Proceedings of the 7th International Conference on Applied Informatics
Eger, Hungary, January 28–31, 2007. Vol. 2. pp. 163–172.

Rapid Web application development and modeling, based on XML and UML technologies

Attila Adamkó

Department of Information Technology, University of Debrecen
e-mail: adamkoa@inf.unideb.hu

Abstract

Recently, many authors have argued that the demand for methods for the development of small and medium sized Web applications has emerged. However, the difficulty of developing these applications increases as the number of technologies they use increases and as the interactions between these technologies become more complex. In this paper we present a design method for Web applications which utilize UML and XML technologies to reduce this complexity. Because there are several different types of Web applications, we are focusing only a smaller part, specifically the Information System class of the Web Information Systems. Our approach takes into account the data-oriented aspects of these applications by creating a UML profile adapted for the problem domain.

Rapid development is enabled by providing roundtrip engineering capabilities with support for automatic code generation. We will explain the role of the XML technologies, how could we apply different transforming stylesheets to transform UML design models to different XML documents to represent platform specific models utilizing the XMI (XML Metadata Interchange) format.

Utilizing these technologies reveals an advantage of our proposed method that several steps can be performed in a semi-automatic way providing rapid development and prototyping.

Categories and Subject Descriptors: D.2.10 [Software Engineering]: Design; D.2.11 [Software Engineering]: Software Architectures; H.4.3 [Information Systems Applications] Communications Applications

Keywords: MDA, Web application, Web modelling, XML, XMI, UML

1. Introduction

With the evolution of technologies the early static web sites are changing into Web based distributed applications. However, the reorganization and the devel-

opment require knowledge and integration of several different technologies. For this reason the development is often performed by teams consisting of graphics designers through software developers.

Because short-time design and implementation are needed in response to the new technologies, development groups often disregard methodological approaches and system plans to reduce time resulting incomplete and difficulty maintainable systems. Moreover, as we could read in [1], *"Most Web developers pay little attention to requirements elicitation and analysis, development methodologies and process, quality, performance evaluation, configuration and project management, and maintainability and scalability. Furthermore, application development heavily relies on the knowledge and experience of individual (or a small group of) developers and their individual development practices rather than standard practices. These systems also lack proper testing and documentation."*

These aspects outline the demand for lightweight methods in the development of small- and medium size Web applications. An important factor in the development for such Web applications is the support of successful communication – using a common language – to avoid misunderstandings and expensive redesign. The UML (Unified Modeling Language) fulfils these requirements by providing a family of intuitive notations and diagrams which could be used to describe software systems at a high level of abstraction. These models have to be focused on the information relevant to the different roles in the development team. In many cases, we can distinguish between the domain expert, who has knowledge about the business processes behind the Web application, the graphic designer, who is in charge of the creation of the user interface, and the developer, who has to build a working software system based on the work of his partners. Furthermore, we have the customer, who has little knowledge about the technical realization of the project, but the diagrams and models helps to synchronize the requests with the software system in the early stages of the development.

Performing early demonstrations is another important factor in the course of the communication with the customer. Therefore, the methodologies should support code-generation, which could shorten production time too. The models should formulate the complete structural description of the website, resulting a working, but incomplete prototype.

Following these guidelines, the presented approach 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. These models make it possible for the team members to investigate the system from different aspects. Naturally, there are several available methodologies for Web applications and a good overview could be found in [2] where the most relevant methods, such as OO-HDM, WebML, UWE and WSDM are described.

However, most of the currently available development tools are based on the J2EE specification for enterprise applications where Web applications have three or four tier architecture. The tools claim to support the whole development pro-

cess but they offer little help for modelling the specialities of Web applications because they only include low level implementation elements like Servlets, Java Server Pages or HTML pages. We need more abstract modelling elements for navigation, presentation and user interaction but they are missing or too special for a given technology. Therefore, our proposed method deals with theses specialities of the Web applications using a UML profile adapted to the problem domain.

1.1. Web Information Systems

The scope and complexity of Web applications could vary from small-scale web sites to large-scale enterprise applications and could be grouped into several different categories. In this paper we are concentrating only a smaller part of Web applications which are defined by the Web Information System term.

