Model Transformations from Requirements to Web System Design

Nora Koch
Ludwig-Maximilians-Universität
Oettingenstr. 67
80538 München, Germany
kochn@pst.ifi.lmu.de

Gefei Zhang
Ludwig-Maximilians-Universität
Oettingenstr. 67
80538 München, Germany
zhangg@pst.ifi.lmu.de

María José Escalona
University of Seville
Av. Reina Mercedes, S/N
41015 Sevilla, Spain
mjescalona@us.es

ABSTRACT

Requirements models are used to specify system functionalities from the customer viewpoint and are the starting point of software development. However, most Web engineering approaches do not provide a systematic method to build design models from requirements specification. We propose an approach using model transformations to close this gap. Our transformation rules are defined in the QVT language – a forthcoming OMG standard, which makes automatic model generation possible. This way design is kept consistent with the customer requirements.

Keywords

Web Engineering, Requirements Engineering, Model-Driven Development, Metamodeling, Model Transformations, UML, QVT.

1. INTRODUCTION

The focus of software development is changing from code writing to the specification of models and model transformations. In the beginning, models were designed for a better understanding of the systems to build and improve developer's communication. Then modeling techniques emerged that make the process of code writing easier, and even allow partial code generation. Currently we are moving towards model-driven processes, whose goal is the development of software at a higher level of abstraction based on models and model transformations.

In principle, model-driven development (MDD) starts at the computation independent level (CIM) with a business model of the requirements on the system. Then this model is transformed into platform independent design models (PIMs), which are used to generate platform specific models (PSMs) and, finally, code. In practice, however, most MDD approaches concentrate on transformations from PIM to PSMs and PSM to code. On the other hand, the relevance of business models is well known in the development of successful Web software systems [12]. Empirical studies demonstrate that efforts invested in a detailed business modeling to capture the customer requirements on the system to built considerably reduce drawbacks in later phases of the development [20].

We focus on an early step of model-driven development: transformations from requirement models to design models. The input for our transformations may be any requirements model of a Web system under construction defined as an instance of the Web requirements metamodel [6]. We use models specified with Navigation Development Technique (NDT, [5]) and UML-based Web Engineering (UWE, [10]) to illustrate our approach. The targets of our transformations are the models that describe the concerns of Web systems – content, navigation and presentation.

Transformations rules are defined as mappings from the Web requirements metamodel to the UWE metamodel. These rules are specified in the forthcoming standard Query View Transformation Language (QVT, [17]). The automatic execution of the rules would be straightforward with appropriate tool support. Such tools are still under development.

The remainder of this paper is structured as follows: Section 2 gives an overview of the role of MDD in the Web domain. Section 3 presents the Web requirements and the UML-based Web engineering metamodels that are the source and target for the transformation rules defined in Section 4. Section 5 provides an overview of related work. Finally, in Section 6 some conclusions and future work are outlined.

2. MODEL-DRIVEN DEVELOPMENT IN WEB ENGINEERING

Model Driven Development (MDD) is becoming a widely accepted approach in different domains of Software Engineering. The basic idea of MDD is to separate the platform independent design and the platform specific implementation of applications, delaying as much as possible the construction of models related to specific technologies. Web Engineering is a concrete domain where MDD may be helpful [11], particularly in addressing the problems of ever-emerging platforms and changing technologies. The Model Driven Architecture (MDA, [15]) of the OMG offers

suitable principles to define model-driven approaches using standard notations. Following [14], Figure 1 shows a possible adaptation of MDA principles to the Web development visualized as a stereotyped UML activity diagram. Models are depicted as objects, and transformations are represented with stereotyped activities (special circular icon).

