling languages. Note that some of these deficiencies have been resolved with the introduction of XMLSchema [21], such as limited set of data types (cf. Section 2.3.1), awkward cardinalities (cf. Section 2.3.4), missing inheritance concept (cf. Section 2.3.6), and no explicit grouping mechanism (cf. Section 2.3.7). A profound comparison between DTD and XMLSchema can be found in Lee et al. [22]. Nevertheless, in context of WebML which is based on DTDs the following shortcomings need to be addressed:

2.3.1 Limited Set of Data Types

In contrast to Ecore, DTDs have a limited set of data types that can not be extended to support, e.g., Integer or Float data types. While the provided data types generally are based on Strings, some other data type may be simulated by defining an enumeration with specific literals. In this way, a Boolean attribute can be simulated by an attribute of type Enumeration having two literals, e.g., true and false. Enumerations, however, can not capture numeric data types such as Integer or Float, which are naturally infinite.

2.3.2 Unknown Referenced Element Type(s)

DTDs referencing mechanism is based on IDREF(S)-typed attributes, which are able to reference any element type having an ID-typed attribute. Unlike Ecore, which provides typed references, it is not possible to identify the element type that may be referenced from an IDREF(S)-typed attribute based on the information given in DTDs. DTDs even allow to reference different element types. These referenced element types potentially have a common super type,

which, however, cannot be specified in the DTD. Due to this peculiarity of DTDs, it is neither possible to determine which element type(s) may be referenced based on the information given in the DTD nor if a certain set of element types may be referenced, only.

2.3.3 No Bi-Directional Associations

While Ecore offers bi-directional associations, in DTDs only uni-directional references can be specified. There is no way to specify that two uni-directional references in combination form a bi-directional association either.

2.3.4 Awkward Cardinalities

DTDs offer a restricted mechanism to specify cardinalities. More specifically, in contrast to Ecore there are no explicit concepts for defining cardinalities having a lower bound greater than one and for defining cardinalities having an upper bound other than '1' or '*'. This can only be simulated in an awkward way by redundantly specifying a certain element type within the content specification of its related (parent) element type according to the required cardinality.

2.3.5 Missing Role Concept

In DTDs, there is no explicit concept to express that an element type can be deployed in different contexts, i.e., a role concept such as in Ecore is missing. Thus, in DTDs this is sometimes bypassed by defining each role as a separate element type each named after the specific role they represent, and redundantly defining the same content and attribute specifications.

2.3.6 Missing Inheritance Concept

DTDs are not able to express inheritance relationships between element types as naturally provided for Ecore. Hence, DTDs can not profit from the typical benefits of inheritance such as reuse.

2.3.7 No Explicit Grouping Mechanism

There is no explicit mechanism to group parts of a DTD that semantically belong together like supported in Ecore. Nevertheless, some designer of DTD languages bypass this deficiency by encapsulating parts of a DTD in separate files and employing parameter entities to import these separated definitions where appropriate.

2.3.8 Missing Constraint Mechanism

A mechanism for defining complex constraints, as is supported in Ecore by using OCL [21], is not provided for DTDs. Thus, even simple XOR constraints, which are often required in meta-models, can not be specified. This

deficiency is specifically problematic, since possible ambiguities in DTDs can not be resolved and XML documents, while valid according to their DTD, might still not represent the domain data correctly.

3. DTD-TO-ECORE TRANSFORMATION FRAMEWORK

On the basis of the discussion of DTD and Ecore concepts as done in the previous section, we are now able to give more insight into our semi-automatic transformation approach, which is based on our previous work [24]. Generally, our transformation approach consists of an automatic phase and a manual phase comprising all together three steps (cf. Figure 4). The first phase is responsible for automatically generating a first version of the WebML meta-model and is performed by a component called MetaModelGenerator (MMG). The meta-model generator employs, in a first step, a set of transformation rules expressing all identified non-ambiguous correspondences between DTD concepts and Ecore concepts (cf. Section 3.1). In a second step of that phase, a set of heuristics is applied, dealing mainly with the aforementioned deficiencies by proposing possible correspondences (cf. Section 3.2). On the basis of these suggestions, in a third step, the user needs to manually validate the generated meta-model and refactor it accordingly in a second phase (cf. Section 3.3). The implementation architecture of our transformation framework is presented in Section 3.4.

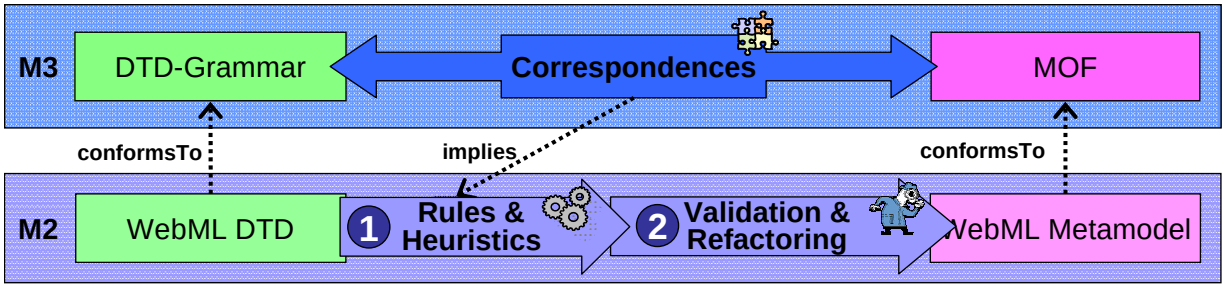

Figure 4: Two Phase Semi-Automatic Transformation Approach

To illustrate our transformation approach we use a small sub-set of the WebML DTD and show the effects of applying transformation rules, heuristics and refactoring steps in terms of the resulting WebML meta-model (M2). In particular we use part of the concepts used by WebML to represent a Web application's content layer which in fact resembles the well-known ER-model [25].

The focus in this section, however, is on the illustration of our transformation approach, i.e., the consecutive application of (some) transformation rules, heuristics, and refactoring steps. We therefore refrained to use an example using concepts for modelling a web application's hypertext since the concepts are too numerous and often

relate to concepts defined for the content layer In this respect, the WebML content layer serves as a self-contained and small example. For those rules and heuristics that cannot be illustrated in the context of WebML's data model context, we provide an abstract example in this section and refer to an illustration of their concrete application in the context of WebML's hypertext model in Section 4.

3.1 Transformation Rules

We have designed a couple of rules for transforming concepts of DTDs into Ecore concepts. Table 1 summarizes these rules by differentiating between rules for `XMLElementTypes`, `XMLAttributes` and XOR-Constraints, denoting DTD concepts on the left-hand side and its Ecore counterparts on the right-hand side. Rules are marked using a decimal numbering schema and may contain sub-rules, further specializing the correspondences between DTD concepts and Ecore concepts. Finally, alternative correspondences, depending on the concrete DTD concept are depicted by a distinction of cases.

	Rule	DTD Concept		Ecore Concept
XML Element Type	**R 1**	*XMLElemType (ET)*		*EClass*
		XMLElemType. Name		*EClass.name*
	(1)	XMLEmptyET		no additional elements required
	(2)	XMLAnyET		no additional elements required
	(3)	XMLAtomicET		add EAttribute EAttribute.name="PCDATA", EAttribute.eAttributeType=EString, EAttribute.defaultValue=XMLAtomicET.value
	(4)	XMLCompositeET ElemContent	If XMLSequence with cardinality=1 and nested=false	add EReference for each XMLElementType in XMLSequence EReference.name=XMLElementType.name, EReference.containment=true
			If XMLChoice with cardinality=1 and nested=false	add EReference for each XMLElementType in XMLChoice EReference.name=XMLElementType.name, EReference.containment=true add OCL constraints restricting the alternative EReferences
			If XMLContentParticle with cardinality>1 or nested=true	add helper EClasses for each XMLSequence or XMLChoice serving as containers for nested XMLContentParticles
	(5)	XMLCompositeETMixedContent		add EReference for each XMLElementType EReference.name=XMLElementType.name, EReference.containment=true add EAttribute EAttribute.name="PCDATA", EAttribute.eAttributeType=EString, EAttribute.defaultValue= XMLCompositeETMixedContent.value
	R1.1	*XMLContentParticle.cardinality*		*EReference.multiplicity*
	(1)	? (Zero-or-one)		EReference.lowerBound=0, EReference.upperBound=1
	(2)	* (Zero-or-more)		EReference.lowerBound=0, EReference.upperBound=-1
	(3)	+ (One-or-more)		EReference.lowerBound=1, EReference.upperBound=-1
	(4)	no symbol		EReference.lowerBound=1, EReference.upperBound=1
XML Attribute	**R2**	*XMLAttribute*		*EAttribute*
		XMLAttribute.name		*EAttribute.name*
	(1)	XMLStringAtt, NMTOKEN(S), IDREF(S)		EAttribute.eAttributeType=EString
	(2)	ID		EAttribute.eAttributeType=EString, EAttribute.id=true
	(3)	XMLEnumAtt		add EEnum EEnum.name= XMLEnumAtt.name+"_ENUM" for each XMLEnumLiteral add EEnumLiteral EAttribute.eAttributeType=EEnum

	R2.1	*XMLAttribute.cardinality*		*EAttribute.multiplicity*
	(1)	default_value	Single-valued	EAttribute.lowerBound=1, EAttribute.upperBound=1, EAttribute.defaultValue=XMLAttribute.default_value
			Multi-valued	EAttribute.lowerBound=1, EAttribute.upperBound=-1, EAttribute.defaultValue=XMLAttribute.default_value
	(2)	#FIXED	Single-valued	EAttribute.lowerBound=1, EAttribute.upperBound=1, EAttribute.defaultValue=default_value, EAttribute.unchangeable=true
			Multi-valued	EAttribute.lowerBound=1, EAttribute.upperBound=-1, EAttribute.defaultValue=default_value, EAttribute.unchangeable=true
	(3)	#REQUIRED	Single-valued	EAttribute.lowerBound=1, EAttribute.upperBound=1
			Multi-valued	EAttribute.lowerBound=1, EAttribute.upperBound=-1
	(4)	#IMPLIED	Single-valued	EAttribute.lowerBound=0, EAttribute.upperBound=1
			Multi-valued	EAttribute.lowerBound=0, EAttribute.upperBound=1
XOR	R3	If XMLElemType is part of several XMLCompositeETElemContent		then add OCL constraint to contained EClass specifying that the produced EReferences are exclusive

Table 1: Transformation rules between DTD and Ecore

Following, we illustrate the application of some of these rules using our running example introduced above.

3.1.1 Rule 1 – Element Type

For each `XMLElemType` an `EClass` is created and the name of the `EClass` is set to the element type name. Depending on the particular subclass of `XMLElemType` additional meta-model elements have to be created in the transformation process, which is outlined in Table 1.