We could find similar names like Web-based Information Systems or Web-based Systems which appear to be the same concept. In [3] we could find several definitions of Web Information Systems (WIS) like the following ones:

"A WIS is an Information System providing facilities to access complex data and interactive services via the Web" [4]

or

"WIS represent a sub-category of mass information systems that are typically support on-line information retrieval and routine task by way of self-service for a large number (thousands or millions) of occasional users who are spread over many locations" [5].

In our point of view the WIS is a computer-supported information system utilizing the technology of the World Wide Web. In [3] we could find further categorization of these systems depending on the direction of the communication and the characteristic of the information.

	Asymmetrical communication	Symmetrical communication
Objective information	**Information Provider**	**Information System**
Persuasive information	**Advertisement**	**Community**

Figure 1: Four perspectives on WIS

As the primary goal of us is to distribute information and concerning with the data manipulation tasks too we shall deal with Information Systems hereafter or as we called them formerly Data-oriented Web applications. The focus here is on the derivation of the data structure and the distribution of the information from the system to the users to support their work.

Using this perspective, typical examples of WISs are on-line systems, timetables, registering systems, flight booking, etc. The objective of the system is to provide a platform for users to reach their goals.

2. Design Strategies

There are numerous different approaches available for modeling Web applications as we mentioned in the introduction. Some of them focus on the modeling notations, while others focus on the development process. Of course, Web application development is not only supported at the conceptual level, but there are several software tools available also, like Together Designer from Borland and Rational Rose from IBM.

However, the methodologies and the tools sometimes do not follow the same way. We could apply a methodology or use a given tool if they meet our requirements, but usually we need to make trade offs because something will not appropriate for us. Moreover, there are several low-level design considerations built into these development processes, so if we choose one our hands will be tied to it.

In contrast, if we would like to achieve greater independence in the development process we need to use tools that could be independent from specific low-level things. Here comes into the picture the MDA (Model Driven Architecture) development process. Within MDA the software development process is driven by the activity of modeling your software system.

2.1. MDA

The MDA development lifecycle is very similar to the traditional lifecycle. The same phases are identified, but the main difference lies in the nature of the artifacts that are created during the development phases. The artifacts are formal models providing a given aspect of the system.

The first MDA model is a model with high level of abstraction that is independent from any implementation detail. This is called a Platform Independent Model (PIM). The PIM models are giving viewpoints of how the system will support the business. These models do not take care of implementation details like relational databases, application servers and so on. They just concentrate on the formalization of the requirements.

The next step is the transformation of these models into one or more Platform Specific Models (PSMs). The transformation process will deal with the available implementation technologies. For each specific technology a separate PSM is generated because most of the systems today span several technologies.

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.

The MDA also defines how these relate to each other. The most complex step is the transformation of the PIM into one or more PSMs. In our point of view – development of WISs – we need to consider the development processes and design strategies of Web applications.

`MVC` We know that several similar problems could occur again and again in the development process and we know that the design patterns could help to reuse successful design and architectural solutions. In the design of Web applications a

useful way is indicated by the Model-View-Controller (MVC) design pattern which is adopted in several frameworks. It can improve the application's usability, the creation of reusable code and helps to understand and clarify the functionality of the program. The MVC pattern is very simple, yet incredible useful. Its importance lies in the clear separation of the functional layers and their functionality.

However, in data-oriented Web applications, the model component becomes a little simpler because small and medium sized Web applications mostly have simpler business logic. Additionally, one of our main purposes is to derive a proper data-structure which could be used in the subsequent steps. Furthermore, in the model construction stage, the primary principle is not the description of the corresponding object's behavior, rather than the effective access and management of the data because we are building systems around the data being managed.

3. The design method

The presently available methodologies are mostly using object-oriented approaches, cleanly separating each component's functionality. The design of Web applications builds on the requirements specification, just like the design of software systems in general. In these systems the conceptual design of the domain is based on use-cases and includes the involved objects that users will perform with the application.

Particular emphasis is placed on the information exchanged between the user and the system. The use case models serve as an input for modeling the content of the application. However, in data-oriented cases it has proved to be an effective strategy to build the object-oriented approach over the data-oriented, i.e. the data structure will be the basis and the application will be built around that structure.

We may have a better understanding of the meaning of data-oriented approach when we think about an existing database that needed to extend with a Web interface. In these cases, the use-cases highly depend from the data model, but the situation is the same when we need to develop new Web applications for managing registration and information systems.