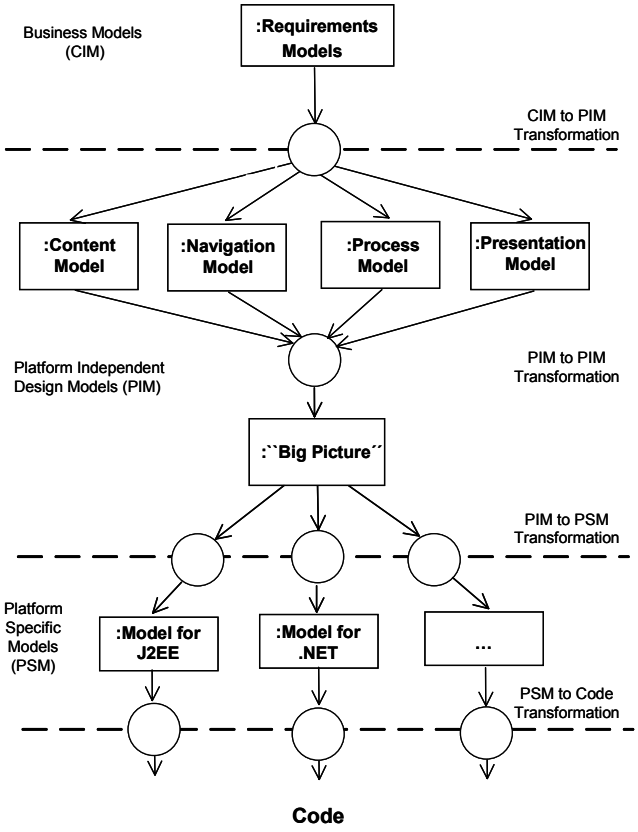

Figure 1: Model-driven approach for Web systems

The process we propose starts with the business model (CIM) level defining requirements models. Platform independent analysis models (PIMs) are derived from the requirements –often based on additional information. On the PIM level, the separate concerns of Web applications: the content, the navigation, the business processes, and the presentation are designed in separate models. These models are integrated into a so-called "big picture" which merges all the concerns together and is used in validation of the design models [9]. Finally, the platform specific models (PSMs) are derived from this validation model, from which program code can be generated. The aim of such an MDD process is automatic model transformation in each step based on rules defined at metamodel level.

Some of the steps of this process have been realized (see [7] [14][1][5]). In the following, we describe our method of obtaining a first draft of the design models from the requirements.

3. METAMODELS AND UML PROFILES

We define the necessary concepts of Web requirements and the different concerns of Web systems at metamodel level. The metamodels are MOF [15] compliant and "profileable" [1], which means that they can be mapped using the extension mechanisms of the UML into a so-called UML profile.

3.1 Metamodel for Web Requirements

Escalona and Koch [6] summarize the concepts used in modeling Web system requirements in the metamodel for Web Requirements Engineering (WebRE). The WebRE metamodel is depicted in Figure 2. Instances of this metamodel are used in several Web Engineering methods for requirements specification, although they do not always use the same terminology and notation.

A *WebUser* is any user who interacts with a Web System and may be either registered or not. The basic use case type is *Navigation*, which comprises a set of browse actions that the Web user performs to reach a target node. Browse is the action of following a link and is represented by an instance of the metaclass *Browse*. The special browse action *Search* models a query that the Web user makes to the Web system. A special kind of the Navigation use case is *WebProcess*, which includes user transactions like checkout or providing credit card data.

The source and the target of browse actions are nodes. *Nodes* are units of information in Web systems. A node is associated to one or more pages, and a page may be associated to one or more nodes (e.g. in case of asynchronous communication). The concept of page is represented by the *WebUI* metaclass. Besides, a node can show different pieces of information. Each piece of information of a Web system is represented as an instance of the metaclass *Content*.

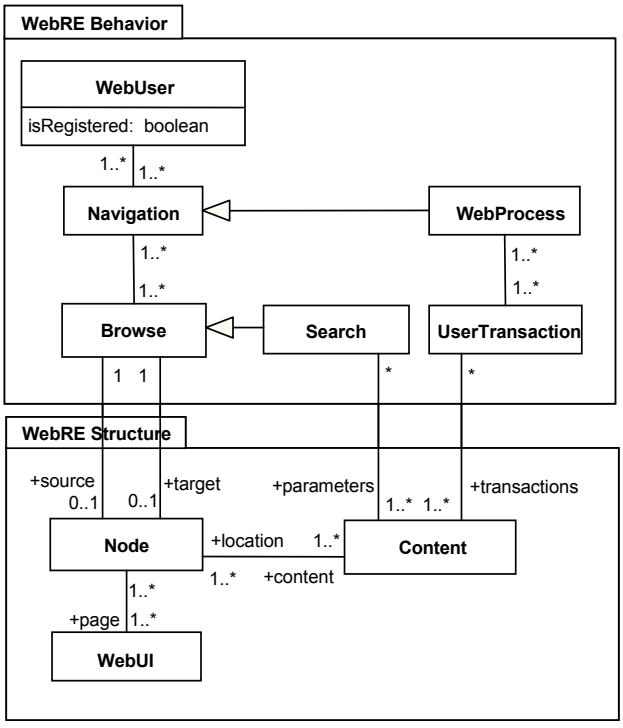

Figure 2: Web requirements metamodel

3.2 UML Profile for WebRE

The WebRE metamodel is the basis for providing a specific and intuitive notation for modeling Web requirements The notation is defined using the standard extension mechanisms of the UML resulting in a so-called UML profile. The elements of the profile are defined as extensions of UML metaclasses. Since the semantics of UML elements is not changed, WebRE is a conservative extension of the UML. Figure 3 shows how the WebRE elements are mapped into the UML by stereotypes.

