Transformation Techniques in the Model-Driven Development Process of UWE

Nora Koch
Ludwig-Maximilians-Universität
Oettingenstr. 67, 80538 München
and FAST GmbH
Arabellastr. 17, 81925 München
Germany

kochn@pst.ifi.lmu.de

ABSTRACT

Development of Web software is still an inefficient and error-prone process. We need integrated techniques and tool support for automated generation of Web systems. The goal of model-driven development (MDD) is to tackle these problems introducing a higher level of abstraction by defining metamodels and model transformations rules. We present the development process of the UML-based Web Engineering (UWE) approach defined as an MDD approach and focus on the model transformation aspects of the process.

Categories and Subject Descriptors

I [**Computing Methodologies**]: I6 Simulation and Modeling, I6.5 Model Development – *modeling methodologies.*

General Terms

Design, Languages, Standardization.

Keywords

Model-Driven Development, Metamodel, Modeling Language, Model Transformation, MDA, Transformation Language, UML, UWE, Web Engineering.

1. INTRODUCTION

Software development techniques are continuously evolving with the goal of solving the main problems that still affect the building and maintenance of software systems: time, costs and error-proneness. Regarding the development process two important trends can be observed: agile and model-driven development.

Model-driven development (MDD) [4] focuses on the construction of models, specification of transformation rules, tool support and automatic generation of code and documentation. Agile development [1] instead focuses on the stakeholder's activities. The central idea of MDD is to separate the platform independent design from the platform specific implementation of applications delaying as much as possible the dependence on specific technologies. Therefore, MDD advocates the construction of platform independent models and the support of model transformations. Consequently, the software development process can be viewed as a chain of model transformations.

Web Engineering is a concrete domain where MDD can be helpful, particularly in addressing the problems of evolution and adaptation of Web software to continuously emerging new platforms and changes in technologies. During the last years the Web engineering community has proposed several languages, architectures, methods and processes for the development of Web applications. In particular, methods for modeling such systems were developed, such as Hera [9], OOHDM [31], OO-H [10], OOWS [33], UWE [19], WebML [5], and W2000 [2]. They focus on the specification of analysis and design models for Web systems, such as the construction of navigation or adaptation models. However, the model transformation aspects were neglected by most of these methods.

We present an overview of the complete MDD process of the UML-based Web Engineering (UWE) approach and focus in this work on the model transformation aspects of the process. The UWE process covers the whole development life cycle of Web systems from the requirements specification to code generation. The difference to other approaches in the Web domain is on the one hand the specification of all models in UML a kind of *lingua franca* for object-oriented specification. On the other hand – and more innovative – is the use of forthcoming transformation languages for the specification of transformation rules in the development process defined by UWE. However, the transformation rules defined in the first development phase of UWE, such as those integrated in the ArgoUWE CASE tool [14], are still tool proprietary. In the more recent phases, we use emerging specification techniques like transformation languages (ATL [13], QVT [28]), and graph transformations.

In this work a set of criteria and values are selected for the classification and comparison of model transformations in the UWE process. These criteria could also be applied to model transformations of other development processes. As far as we know no such analysis and classification has been performed for any other MDD process.

The best-known MDD realization is the Model-Driven Architecture (MDA) of the OMG [25]. The development process of UWE is based on MDA as well as other OMG standards (UML [29], XMI [30], MOF [26], OCL [27]) and the forthcoming standard transformation language QVT ([28]).

The remainder of this paper is structured as follows: Sect. 2 gives a brief overview of the relevant MDD concepts. Sect. 3 presents the UWE process and the description of the UWE models. An analysis of the transformations that are applied to the source models in each step of the UWE process is presented in Sect.4. Sect. 5 provides an overview of related work. Finally, in Sect. 6 we present some conclusions and outline our work in progress and future plans on the implementation of the model transformations.

2. BASIC MDD CONCEPTS

The idea behind MDD is that modeling and transforming is a better foundation for the development and maintenance of systems than programming [24]. The primary goals of MDD are portability, interoperability and reusability trough architectural separation of concerns. A model-driven approach requires languages for the specification of models, the definition of transformations, and the description of metamodels. The concrete techniques developed so far supporting the MDA approach of the OMG include the Unified Modeling Language (UML), Query View Transformation Specification (QVT) and Meta Object facility (MOF).

There are still problems with appropriate tool support and exchange formats, needed for a seamless implementation of the process, but we are observing how research, industry interest and standardization efforts are moving to support the complete MDD process.

2.1 Types of Models

A *model* of a system is a specification of that system and its environment for some certain purpose. Models consist of a set of elements with a graphical and/or textual representation. The idea of MDD is creating different models of a system at different levels of abstraction and using transformations to produce the implementation of the system. MDA suggests building computational independent models (CIM), platform independent models (PIM) and platform specific models (PSM) corresponding to different levels of abstraction or viewpoints [25]. We will use these types to classify the UWE models in Sect. 3.

The *computational independent viewpoint* focuses on the environment of the system, and the requirements the user has on the system; the description provides what the system is expected to do. The details of the structure and processing are hidden or yet undetermined. A computational independent model is sometimes called a domain model or a business model. It should be traceable from the PIM and PSM models that implement the CIM.

The *platform independent viewpoint* focuses on the operation of a system while hiding the details for a particular platform. It shows the part of the complete specification that does not change from one platform to another.

The *platform specific viewpoint* combines the platform independent viewpoint with additional features of a specific platform.

The objective is to postpone in the development process the creation of models that take into account technological aspects of a platform as much as possible. The main advantage is to be able to react efficiently and with low costs to technology changes.

2.2 Transformation Aspects

Model transformation is the process of converting one or more models – called source models – to one output model – the target model – of the same system [25]. The concepts mapping and relation are defined as specializations of transformation. A mapping is defined as a unidirectional transformation in contrast to a relation that defines a bi-directional transformation. Note that the model transformation result is exactly one model.

