MDA-BASED DEVELOPMENT OF DATA-DRIVEN WEB APPLICATIONS

Attila Adamkó and Lajos Kollár
Department of Information Technology, Faculty of Informatics
University of Debrecen, Egyetem tér 1., 4032 Debrecen, Hungary

Keywords: MDA, UML2, Web Information System, XForms, XML Schema.

Abstract: Due to the last decade's technological changes, the more or less static Web sites has been evolving into Web-based distributed applications in the past few years. Many competitive technologies have appeared but when modeling a Web application, details of current technologies should be detached from the model of the application domain since they are evolving very dinamically so they become obsolete relatively soon. We propose an MDA-based method which can be used for designing and implementing Web Information Systems in the above mentioned manner. Web applications are composed of several models (expressed in UML2): structural model describes the underlying data structure, composition model defines an abstract page structure, while navigational paths are modeled by navigational diagrams and use cases capture common user activities. By the help of XML, XSLT and XForms, prototypes can be generated from these UML models rapidly.

1 INTRODUCTION

The evolution of Web technologies increased the presence of the Web in our everyday life: starting from personal home pages through corporate portals and Web shops up to Web applications implementing complex business processes. This diversity of applications makes the selection of the appropriate technology and platform more difficult, in particular, when these applications should collaborate each other. The best practices tells us to keep the modeling of the application domain and the technological details separated.

Performing early demonstrations is an important factor during the communication with the customer. The models should formulate the complete structural description of the Web site, resulting in an incomplete prototype which may be used for testing and demonstrating purposes.

Following these guidelines, the approach presented in Section 2 helps in the development of small and medium sized data-oriented Web applications using UML. Use cases, activity and class diagrams are used to describe the behaviour and the structure of the Web application.

1.1 Model Driven Architecture

OMG's Model Driven Architecture (MDA) provides a framework which allows applications to be described in a platform-independent manner. In MDA, artifacts are formal models describing a given aspect of the system. MDA uses UML as a standard modeling language in order to achieve a common understanding among stakeholders.

In case of a technological change (or evolution), a PIM is not subject to change as it contains the platform-independent model of applications. All what we need is to create a PSM in order to show how platform-independent components manifest on the new platform or technology. A PSM metamodel should be defined (or refined) which can describe the new platform in an abstract way to make the creation of platform-specific models easier. Transformation rules are also needed for mapping PIM metamodel elements to PSM metamodel elements. Applying them to an existing PIM will result PSMs for the new platform. The final step in the development process is the transformation of each PSM to code. Because a PSM fits its technology rather closely this transformation could be done relatively easy.

Adamkó A. and Kollár L. (2008).
MDA-BASED DEVELOPMENT OF DATA-DRIVEN WEB APPLICATIONS.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 252-255
DOI: 10.5220/0001522202520255
Copyright © SciTePress

1.2 UML Profiles

The MDA approach requires a language which uses formal definitions so the tools will be able to transform these models automatically. OMG recommends the use of UML to construct platform-independent models. The power of UML in our case is the modeling of the structural aspects of a system. This is done through the use of class diagrams which enables the generation of PSMs with all structural features.

The semantics of UML models can be extended by applying UML profiles by adding stereotypes and tagged values which is a common way for buliding UML models for particular domains. These profiles are considered to reside in the metamodel layer allowing to define UML “dialects” that can be used by models in the model layer. This mechanism is useful for both PIMs and PSMs.

For PIMs, a new profile should be defined to support a given design methodology. There are several profiles for that purpose (Gómez and Cachero, 2003; Koch and Kraus, 2003; Conallen, 2000) but they might not be appropriate for a different design strategy (see Section 2).

Many popular UML Profiles for specific platforms (e.g., UML Profile for EJB, UML Profile for CORBA, and so on) also exist but due to the extensibility mechanism new ones may also be created. This is how new platforms and technologies should be handled.

Therefore, UML Profiles might serve as a basis for defining a PIM–PSM tranformation. This should be defined using the concepts of the metamodels used for describing PIM (source) and PSM (target).

2 DEVELOPMENT PROCESS

Several years ago, one of the authors of this article has been envolved in a project which aim was to develop a Web-based application for the Doctoral School of Mathematics and Computer Science at the University of Debrecen. The requirements against the system were to keep track of students and teachers of the Doctoral School, and manage data of doctoral programs and courses, as well. Nowadays, almost all of the applied technologies are out-of-date but there are several new requirements that have been arisen since the system’s deployment. For that reason, this legacy system has become difficult to maintain so we considered of applying a model-based solution to reduce development time and maintenance costs.

The first step durig system design is the analysis of requirements to gather and formalize user requests. Using use cases and activity diagrams, we could determine the outline of the system and describe its fundamental functional aspects from the different users’ viewpoints.