In our approach these use-cases will be used to outline the *application* tier in Web applications or if think on the MVC pattern, the Controller module will be formalized with these use-cases because this module is responsible to handle user operations. These operations will be manipulating the data contained in the *data tier*. This data management layer could be described by structural models, typically class diagrams. The class diagrams also could be used automatically to derive the corresponding relational database schema (which is a PSM).

The topmost *presentation tier* of a Web application is created by the Navigation and Presentation model together. The Navigation model is used to describe at each class which related classes can be visited from it (based on the associations). The navigational elements between the classes could be realized with menus and links. In the final step, the user interface is described by the presentation model, i.e. how the requested data could be transformed together with the navigation structure to

the client's language.

3.1. UML as PIM modeling language

The MDA approach requires a language which uses formal definitions so the tools will be able to transform these models automatically. OMG recommends the Uniform Modeling Language to construct the Platform Independent Models. The strongest point in UML in our case is the modelling of the structural aspects of a system. This is done through the use of class diagrams which enable us to generate PSMs with all structural features in place.

In particular, we could extend the PIM's structural class diagram with navigational aspects in order to derive the navigational model too. This could be seen in Figure 2.

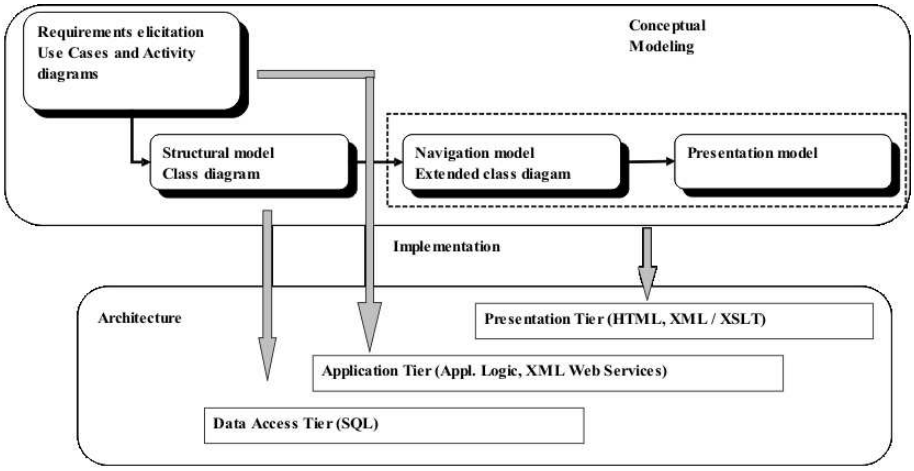

Figure 2: UML as MDA meta-model

3.2. Modeling steps

Our approach follows the Rational Unified Process, so the first step in the system design is the analysis of requirements to gather and form the user requests. Using use-cases and activity diagrams, we could determine the outline of the system, and describe the fundamental functional aspects of the system from the perspectives of different users.

As we have data-oriented approach, the generality of the available tasks are related by data management and manipulating activities. The actors represent the different roles for users of the Web application. Use-cases are used to describe available operations, and have to be detailed by activity diagrams. These activities will produces the entry points of the navigation diagram for each user role and the use-cases will serve the layout for the opening page for each actor (containing the list of available tasks/entry points). The activity diagrams are used to describe

business logic and furthermore, they could be used to derive program modules and code skeletons. Of course, we have a simpler case when we need to develop a system for an existing database. In this situation, the structural model is deeply bounded for the data structure, and we need to focus on the performable data management tasks.

Additionally, there exist further important factors like performance or availability. These aspects could influence our modeling method, and further researches will consider these factors. However, at this time we are focusing on platform independent models leaving untouched state models and page handling scenarios (sequence diagrams).

3.2.1. Structural model

The conceptual design aims to build a domain model trying to take into account as little as possible the navigation paths and presentation aspects. The main modeling elements used in the conceptual model are: class, association and package. For a common Web application the use-cases and the information-flow descriptive activity diagrams could be the base of the conceptual design of the domain.

We could use an incremental approach to identify classes. First, we need to identify the "active" entities in the system. At first glance, the actors identified in the use-case appear to be prime candidates for being listed as potential classes. Next, we need to identify the business domain ("passive") entities in the system. Usually, these business domain classes are mapped to either one or more database tables.