We distinguish the following aspects of transformations: type (based on the type of the models involved), complexity, level of abstraction, use of marks, execution and implementation types. We use these aspects as classification criteria for the transformations of the UWE process presented in Sect. 4. They could also be used to analyze other MDD processes.

2.2.1 Transformation Type

In a model-driven development process model transformations can be of type CIM to PIM, PIM to PIM, PIM to PSM and PSM to code. A computational independent model can be refined, i.e. a CIM can be mapped to another CIM, in the same way PIMs can be refined. Note that transformations from PIM to CIM, PSM to CIM, and PSM to PIM are not possible.

2.2.2 Transformation Complexity

Transformations may combine elements of different source models in order to build a target model. According to the number of source models involved in the mapping process a transformation is named simple or a merge.

2.2.3 Level of Abstraction

Transformations can be defined for elements of different levels: metamodel, model or instance.

A *transformation at metamodel level* is the specification of certain types of source models that are converted to another type of target models, i.e. they apply to all source model elements of a type as represented in the pattern of Bezivin [4] in Figure 1. A *transformation at model level* consists of the identification of particular model elements that will be transformed according to certain rules and a *transformation at instance level* consists on the identification of specific objects that will be transformed in a certain way.
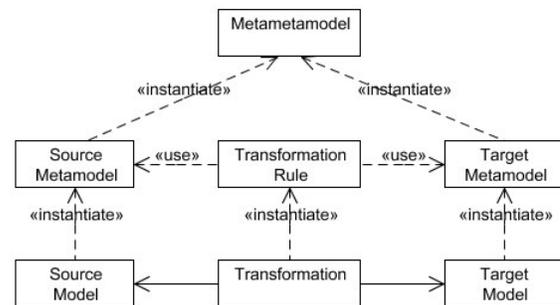

Figure 1: Model transformation pattern [4]

2.2.4 Use of Marks and Additional Information

Transformation rules rely on certain *marks* (types, patterns, templates or UML profile elements) in order to select the elements

to which a rule applies [25]. These marks can be part of the elements or take the form of additional input that does not pollute the source models, i.e. non-intrusive or lightweight extension to models. Examples of marks provided by the model itself are *types* (class or association) and *stereotypes of UML profiles*. In addition, *patterns* identifiable in the source model can also be used as a mark in a transformation rule, as a certain combination of modeling elements.

Other marks instead are only required for the mappings. They do not need to be integrated in the source model [24], such as selection of certain classes or states. This kind of marks are kept in separate *marking models* and combined with the source models during the mapping process. *Templates* are other external providers of input for transformations. They are like patterns but may include more detailed information to guide the transformation.

Additional information can be used to guide the transformation. Often it is drawn from the knowledge the designer has about the application domain or its knowledge on the technology platform. For example a particular architecture style or pattern may be specified.

2.2.5 Execution Type
Transformations are classified in automatic, semi-automatic and manual based on the decisions the designer takes on the source and target models.

A transformation is automatic if it does not require any decision from the user of the system. The transformation is semi-automatic if the user takes the decision of which elements of the source model will be transformed, and manual if the designer produces the results. A model-driven process aims to define transformations rules that allow for automatic model transformations.

2.2.6 Implementation Technique
Transformation rules can be implemented in (1) general programming language as Java, i.e. hard coded in specific tools, or (2) languages for transformations such as ATL [13] or QVT [28]. Transformations are often based on invariants and pre conditions and post conditions specified in languages such as OCL [27].

3. UWE PROCESS AND MODELS
The UWE approach comprises a UML Profile for modeling Web systems, a process and tool support for the development of Web systems. For modeling with UWE and UWE CASE tool we refer the reader to [3], [6], [12], [14], [15], [17] and [19].

The UWE process is a model driven development process following the MDA principles and using the OMG standards ([26], [27], [29], [30]). It consists of a set models and model transformations, which specification is supported by metamodels and transformations rules. The metamodels are the UWE metamodel [20], the Web Requirements Engineering metamodel (WebRE) [6] and the metamodel of the Web Software Architecture approach (WebSA) [23].

3.1 Process Overview
The main characteristic of the UWE process is the systematic, semi-automatic, model-driven and transformation-based (Sect. 4)

support of the development of Web systems. The UWE process is depicted in Figure 2 as a stereotyped UML activity diagram ([23]). Models are represented with object flow states and transformations as stereotyped activities (special circular icon). A chain of transformations then defines the control flow.
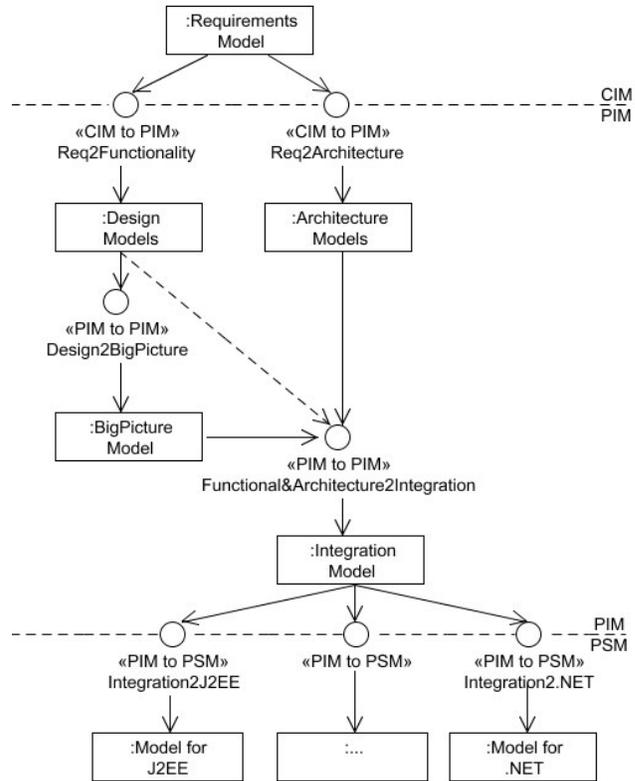

Figure 2: UWE Process Overview

The process starts with the business model (CIM) level defining a requirements model. Platform independent design models (PIMs) are derived from these requirements. The set of design models represents the different concerns of the Web applications. It comprises the content, the navigation, the business logic, the presentation, and the adaptation of the Web system. The different design models are not depicted in the overview shown in Figure 2.

