MDA Transformations Applied to Web Application Development*

Santiago Meliá[1], Andreas Kraus[2], and Nora Koch[2,3]

[1] Universidad de Alicante, Spain
santi@dlsi.ua.es
[2] Ludwig-Maximilians-Universität München, Germany
{kochn,krausa}@pst.ifi.lmu.de
[3] F.A.S.T GmbH, Germany

Abstract. Current Web generation techniques are mainly hard-coded for predefined architectures of Web applications. Consequently, there is a gap between Web design models and the final implementation. We solve this problem, following with our approach the Model-Driven Architecture (MDA) principles of automatic generation of software systems based on model transformations. In this context, we present a transformation process and propose a visual and textual specification for the transformations using the forthcoming OMG standard Query /Views/ Transformations (QVT). Our proposal is illustrated by transformations involving elements of the UML-based Web Engineering (UWE) metamodel and the WebSA metamodel, showing this way how both approaches are integrated.

1 Introduction

Models, modelling approaches and model transformations that follow the key principles defined by the Model-Driven Architecture (MDA) are gaining consensus within many organizations involved in the development of complex software. They are attracted by the final MDA goal that is the automatic generation of a complete software system from a model with as less human interaction in the generation process as possible. Such vision has enormous consequences for the development and maintenance of the increasing amount of Web software that is being produced. However, the current Web generation techniques are totally or partially hard-coded for predefined architectures of Web applications. We propose a generation process using an MDA approach in which the model transformations are driven by different architecture models.

In order to define the transformations between different models, there are several initiatives related to the MDA approach, among others the Request for Proposals for a Query/Views/Transformations (QVT) [11] language. From the received proposals, QVT-P [12] is, in our opinion, the most interesting one as it is a well defined language and it comprises a graphical as well as a textual notation.

In this article we present the WebSA approach [7] based on architectural-centric transformations from design to implementation models. This approach proposes (1) a

* This research has been partially sponsored by the EC 5th FP AGILE (IST-2001-32747) the German BMBF project GLOWA-Danube, and the Spanish METASING (TIN2004-00779)

D. Lowe and M. Gaedke (Eds.): ICWE 2005, LNCS 3579, pp. 465–471, 2005.

development process based on MDA, (2) a set of architectural models and (3) a set of transformations that permit the automatic integration of these architectural models with the functional models of a Web application using the QVT-P notation. The functional models, like navigation are those proposed by any Web design method such as WebML [2], OO-H [3] or UWE [5].

Sections 2 and 3 give an overview of the WebSA development process and the UWE design method, respectively. Section 4 presents the specification of the transformations and finally in section 5 some future steps of the use of WebSA for the development of Web applications are outlined.

2 The WebSA Approach: An Overview

WebSA is a proposal whose main objective is to cover all phases of Web application development focusing on software architecture. It contributes to fill the gap currently existing between traditional Web design models and the final implementation. In order to achieve this, WebSA defines a set of architectural models to specify the architectural viewpoint which complements current Web engineering methodologies [3, 5]. Furthermore, WebSA also establishes an instance of the MDA development process [4], which allows for the integration of the different viewpoints of a Web application by means of transformations between models.

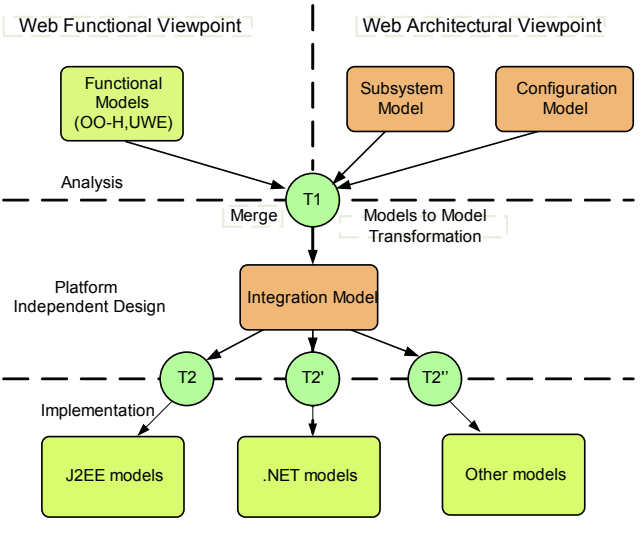

Fig. 1. WebSA Development Process

The WebSA development process is based on the MDA development process in which the artifacts that result from each phase must be models, which represent the different abstraction levels in the system specification. In the analysis phase the Web application specification is vertically divided into two viewpoints, as shown in the diagram flow of Fig. 1. On the one side, the functional-perspective is given by the Web functional models provided by Web methods (see [2, 3, 5]). On the other side, the Subsystem Model (SM) and the Configuration Model (CM) define the software

architecture of the Web Application. The SM and CM architectural models use two different architectural styles to specify a Web application: a subsystem (or layer style) and a component style.

