Assessment of Effort Reduction due to Model-to-Model Transformations in the Web Domain*

Nora Koch[1,2], Alexander Knapp[3], and Sergej Kozuruba[1]

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

Abstract. Model-driven engineering (MDE) approaches provide the well-known advantage of software development at a higher level of abstraction. However, in the web engineering domain such approaches still encounter difficulties mainly due to applications that are continuously evolving and the heterogeneity of web technologies. Instead of fully automated generation, we look at MDE as assisting the web engineer in different phases of the software development life cycle. In particular, we use model-to-model transformations to support the generation of model sketches of the different concerns from a requirements specification. In this work, we present a metric to measure the effort reduction that results from applying this kind of model-driven approach. We use the metric to evaluate models of six web applications in the context of UML-based Web Engineering (UWE).

1 Introduction

Requirements models should be the result of an intensive communication with the customer and provide the representation of the business decisions related to the application to be developed. The more accurate the models produced in this early development phase, the less error-prone are the models and the code generated later. This relationship between the quality of the requirements specification and the implemented system has been analyzed and confirmed several times [4]. However, in practice, many projects start too soon with the technical design and the implementation. Even if requirements are specified, they are often partially ignored. The time invested in the requirements specification is very often seen as partially wasted. Therefore it is important to improve the use of the requirements specification in further development steps and to obtain as much information as possible for the so-called design models. In this work, we assess the utility of modeling the requirements and use these models in a model-driven development process (MDD) instead of manual modeling in all stages. We define a metric for measuring the effort reduction due to automatic generation of models against manual creation. Such an effort reduction would represent a measurable benefit for web productivity [4, Ch. 3]. We propose an assessment strategy that consists of the creation and generation of models, their comparison and calculation of the effort reduction indicator.

We have built the models of six web applications using UWE (UML-based Web Engineering [3]); these applications are a simple address book, a movie database, a music

* 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.

portal, a social network, a publication management system and the UWE website. Our assessment approach analyzes the model elements and aggregates them for the different concern models. Although the requirements models were rather sketchy (estimated degree of details of 53%), the benefit, i.e. the effort reduction reached by having drawn and used them in the MDD process is calculated to be between 26% and 77%. We have tested as well the robustness of our metric and reasoned about the scalability. An empirical evaluation performed by a set of web engineers is planned for the corroboration of our metric. However, in this work, our focus is on the definition of the assessment strategy and to show the plausibility of the approach.

Related Work. Several model-driven web engineering methods were presented during the last decade [5]. Valderas and Pelechano [7] present a detailed analysis of the MDD characteristics of the most relevant methods. Only some of them include explicitly a requirements phase. OOHDM, UWE and WebML defined proprietary notations for the requirements offering only partial MDD tool support. The most complete though rather complex approach is presented by Object-Oriented Web Solutions (OOWS [7]). The requirements analysis phase is also the focus of the Navigational Development Technique (NDT [2]); the approach of textual templates, however, is less appropriate for the specification of navigational aspects of web applications. More recently, the Mockup-driven development process (MockupDD [6]) was defined which enables smooth transformations into e.g. UWE navigation and presentation models.

Software effort models help project managers to take decisions on methods, techniques, and tools. Mendes et al. [4, Ch. 2] present techniques for effort estimates in web software development, but do not particularly analyze the effort reduction implied by an MDD process. Another approach consists in calculating productivity based on size and effort aggregating different size measures that characterize web applications [4, Ch. 3].

2 Assessment Strategy

Our assessment is defined in terms of reduction of modeling efforts, i.e., measuring to which extent models of web software can be generated by transformations. We focus on comparing the results of the automatic generation of design models of rich web applications — just per mouse click — to the work the designer invests in modeling the application from scratch. Both, the manual modeling and the model-driven development processes use requirements models as source for building the design models.

Research Scope and Questions. The requirements models in our assessment are very simple, i.e. without many details. They contain enough information to discuss the web application with the customer, but abstract from details mainly required for the implementation. Hence, these requirements models are insufficient for the generation of complete design models and code. In fact, our goal is to analyze to which extent these simple requirements models can provide substantial help in building design models.