Table 1 shows the icons defined for the WebRE stereotypes [6]. Note that we do not define a specific icon for Web User. Although not depicted in Table 1, the profile also contains an extension of the UML metaclass ObjectNode for Node, Content and WebUI to be used in activity diagrams.

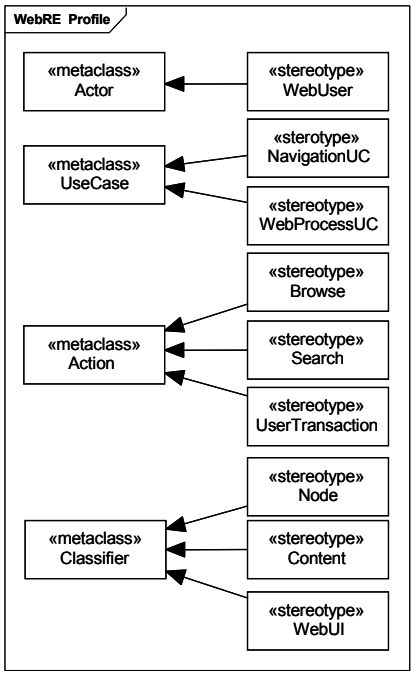

Figure 3: Stereotypes of the WebRE profile

Metaclass	Stereotype	Icon
UseCase	«navigation»	
UseCase	«Web process»	
Action	«browse»	
Action	«search»	
Action	«user transaction»	
Classifier	«node»	
Classifier	«content»	
Classifier	«webUI»	

Table 1: Icons for WebRE stereotypes

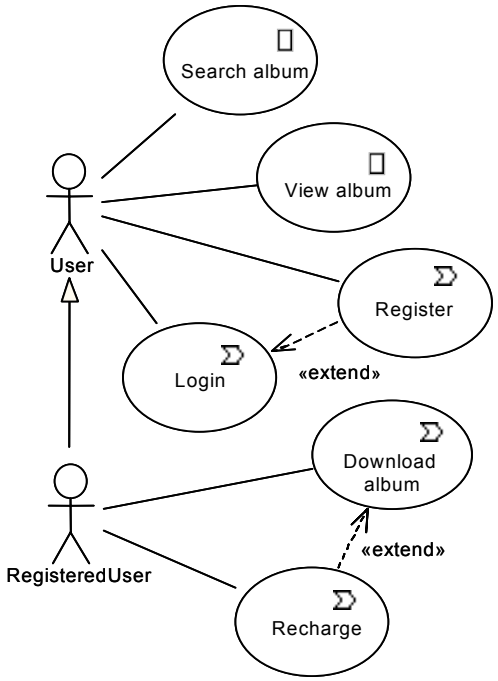

Figure 4: UML use case diagram for the music portal

As an example[1], consider a music portal that provides albums to download. While the standing data such as singer, publisher, etc. is public to everyone, download is reserved for registered users and is only enabled as long as the user has enough credit on his prepaid account. Figure 4 gives a use case diagram of the music portal. The use cases can be refined with additional details. Figure 5 shows an activity diagram that refines the use case *Download album* with more detail.

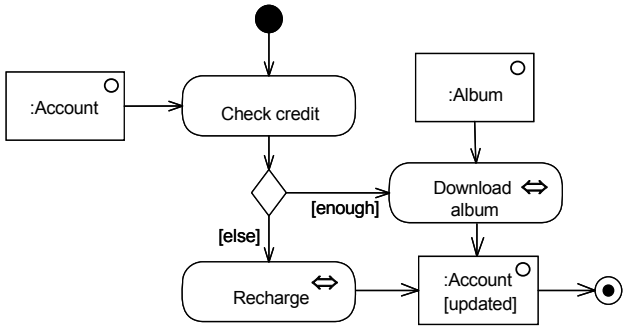

Figure 5: UWE activity diagram for *Download album* use case

An NDT pattern representing this use case is given in Table 2. From the detailed description it can be concluded that

- This is a WebProcess use case;
- The user must be logged in and thus registered to perform it;

[1] This example is inspired by http://www.mp3.com

- The target of this action is a node, whose content includes an album and an account.