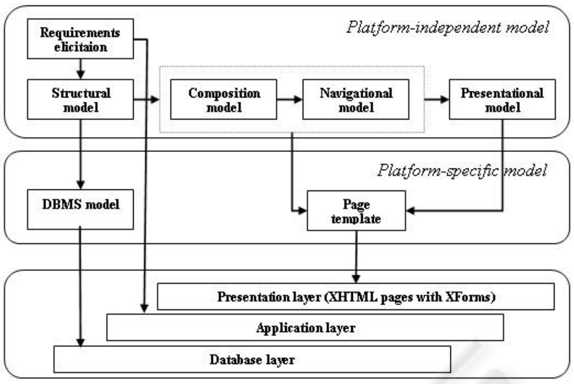

Figure 1: Phases of development.

The second step is the conceptual modeling of data structure and access paths of the application. This is achieved by UML2 class diagrams which are applying our UML profile. At conceptual level, structural, composition, navigational and presentational models are created. Several design methods (OOHDM (Schwabe and Rossi, 1998), WebML (Fraternali, 1999), UWE (Koch and Kraus, 2003), Jim Conallen’s WAE (Conallen, 2000), etc.) follow more or less the same way.

After a platform-independent model has been developed, it is subject to a model transformation which results in a platform-specific model which serves as a starting point for code generation. Our method is illustrated in Figure 1.

2.1 Models of the Design Phase

2.1.1 Structural Model

Main modeling elements of the structural model are classes, associations and packages. For a common Web application, use cases and activity diagrams are the base of the conceptual design of the domain.

However, in case of data-centric applications (which we consider), the structural model of the application domain is typically much more complex than any other models that have been mentioned previously. It is very important to create a class diagram which is of a good quality since many other activities (such as generating some primitive navigational elements like information access along associations) are relying on it. This implies that some kind of quality assurance for these models should be applied which have not been considered by this time.

2.1.2 Composition Model

Composition model is intended to define a mapping between concepts of the structural model and Web pages. One should think of a page as an object of the composition model which is associated with the entities of the structural model. Hence, a page can arbitrarily intermix information originating from multiple entities, and moreover, it is possible to extend the content with derived attributes or relationships.

An element (object) of the compositional model should be considered as the content of a page which should be rendered somehow (described by the presentational model) along with several navigational elements (given using the navigational model). This actually provides a view over the structural model elements which is extremely useful when different user groups are subject to access different page contents since several views might be defined on the same structural elements.

This model is similar to the one defined by WebML (Ceri et al., 2002) but we use UML for describing composition model elements. Classes with `<<page>>` stereotype represent the abstract information content of a Web page. (Recall that we are still not considering any presentational issues.) These classes are in association with classes of the structural model. Such associations has the `<<consists>>` stereotype.

After the structural model has been created, a skeleton for the composition model is generated which can further be used as a starting point for describing complex compositions. This idea reflects the fact that in data-driven applications, the majority of pages is close to be defined as a one-to-one mapping to structural model elements. Retrieving a PhD Student's data or adding a new supervisor are examples of such pages.

2.1.3 Navigational Model

The next stage in the development process is the navigational design. The navigational model specifies which elements of the composition model can be accessed from other parts of the application. In the navigation model's building process, the developer takes crucial design decisions such as what navigation paths are required to ensure the application's functionality. These decisions are based on the composition model, use case diagrams and navigational requirements that the application needs to satisfy.

The navigational diagram is strongly connected to the composition one since it defines the navigational paths among compositional model elements,

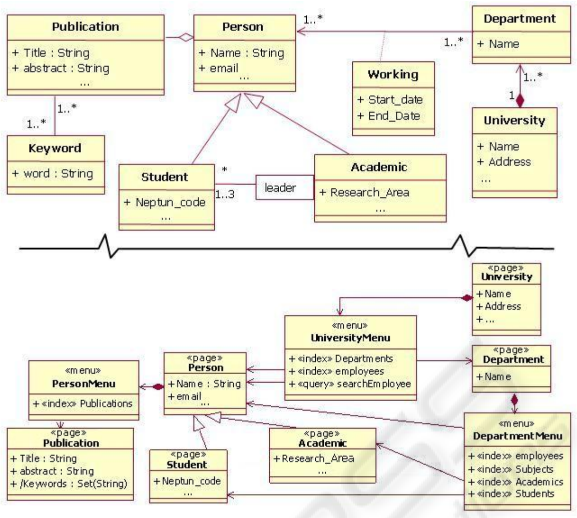

Figure 2: Structural and navigational diagrams.

i.e., abstract page views of the application. In general, new associations might be added for direct navigation to avoid navigation paths of length greater than one. However, there may be some classes in the structural model that are not subject to be a visiting target. Therefore, they should be omitted from the navigational diagram. The navigation inside the application mostly occurs along associations which are used to describe the relation between pages. Typically, these associations appear as either hyperlinks or menus in the user interface.