Example. In Figure 5 (a), the WebML DTD specifies amongst others element types for ENTITY and RELATIONSHIP, since WebML's content model is based on the ER-model. According to Rule 1, two `EClasses` are generated and named after the element types (cf. Figure 5 (b)). In addition, the ENTITY `XMLCompositeETElemContent` contains the RELATIONSHIP `XMLEmptyET`, and with respect to case (4) in Table 1 an `EReference` is produced, specifying RELATIONSHIP as the contained `EClass`.

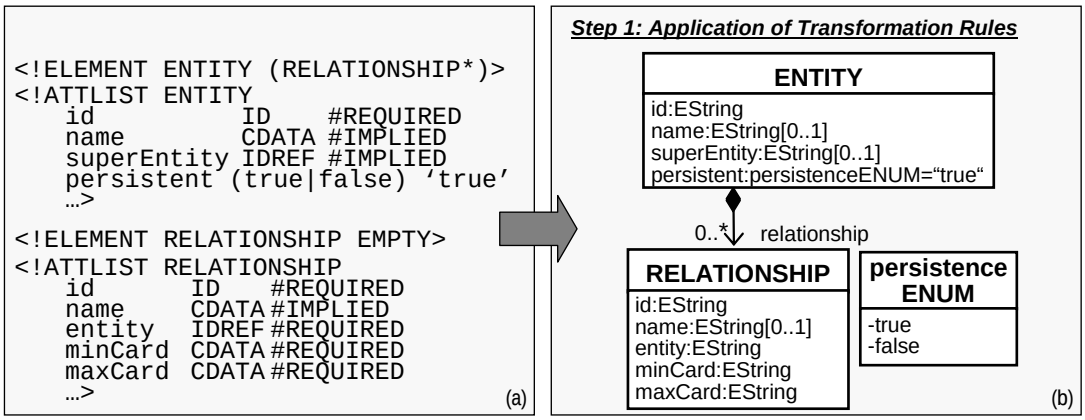

Figure 5: Example of Applying the Transformation Rules (Step 1)

3.1.2 Rule 1.1 – Content Particle Cardinality

Each `XMLContentParticle` may have a certain cardinality, which is represented in meta-models through the `EReference`'s multiplicity in terms of lower and upper bound.

Example. Considering our running example in Figure 5 (b), according to this rule the cardinality of the relationship from ENTITY to RELATIONSHIPs is set to 0...*.

3.1.3 Rule 2 – Attribute

For each `XMLAttribute` an `EAttribute` is created and is attached to the `EClass` representing the `XMLElemType`, which in turn owns the `XMLAttribute`. The name of the `EAttribute` is set to the name of the `XMLAttribute`. The data type of `XMLAttribute` is one of the following: `CDATA`, `NMTOKEN`, `NMTOKENS`, `ID`, `IDREF`, `IDREFS`, and `Enumeration`. Each of these possibilities requires an appropriate transformation as we have outlined in Table 1.

Example. The example in Figure 5 (b), shows that all `XMLAttributes` of type `ID`, `CDATA`, and `IDREF` have been transformed into `EAttributes` of type `EString`, while the `XMLEnumAtt` "persistent" has been transformed to an `EEnum` having two `EEnumLiterals`.

3.1.4 Rule 2.1 – Attribute Cardinality

Attributes in both, DTDs and Ecore have a certain kind of cardinality. In DTDs, the cardinality of an `XMLAttribute` is determined on one hand by the differentiation between single-valued (e.g., `ID`, `CDATA`, `IDREF`, `NMTOKEN`, and `XMLEnumAtt`) and multi-valued (e.g., `IDREFS`, `NMTOKENS`) attributes and on the other hand by the `XMLAttribute` declaration (`#REQUIRED`, `#IMPLIED`, `#FIXED`, and default value). Table 1 illustrates how `XMLAttribute` cardinalities are transformed into `EAttribute` multiplicities.

Example. In Figure 5, all `XMLAttributes` are single-valued meaning that the upper bound is set to one. Only, the `EAttributes` "name" and "superEntity" of `EClass` ENTITY as well as "name" of `EClass` RELATIONSHIP have a multiplicity of 0..1 since their corresponding `XMLAttributes` have been defined `#IMPLIED`. The default value of the `EAttribute` "persistent" is set to one of the `EEnumLiterals`, i.e., true.

3.1.5 Rule 3 – XOR Containment References

An `XMLElemType` can be part of an `XMLContentParticle` of different `XMLCompositeETElemContent`. In the corresponding Ecore-based meta-model an EClass can participate as the contained element in an arbitrary number of containment references. At instance level, the contained object, however, can only be contained by an instance of

only one of the container EClasses at the same time. Hence, this rule adds an OCL constraint to the contained EClass specifying this restriction.

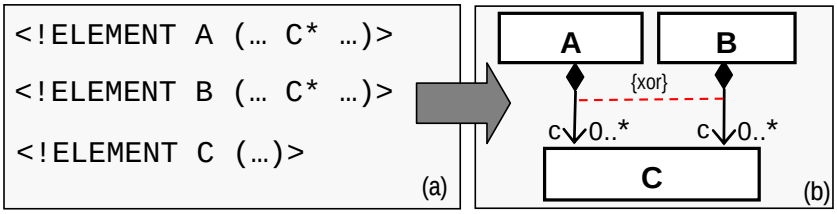

Figure 6: Rule 3 – XOR Containment References

Example. In the abstract example in Figure 6 (a), the XMLElemType C is an XMLContentParticle of XMLElemType A and XMLElemType B. The corresponding meta-model in Figure 6 (b) must ensure that an instance of EClass C is contained either by an instance of EClass A or by an instance of EClass B. Therefore, an XOR constraint is introduced between the relationship c from A to C and the relationship c from B to C. For an example application of Rule 3 in the context of WebML we refer the reader to Listing 3 in Section 4.3.

3.2 Heuristics

As mentioned before, transformation rules are not enough to obtain an Ecore-based meta-model from a specific DTD due to the deficiencies of DTDs described in Section 2.3. Thus, for resolving most of these deficiencies, we propose a set of six heuristics (cf. Table 2) exploiting the assumption that design patterns and naming conventions have been used by DTD designers that have also been found when analyzing the WebML DTDs. This means that the effectiveness of the heuristics, however, is strongly correlated with the quality of the design of the DTDs. For example, the heuristics operate more effectively if naming conventions are used, e.g., for IDREF(S)-typed XMLAttributes, (cf. Heuristic 1 – IDREF(S) Resolution) or a common DTD design pattern [26] for grouping related element types by splitting up a DTD into several external DTDs (cf. Heuristic 3 – Grouping Mechanism).

Heuristic	DTD Concept	Ecore Concept	DTD Deficiency Resolved
H1	If (XMLTokenAtt.kind=IDREF) AND (XMLElemType.name=XMLAttribute.name)	then add EReference to EClass with name= XMLElemType.name, EReference.name=XMLAttribute.name annotate with «Validate IDREF(S)»	Unknown Referenced Element Type(s) (cf. Section 2.3.2)
	else	then annotate EAttribute with «Resolve IDREF(S) manually»	
H2	If XMLEnumAtt has two XMLEnumLiterals and XMLEnumAtt is one of {true, false}, {1, 0}, {on, off}, {yes, no}	then EAttribute.eAttributeType=EBoolean annotate with «Validate Boolean»	Limited Set Of Data Types (cf. Section 2.3.1)
	else if XMLEnumAtt has two XMLEnumLiterals	then annotate EEnum with two EEnumLiterals with «Resolve possible Boolean type manually»	
H3	If DTD imports external DTDs	then add EPackages for each external DTDs to the root EPackage	No Explicit Grouping Mechanism) (cf. Section2.3.7)
H4	If the name of two or more XMLElemTypes in a XMLSequence are equal	then annotate container EClass with «Resolve multiplicity manually»	Awkward Cardinalities (cf. Section 2.3.4)
H5	If XMLElemType has two or more XMLTokenAtts with declaration=#IMPLIED and (kind=IDREF or kind=IDREFS)	then annotate each EAttribute or EReference (cf. Heuristic 1) with «Resolve XOR constraint manually»	Missing Constraint Mechanism (cf. Section 2.3.8)
H6	If XMLElemType is of type XMLAnyET	then annotate EClass with «Resolve XMLAnyET manually»	Missing Inheritance Concept (cf. Section 2.3.6)

Table 2: Heuristics

In any case, these heuristics propose possible correspondences along with annotations guiding the validation and refactoring step in phase 2. In this way, semantically rich language concepts of Ecore such as typed references, data types, and packages can be used to equalize the DTD deficiencies.

Following, the application of some of the heuristics is shown in using our running example in Figure 7.

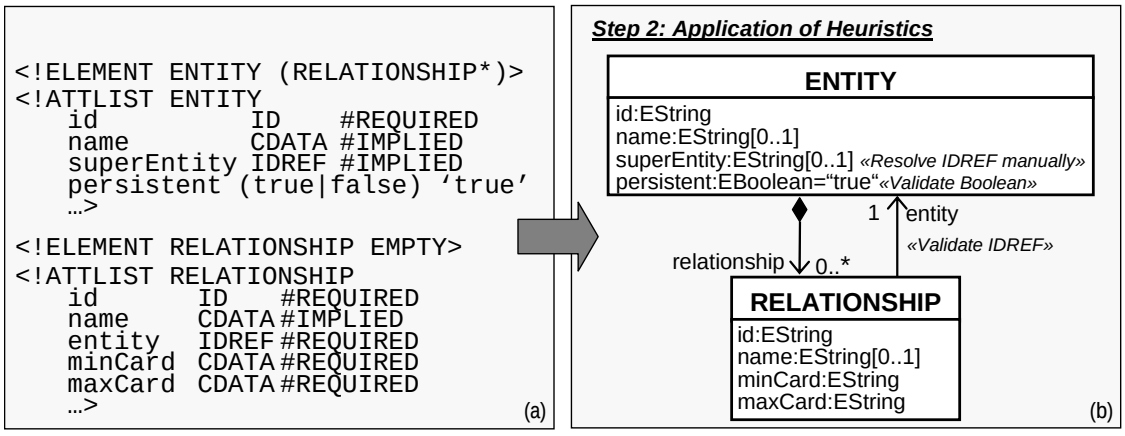

Figure 7: Example of Applying the Heuristics (Step 2)

3.2.1 Heuristic 1 - IDREF(S) Resolution.

The first heuristic is based on the assumption that an IDREF(S)-typed XMLAttribute might be named after the XMLElemType it is intended to reference. Thus, although DTDs lack the possibility to explicitly specify the referenced element types (cf. Section 2.3.1 Unknown referenced element type(s)), it is possible to find them relying on naming conventions of element types and attributes. Heuristic 1 is intended to find such name matches in DTDs. If a match is found, an EReference is generated from the EClass which owns the IDREF(S) attribute to the identified EClass which additionally is also annotated with «Validate IDREF(S)» and the multiplicity of the EReference is set to the multiplicity of the attribute.

