Classification of Model Transformation Techniques used in UML-based Web Engineering[1]

Nora Koch

Ludwig-Maximilians-Universität
Oettingenstr. 67, 80538 München, Germany
and
FAST GmbH
Arabellastr. 17, 81925 München, Germany
kochn@pst.ifi.lmu.de, koch@fast.de

ABSTRACT

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. Model-driven software development (MDD) approaches aim to reduce at least some of these problems providing techniques for the construction of models and the specification of transformation rules, tool support, and automatic generation of code and documentation. The method of resolution of MDD is to first build models, which are independent of the platform, transforming them in later stages to technological dependent models, and to achieve automatic model and code generation based on transformation rules. Web Engineering is a domain where MDD can be used to address evolution and adaptation of Web software to continuously emerging new platforms and changes in technologies. We present an overview of the development process of the UML-based Web Engineering (UWE) defined as an MDD approach. The main characteristic of UWE is the use of standards including the

[1] This article is an extended version of "Transformation Techniques in the Model-Driven Development Process of UWE" presented at the 2nd Workshop on Model-Driven Web Engineering (MDWE`06) at the 6th International Conference on Web Engineering (ICWE 2006).

UML, XMI as model exchange, MOF for metamodelling, MDA, and the transformation language QVT. In this article, we focus on the model transformation aspects of the UWE process.

Keywords

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

1. INTRODUCTION

Model-driven development (MDD) is an approach that promises to solve the main problems the development of software currently has: inefficiency and error-proneness. MDD is based on models like other software development approaches are, but introduces a higher level of abstraction by defining metamodels and model transformation rules [3]. A model is a simplified representation of a software system and is useful if it allows for a better understanding of the system. Models are built to offer different views of a same system. These views need to be refined and integrated and used to produce code, when possible in an automated way, i.e. with the help of transformation rules. Models are represented using a modelling language and following a set of well-formedness rules. These rules can be also represented as a model – a so called metamodel. Metamodels are a prerequisite for the execution of automatic model transformation [24]. The goal of MDD can be summarized as to provide better separation of concerns, automatic generation of models and code, and traceability between code and models.

Development of Web software is not an exception: most Web projects also have budget, time, and quality problems. There is an urgent need of techniques and tool support for automated generation of Web systems. 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. In addition, MDD advocates 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 modelling such systems were developed, for example Hera [8], OOHDM [31], OO-H [9], OOWS [33], UWE [18], WebML [4], and W2000 [1]. They focus on the specification of analysis and design models for Web systems, for instance on the construction of navigation and adaptation models. However, the model transformation aspects were neglected by most of these methods. The UML-based Web Engineering (UWE) is an approach that has continuously been adapted, on the one hand, to new features of Web systems, such as more transaction-based, personalized, context-dependent, and asynchronous applications. On the other hand, UWE evolves to incorporate the state of the art of software engineering techniques. Due to this evolution the software development process of UWE moved towards a model-driven development approach supporting among others model transformation languages, for example the Query-View-Transformation language (QVT), in order to improve design quality.

We present an overview of the complete MDD process of UWE and focus in this article 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. 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. More recently, we use emerging specification techniques like graph transformations and model transformation languages like the ATLAS Transformation Language (ATL) [12] or the Query-View-Transformation (QVT) approach [28].

In addition, we present a classification and comparison of model transformations used in the UWE process. For the classification we selected a set of criteria. These criteria are not UWE specific, thus they 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 in the Web Domain.

The best-known MDD realization is the Model-Driven Architecture (MDA) of the Object Management Group (OMG) [25]. The development process of UWE is based on MDA as well as other OMG standards, i.e. the Unified Modeling Language (UML) [29], XML Metadata Interchange (XMI) [30], Meta-Object Facility (MOF) [26], Object Constraint Language (OCL) [27]), and the forthcoming standard transformation language QVT ([28]).

The remainder of this paper is structured as follows: Sect. 2 introduces the UWE approach and its main characteristics focusing on the MDD process of UWE. Sect. 3 gives a brief description of the UWE models. Sect. 4 presents general criteria for the classification of model transformations, which are used in Sect.5 for an analysis of the model transformations applied in the model-driven process of UWE. Sect. 6 provides an overview of related work. Finally, in Sect. 7 we present some conclusions and outline future plans on the implementation of the UWE model transformations.

2. Model-Driven Development in UWE

The idea behind MDD is that modelling and transforming is a better foundation for the development and maintenance of systems than programming [23]. The primary goals of MDD are portability, interoperability, and reusability through 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 UML, QVT, and 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 UWE Approach

The UWE approach comprises a UML Profile for modelling Web systems, a process and tool support for the development of Web systems. For modelling with UWE and UWE CASE tool we refer the reader to [2], [6], [11], [14], [15], [17] and [18].