Subsets of the structural and navigational diagrams of the Doctoral School are shown in Figure 2. Since the navigational diagram supplements the composition model with some additional associations and classes representing different access structures such as menus, queries and indices, composition model is omitted from the figure.

2.1.4 Presentational Model

Presentational aspects are not dealt in this paper. The main goal of presentational modeling is to map the elements of the composition model to some well-known GUI primitives. This task can also be applied at a conceptual level since it does not hold information about concrete implementation. After the PIM–PSM transformation, some of those primitives might be replaced by others if the target platform does not support the given one (e.g., tree view of some hierarchical data might be replaced by a list with indented sublists).

2.2 PIM–PSM Transformations

For the transformation of the structural model we use Hibernate which provides an implicit PSM (relational model) so we does not deal with issues of this kind of model transformation. Composition and navigational models are used to formalize page templates along with presentational model. These templates are used to generate concrete pages using W3C's XForms standard for handling user inputs. The PSMs are represented with XML documents which allow XSLT to be used for the page generation from templates.

2.3 Code Generation

We use the Eclipse Modeling Framework to create the conceptual models. In the deployed application, all the communication is based on XML documents. When a user interacts with the system by filling and submitting forms, an XML document is created and passed to the server side. However, user-supplied data should be validated against the data model. Since XML documents' validation are based on any of the well-known XML schema languages (e.g., DTD, XML Schema, RelaxNG), a schema was needed. A freely available Eclipse plug-in called *hyperModel* (http://www.xmlmodeling.com/hypermodel) is able to transform a UML2 class diagram into an XML Schema, therefore we applied it in our process.

The generated XML Schema is one of the most important part of our architecture since it is used as the base of the generation of XForms pages. On the other hand, it is applied for validating all the XML documents which are used for intra-system communication, as well. XForms pages might be embedded in more complex XHTML pages which conform to the composition and navigational model defined at the conceptual modeling phase.

For creating models, OMG's XML Metadata Interchange (XMI) gives the possibility of a kind of tool-independence. As XMI is being an industrial standard for exchanging metadata of UML models in XML format, it is supported by all the major modeling tools so—in a collaborative environment—teams might use different tools. First of all, UML diagrams should be saved in an XMI-compliant format. Afterwards, XMI should be imported to Eclipse and let hyperModel do the transformation to XML Schema (`.xsd`). XForms pages are generated directly from the `.xsd` using an XSLT. The whole process is shown in Figure 3.

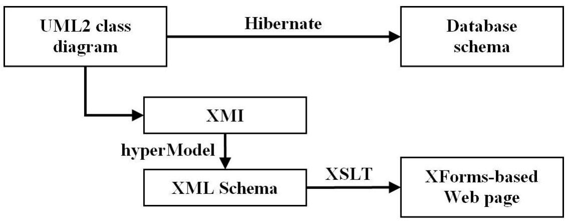

Figure 3: Using UML and XML technologies to create prototypes.

3 FUTURE WORK

The presented approach is applicable for rapid design and development of data-driven Web applications using MDA. We merely focused on building rapid prototypes based on XForms pages generated from an XML Schema.

Obviously, the current state of our work could only be a part of a more comprehensive WIS development framework. Due to the lack of space, we could only enumerate our future plans: include modeling of personalization and presentation (Fraternali, 1999); presentational aspects should cover current technologies like AJAX, OpenLaszlo, etc.; introducing the concepts of page fragments (portlets). Later on, statistical models should be applied to achieve higher quality. For the correct measurement we need to identify operational profiles and applicable statistical models to find the relevant factors and methods.

REFERENCES

Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., and Matera, M. (2002). *Designing Data-Intensive Web Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Conallen, J. (2000). *Building Web applications with UML*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Fraternali, P. (1999). Tools and approaches for developing data-intensive web applications: a survey. *ACM Comput. Surv.*, 31(3):227–263.

Gómez, J. and Cachero, C. (2003). OO-H method: extending UML to model web interfaces. pages 144–173.

Koch, N. and Kraus, A. (2003). Towards a common metamodel for the development of web applications. Cueva Lovelle, Juan Manuel (ed.) et al., Web engineering. International conference, ICWE 2003, Oviedo, Spain, July 14-18, 2003. Proceedings. Berlin: Springer. Lect. Notes Comput. Sci. 2722, 497-506 (2003).

Schwabe, D. and Rossi, G. (1998). An object oriented approach to web-based applications design. *Theor. Pract. Object Syst.*, 4(4):207–225.