It has to be emphasised that since this heuristic is based on name matches, two problematic cases can occur. On one hand, it may falsely resolve references in case IDREF(S) attributes are incidentally named like XMLElemTypes but in fact do not reference them. On the other hand, it may not be able to resolve a reference in case IDREF(S) attributes are not named according to the XMLElemType they shall refer to. Consequently, the user has to validate if the resolution of the IDREF(S) is correct or if another EClass should be referenced.

Example. In our running example the XMLAttribute "entity" in Figure 7 (a) is resolved to an EReference in Figure 7 Figure 7 (b). In case no name match is found the IDREF(S)-typed XMLAttribute is transformed into an EAttribute of type EString annotated with «Resolve IDREF(S) manually», such as the "superEntity" EAttribute in Figure 7 (b).

3.2.2 Heuristic 2 - Boolean Identification

Heuristic 2 is based on the assumption that an XMLEnumAtt consisting of two XMLEnumLiterals might represent an attribute of type Boolean (cf. 2.3.1 Limited set of data types). It recognizes such optimization possibilities and, instead of generating an EEnum, produces an EAttribute of type EBoolean for the following sets of enumeration literals: {true, false}, {1, 0}, {on, off}, and {yes, no}. Furthermore, an annotation «Validate EBoolean» is added to the attribute. In case the two XMLEnumLiterals are not one of the aforementioned sets, the produced EEnum is annotated with «Resolve possible EBoolean type manually» indicating the possibility of replacing the EEnum by the EBoolean data type.

Example. In our running example the XMLEnumAtt "persistent" is identified to be of type Boolean (cf. Figure 7 (a)), thus, in the meta-model the EAttribute persistent is of type EBoolean and no EEnum is generated (cf. Figure 7 (b)).

3.2.3 Heuristic 3 - Grouping Mechanism

Heuristic 3 interprets a parameter entity declaration that points to a further DTD file as a group of related markup declarations (cf. 2.3.7 No explicit grouping mechanism) that can be referenced from within a so called root DTD. A root DTD is equivalent to a root package in a meta-model and external DTDs are equivalent to sub-packages of the root package. The MMG generates a package for each external DTD and one root package for the root DTD.

Example. In the abstract example of Figure 8 (a) a DTD named Root is shown which defines two `ParameterEntityDec` PartA and PartB, both referencing to an external DTD, A.dtd and B.dtd. The DTD Root is transformed into the `EPackage` Root which contains an. `EPackage` A and B for the external DTDs (cf.in Figure 8 (b)). An example application of Heuristic 3 in the context of WebML may be found in Listing 1 in Section 4.1.

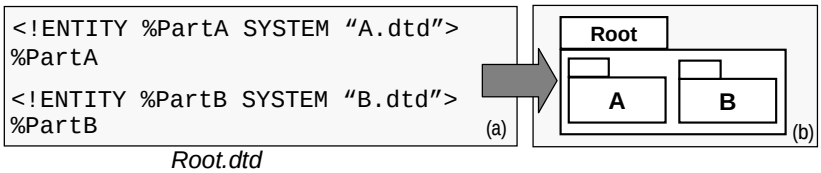

Figure 8: Heuristic 3 – Grouping Mechanism

3.2.4 Heuristic 4 – Cardinalities Identification

This heuristic is based on the assumption that an `XMLSequence` containing element types of the same name has to be interpreted as "one piece of information" (cf. 2.3.4 Awkward cardinalities). This means that instead of producing an `EReference` for each single element type, only one `EReference` should be generated and the cardinality has to be inferred from all element type's cardinalities within the sequence. Heuristic 4 adds an annotation «`Resolve multiplicity manually`» to the `EClass` containing the `EReferences` to indicate that a specific sequence of element types transformed into a set of `EReferences` possibly has to be re-modelled into one `EReference` and the appropriate multiplicity has to be assigned.

Example. In Figure 9 (a) an example for the *awkward cardinality* deficiency of DTDs can be found. The problem is that the first and the second `XMLContentParticle` B of `XMLElemType` A instead of two separate sets of elements together maybe represent one set of elements with a cardinality restriction of 2 or many. Consequently, in the corresponding meta-model (cf. Figure 9 (b)) this ambiguity is marked by an annotation «`Resolve multiplicity manually`». This annotation indicates that in the validation and refactoring step the user has to decide, if `b1` and `b2`

are two separate sets or if `b1` and `b2` should be merged into one set with a cardinality of 2 or many. This heuristic has been applied for example in the context of WebML's hypertext model (cf. Listing 2 in Section 4.3).

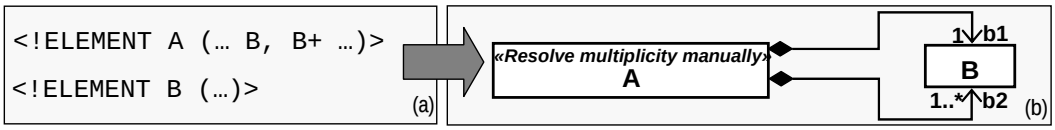

Figure 9: Heuristic 4 – Cardinalities Identification

3.2.5 Heuristic 5 – XOR Constraints Identification

If two `XMLAttributes` within an attribute list declaration of an element type are both of type `IDREF(S)` and have been declared as `#IMPLIED` they might represent two excluding `EReferences` in the meta-model and hence, require an XOR constraint. Heuristic 5 annotates these `EReferences` (or `EAttributes` if the reference could not be resolved automatically by Heuristic 1) with «`Resolve XOR constraint manually`» to indicate to the user the possible need of an XOR constraint.

Example. In Figure 10 (a) an excerpt of a DTD is shown which defines an `XMLElemType` A containing two attributes, namely `b` and `c`. Both attributes are `IDREF`-typed and defined as `#IMPLIED` which means both attributes are optional. In this example we assume that Heuristic 1 can be used to resolve the `IDREF` attributes as `EReferences`. In some cases, however, two optional `IDREF`-typed attributes are mutually exclusive. Therefore, `EReference b` and `c` are annotated with the annotation «`Resolve XOR constraint manually`» indicating that the user must decide if actually an XOR constraint exists between these two relationships or not. For an example application of Heuristic 5 in the context of WebML see Listing 4 in Section 4.3.

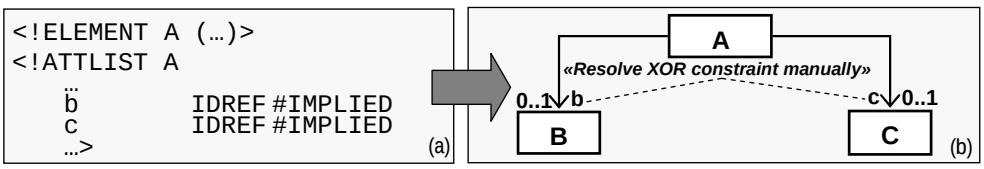

Figure 10: Heuristic 5 - XOR Constraints Identification

3.2.6 Heuristic 6 – Inheritance Identification

This heuristic is based on the assumption that the element type `XMLAnyET` sometimes is used as a container element for other concepts in the described language, i.e., concepts that have similar properties and in this way may represent

sub types of the XMLAnyET element. Hence, Heuristic 6 annotates EClasses resulting from an XMLAnyET with «Resolve XMLAnyET manually» in order to propose a possible candidate for inheritance.

Example. In Figure 11 (a), the abstract example shows a DTD, in which XMLElemType A contains an XMLElemParticle B. XMLElemType B, in turn, is defined as XMLAnyET stating that element B can be any XMLElemType or any text. In Figure 11 (b) an example XML document is shown, in that element A contains element B. which in turn contains elements of type of C and D. In Figure 11 (c) the corresponding meta-model from the DTD definition contains amongst others an EClass B which is annotated with «Resolve XMLAnyET manually». This annotation indicates, that the user must decide, how to express the possibility that an instance of EClass B can contain instances of EClass C and D. In general, this possibility can be expressed as an inheritance relationship, defining EClass B as the super-class of EClass C and D. In the context of WebML's hypertext model, an example application of Heuristic 6 is provided in Listing 5 in Section 4.4.

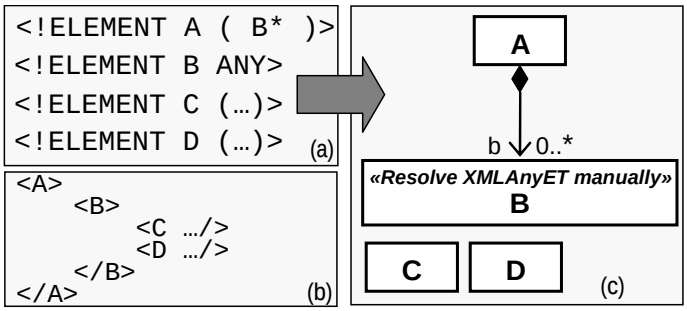

Figure 11: Heuristic 6 – Inheritance Identification

3.3 Manual Validation and Refactoring of the Generated Meta-Model

The second manual phase requires user interaction for validating and refactoring the automatically produced meta-model on the basis of domain-knowledge and specifically on the basis of the suggestions annotated by the applied heuristics during the last step towards a MOF-based WebML meta-model.

In the example showed in Figure 12 (a), two annotations with respect to Heuristic 1 were introduced indicating that the user should validate, on one hand, the directed reference introduced between ENTITY and RELATIONSHIP («Validate IDREF») and, on the other hand, the introduced attribute "superEntity" which was marked with «Resolve IDREF(S) manually». While the directed reference appears to be a correct transformation, the introduced attribute "superEntity" is, in fact, a reference to the ENTITY in its role as super entity used to model inheritance in WebML

and therefore shall be manually refactored accordingly by replacing the `EAttribute` with an `EReference` from ENTITY to itself with the role name "superEntity" (cf. Figure 12 (b)).