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 of models and model transformations, which specification is supported by metamodels and model transformation languages. The metamodels are the Web Requirements Engineering metamodel (WebRE) [6], the UWE metamodel [19], and the metamodel of the Web Software Architecture approach (WebSA) [22] containing elements for modelling requirements, structure and behaviour, and the architecture of Web systems, respectively.

2.2 UWE Process

The main characteristic of the UWE process is the systematic, semi-automatic, model-driven, and transformation-based (Sect. 5) support of the development of Web systems. The UWE process is depicted in Figure 1 as a stereotyped UML activity diagram ([22]). 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 1: Overview of the UWE Process

The process starts with the business model, which MDA calls computational independent model (CIM), defining a requirements model as shown in Figure 1. Platform independent design models (PIMs) are derived from these requirements. The set of functional models represents the different concerns of the Web application. It comprises the content, the navigation, the business logic, the presentation, and the adaptation of the Web system. These different functional models are not depicted in the overview shown in Figure 1 in order to avoid an overloaded diagram. For more details the reader are referred to Figure 11 and Figure 13 of Sect. 5.

Functional models are afterwards integrated into a big picture model mainly for the purpose of e.g. verification [17] (see Figure 14). A merge with architectural modelling 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 a UWE process is automatic model transformation in each step based on transformation rules.

2.3 Running Example

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

3. Models in UWE

A *model* of a system is a specification of that system and its environment. 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. The general 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.

MDA suggests building computational independent models (CIM), platform independent models (PIM), and platform specific models (PSM) corresponding to a business, a design, and an implementation viewpoint, respectively [25]. We will use these types to classify the models UWE propose to build within the scope of the model-driven process.

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.

UWE models are represented by UML diagrams. Whenever appropriate UWE uses the 'pure' UML notation. For modelling specific features of the Web domain, such as navigation nodes and Web pages UWE provides a domain specific UML profile, which is defined using the extension mechanisms provided by the UML: stereotypes and OCL constraints. For further details on the UWE profile refer to [2], [6], [15], [17], [18] and [20].

3.1 Requirements Model

The overall objective of modelling 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 2: Use case diagram of music portal example (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. At least the latter 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 model elements with Web domain specific semantics. Figure 2 depicts the use case diagram for the music portal and Figure 3 shows the activity diagram for the *Download album use* case.

Figure 3: Activity diagram for a (simplified) use case Download Album (CIM)

3.2 Functional 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 [18]. 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 model elements, for example 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 4 and Figure 5 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 4: Content and user model of the music portal example (PIM)

Figure 5: Navigation model (simplified) of the music portal example (PIM)

Process models are visualized as UML 2.0 activity diagrams (see Figure 6). Actions (like *FindUser*) model the actions the user and the system must carry out to complete the business process.

Figure 6: Business process Login (PIM)

UWE proposes to build a presentation model to sketch the layout of the Web application. It uses the UML composition notation for classes that is containment represented by graphical nesting of the symbols. This kind of representation is appropriate for modelling 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 modelling (AOM), extending the UML with concepts such as aspect, pointcut and advice to support AOM [1]. For further details to presentation and adaptation models see [2].

Figure 7: States Home and Song of the ‘big picture’ (PIM)

UWE proposes the generation of an integrated model that merges the separate concerns of a Web system into a big picture (see Figure 1). Using UML state machines as the results of the integration process offers the possibility of applying formal techniques for validation, like model checking [16]. Figure 8 shows the states *Home* and *Song*, both states are part of the ‘Big Picture’ model.

3.3 Architecture and Implementation Models

Information on architectural styles can be merged at different steps in the MDD process of UWE. Following the Web Software Architecture (WebSA) approach [22] we propose to integrate functional and architecture models in a very early development phase. Such an approach is shown in the UWE process overview depicted in Figure 1. Architecture models in the WebSA approach are specified as platform independent models (PIMs). Knapp and Zhang suggest in [17] to merge architecture models with the big picture model, that is 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. Classification Criteria for Model Transformations

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 mappings and relations are defined as specializations of transformations. 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.

Model transformation languages are used to specify model transformations. They are defined at metamodel level that is they specify how certain types of source metamodel elements are converted to another type of the target metamodel. They are applied at model level to transform elements of the source model to elements of a target model, such as represented in the pattern of Bezivin [3] shown in Figure 8.

Figure 8: Model transformation pattern [3]

We distinguish the following aspects of model transformations: MDA type (based on the MDA type of the models involved), complexity, use of marks, level of automation, language diversity, and implementation technique. Mens and Van Gorp [24] and Czarnecki and Helsen [5] use similar criteria for the analysis of existing and proposed model transformation approaches. Our classification criteria were selected for the evaluation of the model transformations of the UWE process presented in Sect. 5. They could also be used to analyze other Web Engineering approaches allowing for a comparative analysis of such approaches.

