Requirements Models as First Class Entities in Model-Driven Web Engineering[*]

Nora Koch[1,2] and Sergej Kozuruba[1]

[1] Ludwig-Maximilians-Universität München, Germany
[2] NTT DATA

Abstract. The relevance of a detailed and precise specification of the requirements is well known; it helps to achieve an agreement with the customer on software functionality, user friendliness and priorities in the development process. However, in practice, modeling of requirements is avoided in many projects, in particular in the Web domain, mainly due to short time-to-market. The objective of this work is to make requirements modeling more attractive providing a win-win situation. On the one hand such models are used to improve the developer-customer communication and on the other hand to generate draft design models, which can be used in further steps of a model-driven development approach, and therefore reduce the developers' efforts. We concretize the approach presenting a domain specific modeling language defined as an extension of the UML-based Web Engineering (UWE) profile and a set of model transformations defined to generate the content, navigation and presentation models of web applications. A social network application is used to illustrate UWE requirements and design models.

1 Introduction

The first steps in a software development project comprise the elicitation, specification and validation of requirements of the new web system to be built. This is also valid in reengineering projects. Mainly elicitation but also the other two activities require intensive communication with the customer in order to reach an agreement on functionality, technologies and priorities. For the specification, different techniques, methods and tools have been developed, such as building models of the application. The more accurate the models produced in this early phase of the software development life cycle (SDLC), the less error-prone the code of the software. This relationship between the quality of the requirements specification and the implemented system has been analyzed and confirmed several times [9]. However, more often than not, only sketches of models are produced and the implementation phase is started too early. Even if requirements are specified, they are often partially ignored by developers. Generally, the time invested in the requirements specification is seen as partially wasted.

In this work we focus on web software and show how to move developers' efforts from the design to the requirements phase of the SDLC. This objective is achieved

[*] This work has been partially sponsored by the EU project ASCENS, FP7 257414, the EU-NoE project NESSoS, GA 256980, and the DFG project MAEWA II, WI 841/7-2.

through two changes in the development process: (1) Building annotated requirements models (more effort) and (2) Generating the design models from the requirements models in a semi-automatic way (less effort). Note that our goal is not the automatic generation of the complete application (proved to also have severe limitations), but instead to use model-driven engineering (MDE) to ease developers' work in several steps, to manage the complexity of web applications, and scalability aspects in the development. The benefits are better requirements models improving the communication between customers and developers, supporting the decision process and resulting in more stable web applications. In our previous work [5] we presented the results of a detailed assessment of the application of our approach showing that the effort reduction reached is calculated to be between 26% and 77%.

We present a domain specific modeling language (DSML), which provides the annotations needed to enrich standard requirements models with web features and to reduce model-complexity. The models specified with the DSML are used in the model transformations of the MDE process to generate the design models of the web applications. Although the approach is generic and the idea of a DSML in an early development phase of web applications could be applied for any model-driven approach, we selected the UML-based Web Engineering (UWE) [6] to illustrate the approach, and as an example we use a social media application, called *Linkbook*.

The remainder of this paper is structured as follows: Section 2 gives an overview of UWE modeling features focusing on requirements models. In Sect. 3 we present the model-to-model transformations showing how draft models are generated. Section 4 describes the tools that support the model-driven process. Finally, Sect. 5 discusses related work and in Sect. 6 we give an outlook on future steps in the use of our model-driven development approach.

2 Modeling Requirements of Web Applications

The specification of web applications focus on building a model of the functional requirements. For the modeling task, different languages can be used, such as BPMN [10], UML [11] or a DSML, such as the Navigation Development Technique (NDT) [3] or the UML-based Web Engineering (UWE) [6] approach. We selected the latter to exemplify our approach.

UWE comprises a notation, a method and tool support. The notation is defined as a UML profile [11], i.e. using the extension mechanism provided by the UML itself, which allows for the refinement of UML in a strictly additive manner by stereotypes, tag definitions and constraints, providing the required additional annotations. The cornerstones of the UWE method are the principle of separation of concerns and a model-driven approach. As UWE tool we use the MagicUWE plugin implemented for MagicDraw (see Sect. 4) and the UWE4JSF eclipse plugin for the code generation.