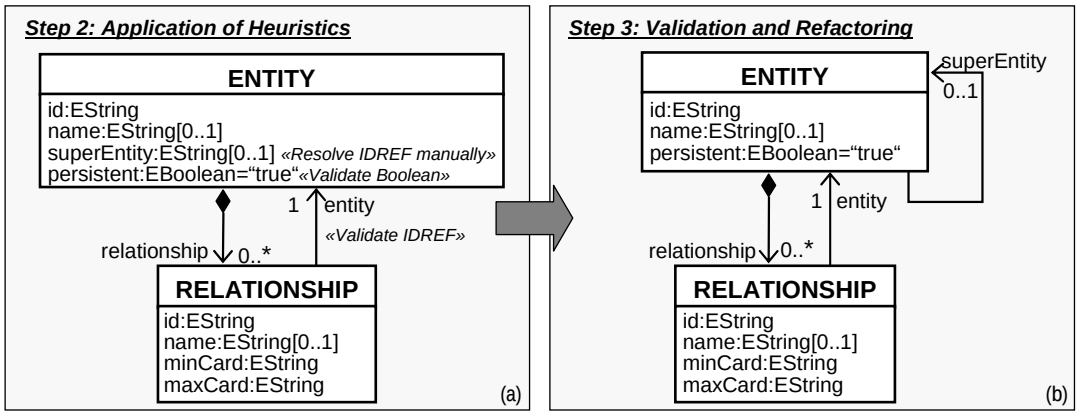

Figure 12: Example of Applying the Manual Refactoring (Step 3)

Furthermore, in the given example the `EAttribute` "persistent" according to Heuristic 2 was annotated with `«Resolve possible EBoolean type manually»` to indicate the need of manual validation; in this case no manual refactoring is necessary.

3.4 Implementation Architecture of the MetaModelGenerator

As already mentioned, the core component of our transformation framework is represented by the MetaModelGenerator. Figure 13shows the details of its implementation architecture.

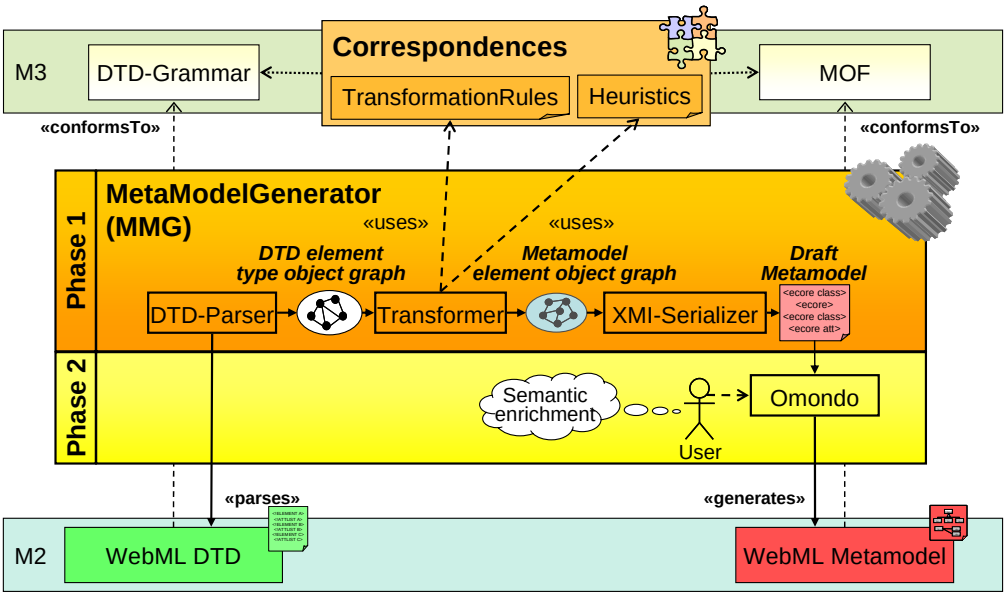

Figure 13: Architecture and Mode of Operation of the MMG

The MMG is based on the Eclipse Modelling Framework (EMF)[4] and on an open source DTD parser[5]. In a first step a specific DTD, in our case, the WebML DTD, serves as input to the DTD parser, which builds a Java object graph of DTD markup declarations in memory. Then each element type in the object graph is visited and transformed according to the transformation rules and heuristics described in Section 3.1 and Section 3.2 respectively. Each transformation rule is implemented as a separate Java method which takes DTD element type objects as input and generates the objects for the corresponding Ecore elements. If a transformation rule uses a heuristic, then the corresponding method calls a helper method which implements that heuristic. As soon as the complete element object graph of the Ecore-based meta-model has been generated, the default XMI Serializer of EMF is activated in order to serialize the meta-model as an XMI file. This XMI file can be loaded into OMONDO[6], a graphical editor for Ecore-based meta-models available as an Eclipse plug-in. In a last step, the annotations created to indicate that a heuristic has been applied should be validated by the user and the meta-model should be refactored accordingly.

4. THE RESULTING WebML META-MODEL

The Ecore-based meta-model for WebML resulting from the application of our DTD-to-MOF transformation framework to the WebML DTD is subject of this section. In particular, it explains the ratio behind some of the manual refactoring decisions taken. Additionally, we will illustrate the WebML meta-model by relating it to a concrete modelling example, i.e., a demo WebML model that is shipped with WebRatio. This way, we want to briefly explain the language and notation to those unfamiliar with WebML and at the same time indicate the relationship between the model and our meta-model specification as well as informally show that our meta-model indeed can be used to produce the same models as the original WebML DTD. A more profound evaluation of the WebML meta-model is provided in Section 5.

Please not that the following figures depicting some of the meta-models packages have been simplified for readability purposes. For the same reason XOR constraints are illustrated in UML syntax.[7]. For an in-depth description of each modelling concept we refer the reader to [6]. Before describing some of the packages in more

[4] http://www.eclipse.org/emf

[5] http://www.wutka.com/dtdparser.html

[6] http://www.omondo.de/

[7] The complete metamodel is available at http://big.tuwien.ac.at/projects/webml/.

detail and explaining some of our refactoring actions (cf. Section 4.2 - 4.6), we give an overview on the package structure.

4.1 Overview

The WebML designers have used parameter entities as a mechanism to structure the WebML's language specification. Thus, the WebML language definition consists of several DTDs with `WebML.dtd` being the root DTD that imports the others, which is expressed in Listing 1.

```
<!-- WebML.dtd -->
<!ENTITY % StructureDTD SYSTEM "Structure.dtd">
%StructureDTD;
<!ENTITY % NavigationDTD SYSTEM "Navigation.dtd">
%NavigationDTD;
<!ENTITY % PresentationDTD SYSTEM "Presentation.dtd">
%PresentationDTD;…
…
```

Listing 1

While Structure.dtd and Navigation.dtd define the main language concepts that have been introduced in [6], other rather tool-related DTDs have been introduced in the WebRatio tool. In contrast to our prior work [24] where our main focus has been WebML's main language concepts, we now consider all of WebML's DTDs. This allows for migrating existing WebML models that have been generated using WebRatio into models conforming to our meta-model specification without loosing any information (cf. Section 5.1) and thus profiting from further MDE techniques such as model transformation.

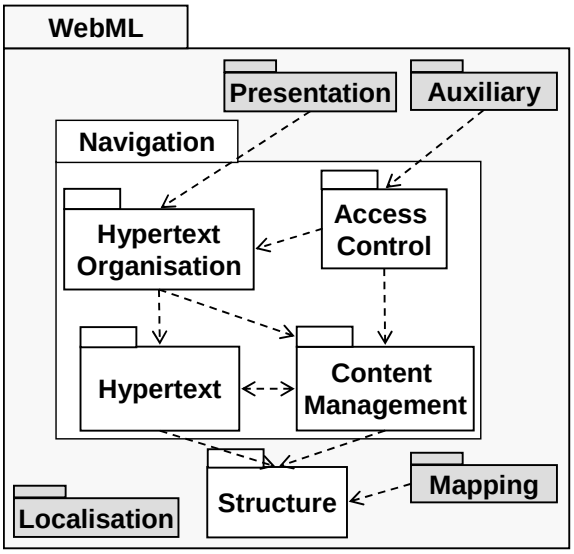

Figure 14: WebML Packages View

In Figure 14, we present a high-level view of the resulting WebML meta-model, i.e., its packages and their interrelationships. This structural organization of the WebML concepts has been automatically generated on the basis of Heuristic 3. While Structure.dtd corresponds to the Structure package in Figure 14 and contains concepts for modelling the content level of a web application, the Navigation package contains modelling concepts for the hypertext level and has been automatically generated from the Navigation.dtd. We have manually reorganized the rather large Navigation package into four packages, namely HypertextOrganisation, Hypertext, ContentManagement, and AccessControl

The additional gray-shaded packages have been generated from the tool-related DTDs: First, the Mapping package imports the RDBMSMapping package (not shown for reasons of briefness) which provides concepts for specifying the mapping of WebML's content model to a relational database, second, the Localisation package offers modelling concepts for multilingual web applications, third the Presentation package defines concepts for modelling the look & feel of web applications, and forth, the graphical illustration and positioning of WebML's notational elements within the WebRatio modelling editor is determined by concepts defined in the Auxiliary package.

In the following we omit a detailed description of the tool-related packages but specifically focus on the Structure and Navigation packages and report on some of the applied refactoring actions. In order to explain the semantics of the meta-model, we will provide the corresponding part of a concrete WebML modelling example for each package. The ACME (A Company Manufacturing Everything) example model is a demo WebML model shipped with WebRatio and represents an E-Store where users can browse, search, and buy products and special combinations of products, respectively. These products and combinations can be edited, extended, and deleted by administrators of the web application.

4.2 Structure Package

The Structure package (cf. Figure 15 (a)) contains modelling concepts that allow modelling the content layer of a web application, which regards the specification of the data used by the application. Since, as already mentioned, WebML's content model is based on the ER-model, it basically supports ER modelling concepts: An Entity represents a description of common features, i.e., Attributes, of a set of objects. Note, that unlike UML class diagrams, ER diagrams model structural features, only.

With respect to manual refactoring actions, we added an XOR constraint to the meta-model in order to specify that Attributes can have either a type, e.g., String, Integer, Float, Date, Time, and Boolean, or a userType, i.e., an enumeration type represented by Domain and DomainValue, respectively. Entities that are associated with each other are connected by Relationships whereby we manually changed the type of the attributes minCard and maxCard of Class Relationship from `EString` to `EInt`.