4.1 MDA Transformation Types

In a model-driven development process two types of model transformations can be identified: model-to-model transformations and model-to-code transformations. Model-to-model transformations can further be classified within the MDA scope into CIM to CIM; CIM to PIM and PIM to PIM. Computational independent models are transformed into platform independent models and both can be refined, that means a CIM can be mapped to another CIM, in the same way that a PIM can be refined into another PIM model. PSM to code transformations belong to the model-to-code category. Note that transformations from PIM to CIM, PSM to CIM, and PSM to PIM are not possible.

Transformations can be also classified into horizontal versus vertical transformations [24]. Horizontal transformations are differentiated from vertical transformation based on whether the source and target models reside at the same abstraction level or at a different level. A typical example for a horizontal transformation is refactoring and examples for vertical transformations are refinement and code generation. All transformations described in the UWE process are vertical transformations with exception of the creation of the big picture, and the integration of the functional and architecture models.

4.2 Complexity of Transformations

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 simple or a merge. Complexity is instead defined on basis of code and tool complexity in [24] where small transformations and heavy-duty transformations are distinguished.

4.3 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, that is 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 model elements. Other marks instead are only required for the mappings. They do not need to be integrated in the source model [23], 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 information that is more detailed in order to guide the transformation.

Other 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 may be specified.

4.4 Levels of Automation

We distinguish between automatic, semi-automatic, and manual model transformations based on the amount of manual intervention a model transformation requires. 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.

4.5 Language Diversity

Models are expressed in some modelling languages, for example UML for design models and programming languages for source code models. The model elements and partially their semantics are represented by the corresponding metamodel. A distinction can be made between endogenous and exogenous transformations. Endogenous transformations are transformations between models expressed in the same modelling language; exogenous are expressed using different languages. If a transformation is a merge, more than two different languages can be involved.

4.6 Implementation Techniques

Transformation rules can be implemented in (1) general programming language as Java, C#, that is rules are hard coded in specific tools, or (2) graph transformation languages as AGG [32] and VIATRA [34], or (3) languages for transformations such as ATL [12] and QVT [28]. Transformations are often based on invariants, pre-conditions, and post-conditions specified in languages such as OCL [27]. Models serialized using XMI can be transformed using XSLT [36].

5. 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 [22] that are MOF-compliant metamodels.

Transformations are classified into three groups: those used to build the functional 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. 4.

Characteristics *Transformation*	MDA Type	Comple-xity	Marks	Level of Automation	Language Diversity	Implementation Technique
Requirements2 Content	CIM to PIM	Simple	WebRE profile	Automatic	UML	QVT
Requirements2 Architecture	CIM to PIM	Simple	-	Manual	UML	-
Requirements2 Navigation	CIM to PIM	Merge	WebRE profile	Automatic	UML	QVT
Requirements2Process	CIM to PIM	Simple	WebRE profile	Semi-automatic	UML	QVT
Content2Navigation	PIM to PIM	Simple	UWE profile & navigation relevance	Semi-automatic	UML	Java (OCL), ATL
NavigationRefinement	PIM to PIM	Simple	UWE profile & patterns	Automatic	UML	Java (OCL)
Process2Navigation	PIM to PIM	Simple	UWE profile	Semi-automatic	UML	Java (OCL)
BusinessLogic2Navigation	CIM to PIM	Merge	WebRE profile	Automatic	UML	Java (OCL)
Navigation2Presentation	PIM to PIM	Simple	UWE profile	Automatic	UML	Java (OCL), ATL
StyleAdjustment	PIM to PIM	Merge	Style guide	Automatic	UML, template-based	Java
Functionality2 BigPicture	PIM to PIM	Merge	Patterns and marks	Automatic	UML	Graph transformations
Architecture Integration	PIM to PIM	Merge	UWE & WebSA profile	Automatic	UML	QVT
Integration2J2EE	PIM to PSM	Merge	Patterns	Automatic	UML, Java	QVT, ATL

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

5.1 Building Functional Models

The first model transformation step of the UWE process consists of the mapping of the Web requirements models to the UWE functional models [20]. The design models are the content, navigation, process, presentation, and adaptation model. There exists a set of dependencies among these functional models, that allow for creation of other models or refinement of models.

Figure 9: Model transformation pattern for metamodels WebRE and UWE

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 8 is applied to the UWE process using the standard Query View Transformation Language (QVT, [28]).

In the UWE process the transformation *Requirements2Content* allows for the construction of the content model; the transformations C*ontent2Navigation, Requirements2Navigation* and *NavigationRefinement* are used to build the navigation model. *Requirements2Process* consists of a refinement of the workflows describing the functionality of the Web system. The presentation model is built in at least two iterations: it is created with the former *Navigation2Presentation* and refined by *StyleAdjustments*. Finally, the adaptation model can also be extracted from the functional requirements models, and the architecture models from the non-functional requirements.

5.2 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 modelling Web systems requirements, objects are used to indicate the need to include certain content information in the Web application (Figure 11).