Design models are afterwards integrated mainly for the purpose of e.g. verification into a big picture model [17]. A merge with architectural modeling features results in an integrated PIM model covering functional and architectural aspects. Finally, the platform specific models (PSMs) are derived from the integration model from which programming code can be generated. The aim of such an MDD process is automatic model transformation in each step based on rule transformations.

3.2 Models in UWE
A set of models of a Web system is built during the UWE development process. Each model belongs to one of the three viewpoints described in Sect. 2.1, i.e. CIM, PIM or PSM (see Figure 2).

UWE models are represented by UML diagrams. Whenever appropriate UWE uses the "pure" UML notation. For modeling specific features of the Web domain, such as navigation nodes and Web pages elements UWE provides a UML profile, which is defined using the extension mechanisms provided by the UML: stereotypes and OCL constraints. For further details on UWE profile refer to [3], [6], [15], [17], [19] and [21].

We illustrate models and model transformations by means of a music Web portal example, inspired by `www.mp3.com`, which provides albums for download. Information about singer, composer, and publisher are available for free, instead only registered users can search albums and download them if they have enough credit on their prepaid account.

3.2.1 Requirements Model
The overall objective of modeling the requirements is the specification of the functionality of the system as a computational independent model (CIM). The specific objectives for Web systems become: (1) the specification of the functional requirements in terms of navigation needs and business processes, (2) the specification of content requirements, and (3) the definition of interaction scenarios for different groups of Web users. .
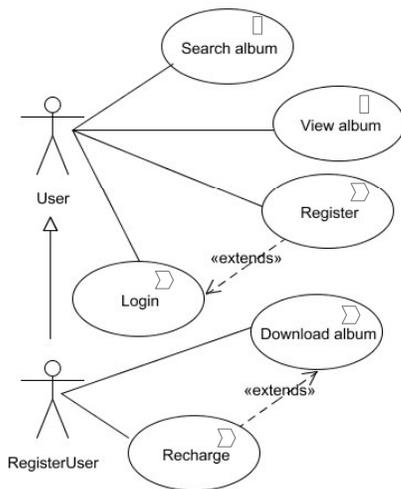
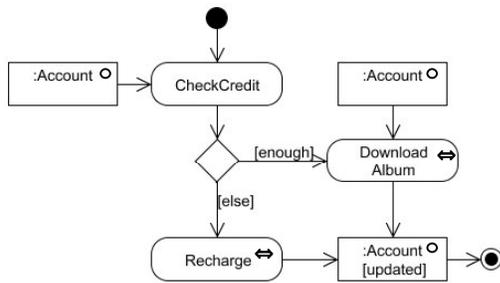

Figure 3: Use case diagram (CIM)

Figure 4: Activity diagram for a (simplified) download album use case (CIM)

UWE models requirements with UML use case diagrams and UML activity diagrams. UWE distinguishes two types of use cases: navigation use cases and use cases describing Web business processes. The latter – following UWE's recommendation –

should be further detailed with activity diagrams. UWE uses the UML profile for Web requirements (WebRE) defined by Escalona & Koch [6], which comprises stereotyped use cases, activities and objects providing modeling elements with Web domain specific semantics. Figure 3 depicts the use case diagram for the music portal and Figure 4 shows the activity diagram for the *Download album use* case

3.2.2 Design Models
At design level UWE follows the separation of concerns widely applied in Web engineering. We build separate models for content, navigation and presentation aspects of Web systems using UML class diagrams for the visual representation [19]. We supplement them with an additional process model for transactional Web applications, and an adaptation model for personalized and context-dependent systems. UWE defines Web domain specific modeling elements e.g. navigation class and menu for the navigation model, and presentation class and anchor for the presentation model.

The UWE profile provides the corresponding stereotypes. Figure 5 and Figure 6 depict the content and navigation model of the music portal. Navigable nodes are represented by instances of the metaclass *NavigationNode* such as *NavigationClass*, *Menu* and *ProcessClass* (stereotypes that extend the UML Class). Links between navigation nodes are represented by instances of *NavigationLink* and *ProcessLink*. In addition, navigation paths are structured by instances of special types of access primitives such as *Index, Query* and *GuidedTour*. Indexes represent choices among instances of a specific navigation class; menus (like *MainMenu*) in contrast represent choices among instances of navigation nodes of different types. A Query (like *SearchAlbum*) models a search action in the Web application, where a user can enter a term and select from the matching results.
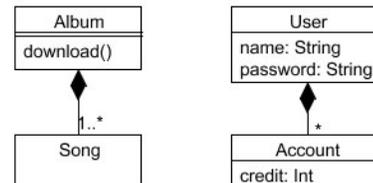

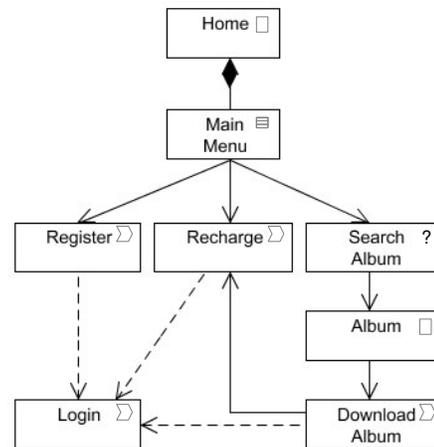**Figure 5: Content and user model (PIM)**