Case Study. To illustrate the modeling features of the DSML and the results of the model transformations, we selected the *Linkbook* rich internet application. This is a social network platform to share favorite web pages with friends, similarly to other social networks that enable the sharing of posts or pictures. The network distinguishes between two kinds of users, guests and registered users, and provides the usual functionality for logging in and out, as well as for registering. The homepage of the application shows a list of favorites website entries (referred to in the diagrams as link infos) grouped by categories, and offers search facilities over the available link info and the user comments. Registered users also have the option to comment link infos and switch to their personal view where they can add new entries as well as sort or remove entries from the list of favorites. The network functionality is provided by managing the list of friends providing access to the list of favorites of all friends.

Fig. 1 depicts a subset of the use cases of the *Linkbook* application. It illustrates the use of the UWE profile to annotate UML model elements supplying them with specific web semantics, e.g. distinguishing between ≪browsing≫ (☐) and ≪processing≫ (Σ) use cases. The former represents pure navigation; the latter, workflow functionality. Examples for these two types of use cases – browsing and processing – are BrowsingLinkInfo and AddFriend, respectively. We introduce groupings of functionality using UML packages, for example, the packages Authentication and New. All of the model elements contained in the package adopt the stereotype of the package, which is then the only one that needs to be made explicit.

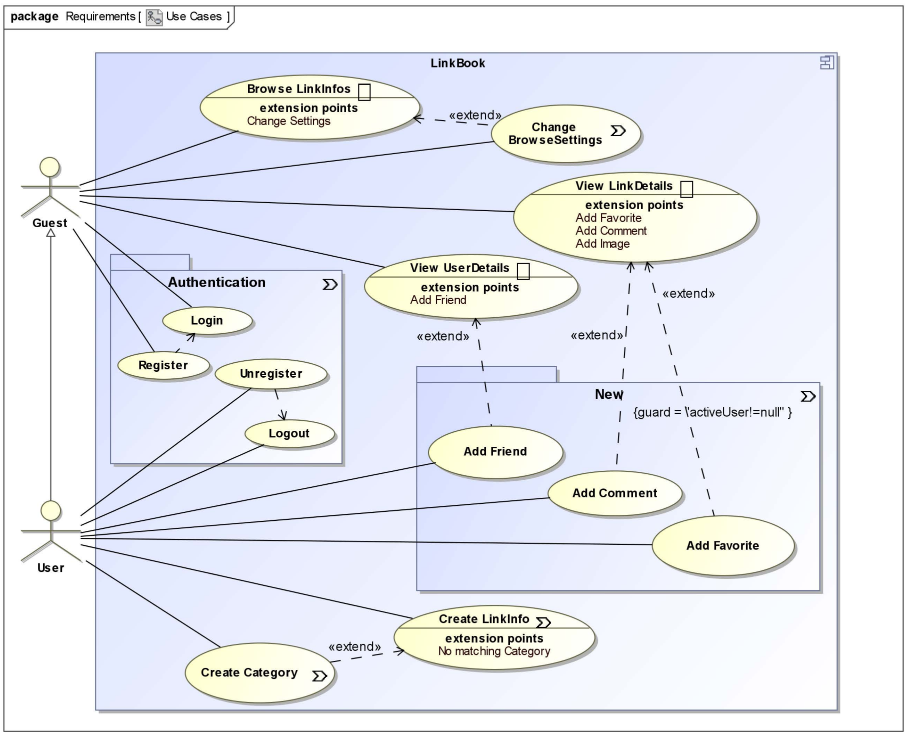

Fig. 1. *Linkbook*: Functional requirements modeled with use cases (excerpt)