The PIM-to-PIM transformation (T1 in Fig. 1) which goes from analysis models to platform independent design models. It integrates the information about functionality and architecture (see sect. 4.1) in a single Integration Model (IM). This transformation type will be called T1. Also, the Integration Model, is the basis on which several PIM-to-PSM transformations, one for each target platform (see e.g. T2, T2' and T2'' in Fig. 1), can be defined. The output of these transformations is the specification of the Web application for a given platform (see sect. 4.2). This transformation type will be named T2 in the rest of the article.

3 A Web Functional Design Method: The UWE Approach

The distinguishing feature of the UML-based Web Engineering (UWE) approach in relation to other Web design methods is its UML compliance. The metamodel of UWE [6] is defined as a conservative extension of the UML metamodel which has a mapping to a UML profile [5]. Similarly to other Web design methods, UWE separates the concerns of a Web application supporting the modelling of different points of view: content, navigation structure, business processes and presentation.

The content of a Web application is modelled in UWE by a conceptual model that is represented as a UML class diagram. The navigation model is based on all conceptual classes that are relevant for the navigation structure and represents the navigation paths of the Web application. The model elements used to build nodes and links are primarily «navigation class» and «navigation link». In addition, access primitives (a special kind of nodes), such as «index» or «guided tour» are used to reach multiple instances of Web nodes.

Navigation models are enriched by «process class»es and «process link»s showing how the workflows are integrated in the navigation structure. These process classes and process links are part of the process model, which deals with the business logic of a Web application. The behavioural aspects of the business logic are modelled by a process flow model represented as a UML activity diagram. In UWE, the presentation model is used to sketch the layout of the Web pages associated to the navigation nodes.

In contrast to many other methods, UWE defines a systematic method, which supports semi-automatic generation of the models described above. Although, until now, UWE has not referred to these automatic generation steps explicitly as a transformation-based "model-driven development" feature, those steps correspond to a model driven development approach. UWE allows e.g. for the generation of the navigation model based on the set of conceptual classes marked as relevant for navigation. Further, indexes and menus are included automatically in the navigation model with additional model transformations that apply on the navigation model. A basic presentation model can be defined by transformations based on the navigation model.

In our case, the WebSA and UWE metamodels play an important role in the WebSA development process, because they contain the information necessary to specify the model transformations T1 and T2.

4 The WebSA Transformation Process

The WebSA transformation policy is defined by a set of transformations in which the first class citizens are the classes of the architectural view. The WebSA development process consists of two types of transformations: T1 and T2 (Fig. 1). T1 merges the elements of the architectural models of WebSA with those of the functional models, and translates them into the Integration Model. T2 maps the platform specific implementation models (e.g. J2EE or .NET) from the Integration Model. Both transformations are complex, i.e. they are built of a set of smaller transformations, which are executed in a deterministic way.

In MDA [9] there are different alternatives to get the information to transform one model into another (e.g. using a profile, using metamodels, patterns and markings, etc). For WebSA we have selected a metamodel mapping approach to specify the transformations. In order to obtain the integration we extend the MDA model transformation pattern of Bézivin [1] for UWE and WebSA models. The metamodels based on the MOF language are the source of the transformation rules that establish the transformation into target metamodel elements. For more details about the metamodels refer to [6] and [8].

The transformation rules are defined in the QVT language [11] which is an MDA standard also based on MOF 2.0. We selected the QVT-P [12] proposal, which comprises a rich graphical and textual notation. Both notations can be used to declaratively define transformations without specifying how a transformation is actually executed. Simple queries can be expressed by a (graphical or textual) pattern matching language that allows matching instances, sets of instances and associations with specific properties. For more complex queries the (additional) use of OCL 2.0 expressions is recommended. QVT-P transformations can be composed and extended by inheritance or overriding which is needed for scalability and reusability. In contrast with other transformation proposals (like graphs, XSLT, etc.), QVT-P has a smaller learning curve because the transformations themselves are models based on standards as MOF and OCL.