Figure 6: Navigation model (PIM)

Process models are visualized as UML 2.0 activity diagrams (see Figure 7). Actions (like *FindUser*) model the actions the user and the system must carry out to complete the business process.
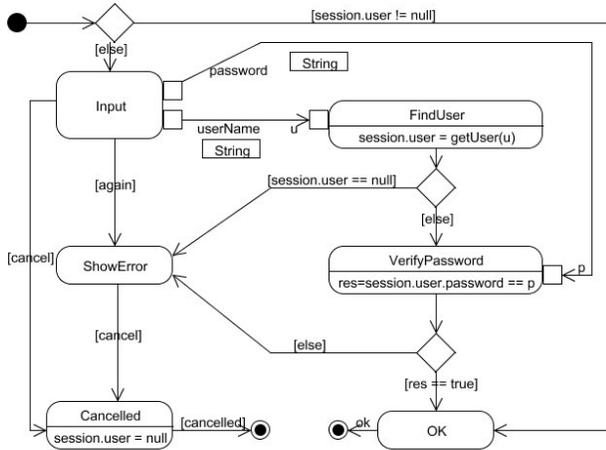

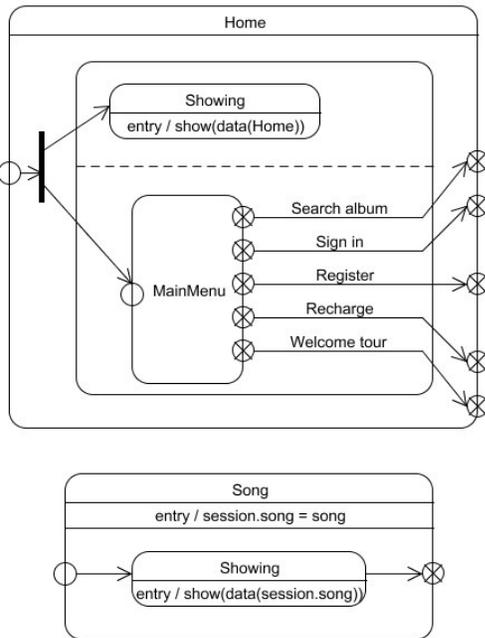**Figure 7: Business process Login (PIM)**

Figure 8: States Home and Song of the "big picture" (PIM)

UWE proposes to build a presentation model to sketch the layout of the Web application. It uses the UML composition notation for classes, i.e. containment represented by graphical nesting of the symbols. This kind of representation is appropriate for modeling user interfaces as it allows for spatial ordering, but has the problem that most standard case tools do not support it. For adaptation models the UWE profile includes stereotypes for different node and link adaptation. The diagrammatic technique used by UWE is aspect oriented modeling (AOM), extending the UML with concepts such as aspect, pointcut and advice to support AOM [1]. Due to space restrictions we do not give further details

to presentation and adaptation models in this paper; the reader is referred to [3] and [18].

UWE proposes the generation of an integrated model that merges the separate concerns of a Web system into a big picture (see Figure 2. Using UML state machines as the results of the integration process offers the possibility of applying formal techniques for validation, like model checking [16]. Currently the big picture is the result of the integration of the UWE content, navigation and business logic models, but it can easily be extended to include features like access control [36] and adaptation [3]. Figure 8 shows the states *Home* and *Song*, both states are part of the "Big Picture" model.

3.2.3 Architecture and Implementation Models
Additional information on architectural styles can be merged at different steps in the UWE MDD process. Following the WebSA approach we propose in [23] to integrate functional design models and architecture models in a very early development phase. Such an approach is shown in the UWE process overview depicted in Figure 2. Architecture models in the Web Software Architecture (WebSA) approach are specified as platform independent models (PIMs). Knapp and Zhang suggest in [17] to merge architecture models with the big picture model, i.e. with the result of the already integrated model of the different concerns (content, navigation and business logic). A third alternative is to introduce the architectural information in the generation of platform specific models.

4. MODEL TRANSFORMATIONS IN UWE
Model transformations are based on the definition of transformation rules, which are defined whenever possible for the metamodel level and written as expressions of transformation languages. Hence, we need the specification of the metamodels of both the source and the target of the transformation. In addition, to the UWE metamodel we use the WebRE metamodel [6] and the WebSA metamodel [23] that are MOF-compliant metamodels.

Transformations are classified into three groups: those used to build the design models, those needed to generate the big picture and the integration model, and finally transformations for the generation of implementation models and code. We summarize the characteristics of each transformation in Table 1 based on the criteria defined in Sect. 2.2.

4.1 Building Design Models
The first model transformation step of the UWE process consists of the mapping of the Web requirements models to the UWE design models [21]. The design models are the content, navigation, process, presentation, and adaptation model. There exists a set of dependencies among these design models themselves that allow for creation of other models or refinement of models.

Transformations rules are defined as mappings from metamodel WebRE to the UWE metamodel and among UWE metamodels. Figure 9 shows for example, how the model transformation pattern of Figure 1 is applied to the UWE process using the standard Query View Transformation Language (QVT, [28]).