Each use case can be refined by a detailed description or a graphical representation of the workflow associated to it. UML activity diagrams can be used for the visual representation as shown in Fig. 2 for the CreateLinkInfo workflow. Here we also use different stereotypes of the UWE profile to enrich the semantic of the activity diagram with web specific concepts. The objective is to specify:

- the actions which are part of the workflow, i.e. ShowForm, EnterData, sub-workflow CreateCategory and SaveLinInfo in our example;
- input and output information, given by pins (like name, address, description, category) or objects (linkinfo);
- decisions (not present in this example);
- features regarding the richness of the user interface, like the tag live validation for the input fields name and address;
- kind of visualization, e.g. the tag lightbox for the action ShowForm; and
- type of user-system interaction, indicating additional semantics such as validation and confirmation of the user input by the tag validated for EnterData and the tag confirmed for SaveLinkInfo.

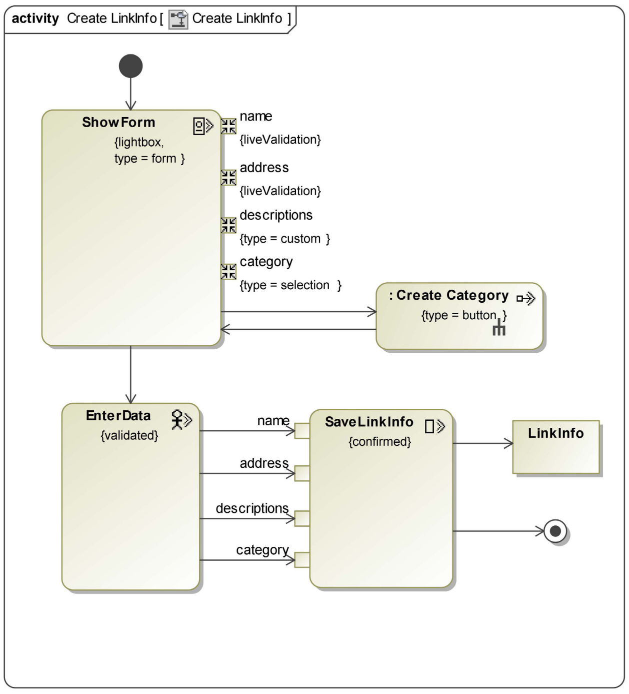

Fig. 2. *Linkbook*: Example of workflow represented as UML activity diagram

This CreateLinkInfo workflow depicts three different stereotyped actions: (1) ≪display-Action≫ (), used to visualize explicit presentation of elements; (2) ≪userAction≫ ()

that defines a point in the process flow when the user is asked to input data; and the (3) ≪systemAction≫ (▯») that indicates a step of the process flow where the application is processing some data.

3 Transformations and Model Generation

In the previous section we described the source models of the transformations, i.e. the requirements models. In this section we present the target, i.e. the design models and the model transformations that generate design models from requirements specification.

Our modeling approach for the design phase follows the principle of "separation of concerns" building separate models for views of the navigation, content, presentation, processes, etc. in the same way other web development methods do, such as OOHDM [14], OOHRIA [8], OOWS [15] and WebML [2], among others. The set of model types is a highly flexible and modular modeling framework providing the basis for the model-driven engineering (MDE) development process. Each model type has clear aims:

Content. The content model represents the domain concepts and the relationships between them.

Navigation. The navigation model is used to represent navigable nodes and the links between nodes.

Presentation. The presentation model provides an abstract view on the user interface (UI) of a web application. It is a platform-independent specification without considering concrete aspects like colors, fonts, and position of UI elements.

Process. The process model visualizes the workflows of the processes which are invoked from certain navigation nodes.

Our MDE approach comprise the generation of draft models of each concern, i.e. initial versions that require further refinement. In the following paragraphs we sketches the main modeling elements for the main concerns and gives an overview to the model transformations (informal description).