Note that the object nodes in the activity diagram are marked with the icons of the corresponding stereotyped classifier of the profile. For more details of mapping between WebRE metaclasses and use case description see [6].

FR-02	Download album	
Description	The system actions when the user selects the link "download album"	
Preconditions	This use case is only enabled when the use case "RF-03.View album" was executed	
Actors	AC-01. RegisteredUser	
Normal sequence	**Step**	**Action**
	1	The user selects to download the album
	2	The system checks his/her credit
	3	The system downloads the album
Exceptions	**Step**	**Action**
	3	If there is not enough credit, the system leads the user to the "RF-04.Recharge"
Comments	This use case has two activities of type UserTransaction	

Table 2: NDT pattern for use case *Download album* of the music portal example

Although both methods NDT and UWE use the same modeling concepts, they use a completely different representation for these concepts. The metamodel instead offers an abstract common base for both approaches.

3.3 Metamodel for the Design Phase

After the requirements of a Web application are laid down with requirements models, its design is performed in platform independent models, where the content, the navigation structure, the business processes and the presentation are defined on an abstract level without considering technical details of implementations. The concepts required for modeling are defined in the UWE metamodel [10]. This metamodel includes a package for each of these concerns and is defined as a conservative extension of the UML metamodel [16]. See Figure 6 for the package navigation. The metamodel is complemented with well-formedness rules formulated in the Object Constraint Language (OCL). Thus, this metamodel is an instance of the MOF metamodel.

The content of a Web system is modeled in a content model built with UML class diagrams and "pure" UML modeling elements. In the navigation model, navigable nodes are represented by instances of subclasses of *Node*, which is derived from the UML metaclass *Class*. Direct links between navigation nodes are modeled by instances of *NavigationLink,* a subclass of the UML metaclass *Association*. There are several kinds of nodes defined: navigation classes represents the navegable information units of the Web application; menus model the common starting point of alternative links leaving a node; access primitives are used to represent special constructs in Web navigation: indexes, queries and guided tours.

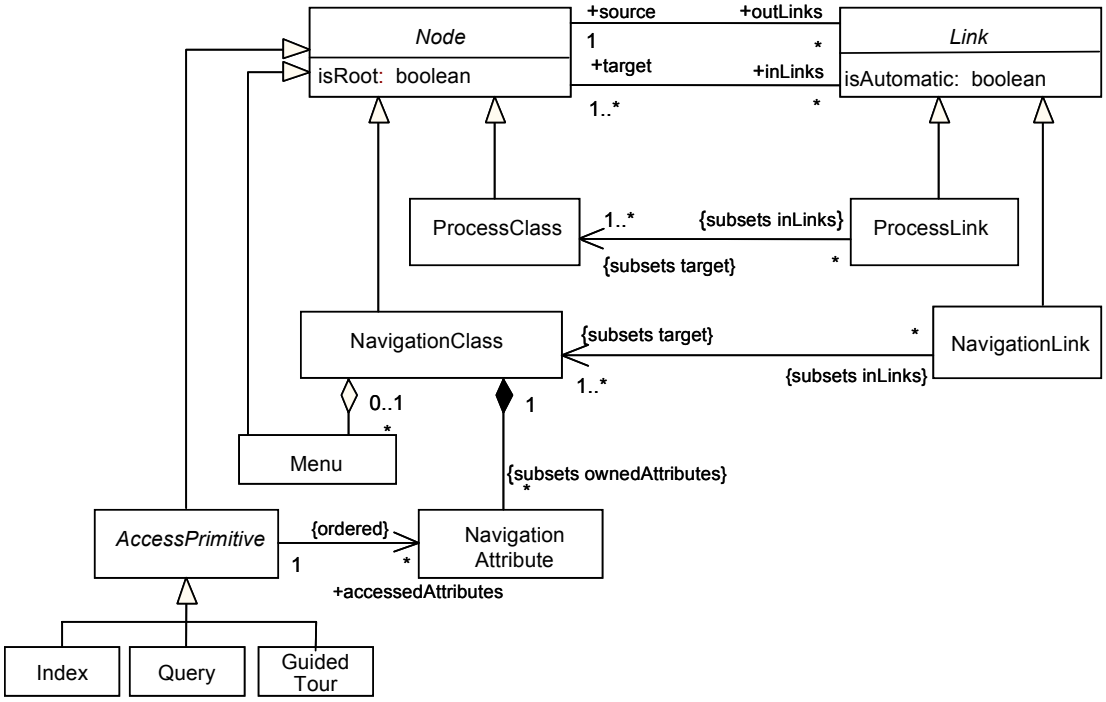

Figure 6: UWE metamodel: Navigation package

The attributes of access primitives are used in indexes to represent how the target instances should be indexed and to specify the search criteria in queries. The business processes of Web systems are visualized in process models. Processes are integrated into the navigation model by process classes that represent the process entry points, and process links that represent the navigation paths leading from and to them. For more details, see [10].