Transfor-mation	Req2 Content	Req2 Architec-ture	Content2 Navigation	Navigation Refinement	Req2 Navigation	Navigation2 Presentation	Style Adjust-ment	Design2 BigPicture	Integrating Architecture	Integration2 J2EE
Type	CIM to PIM	CIM to PIM	PIM to PIM	PIM to PIM	CIM to PIM	PIM to PIM	PIM to PIM	PIM to PIM	PIM to PIM	PIM to PSM
Complexity	simple	simple	simple	simple	merge	simple	merge	merge	merge	merge
Level of abstraction	metamodel	-	model	metamodel	metamodel	metamodel	model	metamodel	metamodel	metamodel
Marks	WebRE profile	-	UWE profile & navigation relevance	UWE profile & patterns	WebRE profile	UWE profile	style guide	patterns and marks	UWE & WebSA profile	patterns
Execution	automatic	manual	semi-automatic	automatic	automatic	automatic	automatic	automatic	automatic	automatic
Techniques	QVT	-	Java (OCL) ATL	Java (OCL)	QVT	Java (OCL) ATL	Java	graph transfor-mations	QVT	QVT, ATL

Table 1: Characteristics of model transformations in the UWE model-driven development process
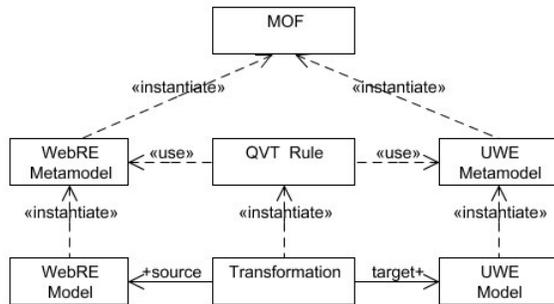

Figure 9: Model transformation pattern for metamodels WebRE and UWE

In the UWE process the transformation *requirements to content* allows for the construction of the content model; the transformations *content to navigation, requirements to navigation* and *navigation refinement* are used to build the navigation model. The presentation model is built in at least two iterations: it is created with the former *navigation to presentation* and refined by *style adjustments*. Last but not least the adaptation model can also be extracted from the functional requirements models, and the architecture models from the non-functional requirements.

4.1.1 Transforming Requirements to Content
Web activities, such as browse, search or transactions are related to objects that are either required as input or produced as results. These objects can be included in activity diagrams by means of object flows. In the particular case of modeling Web systems requirements, objects are used to indicate the need to include certain content information in the Web application (Figure 10).

Figure 10: From requirements to content model

We use the QVT language to specify the transformation from elements of the requirements model to elements of the content model (*Req2Content*). The transformation rule defines the mapping of the metaclass *Content* of the WebRE metamodel to classes of the UWE content model; the QVT specification of the transformation is shown in Figure 11. For further details refer to [21]. The characteristics of the *Req2Content* transformation are summarized in Table 1.

```
transformation ReqContent2ContentClass (webre:WebRE, uwe:UWE) {
    top relation R1 {
        checkonly domain webre  c:Content { name = n };
        enforce domain uwe cc: Class { name = n };  }
    top relation R2 {
        cn: String;
    checkonly domain webre p: Property { namespace=c:
                Content {}, name = cn};
        enforce domain uwe p1:Property { namespace = cc: Class{};
                name = cn}
        when {R1 (c,cc); }
    }
}

                name = cn}
        when {R1 (c,cc); }
    }
}
```

Figure 11: Transformation requirements elements to content elements (QVT textual notation)

4.1.2 Transforming Requirements to Architecture
A mapping of non-functional requirements to architectural model elements is subject of future work. Currently, the designer includes architectural elements manually. A metamodel of non-functional requirements for Web applications is still missing.

4.1.3 Transforming Content to Navigation
In UWE a first navigation model (see Figure 12) is generated based on classes of the content model marked as navigation relevant, i.e. the transformation *Content2Navigation* is defined for certain model elements. From one content model different navigation views can be obtained, e.g. for different stakeholders of the Web system like anonymous user, registered user and administrator [19].

The generation of each navigation view requires a set of marks on elements of the content model, which comprise a so-called *marking model*, kept separately from the content model. Hence, the development process cannot be completed in an automatic way, as the designer has to take the decision about the "navigation relevance" marks. Once the marks have been set, the transformation is applied. It is defined as an OCL constraint and implemented in Java in the CASE tool ArgoUWE [15].
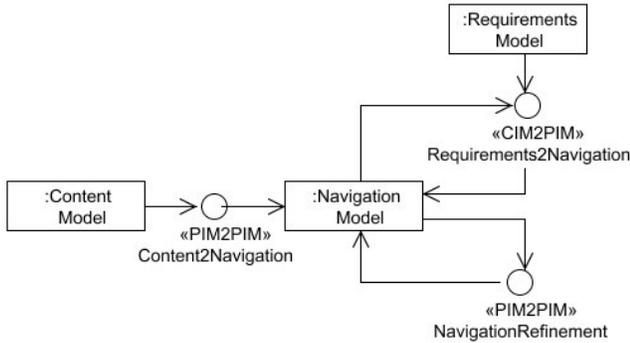

Figure 12: Transformations to build navigation model

4.1.4 Adding Requirements to Navigation

The requirements model contains information that is useful for the enrichment of the navigation model. For example, UWE distinguishes in the requirements model among different types of navigation functionality: browse, search and transactional activities. On the one side, *Browse* actions can be used to verify the existence of a navigation path between source and target nodes. On the other side, e.g. an action of type *Search* indicates the need of a *Query* in the navigation model in order to allow for user input of a term and the system responding with a resulting set matching this term. Figure 13 shows the Search2Query transformation specified in the QVT graphical notation [21].

The transformation *Req2Navigation* is a merge and is based on the WebRE profile (see Table 1). Figure 12 shows that the transformation rule *Req2Navigation* can be applied after the transformation rule *Content2Navigation*, but there is no restriction related to the order in which the *Req2Navigation* rule and the *NavigationRefinement* has to be applied.

4.1.5 Refining the Navigation Model

The navigation model generated on the content model contains itself valuable information that allows for reasoning and improving the navigation model [12]. The following constrains (informally described) define e.g. such transformation rules:

1. An index is added for all associations of the navigation model that have multiplicity greater than one at the directed association end.

2. All navigation classes that have at least one outgoing association require a menu class with menu items defined on basis of the association ends of the associations.