Content models are represented in UWE as plain UML class diagrams (see Fig. 3). A first draft of a content model is obtained by a set of model-to-model transformations using as source models the use cases and the corresponding workflows (graphically represented as activity diagrams). These transformations are:

- Objects nodes that model the data used in the workflows are translated into content classes using the name of the object note as the class name.
- If an action pin is connected to an object node directly or through an action, then it can be assumed that this pin represents a property of the class modeled by the object node. In that way, the name of the pin is used to determine whether an attribute or association is created by comparing the name with existing content classes.

Figure 3 shows the content model of the *Linkbook* web application that results from the above described transformations applied to the uses cases of Fig. 1.

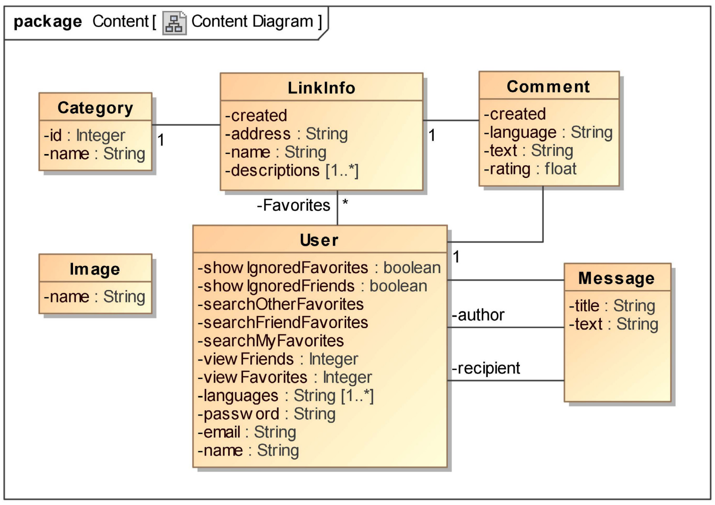

Fig. 3. *Linkbook*: Generated content model

Navigation models describe the navigation structure of a web application using a set of stereotyped classes defined for the web domain, such as navigation classes and links, menus, etc. Figure 4 depicts a first approach to the *Linkbook* navigation model which was generated based on the requirements models.

The following is a very brief overview of some modeling elements part of the UWE profile. A ≪navigationClass≫ (visualized as □) represents a navigable node of the hypertext structure; a ≪navigationLink≫ shows a direct link between navigation classes. Alternative navigation paths are handled by ≪menu≫ (▤) and the so-called access primitives are used to reach multiple instances of a navigation class (≪index≫, ≡), or to select items (≪query≫, ⍰). Web applications frequently support business logic as well. An entry and/or exit points of the business processes is modeled by a ≪processClass≫ (Σ) in the navigation model, the linkage between each other and to the navigation classes is modeled by a ≪processLink≫.

The model transformations from requirements (use cases and workflows) to the navigation structure model encompass the following steps:

- Creation of ≪navigationClass≫es for ≪browsing≫ use cases; ≪processing≫ use cases are transformed into ≪processClass≫es.
- Tagged values of the use cases are transformed into equally named tags of the generated classes.
- Relationships between use cases are translated into associations between created navigation and process classes. The associations are stereotyped with ≪processLink≫ if at least one related class is a ≪processClass≫ and ≪navigationLink≫ otherwise.