Our empirical method was designed to answer the following questions: (Q1) How much of the modeling effort can be reduced through automatic generation of design models of web applications? (Q2) Is it worth for the designer to focus on the modeling of the requirements in terms of effort reduction for design models? (Q3) How could web specific modeling tools provide more assistance through partial model-driven support?

Table 1. Similarity scale with associated benefit

Kind (k)	**Description**	**Benefit** ($b(k)$)
$identical$	both model elements have exactly the same features	1
$similar$	some features are identical; others are missing or erroneous	0.5
$erroneous$	contains features not included in the original one	-0.25
$missing$	is not included in the model while the original is included	0

Table 2. Weights for UML model element types

UML model element type (t)	**Weight** ($w(t)$)	**UML model element type** (t)	**Weight** ($w(t)$)
Class	1	Action	1
Attribute	0.5	Object Node	0.75
Association/Dependency	0.5	Pin	0.5
Tag	0.25	Control/Object Flow	0.25
Property	0.75	Use Case	1

Table 3. Effort reduction indicators and scope factor

(1) $E(t) = (\sum_{k=1}^{m} G(k,t) \cdot b(k))/M(t)$ effort reduction per model element type

(2) $E(c) = (\sum_{t=1}^{n} E(t) \cdot w(t))/\sum_{t=1}^{n} w(t)$ effort reduction per concern

(3) $E = (\sum_{c=1}^{v} E(c))/v$ effort reduction for the application

(4) $s = (\sum_{t=1}^{n} R(t) \cdot w(t))/(\sum_{t=1}^{n} M(t) \cdot w(t))$ scope factor

Assessment Process. The methodology for the assessment has as input the requirements models of the application and consists of the following steps: (1a) manual creation of the design models following the principle of separation of concerns, (1b) generation of the basic design models using transformations; (2) comparison and classification of the model elements of the manually created and automatically generated models; and (3) calculation of the *effort reduction* indicator and interpretation of results.

The notation used for our case studies is the UML profile UWE, the tool is MagicDraw and the MagicUWE plugin with its model-to-model transformations. The requirements of the web applications are modeled with use case diagrams for the functional system properties and activity diagrams for the navigational paths and processes. The design models express the different aspects of content, navigation, presentation, and processes of web applications. These design models are produced twice for our evaluation: in step (1a) manually by the designer, in (1b) automatically by model-to-model transformations from the requirement models (see Sect. 3).

In step (2) the generated model elements, such as classes, attributes, and actions are compared to the manually designed model elements. We distinguish $m = 4$ *kinds of similarity* k for generated model elements: $identical$, $similar$, $erroneous$, and $missing$ (see Tab. 1). The *benefit factor* $b(k)$ of kind k tells how much the generation contributes to the work of the web engineer. This ranges from $b(identical) = 1$, when nothing has to be changed, to $b(erroneous) = -0.25$, when elements need to be removed.

Finally, in step (3) the *effort reduction* indicator is calculated for model types and aggregated for each concern and for the application; see Tab. 3. The indicator $E(t)$ for a model element type t, like class, attribute, etc., is the sum of the number of the generated

elements $G(k,t)$ of a similarity kind k and of the type t weighted with the benefit factor $b(k)$ and divided by the corresponding number of manually generated model elements $M(t)$ (Tab. 3(1)). The effort reduction indicator $E(c)$ for a concern c, such as content, navigation, process, and presentation, is calculated as a linear additive weighted formula of the effort reduction indicator $E(t)$ of the n individual model element types (Tab. 3(2)). The *weight* $w(t)$ in Tab. 2 expresses the relevance a model element type t has for the designer. For example, classes are first-class citizens and attributes are not, as they belong to classes. The effort reduction indicator E for the entire web application is given by the average over all v concerns that are modeled for the web application (Tab. 3(3)). We assume that for the designer all concerns have the same relevance.

With E we provide an estimation of the amount of spared effort when we focus on modeling requirements of a web application and partially generate the design models. We need, however, to complete these draft models with some *additional effort* in order to achieve the same objective as when modeling the different concerns manually. In terms of project productivity each activity in the development process has a measurable, positive cost, with exception of the automated model transformations (we neglect the implementation costs of the transformations as they are reusable for many projects).