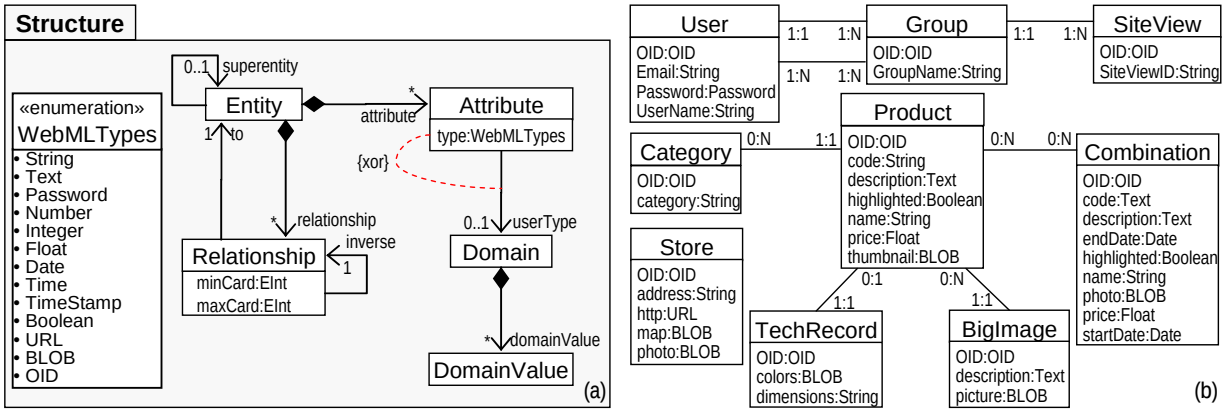

Figure 15: Content Package

In Figure 15 (b) we show the WebML content model of the ACME web application. Products belong to one Category and can be described by a TechnicalRecord and several BigImages. Furthermore, Products can be offered within several Combinations with other Products. In addition, the web application provides information of available Stores. The User, Group, and SiteView entities are used for user management (e.g., normal users and administrators) and access control purposes.

4.3 HypertextOrganisation Package

The HypertextOrganisation package includes concepts for structuring the hypertext, i.e., it offers concepts for organizing modeling concept from the Hypertext package (cf. Section 4.4). More specifically, the Page concept is used to organize and structure information from the content level, e.g., ContentUnits from the Hypertext package.. SiteViews and Areas in turn group Pages as well as operations on data from the content level, e.g., OperationUnits from the ContentManagement package (cf. Section 4.5). More specifically, SiteViews represent groups of areas and/or pages devoted to fulfilling the requirements of one or more user groups, while Areas are containers of Pages or nested sub-areas related to a homogeneous subject and are used to hierarchically organize the web application. These concepts are encapsulated within the HypertextOrganisation package (cf. Figure 16 (a)).

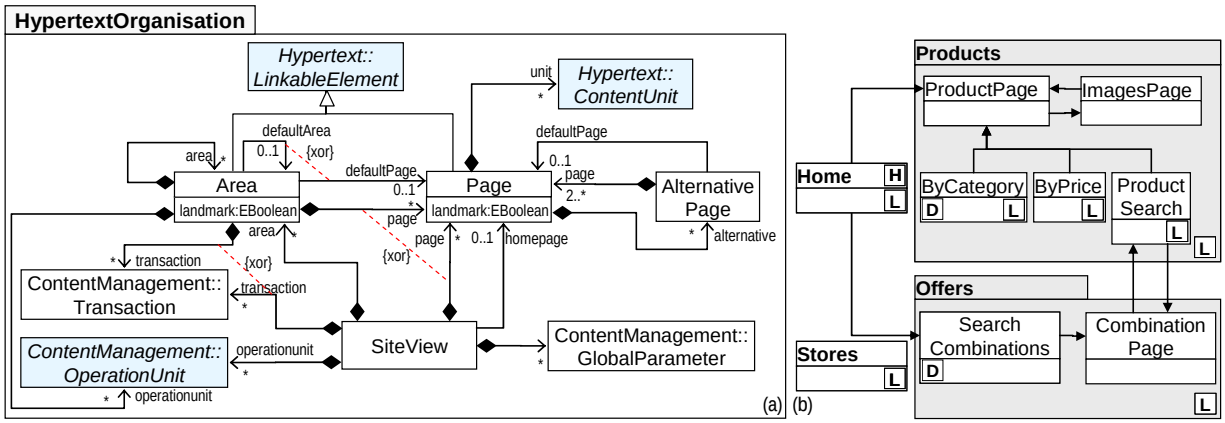

Figure 16: HypertextOrganisation Package

With respect to refactoring actions, we were able to identify an example of the awkward cardinalities problem (cf. Section 2.3.4) based on `EAnnotations` created by Heuristic 4. The definition of the AlternativePage concept requires the AlternativePage to have at least two sub-pages, which is expressed in the WebML DTD as is depicted in Listing 2.

```
<!ELEMENT AlternativePage (Page, Page+)>
```

Listing 2

Yet, this definition found in the DTD might be interpreted differently in a meta-model. One possible interpretation is that the first `XMLContentParticle` represents a special page, e.g., a default page. Nevertheless, the correct interpretation in the context of WebML [6] is that the first and the second `XMLContentParticle` together represent one set of alternative Pages, i.e., one containment `EReference`, but with special restrictions on their cardinalities, i.e., 2..*. In meta-models, this constraint can be expressed unambiguously, which is shown by the AlternativePage.page reference in Figure 16 (a).

While Rule 3 already detected, that Pages and Transactions can be contained by either a SiteView or an Area (cf. Listing 3), Heuristic 5 identified further possible candidates for XOR constraints in the HypertextOrganisation Package.

```
<!ELEMENT SiteView (… Page* …)>
<!ELEMENT Area (… Page* …)>
```

Listing 3

An Area can have either a defaultArea or a defaultPage, but not both at the same time.

```
<!ELEMENT Area (…)>
<!ATTLIST Area
        defaultPage IDREF #IMPLIED
        defaultArea IDREF #IMPLIED
        …>
```

Listing 4

In the DTD the attribute list declaration of Area is not able to ensure this constraint at the instance level. Therefore, we introduce an XOR constraint to specify that either the defaultPage `EReference` or the defaultArea `EReferences` occurs at the instance layer:

```
context Area inv: defaultArea.oclIsUndefined()<>defaultPage.oclIsUndefined()
```

In the ACME WebML model separate SiteViews for users and administrators have been designed. The first one, i.e., the Web SiteView is depicted in Figure 16 (b). The Products Area groups all Pages presenting some information about products and the Home Page (H) acts as the entry point of the SiteView. The default page of an Area (D) such as the ByCategory Page of the Products Area is the one displayed when the Area is entered. Furthermore, Pages and Areas declared as landmark (L) are reachable from all other Pages or Areas within their enclosing SiteView or enclosing Area. In this respect, the landmark represents a compact way of specifying a set of links to a Page or Area, respectively.

4.4 Hypertext Package

The hypertext layer represents a view on the content layer of a web application, only, and thus, the Hypertext Package reuses concepts from the Structure Package, namely, Entity, Relationship, and Attribute.

The Hypertext Package (cf. Figure 17 (a)) summarizes ContentUnits used, for example, to display information from the content layer, which may be connected by Links in a certain way.

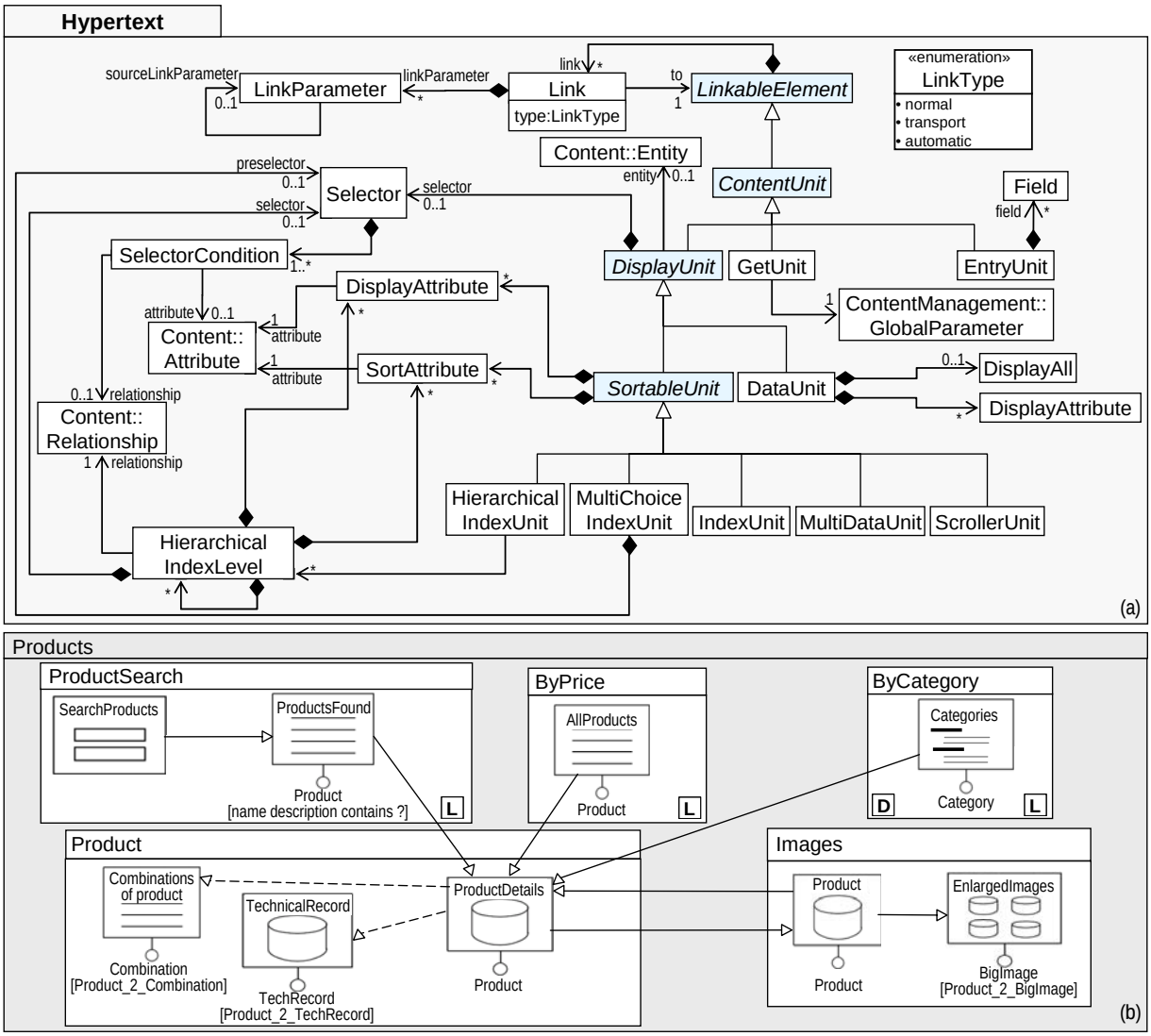

Figure 17: Hypertext Package

Based on Heuristic 6, we identified a candidate EClass for introducing inheritance. In WebML, Pages contain different kinds of ContentUnits:

```
<!ELEMENT Page (ContentUnits,…)>
<!ELEMENT ContentUnits ANY>
```

Listing 5

The XMLAnyET, however, does not restrict which element types are allowed, i.e., only ContentUnits, and which are not allowed at the instance layer. Again, these constraints have to be ensured by the WebRatio tool. In the meta-model, we were able to resolve this problem by manually introducing a generalisation hierarchy, which includes the additional abstract classes ContentUnit, DisplayUnit, and SortableUnit. In this way, we can ensure that Pages contain

subclasses of ContentUnit, only, and handle the large amount of different kinds of ContentUnits more easily by reducing redundant structural feature definitions.