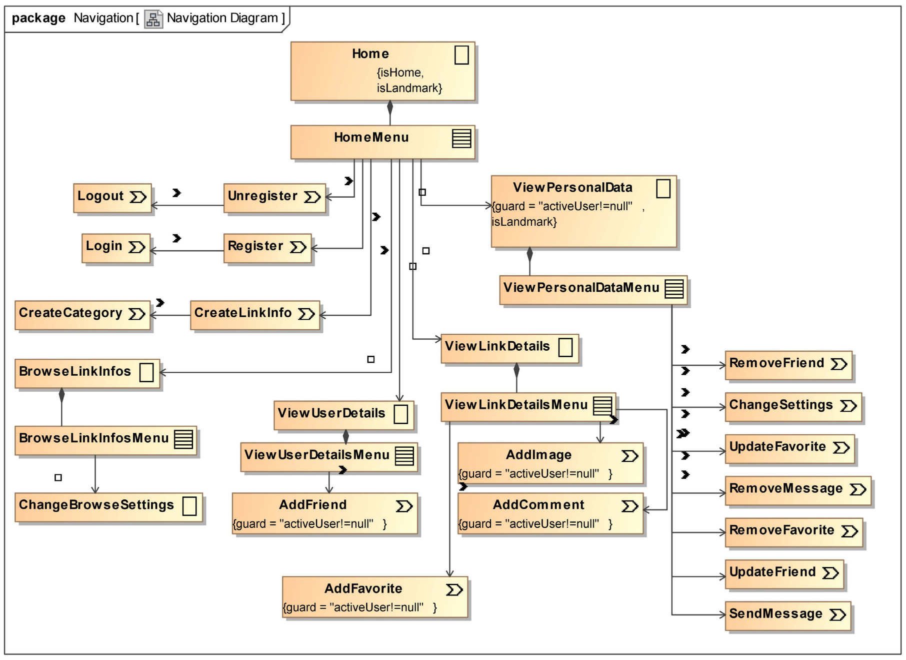

Fig. 4. *Linkbook*: Generated navigation model

– A ≪menu≫ is introduced whenever a navigation class has several outgoing links. The source of the links is changed to the ≪menu≫, which is connected to the navigation class by a composition.
– A navigation class can be created to serve as home of the application, if it has not been modeled explicitly.

In addition, each process class included in the navigation specification can be modeled as a detailed workflow in the form of a UML activity diagram (not included in this work). It is the result of a refinement process that starts from the workflow of the requirements model.

Presentation models are designed based on the information provided by the navigation models and the information available in workflows of the requirements models, e.g. rich UI features. A UML nested class diagram is selected as visualization technique. The presentation model describes the basic structure of the user interface, i.e., which UI elements (e.g. text, images, anchors, forms) are used to represent the navigation nodes.

The basic presentation modeling elements are the ≪presentationGroup≫ which are directly based on nodes from the navigation model, i.e. navigation classes, menus, access primitives, and process classes. A presentation group (▣) or a ≪form≫ (⊟) are used to include a set of other UI elements, like ≪text≫ (≋), ≪textInput≫ (▭), ≪button≫ (●), ≪selection≫ (↖), etc.

The top level elements of the presentation model are classes with the stereotype ≪presentationGroup≫. The second level of presentation elements consists of input and output elements. The presentation model similarly to the navigation model requires a main class, which is not modeled explicitly during the requirements specification. This presentation group is named ≪Home≫ and contains all presentation groups created from use cases inside a class ≪presentationAlternatives≫ and an anchor for every presentation group.