These transformations are defined as OCL constraints in UWE and implemented in Java in the CASE tool ArgoUWE [15]. See Table 1 for the characteristics of these transformation rules.
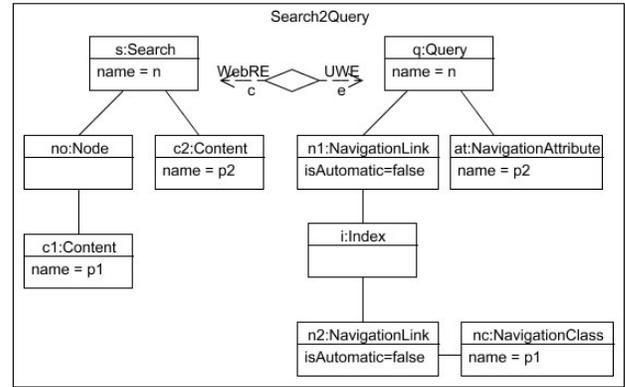

Figure 13: Search2Query transformation (QVT graphical notation)

4.1.6 Transforming Navigation to Presentation and Adjusting to Presentation Style.

Presentation elements are generated based on navigation elements of the navigation model and merged then with style guide information (Figure 14). For example for each link in the navigation model an adequate anchor is required in the presentation model The main difficulty is the introduction of the look and feel aspect.
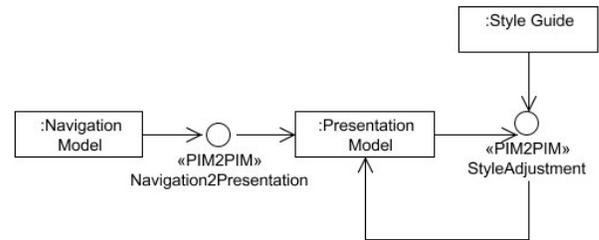

Figure 14: Transformations to build presentation model

ArgoUWE implements the *Navigation2Presentation* rule in Java. *Style2Adjustment* rules are not implemented by the time this work was written. Table 1 characterizes both, the *Navigation2 Presentation* and the *Style2Adjustment*.

4.2 Creation of an Integrated Model

The aim of this phase in the UWE MDD process is the creation of one model that allows both seamless creation of platform specific models (PSMs) and validation of correctness of the models by model checking. The UWE process comprises two main integration steps: the integration of all functional models and the integration of functional and non-functional aspects; the latter related to architectural design decisions.

4.2.1 Building the "Big Picture"

Though from different viewpoints, the different design models represent the Web application as a whole. They are integrated into another platform independent model that we call the big picture (Figure 15). This model is used to validate the interaction of the separated models using model checking and to

generate the Web application automatically. The target model is a UML state machine, representing the navigation structure and the business processes of the Web application. The big picture model can be checked by the tool Hugo/RT – a UML model translator for model checking and theorem proving [17].
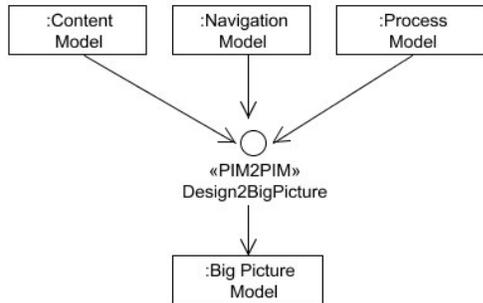

Figure 15: Transformations to build "big picture" model

The transformation *Design2BigPicture* form a metamodel-based graph transformation system. An example of the graph transformation of a navigation node to a navigation state in the big picture is depicted in Figure 16. Source models are the content, business process and navigation models of UWE. Big Picture transformation rules are defined within the scope of UWE as graph transformation rules. Work in progress is the implementation of these transformation rules in AGG [32] (a non-Web specific tool for graph transformations). Other characteristics of the model transformation *Design2BigPicture* are outlined in Table 1.
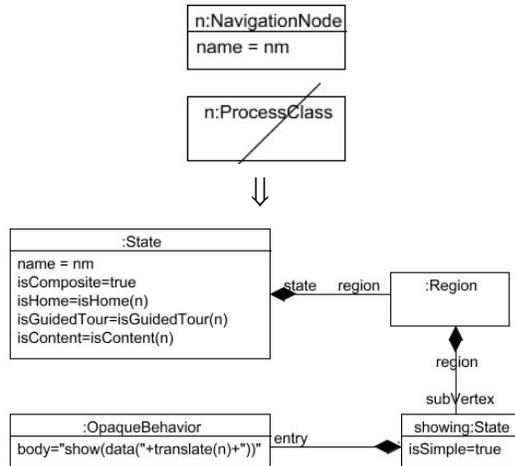

Figure 16: Mapping navigation node to state in "big picture" (graph transformation)

4.2.2 Integrating Architectural Features

Functional models defined so far (e.g. navigation, presentation, process) can be merged with architecture models (defined as PIMs) as shown in Figure 17. WebSA provides a layer-view and a component-view of the architecture, which are also defined as PIMs. Transformation rules are defined based on the UWE and WebSA metamodels (for further details see [23]). The characteristics of the rules of type *IntegratingArchitectural Features* are outlined in Table 1.
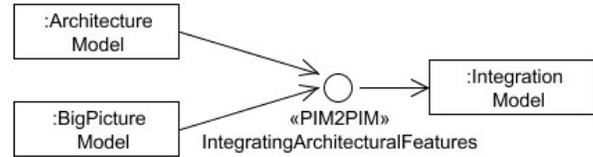

Figure 17: Integration of architectural features

4.3 Generation of Platform Specific Models and Code

To transform technology independent models into platform specific models additional information about the platform is required. It can be provided as an additional model or it is implicitly contained in the transformations. For mappings from UWE design models (PIMs) to PSMs for Web applications (Figure 18) we performed a set of experiments with the recently developed model transformation languages. The Query View Transformations languages used are the Atlas Transformation Language (ATL) [13], QVT-P and QVT [28]. For example, the transformation depicted in