We use the QVT language to specify the transformation from elements of the requirements model to elements of the content model (*Requirements2Content*). 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 10. The application of this transformation rule to the content elements *Account* of the activity diagram of the music portal example (see Figure 3) generates the class *Account* of the content model of this Web system (see Figure 4). For further details, refer to [20]. The characteristics of the *Requirements2Content* 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); }
      }
 }
```

Figure 10: Transformation requirements elements to content elements (QVT textual notation)

5.3 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.

5.4 Transforming Content to Navigation

In UWE a first navigation model (see Figure 11) is generated based on classes of the content model marked as navigation relevant, that is the transformation *Content2Navigation* is applied to the selected model elements. From one content model different navigation views can be obtained, such as for different stakeholders of the Web system like anonymous user, registered user and administrator [18]. For example, in the music portal, a transformation will generate a navigation class *Album* based on the classes *Album* of the content model (Figure 4 and Figure 5).

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 11: Transformations to build functional from requirements models

5.5 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, for example 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 12 shows the *Search2Query* transformation specified in the QVT graphical notation [20].

The transformation *Requirements2Navigation* is a merge and is based on the WebRE profile (see Table 1). Figure 11 shows that the transformation rule *Requirements2Navigatio*n has to be applied after the transformation rule *Content2Navigatio*n, but there is no restriction related to the order in which the *Requirements2Navigatio*n rule and the *NavigationRefinement* has to be applied.

5.6 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 [11]. The following constrains (informally described) define 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, for example *IndexAlbums* in the navigation model of the music portal (Figure 5).
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, for example *MainMenu* (Figure 5).

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.

5.7 Adding Business Logic to Navigation

The business logic described in the activity diagrams of the requirements model is included in the navigation and process model. For example, the *Download Album* activity of the requirements model (see Figure 3) is transformed to a process class that is included as navigation node in the navigation model of the music portal. The designer takes the decision which navigation classes this node will be related with through corresponding process links. The transformation rule *Process2Navigation* (see Figure 11) is implemented in Java in the CASE tool ArgoUWE.