3.4 UWE Profile

The elements of the UWE metamodel are mapped to a UML profile using stereotyped classes. Table 3 shows the mapping rules and the icons defined for the non-abstract classes of the navigation model and the process model. For navigation attribute, navigation link and process link there are no icons defined.

Metaclass	Stereotype	Icon
GuidedTour	«guided tour»	
Index	«index»	
Menu	«menu»	
NavigationAttribute	«navigation attribute»	
NavigationClass	«navigation class»	
NavigationLink	«navigation link»	
ProcessClass	«process class»	
ProcessLink	«process link»	
Query	«query»	

Table 3: Elements of the UWE profile

Figure 7 shows the navigation diagram for the music portal example. From the *MainMenu* the user can register, search for an album to download, or recharge his prepaid account. The dependency relationships show that the user must first login before he can perform the processes *Recharge* and *DownloadAlbum*.

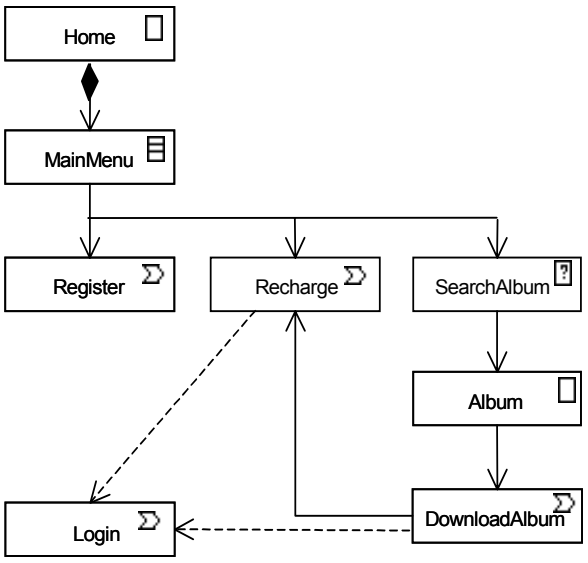

Figure 7: UWE navigation model for the music portal example

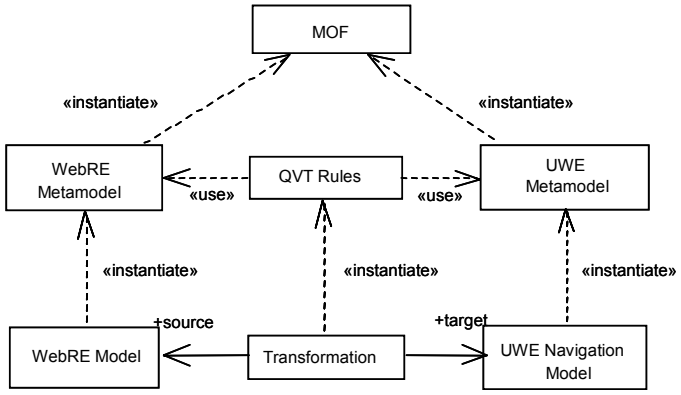

Figure 8: WebRE model transformation pattern

4. WebRE TRANSFORMATION PROCESS

Based on the two metamodels given in Section 3, we define our approach of deriving draft design models of a Web system from their requirements. The approach is based on metamodel mappings, i.e. transformations rules are defined to map UWE metamodel elements from the WebRE metamodel elements. The transformation implements the MDA model transformation pattern of Bézivin [2] as shown in Figure 8. Both metamodels are specified using the MOF language, which is also an OMG standard [16].

Figure 8 shows how a UWE navigation model is derived from a WebRE model by means of the metamodel-based transformations. Note that the navigation model generated is a first draft as its completion may require additional information, partially depending on the developer's decisions. For more details on further enhancements of the generated navigation model see [10].

Transformation rules are defined in the QVT language [17], which is a forthcoming OMG standard and based on the MOF metamodel. In QVT a transformation is defined as a list of relations. A relation defines a mapping from the source domain (*checkonly*) onto the target domain (*enforce*). New elements are created in the target domain, if necessary, in order to hold the relation. The QVT language comprises a textual and a graphical notation for simple transformations. Both notations can be used to declaratively define transformations without specifying how a transformation is actually executed. For more complex transformations the additional use of OCL 2.0 expressions is recommended. QVT itself has a MOF metamodel fitting in the central concept of MDA, which stresses that transformations themselves are models.