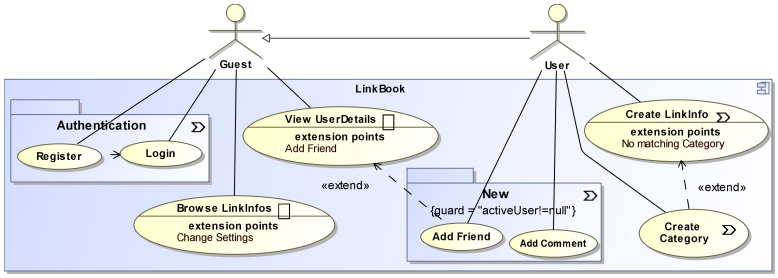
Until now, we only assumed that the same requirements model were used for both the manual and the automatically generated design models, but disregarded the quality of the requirements model. We introduce a *scope* factor that gives a very rough estimation of the degree of detail to which the requirements are modeled. This scope factor is calculated as the ratio between the linear additive weighted expression of the number of requirements elements $R(t)$ of a model element type t and the number of design elements $M(t)$ (Tab. 3(4)). We use it to normalize the values obtained for the effort reduction of a web application making different web applications comparable.

3 Model-based Development of Web Applications in UWE

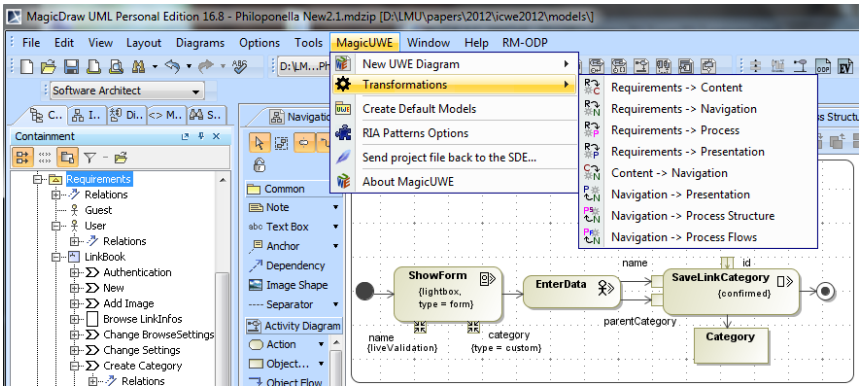
The assessment strategy for effort reduction in MDD is independent of the approach for developing web applications; only support for modeling the requirements and the different design concerns is required. In this work, we use the UML-based Web Engineering (UWE [3]) notation, method, and tool support for evaluating the strategy. The UWE notation is defined as a UML profile. The cornerstones of the UWE method are the principle of separation of concerns and the model-driven approach. As UWE tool we use the MagicUWE plugin implemented for MagicDraw [1].

We illustrate the modeling process and the results of the model transformations in UWE by a web-based social network application for sharing favorite web links with friends: *Linkbook* is accessible to guests and registered users, providing logging in/out and (un)registering functionality. The homepage shows a list of favorite links grouped by categories and offers search facilities for links and user comments. Registered users can comment links and switch to their personal view for managing their links. Network functionality is offered by a list of friends, giving access to the friends' favorites.

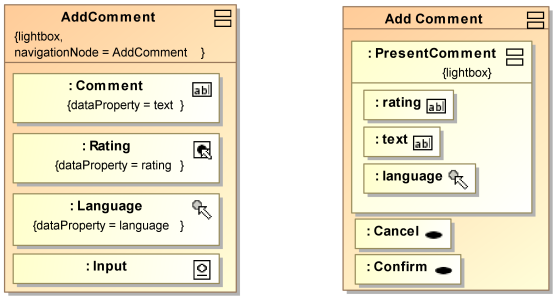
Requirements Modeling. In UWE, a web application's functional requirements are captured by use cases and activities. Figure 1(a) depicts a subset of the *Linkbook* use cases. The UWE profile supports web-specific annotations by stereotypes for use cases, like



(a) Functional requirements modeled with use cases (excerpt)



(b) MagicUWE tool with requirements workflow for CreateCategory



(c) Presentation elements: manually modeled (M, left) and automatically generated (G, right)

Fig. 1. *Linkbook* example in MagicUWE

≪browsing≫ (□, pure navigation) and ≪processing≫ (Σ, workflow functionality). Use cases in packages inherit the stereotype of the package.