Figure 12: Search2Query transformation (QVT graphical notation)

5.8 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 13). 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 13: Transformations to build presentation model

ArgoUWE implements the *Navigation2Presentation* rule in Java. *Style2Adjustment* rules are planned to be implemented by the time this work was written. Table 1 characterizes both, the *Navigation2 Presentation* and the *Style2Adjustment.*

5.9 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.

5.10 Building the 'Big Picture'

Though from different viewpoints, the different functional models represent the Web application as a whole. They are integrated into another platform independent model that we call the big picture (Figure 14). 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 [37] and adaptation [2]. 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 14: Transformations to build "big picture" model

Figure 15: Mapping navigation node to state in "big picture" (graph transformation)

The transformation *Functional2BigPicture* forms 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 15. 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 *Functional2 BigPicture* are outlined in Table 1.

5.11 Integration of 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 1. 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 [22]). The characteristics of the rules of type *IntegratingArchitectural Features* are outlined in Table 1.

5.12 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 is implicitly contained in the transformation. For the generation of platform specific models mappings from UWE functional models (PIMs) to PSMs for Web applications (see Figure 1) were defined. 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) [12], QVT-P and QVT [28]. For example, the transformation depicted in Figure 16 tackles the generation of J2EE elements from Server Pages of the Integration Model. The rule is written in QVT-P language.

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

Another example is shown in Figure 17. The ATL code exemplifies a transformation rule that maps the element *Anchor* of the UWE integration model to a JSP element. This element anchor is incorporated in the presentation model based on the existence of a link in the navigation model. 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 17: Generation of JSP elements based on the integration model (ATL language)

6. 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 methods.

WebSA is an approach that focuses on architectural models and transformations specified in a QVT like language called UML Profile for Transformations (UPT) [21]. UPT is a graphical transformation language. UPT-tool is a transformation engine that translates UML source models in UML target models and is implemented as a Web application. The architecture models are partially integrated in the UWE process [22]. Baresi and Mainetti [1] propose to use transformation techniques for the verification of correctness and adaptability of functional models developed by W2000. The approach is based on a work on graph transformations [10]. OOWS [33] uses graph transformations to automate its CIM to PIM transformation.