In the following subsections we present some transformation rules, which are analyzed in detail and applied to our music portal example.

4.1 From Requirements to Content Model

In Web requirements models use cases are refined by activity diagrams containing activity nodes of type *Browse*, *Search* or *UserTransaction.* Actions may be related to objects that are either required as input or produced as results. These objects are included in activity diagrams by means of object flows. In the special case of modeling Web systems, objects are used to model source and target of the navigation, parameters needed for searches or information that is modified due to transactions performed by the user. Such parameters or transaction data are modeled by objects of *Content*.

Our first transformation maps the instances of the metaclass *Content* of the requirements model into classes in the UWE content model. The target class must have the same name and the same properties as the source content. The transformation rule *Content2Class* is specified in the QVT language. We choose the textual form of QVT as shown in Figure 9. Applying this rule to the use case *Download album* of our music portal example, the classes *Account* and *Album* are created.

```
transformation Content2Class (webre:WebRE, uwe:UWE) {
    top relation R1 {
          n: String;
          checkonly domain webre c:Content { name = n };
          enforce domain uwe cc: Class { name = n };
    }
    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 9: Transformation rule Content2Class

4.2 From Requirements to Navigation Model

Not only the content model can be derived from the requirements, but the information provided by the action of the requirements models, i.e. activities of type *Browse*, *Search* and *User Transaction* is also used in further transformations to generate navigation classes or access structures of the navigation model. Each *Browse* action implies the existence of a navigation class and a link to this target in the navigation model. The rule transformation *Browse2NavClass&Link* shown in Figure 10 specifies this mapping in QVT textual notation.

```
transformation Browse2NavClass&Link (webre: WebRE, uwe: UWE) {
    top relation R3{
          c: String;
          checkonly domain webre b:Browse {target = no:Node
                {content = c}   };
          enforce domain uwe nc: NavigationClass {};
          enforce domain uwe link: Link { isAutomatic = false,
                target = nc; };
          where { R4 (c,nc); }
    }
    relation R4 {
          n: String;
          checkonly domain webre c: Content {name = n };
          enforce domain uwe nc: NavigationClass {name = n };
    }
}
```

Figure 10: Transformation rule Browse2NavClass&Link

Applying this rule to the use case *Download album* of our music portal example the result is the following: For the browse action based on the use case *View album* a navigation class *Album* and a navigation link pointed to the navigation class are generated.

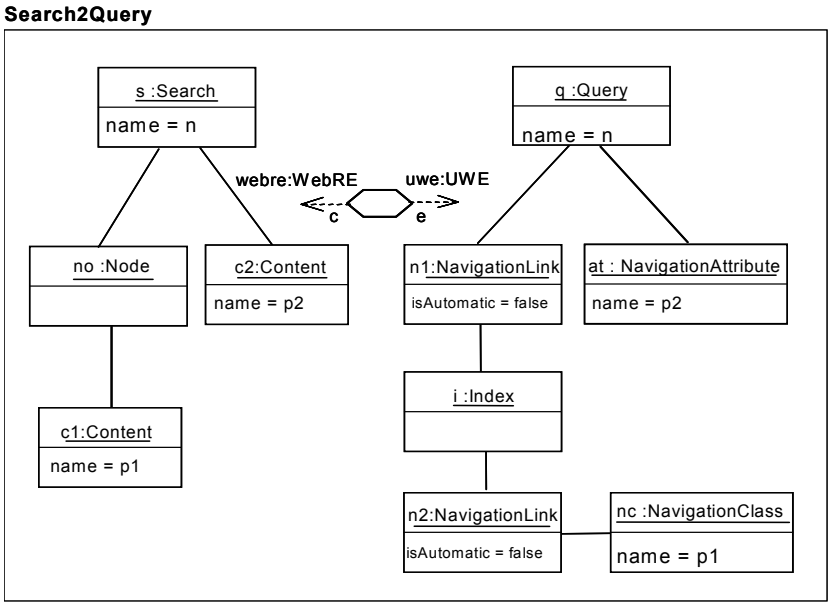

Figure 11: Transformation rule Search2Query

Each *Search* action in the requirements model is transformed into a *Query,* an *Index*, a *NavigationClass* and two *Link*s relating these three elements. The navigation class represents in the navigation model the results of the query. The parameters of the search are transformed to attributes of the query. The rule transformation Search2Query is formally defined in QVT as shown in Figure 11.

Declarative definition of the transformation rules is the basis for the QVT operational specification, in which e.g. the need of an index instance depending on the number of outcomes of the query can be considered.

The music portal example includes the «navigation*n*» use case *SearchAlbum.* It is associated with a «search» action of the same name, and objects of class *Album.* Using the transformation rule *Search2Query*, the navigation model will be enforced to include a Query named *SearchAlbum* in the navigation model, which will point with a link to an *Index* of *Album*. Each index item is a link to one instance album that matches the query.

Another transformation rule that we named *UserTransaction2-Process* derives UWE process elements from the requirements model. The source of the transformation is a «*user transaction*» action together with a related instance of the class *Content* in the requirements model. The instance of the class *Content* indicates the transactional data affected by the user transaction. The target of the transformation is instances of classes belonging to the UWE process package, i.e. *ProcessLink* and *ProcessClass*. This rule is shown in Figure 12 using the graphical notation of the QVT language.

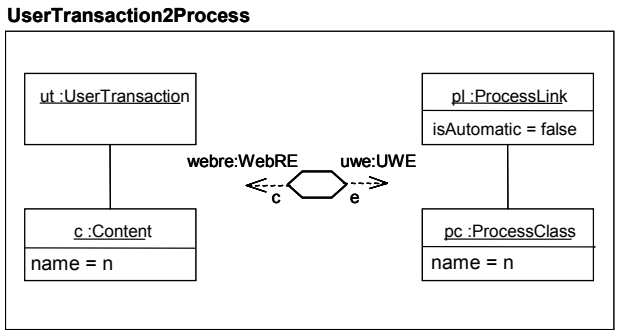

Figure 12: Transformation rule UserTransaction2Process

Finally, for all activities related to use cases that only can be triggered by a registered user, i.e., where the attribute *isRegistered* is true, there must exist a dependency relationship in the navigation model between its corresponding process class and the "Login" process class. Examples are the activities of downloading an album or recharging the account in the music portal. The rule for creating such dependency relations is given in Figure 13.