Each ≪processing≫ use case is refined by a workflow using UML activity diagrams, for CreateCategory see Fig. 1(b) (lower right). The workflow specifies the process actions, input/output information with pins or objects, decisions, and rich user interface features. Web-specific annotations can be added to actions, like ≪displayAction≫ (✓) for the explicit presentation of elements, ≪userAction≫ (웃) for asking the user for input, or ≪systemAction≫ (▯) indicating the processing of data.

Creating Design Models. The UWE method for the design phase follows the principle of "separation of concerns": A content model represents the domain concepts and the relationships between them. A navigation model is used to represent navigable nodes and the links between nodes. A presentation model provides an abstract view on the user interface (UI). A process model contains the workflows of the processes which are invoked from certain navigation nodes.

A navigation model consists of navigation classes for the navigable nodes and process classes for accessing business processes. Alternative navigation paths are handled by menus, multiple instances of navigation paths by indexes, menus, and queries. 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 ≪textInput≫ or ≪selection≫. Figure 1(c) (left) shows the presentation model for the process AddComment which is related to the ≪processing≫ use case with the same name.

Generating Design Models. On the right side of Fig. 1(c) the presentation model of the same AddComment form is shown, but this one was automatically generated by model-to-model transformations. A set of model transformations is defined in MagicUWE with the goal to benefit from the efforts invested in the requirements models and produce initial versions of all design models, i.e., content, navigation, process, and presentation (see Fig. 1(b)). To describe in detail each of these transformations is out of the scope of this work. For the requirements-to-presentation transformation mainly information from the requirements workflow with its action stereotypes is used, like ≪userAction≫ asking the user for input.

4 Evaluation Results

Once the modeling part of the process described in Sect. 2 is completed, the evaluation of the MDD approach can be started (steps 2 and 3). We first provide details on how the effort reduction calculations are applied to the *Linkbook*, then we present the assessment results for a set of six web applications and discuss the robustness of our findings.[1]

Assessing Effort Reduction for Linkbook. A counting of model elements of both design models resulting from the requirements, the manually crafted and the generated model, is performed for each concern. The generated elements are classified into the kinds of of Tab. 1. For example, the excerpt of the manually crafted presentation models of *Linkbook* shown in Fig. 1(c) contains 10 model elements (1 class, 4 properties and 5 tags); the generated counterpart contains 7 model elements (1 class, 6 properties and 1 tag); there is no *identical* element, 4 *similar* and 3 *erroneous*. Table 4 presents the results of the counting and categorization of all model elements of the presentation concern. These modeling elements are the presentation group (including those that inherit from it, like input form), the interactive elements (like button, input text, selection), and the output properties (such as text, images).

[1] Further details on the example applications (*Address Book*, *Music Portal*, *Movie Database*, *Publication Management System* and the *UWE website*) as well as download links of their models can be found at the UWE website: `http://uwe.pst.ifi.lmu.de`.

Table 4. Linkbook application: Effort reduction indicators for the presentation concern

Linkbook	Manually modeled	Generated			Effort reduction indicator (E)
		Identical	Similar	Erroneous	
Presentation Groups (Class)	38	13	15	12	0.46
Interactive Elements (Property)	54	32	18	26	0.64
Output Elements (Property)	25	17	8	11	0.73
Presentation Model					0.59

Table 5. Overview on Assessment Results

Web application	Content	Navigation	Presentation	Process	E	Scope	E normalized
Address Book	0.56	0.45	0.51	0.66	0.54	0.75	0.39
Linkbook	0.45	0.57	0.59	0.63	0.56	0.84	0.35
Movie DB	0.25	0.30	0.10	0.33	0.25	0.51	0.26
Music Portal	0.59	0.52	0.46	0.78	0.59	0.41	0.77
Publications MS	0.13	0.33	0.25	0.17	0.22	0.36	0.33
UWE Website	0.40	0.01	0.53	0.78	0.43	0.32	0.72
Average						0.53	0.47

Table 6. Statistics on Linkbook models

Model	Class	Use Case	Action	Attribute	Assoc./ Depend.	Pin	Property	Contr./Obj. Flow	Object Node	Tag	Weighted Average
Req.	0	22	93	0	15	131	0	179	16	93	63.06
Design	80	0	46	33	70	57	135	120	19	114	75.11

The last column of Tab. 4 shows the values of the effort reduction indicators calculated using the equations in Tab. 3: $E(t)$ for each model element type t and the effort reduction indicator $E(c)$ for the entire presentation concern, which is 59%. The effort reduction indicators for the content, navigation and process concerns are computed similarly (see second row of Tab. 5). The effort reduction indicator E for the entire web application *Linkbook* is 56%. As additional modeling effort required after the execution of the model transformation 48 model elements have to be built.