The abstract class LinkableElement has been manually introduced in order to cope with other language concepts that can also be connected by Links. This was necessary, since the IDREF-typed attribute "to" of the Link element type declaration does not restrict the referenced elements to those that the designer originally intended to reference:

```
<!ELEMENT Link (…)>
<!ATTLIST Link
        to IDREF #REQUIRED
        type (normal|automatic|transport) 'normal'
        …>
```

Listing 6

Furthermore, besides ContentUnits, there are other LinkableElements in the HypertextOrganisation package (cf. Section 4.3), namely Page and Area, as well as in the ContentManagement package (cf. Section 4.5), namely OperationUnits. More specifically, three disjoint Link types are available in WebML, i.e., normal Link, automatic Link, and transport Link (cf. Figure 17 (a)).Besides these Link concept, there are also the OKLink and KOLink modelling concepts from the ContentManagement package, which are specifically used to define Links from OperationUnits to other LinkableElements. Consequently, there are multiple sourceElement–link–targetElement tuples of which some are allowed in WebML, only (cf. Table 3).

Table 3: Linking Possibilities in WebML

From\To	Content Unit	Operation Unit	Page	Area
Content Unit	*normal* *automatic* *transport*	*normal* *transport*	✗	✗
Operation Unit	*transport* *OK* *KO*	*transport* *OK* *KO*	*transport* *OK* *KO*	*transport* *OK* *KO*
Page	✗	*normal* *transport*	*normal* *transport*	*normal*

These sourceElement–link–targetElement tuples, however, are not restricted by WebML DTD but are implicitly ensured within the WebRatio tool. Aiming at a precise definition of sourceElement–link–targetElement tuples in the WebML meta-model, the introduction of the LinkableElement concept, which acts as a super class for all possible sources and targets, is not enough. Consequently, we have additionally introduced appropriate OCL constraints

restricting the sourceElement–link–targetElement tuples to those that are allowed in WebRatio (cf. Table 3). For example, a Page cannot link ContentUnits, which can be specified with the following OCL constraints:

```
context Page inv: self.link->forAll(l | not l.to.oclIsTypeOf(ContentUnit))
```

In Figure 17 (b), we show a refined view of the Web SitView of the ACME example depicting in detail the Products Area: While the ByPrice Page uses the IndexUnit AllProducts for listing links to all products in ascending order according to their price, the ByCategory Page displays a linked list of all products organised according to their categories using a HierarchicalIndexUnit. The ProductSearch Page provides an EntryUnit SearchProducts with one Field where the user can enter a keyword and displays the found products as an IndexUnit. A single SelectorCondition of the IndexUnit's Selector defines that only those products where the keyword is part of the name or the description of the product are to be retrieved. A specific product is shown by the Product Page, which is linked by all other Pages via Links of type normal. The ProductDetails DataUnit represents one product entity from the content model and displays the specified DisplayAttributes only. Furthermore, an additional DataUnit retrieves the technical record of the product and an additional IndexUnit displays a linked list of combinations where the specific products is part of. The information about what technical records and what combinations to retrieve is transported via LinkParameters of Links of type transport (dashed arrows), which are neither navigable by nor visible to users. Finally, the Images Page again shows some details of a product using a DataUnit and a set of images of the product using a MultiDataUnit.

4.5 ContentManagement Package

The ContentManagement package contains modelling concepts that allow the modification of data from the content layer. Similar to the generalisation hierarchy in the Hypertext package, we also introduce additional abstract classes in the ContentMangagement package based on EAnnotations created by Heuristic 6 (cf. Figure 18 (a)), i.e., OperationUnit, ContentManagement-Unit, EntityManagementUnit, and RelationshipManagementUnit. In particular, the introduction of the OperationUnit EClass ensures that Areas and Siteviews from the HypertextOrganisation package contain subclasses of OperationUnit, only. Since the specific ContentManagementUnits are able to create, modify, and delete Entities as well as establish or delete Relationships between Entities from the content layer, the ContentManagement package reuses concepts from the Structure Package, namely Entity and Relationship.

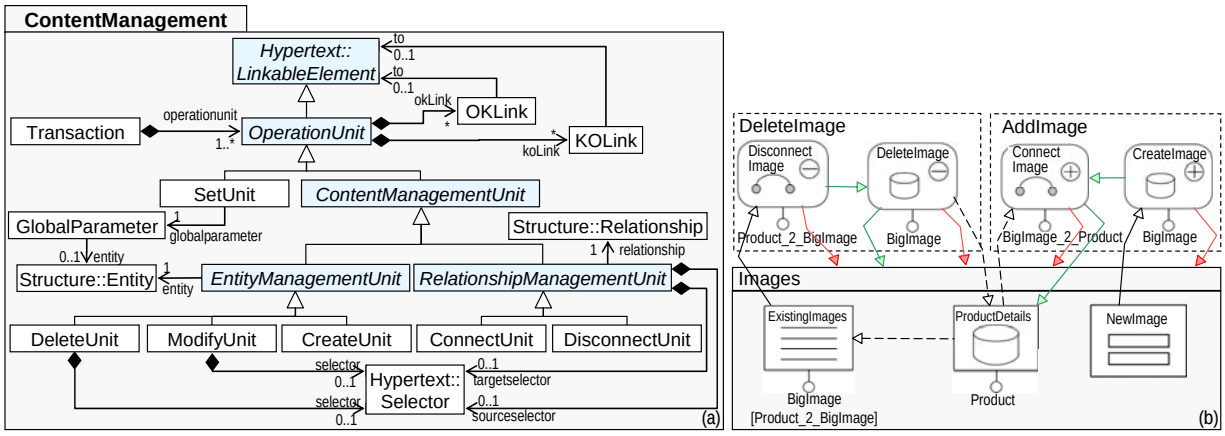

Figure 18: ContentManagement Package

Furthermore we have identified roles, i.e., redundant definitions of the same concept, namely Selector. As an example, a RelationshipManagementUnit may have two Selectors, with one being used in the role of a Sourceselector and the other one being used as a Targetselector. In the WebML DTD, this is expressed as follows:

```
<!ELEMENT DisconnectUnit (Sourceselector?, Targetselector?,…)>
<!ELEMENT Selector(SelectorCondition+)>
<!ELEMENT Sourceselector (SelectorCondition+)>
<!ELEMENT Targetselector (SelectorCondition+)>
```

Listing 7

Since the Targetselector and Sourceselector element type declarations are identical to the Selector element type declaration, one can conclude that they represent the same concept as the Selector but are used in a special context. In contrast, in meta-models this context information can be incorporated by the EReferences' names. Therefore, the WebML meta-model only contains the Selector EClass, which is referenced by the RelationshipManagementUnit as a sourceselector and targetselector, respectively (cf. Figure 18 (a)). A similar example can be found in the Hypertext package, where a Selector can act as preselector for MultiChoiceIndexUnits (cf. Figure 17 (a)).

In the Administrator SiteView of the ACME web application, administrators can add, edit, and delete products, combinations, and stores. The Images Page in Figure 18 (b) is part of the ProductEditing Area and allows adding and deleting images of a specific product. The Page displays product details in a DataUnit, an IndexUnit of existing images for the product, and an EntryUnit allowing the upload of further images. Selecting an image from the IndexUnit activates the Transaction DeleteImage, which has similar semantics as typical database transactions. First a DisconnectUnit disconnects the image and the products by deleting the specific instance of the relationship, then the green OKLink is followed, the image is deleted using a DeleteUnit, and via a second OKLink the Images Page is

reached again. In case of an error, the red KOLinks are followed to the Images Page. The AddImage Transaction is activated when the user uploads a new image and first creates a new image (CreateUnit) and connects it to the specific product (ConnectUnit).

4.6 AccessControl Package

In Figure 19 (a), the AccessControl Package groups concepts for controlling the access to SiteViews, namely LoginUnit, LogoutUnit, and ChangeGroupUnit.

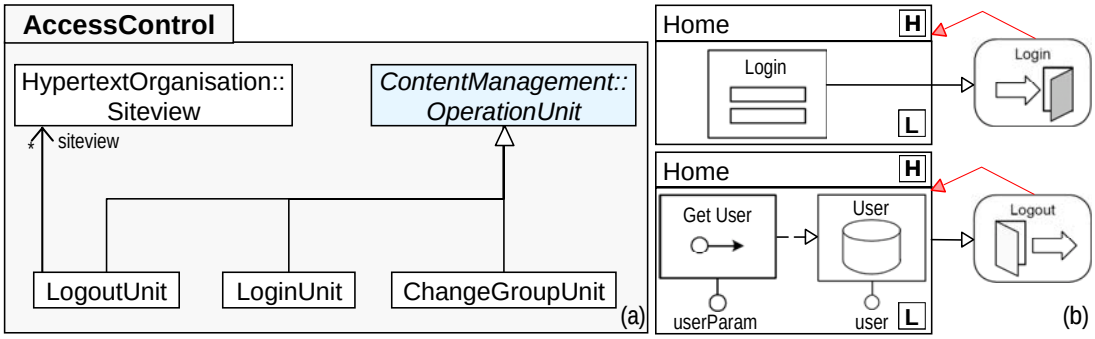

Figure 19: AccessControl Package

The example shows the SiteView, i.e., the Home Page of normal users (cf. Figure 19 (b)). Administrators have to log in via the EntryUnit Login. The LoginUnit verifies username and password and switches to the user's default SiteView, i.e., the Administrator Siteview. In the Home Page of the Administrator SiteView user information is displayed with the User DataUnit. The respective user is obtained from the session with a GetUnit (cf. Section 4.4). A user logs out via LogoutUnit, which forwards the user back to the Web SiteView for normal users.

5. EVALUATION OF THE GENERATED WebML META-MODEL

In the following we provide a discussion on the evaluation of the generated WebML meta-model, which gives an indication on the applicability of our semi-automatic transformation approach. This evaluation is conducted, first, with respect to the meta-model's completeness compared to the language concepts defined in the original WebML DTD (cf. Section 5.1) and, second, on the basis of certain quality metrics (cf. Section 5.2).