```
transformation RegisteredUser2Dependency (webre: WebRE, uwe:
UWE){
    top relation R8 {
        checkonly domain webre n: Navigation
                {webUser = w: WebUser{isRegistered = true}};
        enforce domain uwe p: ProcessClass {};
        enforce domain uwe d: Dependency {
              supplier = ps: ProcessClass {name = "Login"},
              client = p};
    }
}
```

Figure 13: Transformation rule RegisteredUser2Dependency

5. RELATED WORK

Model-driven development is applied successfully by several Web engineering methods. In OO-H, WebML, OOHDMDA etc. models are used to separate the platform independent design of Web systems from the platform dependent implementations as much as possible [7], [18], [13]. However, these methods do not include support for obtaining design models from requirement specifications.

OOWS [21] is to the authors' knowledge the only Web engineering method that provides automatic generation of navigation models from requirements. Our approach is novel in that it is underlined by a well-defined requirement metamodel, defined as a UML profile, and that we also generate elements of the content model from the requirements. Another feature that distinguishes our approach is the use of the transformation language QVT.

6. CONCLUSIONS AND FUTURE WORK

Defining transformation rules at metamodel level we achieve a model driven development approach. We present such transformations rules for an early phase in the development life cycle of Web system, which is the basis for an automated generation of design models from requirements specification.

The source modeling elements for our transformations are instances of any requirements model of a Web system that is built with modeling elements of the Web requirements metamodel [6]. For this work, we use models specified with NDT and UWE to illustrate our approach. The targets of our transformations are the models that describe any of the concerns of Web systems – content, navigation and presentation. We choose the UWE models, for illustration purpose, as transformation target. The transformation rules are specified in the forthcoming standard Query View Transformation Language (QVT, [17]).

The success of an MDD approach strongly depends on the tool support. We are looking forward to tools supporting the QVT transformation language. In the meantime we are gathering experience with other transformation languages and techniques, such as ATL and graph transformations.

Currently, we are analyzing the possibilities to extend the open source CASE tools ArgoUWE[2] and NDT-Tool to support the set of transformation rules defined so far. Further, we plan to define transformations for the automatic generation of test cases also based on requirements specified as WebRE models.

7. ACKNOWLEDGMENTS

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).

We thank Alexander Knapp, Andreas Kraus, Santiago Meliá and Andreas Schroeder for their helpful comments on a draft version of this paper.

8. REFERENCES

[1] Baresi, L., Garzotto, F., Mainetti, L., Paolini, P. Meta-modeling Techniques Meet Web Application Design Tools. In Proc. 5th International Conference on Fundamental Approaches to Software Engineering (FASE 2002), 294-307, Grenoble, France, 2002.

[2] Bézivin, J. In Search of a Basic Principle for Model Driven Engineering , Novática No.1, 21-24, 2004.