Figure 19 tackles the generation of J2EE elements from Server Pages of the Integration Model. The rule is written in QVT-P language.
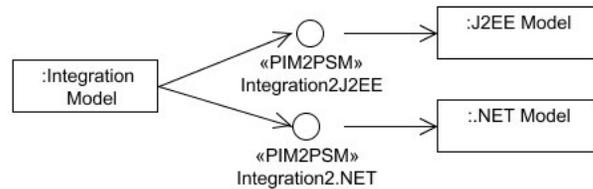

Figure 18: Generation of platform specific models

```
relation ServerPage2J2EE {
   domain { (IM.IntegrationModel) [ (ServerPage) [name=nc,
      services = {(WebService) [name=on, type=ot]}, views = {(View)
[name = vn]}]] }
domain { (JM.J2EEModel) [ (JavaServerPage) [name=nc,
forms = {(Form) [name=on, type=ot]}, beans = {(JavaClass) [name =
vn]}]] }
when { services -> forAll (s | WebService2Form (s, F1set.toChoice()) )
views-> forAll (v | View2Bean (v, J1set.toChoice()) )) }}
}
```

Figure 19: Generation of J2EE model elements based on the integration model (QVT-P language)

Another example is shown in

Figure 20. The ATL code exemplifies a transformation rule that maps the element *Anchor* of the UWE presentation model to a JSP element. Note that the transformation rule also involves elements of the navigation model (*NavigationLink*) and content model (*ContentNode*).

```
rule Anchor2JSP {
from uie : UWE!Anchor (
  to jsp : JSP!Element (
    name <- 'a',
    children <- Sequence { hrefAttribute, contentNode } ),
  hrefAttribute : JSP!Attribute (
    name <- 'href',
    value <- thisModule.createJSTLURLExpr(
                    uie.navigationLink.target.name, 'objID' ) ),
  contentNode : JSP!TextNode (
                    value <- uie.name )
}
```

**Figure 20: Generation of JSP elements based on the
presentation model (ATL language)**

5. RELATED WORK

The MDD approach of UWE focuses on model transformations defined at metamodel level and specified in general purpose transformation languages, such as QVT and graph transformations. Transformation languages are also used by some other Web design methods.

WebSA is an approach that focuses on architectural models and transformations specified in QVT. It is partially integrated in the UWE process [23]. Baresi and Mainetti [2] propose to use rule transformation techniques for the verification of correctness and adaptability of models designed by W2000. The approach is based on a work on graph transformations [11]. OOWS [33] uses graph transformations to automate its CIM to PIM transformation.

WebML follows an MDD approach for mapping modeling elements of WebML to architecture components of MVC2, which can be transformed into components for different platforms [5]. OO-H [10] supports a transformation-based construction of a presentation model based on modeling elements of the navigation model and code generation based on the conceptual, navigation and presentation models. Both, WebML and OO-H transformation rules are proprietary part of their CASE tools. Hera – an approach centered on the Semantic Web–RDF technology – instead applies MDD only to the creation of a model for data integration [35].

The approaches of Engels et al [8] and Varró and Pataricza [34] are interesting although they do not consider Web domain specific characteristics but they define a generic approach with focus on formal definition of transformation semantics.

6. CONCLUSIONS AND FUTURE WORK

We presented the development process of the UML-based Web Engineering (UWE) approach defined as a model-driven development approach. We outlined all the models and model transformations that comprise the MDD process, focusing on the classification of the model transformations in terms of: MDA type, complexity, level of abstraction of the rule definition, number of source models, involvement of marking models, implementation techniques and execution type.

We use different specification techniques for the transformations like ATL, QVT, graph transformations, and hard-coded in Java.

We need to redefine some of them, e.g. those that are hard coded in the CASE tool, in order to benefit from transformation rules defined at a higher abstraction level, e.g. using graph transformations or transformation languages.

By the time this paper was written, the main problem still is the tool support for model transformations. A detailed analysis of the requirements of such tools is beyond the scope of this paper. We plan to concentrate on the most promising and adequate approach and those that provide a user-friendly tool environment. For example, we plan to use the AGG [32] and the apache struts technology (www.apache.org) to produce results that can then be integrated with the tool environment HUGO/RT [16] for model checking purposes. We plan to use our research results in this area for the automatic generation of test cases.

REFERENCES
[1] Amber, S. Agile Model-Driven Development with UML 2.0. Cambridge University Press, 2004.

[2] Baresi, L. and Mainetti, L. Beyond Modeling Notations: Consistency and Adaptability of W2000 Models. In Proc. of SAC'05, ACM Symposium on Applied Computing, Santa Fe, USA, 2005

[3] Baumeister, H., Knapp, A., Koch, N. and Zhang, G. Modelling Adaptivity with Aspects. In Proc. 5th Int. Conf. on Web Engineering (ICWE 2005), LNCS 3579, Springer, July 2005.

[4] Bézivin, J. In Search of a Basic Principle for Model Driven Engineering. UPGRADE V(2), Novótica, April 2004.

[5] Ceri, S., Fraternali, P. and Matera, M. Conceptual Modeling of Data-Intensive Web Applications, IEEE Internet Computing 6(4), July/August 2002.

[6] Escalona, M. J. and Koch, N. Metamodeling Requirements of Web Systems. In Proc. 2nd Int. Conf. on Web Information System and Technologies, Portugal, April 2006.

[7] Escalona, M. J., Mejías, M. and Torres, J. Developing System with NDT & NDT-Tool. In Proc. of 13th Information System Development (ISD 2004), Lithuania, 2004.

[8] Engels, G., Hausmann, J-H, Heckel, R. and Sauer, S. Dynamic Meta Modeling: A Graphical Approach to Operational Semantics of Behavioral Diagrams in UML. In

Proc. of 3rd Int. Conf. on the Unified Modeling Language (UML 2000), LNCS 1939, Springer, October 2000.