5.1 Completeness Criteria

The completeness criteria is fulfilled at the meta-level M2 if the generated WebML meta-model contains all concepts defined within WebML's DTD and the WebRatio tool. At the model level M1 this means that the models as

instances of both the WebML meta-model and as XML conforming to the WebML's DTD can be exchanged in a lossless way.

Although WebML does provide a formal definition of the semantics of its concepts [27], [28], a formal verification of the completeness criteria is not an option. This is due to the fact that currently within EMF the definition of semantics is not provided without executing the model itself which is unfortunately not applicable in the scope of the paper. Nevertheless, a first prerequisite for completeness at the M2 level is provided by the fact that each WebML concept present in the DTD is dealt with by at least one transformation rule of our framework, which in turn assures for each WebML concept there exists at least one counterpart in the meta-model.

In addition, completeness at the M2 level can be further underpinned by considering the M1 level.

Taking a first step towards evaluating completeness at the M1 level, we followed an "example-based" strategy, i.e., we remodelled an existing WebML reference example on the basis of our generated meta-model. For this, we have generated a tree-based modelling editor from our meta-model using EMF and were able to completely re-model WebRatio's demo example, the ACME E-Store, which we additionally extended by those WebML language concepts not covered in the original example[8].

In a second step our tree-based modelling editor was enhanced with an import/export facility to demonstrate whether models could be exchanged with the WebRatio tool in a lossless way[9]. With that facility we were able to import the extended ACME example from WebRatio into our modelling editor and subsequently to export the model back into a WebRatio XML document. A comparison of the original XML document defined by WebRatio and the exported XML document from our modelling editor with the XML Differencing facility of StylusStudio[10] demonstrated that both were equivalent.

Admittedly, it has to be noted that this is only a first step towards justifying the semantic equivalence of our WebML meta-model and the original language specification not least since the evaluation shall comprise a larger set of more complex examples.

---

[8] The modelling editor and the ACME example are available at http://big.tuwien.ac.at/projects/webml/

[9] Note that currently the import/export facility supports the WebML content model only.

[10] http://www.stylusstudio.com/

5.2 Quality Metrics

The WebML meta-model and its quality characteristics in terms of expressiveness, accuracy, and understandability have evolved considerably during our three-step transformation process. In order to illustrate this evolution, we have applied a set of metrics inspired by [27] to the meta-model versions resulting from each step of the transformation process, i.e. application of transformation rules, employment of heuristics, and manual validation and refactoring. The results of applying these metrics are summarized in Table 4.

Metrics		Phase 1 Automatic Transformation		Phase 2 Manual Refactoring
		Step 1	Step 2	Step 3
All Modelling Concepts (excl. EAnnotations)		707	707	573
EPackage		1	7	11
nested EPackage depth (Heuristic 3)		1	3	3
EClass		96	96	102
abstract		0	0	11
inheriting from multiple EClasses		0	0	5
maximum inheritance depth		0	0	4
average inheritance depth		0	0	0,86275
annotated with «Resolve XMLAnyET manually» (Heuristic 6)		-	3	-
annotated with «Resolve Multiplicity manually» (Heuristic 4)		-	1	-
EClasses/EPackage	MIN	96	1	1
	MAX	96	53	27
	AVG	96	13	9
EAttribute		338	338	260
EString		278	278	207
annotated with «Resolve IDREF manually» (Heuristic 1)		-	51	-
annotated with «Resolve IDREFS manually» (Heuristic 1)		-	5	-
annotated with «Resolve XOR manually » (Heuristic 5)		-	17	-
EBoolean (Heuristic 2)		0	46	40
EEnum		50	14	11
EInteger		0	0	2
EReference	annotated with «Validate IDREF» (Heuristic 1)	-	39	53
	annotated with «Validate IDREFS» (Heuristic 1)	-	0	
	annotated with «Resolve XOR manually » (Heuristic 5)	-	5	-
Containment EReference		234	234	158
EEnum		12	12	8
annotated with «Resolve possible Boolean type manually » (Heuristic 2)		-	6	-
OCL constraints	XOR constraint	4	4	10
	other constraints	-	-	20
Identified Roles		-	-	3

Table 4: Meta-model Metrics

Interpreting these metrics, one can observe that the introduction of a package structure, inheritance, and roles as well as the resolution of the awkward cardinalities deficiency has had a great impact on the understandibility and readability of the meta-model. In particular, the introduction of inheritance through 11 abstract `EClasses` has helped to decrease complexity by reducing redundant `EAttributes` and `EReferences`. The identification of 3 roles has diminished the number of `EClasses`, while the resolution of awkward cardinalities has diminished the number of `EReferences`. All in all, the number of 707 modeling elements in the WebML DTD could be reduced to 537

modelling concepts in Ecore. The application of grouping mechanisms according to Heuristic 3 and further manual refactorings also had a positive effect on the language's readability in terms of an introduced package structure and a reduced ratio of `EClasses` per `EPackage`. After manual refactoring, the maximum number of `EClasses` per `EPackage` decreased from 53 to 27.

Concerning accuracy, the resolution of `IDREF(S)`-typed `XMLAttributes` into `EReferences`, the introduction of `EBoolean`-typed `EAttributes` instead of enumerations and the definition of constraints have considerably contributed to a more precise language. E.g., Heuristic 1 already correctly resolved 39 `IDREF`-typed `XMLAttributes` into `EReferences` making the relationship between `EClasses` explicit. Further 56 `EStrings` had to be resolved manually into `EReferences`. From 50 enumeration-typed `XMLAttributes`, 46 could be resolved correctly by Heuristic 2 as `EBooleans`. Moreover, further 4 `EEnums` could be eliminated, the respective `EAttributes` could be refactored to `EBooleans` and 26 additional constraints could be defined, thus, achieving a more precise WebML meta-model.

6. RELATED WORK

With respect to our approach of a semi-automatic generation of a MOF-based meta-model for WebML, we basically distinguish two areas of related work. First, approaches aiming at the design of meta-models for Web modelling languages, and second, approaches dealing with the transformation of DTDs to MOF-based meta-models.

Meta-models for Web Modelling Languages. To the best of or knowledge, there is currently just one closely related approach focusing on the definition of a UML 2.0 Profile for WebML [30]. The motivation of this approach is to facilitate the interoperability of the WebRatio tool with existing UML modelling tools. More specifically, WebML has been manually remodelled using MOF and in a second step a UML profile has been inferred from it. Our work differs from this approach in three ways. First, we strive for a domain-specific language, for which today tool support can easily be provided, e. g., based on EMF. Second, our WebML meta-model has been semi-automatically generated from WebML's DTD-based language specification. Finally, our approach provides the prerequisite of migrating existing WebML models to MOF, while the WebML profile requires developers to re-model existing WebML models from scratch.

Besides this closely related work in the context of WebML, we are aware of three other Web modelling approaches which are currently defined on top of a meta-model, namely W2000 [9], UWE [7], and Muller et al. [14]. W2000 [9], a successor of HDM [31], originally has been defined as an extension to UML. In [13], the provision of a meta-model based on MOF 1.4 [32] has been motivated and adopted as a necessity for providing tool support for an

evolving language definition. The meta-model of UWE [7] has been designed as a conservative extension to the UML 1.4 meta-model [33], and thus is implicitly based on MOF 1.4. It is intended as a step towards a future common meta-model for the Web application domain, which envisions supporting the concepts of all existing Web modelling methods. Similar to W2000, a language definition already exists as UML Profile. Muller et al. [14] present a model-driven Web application design and development approach through the Netsilon tool. The tool is based on a meta-model specified with MOF 1.4 and the Xion action language. The decision for a meta-model-based approach has been motivated by the fact that in the Web application domain the semantic distance between existing modelling elements (e.g., UML) and newly defined modelling elements is becoming too large.

Our work is complementary to W2000 and UWE in that we propose a meta-model for another prominent Web modelling language, i.e., WebML. Furthermore, in contrast to these approaches we generated the WebML meta-model semi-automatically, instead of manually deriving it from an existing language definition. Finally, our resulting WebML meta-model is based on Ecore and thus, basically corresponds to MOF 2.0, while the meta-models of the other approaches are based on MOF 1.4.

Transforming between DTDs and Meta-Models. There have already been several approaches focusing on the transformation between XML and meta-models (for details on 13 approaches we refer to our survey [34]). Summarizing these results, the approaches can be classified according to the direction of the transformation and the concrete formalisms used as source/target of the transformation. Considering the direction, one can distinguish between forward and backward approaches, regarding the used formalisms the approaches focus on the XML side either on DTDs or XML Schema and on the model side either on MOF, UML or ER, respectively.

In the context of our approach proposed in this paper, especially those approaches conducting a forward transformation from DTD to MOF are closely relevant. To the best of our knowledge, currently there is no such approach but there are two approaches [35], [36] transforming DTDs into UML models which are also closely related, not least since UML is based on MOF. There are, however, two differences to our approach. First, we do not only provide a straight-forward transformation on basis of the correspondences between the two formalisms but rather extend this by employing a set of heuristics dealing with potential ambiguous correspondences, thus facilitating a manual refactoring of the resulting meta-model. Second our approach is based on a higher-level of abstraction meaning that we consider the WebML DTDs at the meta-level M2 whereas the other approaches relate domain DTDs to the model-level M1. Because of this higher-level of abstraction we are able to transform WebML

models in terms of XML documents conforming to the WebML DTD into instances of the corresponding WebML Ecore meta-model (representing in fact a so called linguistic instantiation according to [37] and to validate if these models indeed fully conform to the WebML Ecore meta-model which is not facilitated by the other approaches. Note that, with respect to UML models, an XML document could in principle be mapped onto an object model, which represents an ontological instantiation [37] at M1. However, the problem is that the object model must not fulfil the constraints given by the UML model and thus, the “conforms to”-relationship between the XML document and the DTD is lost.

7. CONCLUSION AND FUTURE WORK

In this paper we aimed at bridging WebML, a prominent Web application development language, to MDE for exploiting its benefits such as standardized storage, exchange, and transformation of models. For this we reused WebML's language specifications partly available in the form of a DTD and partly hard-coded in WebML's modelling tool and generated a MOF-based WebML meta-model in terms of EMF's Ecore through a semi-automatic transformation process. As a prerequisite for this, we identified the general correspondences and deficits of DTDs with respect to Ecore. On this basis, we defined a set of transformation rules as well as heuristics at the meta-meta-level and applied these to WebML's DTDs. The resulting WebML meta-model has been evaluated concerning its completeness and quality giving in particular indication on the applicability of our semi-automatic transformation approach. For evaluating completeness we followed an "example-based" strategy in that we remodelled a WebML reference example on the basis of a tree-based modelling editor supporting our generated meta-model. An appropriate import/export facility of that editor was used to demonstrate that models could be exchanged with the WebRatio tool in a lossless way. Finally, for evaluating the quality of the meta-model, a set of quality metrics was applied to show the improvement of the meta-model during the semi-automatic transformation process.