WebML follows an MDD approach for mapping model elements of WebML to architecture components of MVC2, which can be transformed into components for different platforms [4]. OO-H [9] supports a transformation-based construction of a presentation model based on model 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 [7] and Varró and Pataricza [34] are interesting although they do not consider Web domain specific characteristics but define a generic approach with focus on formal definition of transformation semantics.

Czarnecki and Helsen present in [5] a classification of model transformations focusing on the characteristics of existing and proposed model transformation languages. They analyze properties of transformation languages, such as. type of transformation rules, rule application strategy and rule organization. Another taxonomy of model transformations is presented by Mens and Van Gorp [24]. They also discuss the commonalities and differences between existing model transformation approaches providing MDD developers with criteria for the selection of model transformation. Our characterization is on the one hand based on these works, but on the other hand focuses on Web engineering model-driven processes and the transformations defined in these particular processes.

7. CONCLUDING REMARKS

We presented the development process of UWE (UML-based Web Engineering) defined as a model-driven development approach. We outlined and classified the models and model transformations specified in the UWE process. Models are classified – according to the MDA characterization – in computational independent models (CIM), platform independent models (PIM), and platform specific models (PSM). In addition, we classified the model transformations defined in UWE focusing on a classification in terms of type, complexity, number of source models, involvement of marking models, implementation techniques and execution type.

The main contribution of this work is the specification of the classification criteria for model transformations in the Web engineering domain based mainly on the approaches of Mellor et al. [23] and Czarnecki et al. [5] and Mens et al. [24]. The value of such a set of criteria is the applicability to other model-driven methods to classify their model transformations. Thus, and even more important, it would allow for a comparative analysis of the model transformation aspect of the different development processes in the Web Engineering field. We validate the proposed classification criteria applying it to the UML-based Web Engineering approach.

8. FUTURE WORK

Of general interest for the Web Engineering community would be the analysis of the model transformations used in each model-driven approach. For this analysis and with comparative effects, the classification criteria defined in this work can be applied.