Comparing Effort Reductions of Multiple Applications. Table 5 gives an overview of the effort reduction indicators for all six web applications of the assessment study. The evaluation results show that the execution of the transformations and the resulting first drafts of the different models for the content, navigation, presentation, and process concerns imply an effort reduction between 22% and 59%, but irrespective of the amount of effort that has been invested in the requirements modeling. To correct this bias, we use the scope factor s of Tab. 3(4) based on a relationship between amount and type of model elements used at requirements and design level; see next-to-last column of Tab. 5. The scope factor is then applied to normalize the effort reduction indicator of each application. The normalized values of E (last column) are comparable and are situated in the range between 26% and 77% with an average effort reduction of 47%.

These results allow the following answers to the questions in Sect. 2: (Q1) The modeling effort can be reduced in average by 47% if the degree of detail of the require-

ments models is estimated in 53%; assuming linearity this would imply that complete requirements models (100%) would lead to an automatic generation of 88% of the design models. (Q2) The effort reduction values confirm that it is worth to invest in the requirements modeling. (Q3) Tools should allow for separate execution of model transformations for each concern enabling the modeler to select appropriate transformations.

Robustness of the Assessment. We recalculated the effort reduction indicator changing the weights of model elements in Tab. 2. A modification of 0.25 for a navigation element type changes the effort reduction indicator of the concern by max. 3%, in average only 2%. Similarly, in the presentation model changes of 0.25 in average only affect the value of the indicator by 1%, max. 4%. Although these results sound encouraging, there are still some difficulties to be solved. The most important methodological issue is the regeneration of models after changes in the target models have been performed, i.e. how to merge models and identify conflicts. A more technical and tool related problem is the graphical representation of the diagrams corresponding to generated models.

5 Conclusions

We have presented an assessment process and a metric for measuring the effort reduction resulting from using model-transformations instead of manual creation of design models based on the requirements models of web applications. The proposed assessment strategy has been applied to six web applications, whose requirements have been specified using the UWE approach. Our evaluation shows that the MDD approach reduced the effort in more than 45%, which could even be improved if the degree of detail of the source models is increased. We plan to corroborate the results of our evaluation with empirical data obtained by groups of students that will create the models and use the same tool for generating these web applications.

References

1. M. Busch and N. Koch. MagicUWE — A CASE Tool Plugin for Modeling Web Applications. In *Proc. 9th Int. Conf. Web Engineering (ICWE'09)*, volume 5648 of *Lect. Notes Comp. Sci.*, pages 505–508. Springer, 2009.
2. M. J. Escalona and G. Aragón. NDT. A Model-Driven Approach for Web Requirements. *IEEE Trans. Softw. Eng.*, 34(3):377–390, 2008.
3. N. Koch, A. Knapp, G. Zhang, and H. Baumeister. UML-Based Web Engineering: An Approach Based on Standards. In Olsina et al. [5], chapter 7, pages 157–191.
4. E. Mendes and N. Mosley, editors. *Web Engineering*. Springer, Berlin, 2006.
5. L. Olsina, O. Pastor, G. Rossi, and D. Schwabe, editors. *Web Engineering: Modelling and Implementing Web Applications*. Springer, 2008.
6. J. M. Rivero, J. Grigera, G. Rossi, E. R. Luna, and N. Koch. Towards Agile Model-Driven Web Engineering. In *Proc. 23rd Int. Conf. Advanced Information Systems Engineering (CAiSE'11)*, Lect. Notes. Bus. Inf. Proc. Springer, 2012. To appear.
7. P. Valderas and V. Pelechano. A Survey of Requirements Specification in Model-Driven Development of Web Applications. *ACM Trans. Web*, 5(2):10, 2011.