With respect to future work, first of all, we aim at providing a better-founded evaluation by using a larger set of more complex examples to better demonstrate the completeness of our WebML meta-model. Second, since WebML recently has been extended by additional concepts addressing context-awareness [38], service-enabled [39], and workflow-based [40] Web applications which not yet have been represented explicitly in the WebML' DTDs we plan to extend our meta-model to also capture those WebML concepts. Particularly, with respect to context-awareness we currently investigate how aspect-oriented modelling techniques [41] could be employed to better represent those concepts, since we regard context-awareness to be a cross-cutting concern within Web applications. Third, to exploit

the benefits of MDE for WebML we also envision to provide, on the basis of the transformations currently done at the meta-level, derived transformations at the model level (i.e., of WebML models) into MDE-compliant models which would allow reusing existing WebML models in MDE. For this we intend to apply findings of our ModelCVS project [42] which shall facilitate seamless exchange of models based on meta-model transformations and semantic technologies. Furthermore, we plan to apply ModelCVS for other existing web modelling approaches in order to integrate their concepts in a common domain ontology for web modelling. Finally, we presume that the proposed transformation rules and heuristics, extended by the currently disregarded DTD concepts, could also be successfully applied to transform arbitrary DTDs to MOF-based meta-models, thus going beyond the current focus on WebML's DTDs. Although first experiments with e.g., XHTML, has strengthened this thesis more extensive investigations with respect to the genericity of our approach will be required in the future.

ACKNOWLEDGMENTS

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.

REFERENCES

[1] Bézivin, J.: ‘On the Unification Power of Models’, Software and Systems Modeling, 2005, 4, (2), pp. 171-188

[2] Object Management Group (OMG): ‘Meta Object Facility (MOF) 2.0 Core Specification Version 2.0’, 2004, Available at: http://www.omg.org/docs/ptc/04-10-15.pdf

[3] Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., and Grose, T.J.: ‘Eclipse Modeling Framework’ (Addison-Wesely, 2004, 1st edn.)

[4] Object Management Group (OMG): ‘MOF 2.0/XMI Mapping Specification, v2.1’, 2005, Available at: http://www.omg.org/docs/formal/05-09-01.pdf

[5] QVT-Merge Group: ‘Revised submission for MOF 2.1 Query/View/Transformation’, 2005, Available at: http://www.omg.org/docs/ad/05-07-01.pdf

[6] Ceri, S., Fraternali, P., Bangio, A., Brambilla, M., Comai, S., and Matera, M.: ‘Designing Data-Intensive Web Applications’ (Morgan-Kaufmann, 2003, 1st edn.)

[7] Koch, N., and Kraus, A.: ‘Towards a Common Metamodel for the Development of Web Applications’, Proc. Int. Conf. on Web Engineering, Oviedo, Spain, July 2003, pp. 497-506

[8] Garrigós, I., Casteleyn, S., and Gómez, J.: 'A Structured Approach to Personalize Websites using the OO-H Personalization Framework', Proc. Asia-Pacific Web Conf., Shangai, China, March-April 2005, pp. 695-706

[9] Baresi, L., Colazzo, S., Mainetti, L., and Morasca, S.: 'W2000: A Modeling Notation for Complex Web Applications', in Mendes, E. and Mosley, N. (Ed.): 'Web Engineering: Theory and Practice of Metrics and Measurement for Web Development' (Springer, 2006, 1st edn.), pp. 335-364

[10] 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-November 2005, pp. 700-709

[11] Rossi, G., and Schwabe, D.: 'Model-Based Web Application Development', in Mendes, E. and Mosley, N. (Ed.): 'Web Engineering: Theory and Practice of Metrics and Measurement for Web Development' (Springer, 2006, 1st edn.), pp. 303-334

[12] Pastor, O., Fons, J., Pelechano, V., and Abrahao, S.: 'Conceptual Modelling of Web Applications: The OOWS Approach', in Mendes, E. and Mosley, N. (Ed.): 'Web Engineering: Theory and Practice of Metrics and Measurement for Web Development' (Springer, 2006, 1st edn.), pp. 277-302

[13] Baresi, L., Garzotto, F., and Maritati, M.: 'W2000 as a MOF Metamodel', Proc. World Multiconference on Systemics, Cybernetics and Informatics - Web Engineering track, Orlando, USA, July 2002.

[14] Muller, P.-A., Studer, P., Fondement, F., and Bézivin, J.: 'Platform independent Web application modeling and development with Netsilon', Software and Systems Modeling, 2005, 4, (4), pp. 424-442

[15] World Wide Web Consortium (W3C): 'Extensible Markup Language (XML) 1.1 (Second Edition)', 2006, Available at: http://www.w3.org/TR/xml11/

[16] World Wide Web Consortium (W3C): 'XSL Transformations (XSLT) Version 1.0', November 1999. Available at: http://www.w3.org/TR/xslt/

[17] Jouault, F., Kurtev, I.: 'Transforming Models with ATL'. Proc. MoDELS Satellite Events 2005, Montego Bay, Jamaica, October 2005, pp. 128-138

[18] Object Management Group (OMG): 'UML Specification: Superstructure Version 2.0', 2005, Available at http://www.omg.org/docs/formal/05-07-04.pdf

[19] Kappel, G., Kapsammer, E., and Retschitzegger, W.: 'Integrating XML and Relational Database Systems', World Wide Web Journal, 2004, 7, (4), pp. 343-384

[20] Lämmel, R., and Meijer, E.: ‘Mappings make data processing go ‘round – An inter-paradigmatic mapping tutorial’, Post-proc. Generative and Transformation Techniques in Software Engineering, Braga, Portugal, July 2006, to appear

[21] World Wide Web Consortium (W3C): ‘XML Schema Part 0: Primer (Second Edition)’, 2004. Available at: http://www.w3.org/TR/xmlschema-0/

[22] Lee, D., and Chu, W.: ‘Comparative Analysis of Six XML Schema Languages’, In: ACM SIGMOD Record, 2000, 29, (3). pp. 76-87

[23] Object Management Group (OMG): ‘OCL Specification Version 2.0’, 2005, Available at: http://www.omg.org/docs/ptc/05-06-06.pdf

[24] Schauerhuber, A., Wimmer, M., and Kapsammer, E.: ‘Bridging existing Web Modeling Languages to Model-Driven Engineering: A Metamodel for WebML’, Proc. Workshop on Model-Driven Web Engineering, Palo Alto, USA, July 2006, Article No. 5

[25] Chen, P.P.: ‘The Entity-Relationship Model – Toward a Unified View of Data’, ACM TODS, 1976, 1, (1), pp. 9-36

[26] Vitali, F., Di Iorio, A., and Gubellini, D.: ‘Design Patterns for Descriptive Document Substructures’, Proc. Extreme Markup Languages 2005, Montréal, Canada, August 2005, Available at: http://www.idealliance.org/papers/extreme/Proceedings/html/2005/Vitali01/EML2005Vitali01.html

[27] Comai, S., and Fraternali, P.: ‘A semantic model for specifying data-intensive Web applications using WebML’, Semantic Web Workshop, Stanford, USA, July, 2001, pp. 566-585

[28] Brambilla, M., Comai, S., and Fraternali, P.: ‘Hypertext Semantics for Web Applications’, SEBD Italian National Conference on DataBase Systems, Portoferraio, June, 2002, pp. 73-86

[29] Ma, H., Shao, W., Zhang, L., Ma, Z., and Jiang, Y.: ‘Applying OO Metrics to Assess UML Meta-models’, Proc. Int. Conf. on the Unified Modelling Language, Lisbon, Portugal, October 2004, pp. 12-26

[30] Moreno, N., Fraternali, P., and Vallecillo, A.: ‘A UML 2.0 Profile for WebML Modeling’, Proc. Workshop on Model-Driven Web Engineering, Palo Alto, USA, July 2006, Article No. 4

[31] Garzotto, F., Paolini, P., and Schwabe, D.: ‘HDM - A Model-Based Approach to Hypertext Application Design’, ACM TOIS, 1993, 11, (1), pp. 1-26

[32] Object Management Group (OMG): ‘Meta Object Facility (MOF) Specification Version 1.4’, 2002, Available at: http://www.omg.org/docs/formal/02-04-03.pdf

[33] Object Management Group (OMG): ‘UML Specification Version 1.4’, 2001, Available at: http://www.omg.org/docs/formal/01-09-67.pdf,

[34] Wimmer, M., Schauerhuber, A., Kapsammer, E.: ‘From Document Type Definitions to Metamodels – The WebML Case Study’. Vienna University of Technology, March 2006, Available at: http://big.tuwien.ac.at/projects/webml/

[35] Booch, G., Christerson, M., Fuchs, M., and Koistinen, J.: ‘UML for XML Schema Mapping Specification’. Rational Software and CommerceOne, 1999, Available at: http://xml.coverpages.org/fuchs-uml_xmlschema33.pdf

[36] Rational Software Corporation. ‘Migrating from XML DTD to XML Schema using UML. Rational Software White Paper’, 2000, Available at: http://www.rational.com/media/whitepapers/TP189draft.pdf

[37] Atkinson, C., and Kühne, T.: ‘Model-Driven Development: A Metamodeling Foundation’, IEEE Software, 2003, 20, (5), pp. 36-41

[38] Ceri, S., Daniel, F., Matera, M., Facca, and F.: ‘Model-driven Development of Context-Aware Web Applications’, ACM TOIT, 2007, 7, (2), to appear

[39] Manolescu, I., Brambilla, M., Ceri, S., Comai, S., and Fraternali, P.: ‘Model-Driven Design and Deployment of Service-Enabled Web Applications’, ACM TOIT, 2005, 5, (3), pp. 439-479

[40] Brambilla, M., Ceri, S., Fraternali, P., and Manolescu, I.: ‘Process Modeling in Web Applications’, ACM TOSEM, 2006, in print

[41] Schauerhuber, A.: ‘aspectUWA: Applying Aspect-Orientation to the Model-Driven Development of Ubiquitous Web Applications’, Int. Conf. on Aspect-Oriented Software Development: Poster Event , Bonn, Germany, March 2006, Available at: http://www.wit.at/people/schauerhuber/

[42] Kappel, G., Kapsammer, E., Kargl, H., Kramler, G.; Reiter, T., Retschitzegger, W., Schwinger, W., Wimmer, M.: ‘Lifting metamodels to ontologies - a step to the semantic integration of modeling languages’, Proc. Int. Conf. on Model Driven Engineering Languages and Systems, Genova, Italy, October 2006, pp. 528-542