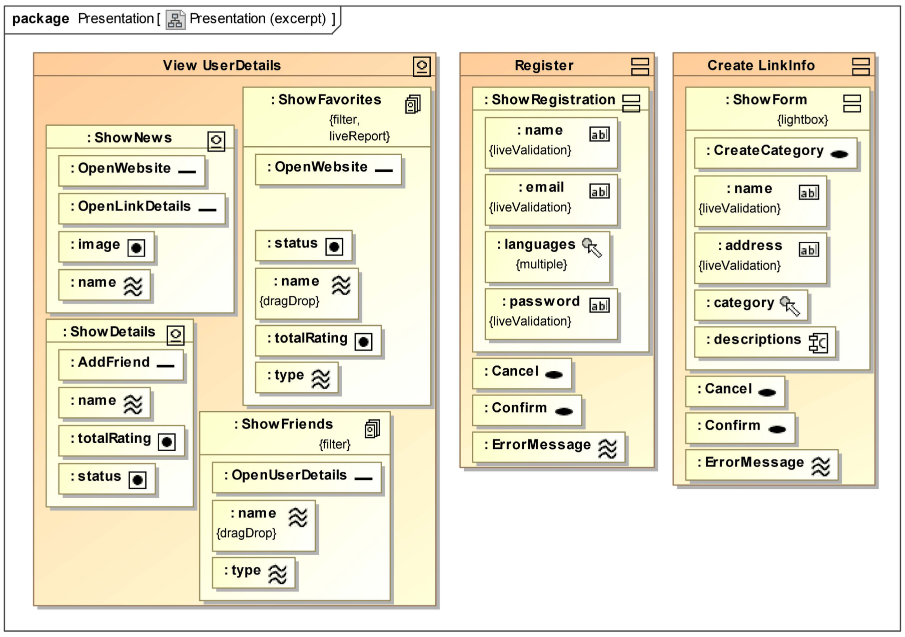

Fig. 5. *Linkbook*: Presentation model generated by transformations (excerpt)

The following model transformations are defined to transform requirements (use cases and workflows) into a presentation model:

- Creation of ≪presentationGroup≫es for ≪browsing≫ use cases; ≪processing≫ use cases are transformed into ≪inputForm≫s.
- An ≪inputForm≫ is also created for each ≪displayAction≫.
- Elements of type ≪displayPin≫ and ≪interactionPin≫ are translated as presentation properties part of the ≪inputForm≫s that were generated for the ≪displayAction≫; the stereotype of the presentation class is computed from the type tag.
- Tags (with exception of type), mainly used for modeling RIA features, are added to the corresponding input elements.

– An action of type ≪systemAction≫ with a tagged value confirmed set to true is translated into two classes of stereotype ≪button≫ named OK and Cancel; an action of type ≪userAction≫ with a tagged value validated set to true is translated to a class of stereotype ≪text≫ to show the errors of the validation.

Fig. 5 shows an excerpt of the presentation model including the presentation groups ViewUserDetails, Register and CreateLinkInfo that were automatically generated by the model-to-model transformations defined above. The first presentation group is related to the ≪navigation≫ use case and the other two to the ≪processing≫ use cases with the same name.

4 Tools Supporting the Model-Driven Engineering Process

UWE models can be designed using all UML development environments that enable the use of (almost all) profiles and mainly those offering visual modeling facilities. However, the frequent use of stereotypes and tagged values as well as certain domain-specific modeling characteristics suggested the idea of a tool supporting frequently used features of UWE. Conversely to the development of a proprietary tool, the goal is to extend existing CASE tools in order to benefit from UML compliance.

The following tools were developed as plugins of available UML development environments to support the UWE approach : ArgoUWE is based on ArgoUML, MagicUWE extends MagicDraw [1] and the TopUWE-plugin has been developed for TopCased. The first and third tool have the advantage of being based on open-source projects; the second one builds on a commercial tool whose new releases always consider UML improvements. The implementation of new features in ArgoUWE was discontinued as ArgoUML was not migrated to UML2. Currently, the second plugin provides full support and the third one is work in progress.

We therefore started to define a set of model transformations in MagicUWE to benefit from the efforts invested in the requirements models and to produce initial versions of all design models, i.e. content, navigation and presentation models (see Fig. 6). The set of transformations implemented are:

– requirements to content,
– requirements to navigation,
– requirements to process, and
– requirements to presentation.

The goal of these plugins are the computer aided design of web applications using the UWE approach. They offer to the designer, in addition to the use of the UWE profile, aid for the selection of the model elements, transformations for the automatic generation of sketches of models (see transformations options in Fig. 6) or the refinement of certain parts (aspects) of the models. Thus, the UML CASE tools are customized to the specific modeling domain of web software by specific plugins. Code generation is supported by the UWE4JSF tool [7].