Within the scope of improvements planned for UWE, we aim to provide a more uniform set of model transformations. Therefore we plan to redefine those that are hard coded in the CASE tool using specification techniques for the transformations like ATL, QVT or graph transformations. We would therefore benefit from model 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. Our future work will focus on the most promising and adequate approaches, mainly 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. In addition, our aim is to validate our approach with further case studies and to use the research results for the automatic generation of test cases.

ACKNOWLEDGMENTS

We would like to thank the Web Engineering team of the LMU for fruitful discussions on the draft version of this article. We also thank the organizers, the program committee and participants of the Model-Driven Web Engineering (MDWE`06) Workshop at ICWE 2006 for their feedback, which allowed for the improvement of the paper presented at the workshop . This research has been partially supported by the project MAEWA "Model Driven Development of Web Applications" (WI841/7-1) of the Deutsche Forschungsgemeinschaft (DFG), Germany and the EC 6th Framework project SENSORIA "Software Engineering for Service-Oriented Overlay Computers" (IST 016004).

REFERENCES

[1] 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

[2] 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.

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

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

[5] Czarnecki K. and Helsen S., Classification of Model Transformation Approaches. OOSPLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture, Anaheim, USA, 2003.

[6] Escalona, M. J. and Koch, N. Metamodeling Requirements of Web Systems. In Proc. 2nd Int. Conf. on Web Informa-tion System and Technologies (WEBIST'06), Portugal, April 2006.

[7] 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.

[8] 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.

[9] 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.

[10] Heckel R. and Lohmann M. Model-based development of Web applications using graphical reaction rules. In Proc. Fundamental Approaches to Software Engineering, Warsaw, Polen, LNCS 2621, Springer, April 2003.

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

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

[13] Koch, N. Transformation Techniques in the Model-Driven Development Process of UWE. In Proc. of the 2nd Workshop on Model-Driven Web Engineering (MDWE´06), ACM Portal, Vol. 155, Palo Alto, USA, July 2006.

[14] Knapp A., Koch N., Moser F. and Zhang G. ArgoUWE: A CASE Tool for Web Applications. In Proc. of 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. Conf. 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. and Kraus, A. The expressive Power of UML-based Web Engineering. 2nd Int. Workshop on Web-oriented Software Technology (IWWOST02). Málaga, Spain. June, 2002.

[19] 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.