Next, we present an example of a T1 transformation using the graphical notation of QVT-P and also an example of a T2 transformation in the textual notation of QVT-P.

4.1 Transformation T1: Merging Web Functionality and Architectural Models

Due to the complexity of the T1 transformation, it is helpful to build a map of transformations that indicates the flow of execution and avoids redundancies in the specification. In the transformation map each transformation is related to the rest by means of three different relationships: (1) Composition – A transformation can be composed by one or more transformations (2) Dependency – A transformation must be executed before another transformation (3) Inheritance – A transformation extends or overrides another transformation. We defined a simple UML profile to represent the transformation map where a transformation is defined as a class stereotype and it is represented by a circle (Fig. 2). The first transformation shown in the T1 map of Fig. 2 (*SM2IM*) goes from Subsystem Model to Integration Model.

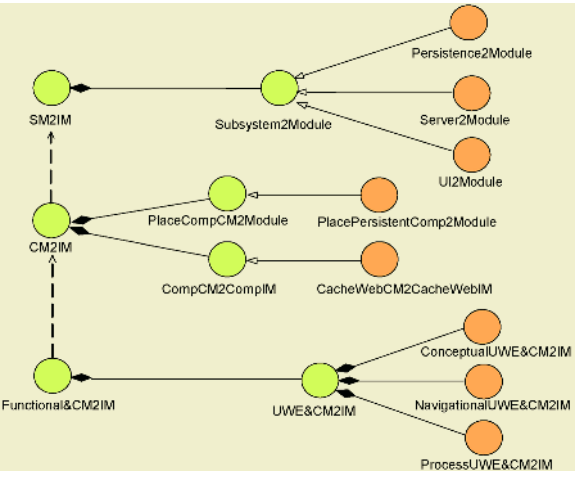

Fig. 2. Transformation MAP of T1

The second transformation (CM2IM) maps from Configuration Model to Integration Model. It is composed by a set of two types of transformations. The first one places components into the modules (PlaceComp2Modules), and the second one transforms each configuration component into one or more integration components (CompCM2Comp IM). The last transformation Functional&CM2IM merges the functional UWE models with the Configuration Model and introduces the functional aspects into the components of the Integration Model.

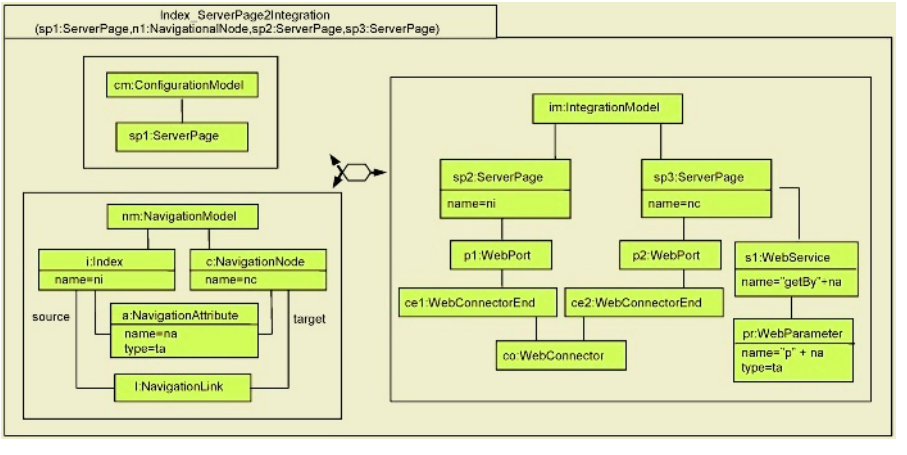

Fig. 3. Example of T1: NavigationalUWE&CM2IM

An example using the QVT-P graphical notation for the transformation *Index-Server Page2Integration,* which merges the navigation and configuration models, is shown in Fig. 3. This transformation specifies how links between index nodes and other navigation nodes in the UWE navigation model are merged into the WebSA configuration model and it results in a corresponding part of the integration model.

A more general transformation which is not depicted here states that every navigation node is merged into a *ServerPage* element. The more specialized transformation of the example additionally generates for every *NavigationAttribute* of an index element a *WebService* element with a *WebParameter* element corresponding to the *Navigation Attribute*. The *ServerPage* elements corresponding to the Index and the *NavigationNode* element are linked by a *WebConnector* element via *Web- ConnectorEnd* and *WebPort* elements, respectively.