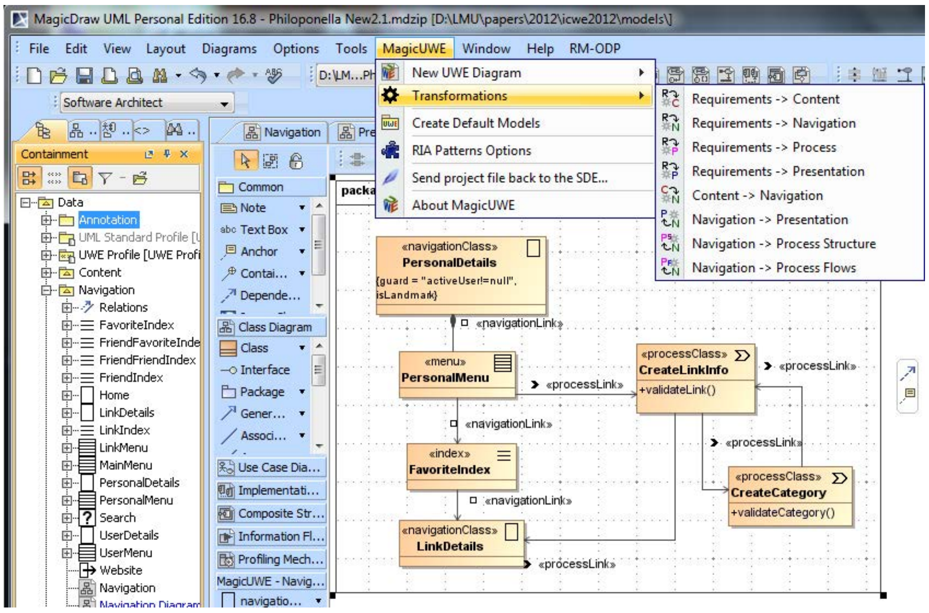

Fig. 6. MagicUWE: Tool support for modeling and transforming

5 Related Work

Several model-driven web engineering methods have been put forward during the last decade, only some of them include explicitly requirements specification in their software development process. The survey of Valderas and Pelechano [16] presents a detailed analysis of the model-driven characteristics of the most relevant methods.

OOHDM [14] defines a proprietary notation called user interaction diagrams used to refine use cases. Only UIDs are used to derive conceptual models, but there is no tool supporting the MDE process. Similarly, the previous version of UWE [4] that included a notation for requirements specification called WebRE, did not provide tool support for model transformations, but for the modeling as it is UML compliant. The Web Modeling Language (WebML) is supported by WebRatio, a commercial tool that is in use in many real projects. This implies a lot of experience in requirements specification, but the requirements models – use cases and textual specification – proposed by WebML [2] are not fully integrated in the automated generation of the web applications. The most complete approach is presented by Object-Oriented Web Solutions (OOWS) [16], which includes a task taxonomy, description of user tasks and system data. The notations used are task trees and activity diagrams, and the MDE process is fully supported by a graph-transformation-based tool. The drawback of this approach is the complexity of the requirements model and the need of proprietary tool support due to the use of a mix of techniques. The main focus of the Navigational Development Technique (NDT) is the requirements analysis phase [3]. The NDT Suite has been developed to support this very detailed template-based approach. Although NDT is useful for the requirements elicitation, the approach of textual templates are less appropriate for the specification of navigational aspects of web applications.

More recently, the Mockup-driven development process (MockupDD) of Rivero *et al.* [13] was defined using user interface mockups. Digital mockups are constructed with open-source mockup tools and afterwards enriched with annotations enabling smooth transformations into e.g. UWE navigation and presentation models. The advantage of the approach is the use of graphical user interface prototypes, easing communication with customers and designers. But the use of more than one CASE tool requires the export and import of models with the usual problem of visualizing these models.

6 Conclusions

We presented a model-driven engineering (MDE) approach that moves the focus of modeling from a late to an early phase in the software development life cycle (SDLC), i.e. from design to requirements. The approach consists of the specification of requirements models (source models)(1) using the UML-based Web Engineering domain specific modeling language(2), transforming these models to the target models (3), i.e. first approaches of UWE design models (content, navigation, presentation).