[9] Frasincar F., Houben, G.J. and Vdovjak R. An RMM-Based Methodology for Hypermedia Presentation Design. In Proc. of Advances in Databases and Information Systems (ADBIS 2001) Vilnius, Lithuania, Springer, LNCS 2151, 2001.

[10] Gomez, J., Cachero, C. and Pastor, O. Extending a Conceptual Modelling Approach to Web Application Design. In Proc. 2nd CaiSE´00, LNCS 1789, Springer Verlag, Stockholm, June 2000.

[11] Heckel R. and Lohmann M. Model-based development of Web applications using graphical reaction rules. In Proc. Fundamental Approaches to Software Engineering, Warsaw, Springer.

[12] Hennicker, R. and Koch, N. A UML-based Methodology for Hypermedia Design. In Proc. of the Int. Conference, editors, UML'2000 - The Unified Modeling Language - Advancing the Standard, LNCS 1939, York, Springer, October 2000.

[13] Jouault, F., and Kurtev, I. Transforming Models with ATL. In Proc. of the Model Transformations in Practice Workshop at MoDELS 2005, Jamaica.

[14] Knapp A., Koch N., Moser F. and Zhang G. ArgoUWE: A CASE Tool for Web Applications. In 1st Int. Workshop on Engineering Methods to Support Information Systems Evolution (EMSISE03), September 2003.

[15] Knapp, A., Koch, N., Zhang, G. and Hassler, H.-M. Modeling Business Processes in Web Applications with ArgoUWE. 7th Int. Conference on the Unified Modeling Language (UML 2004). LNCS 3273, Lisbon, 2004.

[16] Knapp, A., Merz, S. and Rauh, C. Model Checking Timed UML State Machines and Collaborations. In Proc. 7th Int. Symposium Formal Techniques in Real-Time and Fault Tolerant Systems, LNCS 2469, Springer, Berlin, 2002.

[17] Knapp, A. and Zhang, G. Model Transformations for Integrating and Validating Web Application Models. In Proc. of Modellierung 2006, Innsbruck, March 2006.

[18] Koch, N. Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process. Uni-Druck Verlag, 2001. PhD. Thesis, LMU München.

[19] Koch, N. and Kraus, A. The expressive Power of UML-based Web Engineering. 2nd Int. Workshop on Web-oriented Software Technology (IWWOST02). Málaga, Spain. June, 2002.

[20] Koch, N. and Kraus, A. Towards a Common Metamodel for the Development of Web Applications. In 3rd Int. Conf. on Web Engineering (ICWE 2003), LNCS 2722, Springer, July 2003.

[21] Koch, N., Zhang, G. and Escalona, M. J. Model Transformations from Requirements to Web System Design, 2006, 6th Int. Conf. on Web Engineering, Palo Alto, USA, July 2006.

[22] Manifesto for Agile Software Development http://agilemanifesto.org/, last retrieved March 22, 2006.

[23] Melía, S., Kraus, A. and Koch, N. MDA Transformations Applied to Web Application Development. In Proc. 5th Int. Conf. on Web Engineering (ICWE 2005), Sydney, Australia, LNCS 3579, Springer, July 2005.

[24] Mellor, S., Scott, K., Uhl, A. and Weise, D. MDA Distelled; Principles of Model-Driven Architecture, Addison Wesley, 2004.

[25] Object Management Group (OMG). MDA Guide Version 1.0.1. omg/2003-06-01, http://www.omg.org/docs/omg/03-06-01.pdf.

[26] Object Management Group (OMG). Meta Object Facility (MOF) Core Specification, v2.0, 2006-01-01, http://www.omg.org/cgi-bin/doc?formal/2006-01-01

[27] Object Management Group (OMG). UML 2 Object Constraint Language (OCL), http://www.omg.org/cgi-bin/doc?ptc/2005-06-06

[28] Object Management Group (OMG). Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification Final Adopted Specification, ptc/05-11-01. http://www.omg.org/docs/ptc/05-11-01.pdf, November 2005.

[29] Object Management Group (OMG). Unified Modeling Language (UML): Superstructure, version 2.0. Specification, http://www.omg.org/cgi-bin/doc?formal/05-07-04.

[30] Object Management Group (OMG). XML Metadata Interchange (XMI), v2.1, 2005-09-01, http://www.omg.org/technology/documents/formal/xmi.htm

[31] Schwabe D.and Rossi G. Developing Hypermedia Applications using OOHDM. Workshop on Hypermedia Development Process, Methods and Models, Hypertext´98, Pittsburg, USA, 1998.

[32] Taenzer, G. AGG: A Graph Transformation Environment for System Modeling and Validation. Proc. Tool Exihibition at "Formal Methods 2003", Pisa, Italy, September 2003.

[33] Valderas P., Fons J. and Pelechano V. From Web Requirements to Navigational Design – A Transformational Approach. In Proc. 5th Int. Conf. on Web Engineering Engineering (ICWE 2005), Sydney, Australia, LNCS 3579, Springer, July 2005.

[34] Varró, D. and Pataricza A. Generic and Meta-transformations for Model Transformation Engineering. In Proc. 7th Int. Conference on the Unified Modeling Language (UML 2004), LNCS 3273, Springer, 2004.

[35] Vdovjak, R. and Houben G.J.A Model-Driven Approach for Designing Distributed Web Information Systems. In Proc. of 5th Int. Conference on Web Engineering (ICWE 2005), Sydney, Australia, LNCS 3579, Springer, July 2005.

[36] Zhang, G., Baumeister, H., Koch, N. and Knapp, A.. Aspect-Oriented Modeling of Access Control in Web Applications. In 6th Int. Workshop Aspect Oriented Modeling (AOM'05), Chicago, USA, 2005.