4.2 Transformation T2: Transforming from a PIM to a PSM

Once the transformation T1 is completely executed, the functionality is interwoven into the architectural aspects in the Integration Model. Now, we can tackle the final step of the WebSA development process, defining a set of PIM-to-PSM transformations for each target platform such as J2EE, .NET or CORBA from the Integration Model. As is specified in [9], in order to make a transformation from PIM-to-PSM, design decisions must be made. These decisions are specified in the transformation T2 and taken in the context of a specific implementation design. Therefore, T2 is made up by a set of simple transformations in which one Integration Model component is transformed into a platform specific component. To specify T2, it is necessary to have the metamodels of the target platforms (e.g. the J2EE metamodel [10]).

Fig. 4 shows a QVT-P example of transformation T2 for J2EE using the textual notation. It transforms each *ServerPage* component of the Integration Model specified in the first *domain* into a *JavaServerPage* specified in the second *domain*. Furthermore, this *ServerPage* has a set of *WebServices*, each one of them translatable into a Java method, a Javascript method or an HTML form. In this example, we have chosen a translation into an HTML form by the *WebService2Form* transformation defined in the *forall* OCL sentence of the *{when}* part. In the same way, each View element related to the *ServerPage* is translated into a *JavaBean* through the *View2Bean* transformation. The PSMs obtained from the WebSA process are considered an implementation, because they provide all the information needed to construct an executable system.

```
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()) ))  }}
```

Fig. 4. Example of T2: ServerPage2J2EE

5 Conclusions and Future Work

Using an MDA approach with a transformation component in WebSA we achieve a more automated process for the development of Web applications with a strong focus on architecture modelling. WebSA complements the existing methodologies for the design of Web applications. In this paper we present the development process of WebSA and describe how models are integrated and generated based on model trans-

formations. For the specification of the transformations we choose QVT-P that allows for visual and textual description of the mapping rules. Currently, we are analyzing the possibilities to extend the Web development environments VisualWADE[1] and ArgoUWE[2] to support architectural modelling and model transformations. Further, we plan to test transformation specification and model generation for complex Web applications addressing the scalability of the approach.

References

1. J. Bézivin. In Search of a Basic Principle for Model Driven Engineering, Novática nº1, June 2004, 21-24
2. S. Ceri, P. Fraternali, M. Matera: Conceptual Modeling of Data-Intensive Web Applications, IEEE Internet Computing 6 (4), July/Aug. 2002, 20–30
3. J. Gomez, C. Cachero, O. Pastor: Extending a Conceptual Modelling Approach to Web Application Design. In Proc. 12th CAiSE '00, LNCS 1789, Springer, 2000
4. A. Kleppe, J. Warmer, W. Bast: MDA Explained: The Model Driven Architecture, Practice and Promise, Addison-Wesley, 2003
5. N. Koch, A. Kraus. The expressive Power of UML-based Web Engineering. In 2nd IWWOST02, CYTED, June 2002, 105-119
6. N. Koch, A. Kraus: Towards a Common Metamodel for the Development of Web Applications. In Proc. 3rd ICWE 2003, LNCS 2722, Springer Verlag, July 2003, 497-506
7. S. Meliá, C. Cachero. An MDA Approach for the Development of Web Applications, In Proc. of 4th ICWE'04, LNCS 3140, July 2004, 300-305
8. S. Melía.. The WebSA Composition Model Profile. Technical Report TR-WebSA2, http://www.dlsi.ua.es/~santi/pPublicaciones.htm, Nov. 2004
9. OMG. MDA Guide, OMG doc. ab/2003-05-01
10. OMG. UML Profile for Enterprise Distributed Object Computing Specification. OMG doc. ad/2001-06-09
11. OMG. Request for Proposal. MOF 2.0 Query/Views/Transformations, OMG ad/2002-04-10
12. QVT Partners. Initial Submission for MOF 2.0 Query/View/Transformations RFP, QVT-Partners, http://qvtp.org/downloads/1.1/qvtpartners1.1.pdf, Aug. 2003

[1] VisualWADE: http://www.visualwade.com

[2] ArgoUWE: http://www.pst.informatik.uni-muenchen.de/projekte/uwe/argouwe.shtml