The benefits of such an approach are that the developer can focus on requirements modeling providing a better tool for discussions and agreements with the customer. On the other hand the generation of basic design models provides an effort reduction of the time consuming task of building these design models. Although the approach is generic and the idea of a DSML in an early development phase of web applications could be applied for any model-driven approach, we selected the UML-based Web Engineering (UWE) [6] to illustrate the approach. As an example we use a social media *Linkbook* application. Model transformations are tool supported in the CASE tool MagicUWE.

An evaluation of the approach comparing automatic generated and manually created design models was performed; the results are included in our previous work [5]. We plan to corroborate the evaluation results with empirical data obtained by groups of students that will manually create the design models. A future task would be the implementation of the model transformations as plugin of an open source tool, probably TopCased.

References

1. Busch, M., Koch, N.: MagicUWE – A CASE Tool Plugin for Modeling Web Applications. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 505–508. Springer, Heidelberg (2009)
2. Ceri, S., Brambilla, M., Fraternali, P.: The History of WebML: Lessons Learned from 10 Years of Model-Driven Development of Web Applications. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 273–292. Springer, Heidelberg (2009)
3. Escalona, M.J., Aragón, G.: NDT. A Model-Driven Approach for Web Requirements. IEEE Trans. Softw. Eng. 34(3), 377–390 (2008)
4. Escalona, M.J., Koch, N.: Metamodelling the Requirements of Web Systems. In: Rev. Sel. Papers Int. Conf. Web Information Systems and Technologies (WEBIST 2005–2006). LNBIP, vol. 1, pp. 267–280 (2007)

5. Koch, N., Knapp, A., Kozuruba, S.: Assessment of Effort Reduction due to Model-to-Model Transformations in the Web Domain. In: Brambilla, M., Tokuda, T., Tolksdorf, R. (eds.) ICWE 2012. LNCS, vol. 7387, pp. 215–222. Springer, Heidelberg (2012)
6. Koch, N., Knapp, A., Zhang, G., Baumeister, H.: UML-Based Web Engineering: An Approach Based on Standards. In: Olsina, et al. [12], ch. 7, pp. 157–191
7. Kroiss, C., Koch, N., Knapp, A.: UWE4JSF: A Model-Driven Generation Approach for Web Applications. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 493–496. Springer, Heidelberg (2009)
8. Meliá, S., Gómez, J., Pérez, S., Díaz, O.: A Model-Driven Development for GWT-Based Rich Internet Applications with OOH4RIA. In: Proc. 8th Int. Conf. Web Engineering (ICWE 2008). LNCS, vol. 5648, pp. 13–23. Springer (2008)
9. Mendes, E., Mosley, N. (eds.): Web Engineering. Springer, Berlin (2006)
10. Object Management Group. Business Process Model and Notation, version 2.0. Specification, OMG (January 2011), `http://www.omg.org/spec/BPMN/2.0/`
11. Object Management Group. Unified Modeling Language: Superstructure, version 2.4. Specification, OMG (August 2011), `http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/`
12. Olsina, L., Pastor, O., Rossi, G., Schwabe, D. (eds.): Web Engineering: Modelling and Implementing Web Applications. Human-Computer Interaction Series, vol. 12. Springer (2008)
13. Rivero, J.M., Grigera, J., Rossi, G., Luna, E.R., Koch, N.: Towards Agile Model-Driven Web Engineering. In: Nurcan, S. (ed.) CAiSE Forum 2011. LNBIP, vol. 107, pp. 142–155. Springer, Heidelberg (2012)
14. Rossi, G., Schwabe, D.: Modeling and Implementing Web Applications with OOHDM. In: Olsina, et al. [12], ch. 6, pp. 109–155
15. Valderas, P., Fons, J., Pelechano, V.: From Web Requirements to Navigational Design – A Transformational Approach. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 506–511. Springer, Heidelberg (2005)
16. Valderas, P., Pelechano, V.: A Survey of Requirements Specification in Model-Driven Development of Web Applications. ACM Trans. Web 5(2), 10 (2011)