[3] Ceri, S. Fraternali, P., Bongio, A., Brambilla M., Comai S., Matera M. Designing Data-Intensive Web Applications. Morgan Kaufman. 2003.

[4] Cengarle, M. V., Knapp A. OCL 1.4/1.5 vs. OCL 2.0 Expressions. Formal Semantics and Expressiveness. Software and System Modeling, Vol. 3 No. 1, 9-30, 2004.

[5] Escalona, M.J., Mejías, M., Torres, J. Developing System with NDT & NDT-Tool. In Proc. 13th International Conference on Information System Development (ISD 2004), 149-159, Lithuania, 2004.

[6] Escalona, M.J., Koch, N. Metamodeling Requirements of Web Systems. In Proc. International Conference on Web Information System and Technologies (WEBIST 2006), INSTICC, 310-317, Setúbal, Portugal. 2006.

[7] Gómez, J., Cachero, C. OO-H: Extending UML to Model Web Interfaces. Information Modeling for Internet Applications. Idea Group Publishing, 2002.

[8] Knapp A., Koch N., Zhang, G., Hassler, H.-M.Modeling Business Processes in Web Applications with ArgoUWE. In Proc. 7th International Conference on the Unified Modeling Language (UML 2004). LNCS 3273, 69-83, Lisbon, Portugal, 2004.

[9] Knapp, A., Zhang, G. Model Transformations for Integrating and Validating Web Application Models. In Proc. Modellierung 2006 (MOD 2006), Vol. P-82, Lect. Notes in Informatics, 115-128, Innsbruck, Austria, 2006.

[10] Koch, N., Kraus, A. The expressive Power of UML-based Web Engineering. In Proc. 2nd International Workshop on Web-oriented Software Technology (IWWOST 2002), 105-119, Málaga, Spain, 2002.

[11] Koch, N., Vallecillo, A., Rossi, G. (Eds.) In Proc. Workshop on Model-Driven Web Engineering (MDWE 2005), Sydney, Australia, 2005.

[12] Lowe, D., Eklund, J. Client Needs and the Design Process in Web Projects. Journal on Web Engineering. Vol. 1, No. 1, 23–36, 2002.

[13] Matera, M., Maurino, A., Ceri, S., Fraternali, P. Model-Driven Design of Collaborative Web Applications. Software--Practice and Experience. 33(8), 701-732. 2003.

[14] Meliá, S., Kraus, A., Koch, N. MDA Transformations Applied to Web Application Development. In Proc.5th International Conference on Web Engineering (ICWE 2005), LNCS 3579, 465-471, Sydney, Australia, 2005.

[15] OMG: MDA Guide, http://www.omg.org/docs/omg/03-06-01.pdf. Version 1.0.1, 2003.

[16] OMG. Unified Modeling Language: Superstructure, Version 2.0. Specification, Object Management Group. http://www.omg.org/cgi-bin/doc?formal/05-07-04, 2005.

[17] Query QVT-Merge Group, Revised submission for MOF 2.0 Query/View/ Transformation RFP. Object Management Group, http://www.omg.org/cgi-bin/apps/doc?ad/04-04-01.pdf, 2004.

[18] Schmid, H.A., Donnerhak, O. OOHDMDA-An MDA Approach for OOHDM. In Proc. 5th International Conference on Web Engineering (ICWE 2005). LNCS 3579, 569-574, Sydney, Australia, 2005.

[19] Schwabe, D., Rossi,G. An Object Oriented Approach to Web-Based Application Design. Theory and Practice of Object Systems 4(4). Wiley and Sons, USA, 1998.

[20] Sommerville, I., Ransom, J. An Empirical Study of Industrial Requirements Engineering Process Assessment and Improvement. ACM TOSEM, Vol. 14, No. 1, 85-117, 2005.

[21] Valderas, P., Fons, J., Pelechano, V. From Web Requirements to Navigational Design - A Transformational Approach. In Proc. 5th International Conference on Web Engineering (ICWE 2005). LNCS 3579, 506-511, Sydney, Australia, 2005.

[2] ArgoUWE: http://www.pst.ifi.lmu.de/porjekte/argouwe.shtml