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

[21] Melía S., Gómez J. and Serrano J.L. UPT: A Graphical Transformation Language based on a UML Profile, In European Workshop on Milestones, Models and Mappings for Model-Driven Architecture (3M4MDA) at ECMDA 2006, Bilbao, Spain, July 2006.

[22] 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.

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

[24] Mens, T. and Van Gorp, P. A Taxonomy of Model Transformation. In Proc. of Int. Workshop on Graph and Model Transformation, 2005 (GraMoT 2005). ftp://ftp.umh.ac.be/pub/ftp_infofs/2005/GraMOT-taxonomy.pdf

[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. Conf. 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. Conf. on Web Engineering (ICWE 2005), Sydney, Australia, LNCS 3579, Springer, July 2005.

[36] W3C, XSL Transformations (XSLT) http://www.w3.org/TR/xslt, June 2006.

[37] 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.

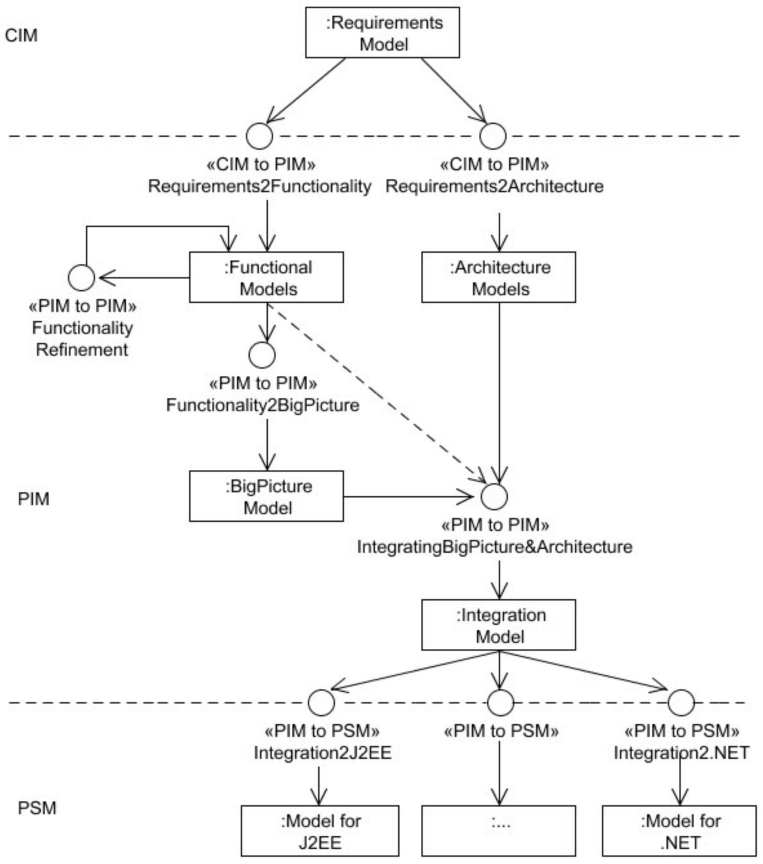
CIM
:Requirements Model
«CIM to PIM» Requirements2Functionality
«CIM to PIM» Requirements2Architecture
:Functional Models
«PIM to PIM» Functionality Refinement
:Architecture Models
«PIM to PIM» Functionality2BigPicture
PIM
:BigPicture Model
«PIM to PIM» IntegratingBigPicture&Architecture
:Integration Model
«PIM to PSM» Integration2J2EE
«PIM to PSM»
«PIM to PSM» Integration2.NET
PSM
:Model for J2EE
:...
:Model for .NET

Figure 18: Overview of the UWE Process

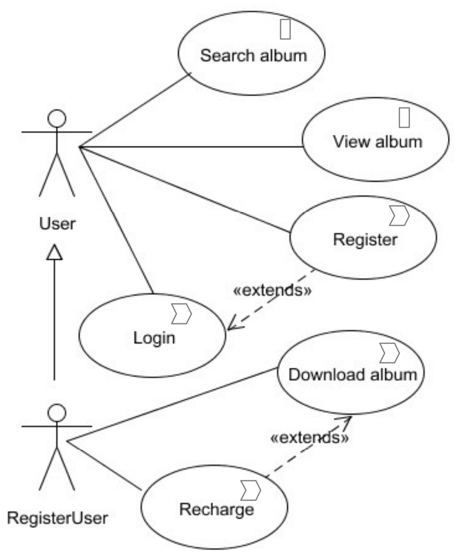

Figure 19: Use case diagram of music portal example (CIM)

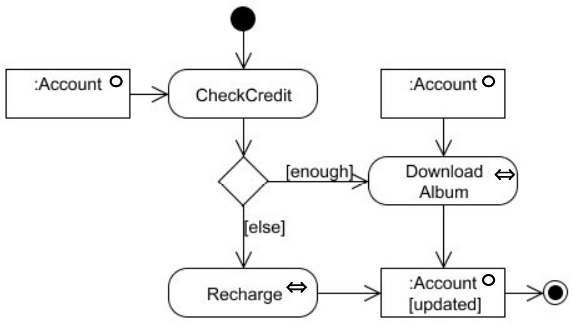

Figure 20: Activity diagram for a (simplified) use case Download Album (CIM)

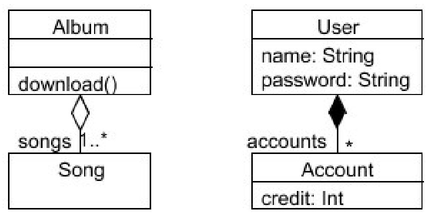

Figure 21: Content and user model of the music portal example (PIM)

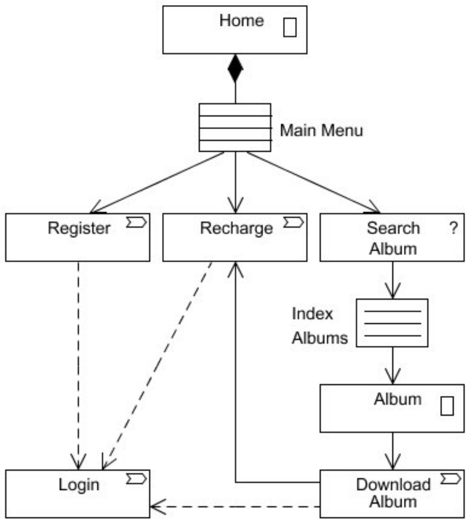

Figure 22: Navigation model (simplified) of the music portal example (PIM)

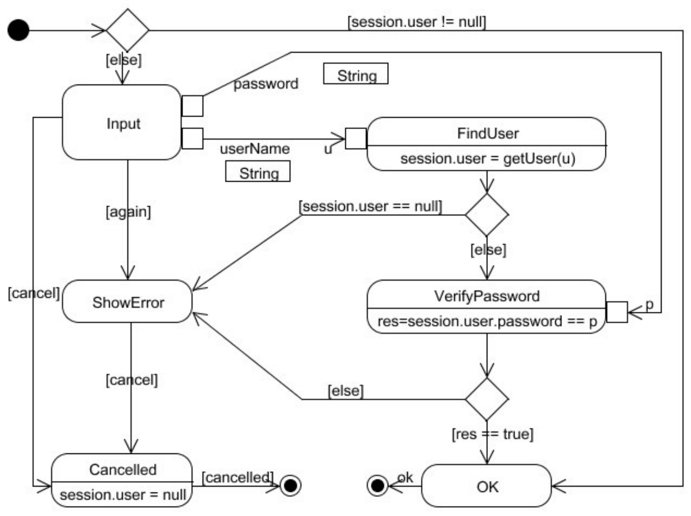

Figure 23: Business process Login (PIM)

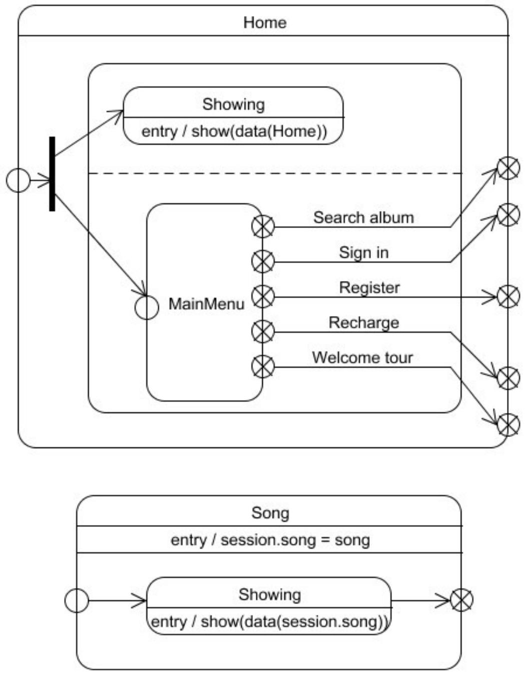

Figure 24: States Home and Song of the ‘big picture’ (PIM)

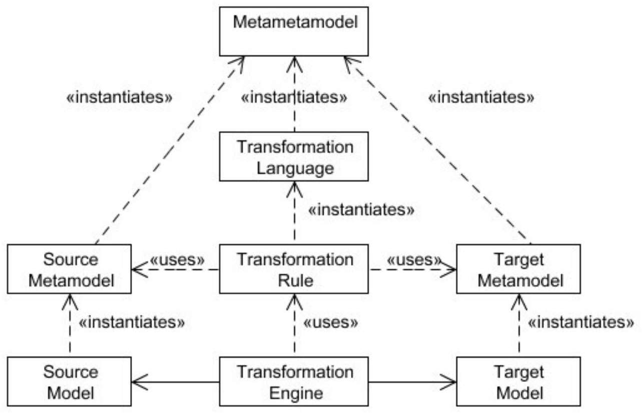

Figure 25: Model transformation pattern [3]

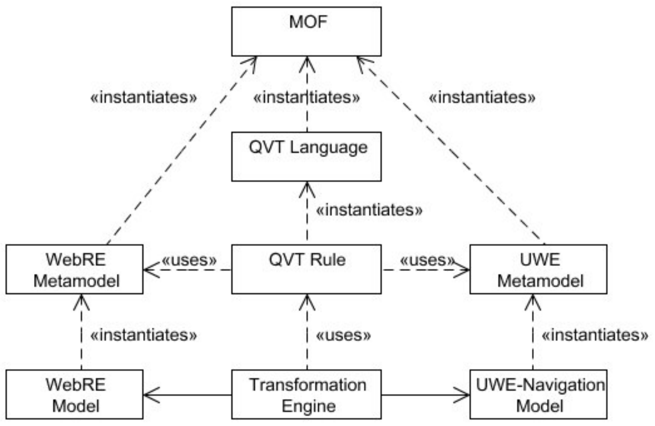

Figure 26: Model transformation pattern for metamodels WebRE and UWE

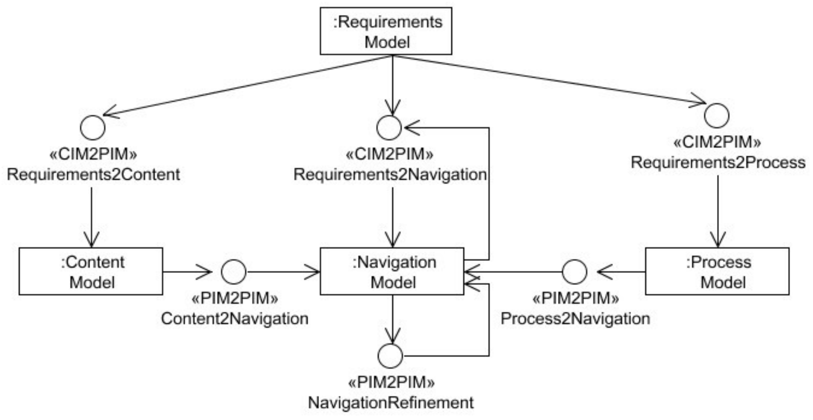
:Requirements Model
«CIM2PIM» Requirements2Content
«CIM2PIM» Requirements2Navigation
«CIM2PIM» Requirements2Process
:Content Model
:Navigation Model
:Process Model
«PIM2PIM» Content2Navigation
«PIM2PIM» Process2Navigation
«PIM2PIM» NavigationRefinement

Figure 27: Transformations to build functional from requirements models

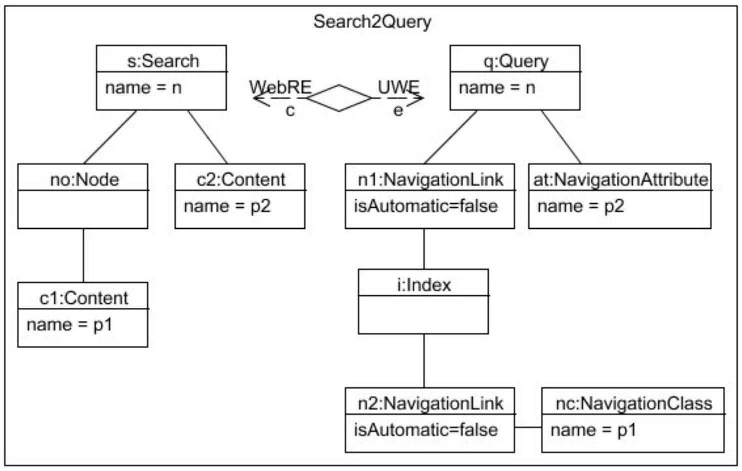

Figure 28: Search2Query transformation (QVT graphical notation)

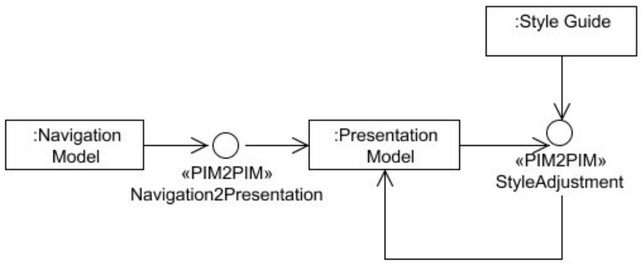

Figure 29: Transformations to build presentation model

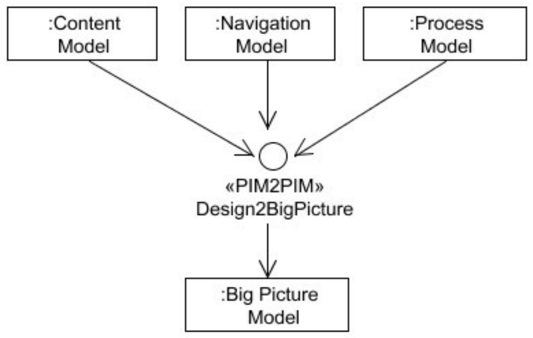

Figure 30: Transformations to build “big picture” model

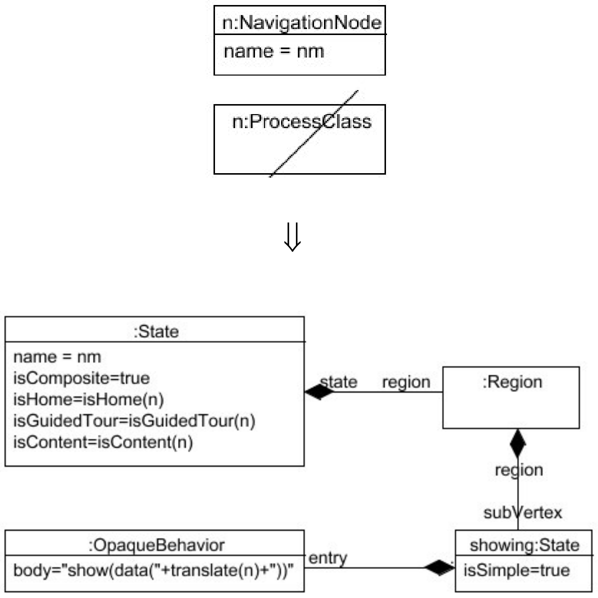

Figure 31: Mapping navigation node to state in “big picture” (graph transformation)