However, in data-oriented cases the base of the conceptual model should be the managed data, not the typical user activities and related use cases. So the modeling sequence will be modified a little. At first, the information carrier classes and attributes are determined, and only after this point will be detailed the use-cases describing the user activities. The result of the first step will be a class diagram which will determine the structure of the system, the classes and their relations (like association, aggregation, ...). In this stage we need to ensure only the data management activities to the users, so the tasks list will be shorten.

3.2.2. Navigational model

The next stage in the development process is the navigation model design. The navigation model specifies which objects can be visited in a Web application. Moreover, it defines the availability of the objects. In the navigation model's building process the developer takes crucial design decisions, such as which view of he conceptual model is needed and what navigation path are required to ensure the application's functionality. The decisions are based on the conceptual model, use-case model and navigation requirements that the application must satisfy.

The navigation diagram takes a structural diagram copy as its starting-point, which could be extended with additional associations. Generally, these new associations should be added for direct navigation to avoid navigation path of length

greater than one. However, there could be some conceptual classes that are not a visiting target in the use-case model. It is irrelevant in the navigation model and therefore it is omitted in the navigation diagram. The navigation inside the Web application occurs along the associations, which are used to describe the relation between navigation classes. These associations will appear as hyperlinks in the user interface.

We could design proper models if we extend our navigation model with some notations, like index, query and menu as we can found in UWE [6]. This new model could serve as a background of the automatically generated navigation system. Unlike UWE, we do not use new graphical notations in the diagrams; instead we pick up new attributes extended with stereotypes to describe their functionality in the navigation.

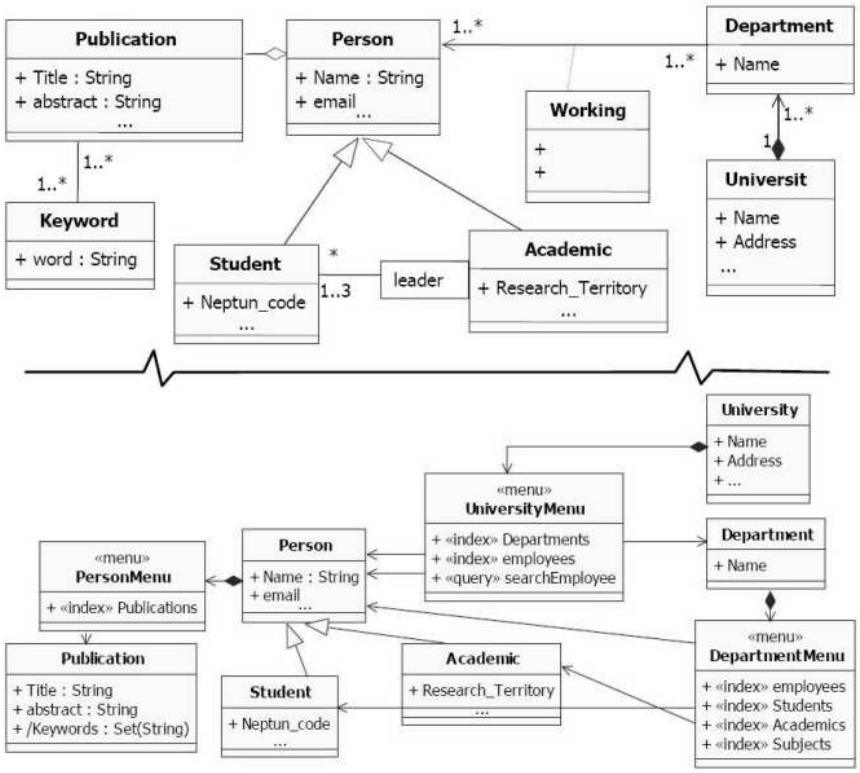

Figure 3: Structural and Navigational diagram

3.2.3. Presentation model

Our methodology does not deal with presentation aspects because the answer for each request could be presented with an XML document containing the result and the navigation structure starting from this entity. This XML document could

be transformed to the desired output format using XSLT. The structure of the opening page for each user also could be derived from to related use cases.

4. Code generation

These models help to comprehend the problem domain, but these models would offer more complex support if we could generate from them a working prototype of the desired Web application. Naturally we agree that UML models are supposed to be abstract, but it is not uncommon for a UML model to capture almost as much technical information (transforming PIM to PSM). Our approach could be seen in Figure 4.

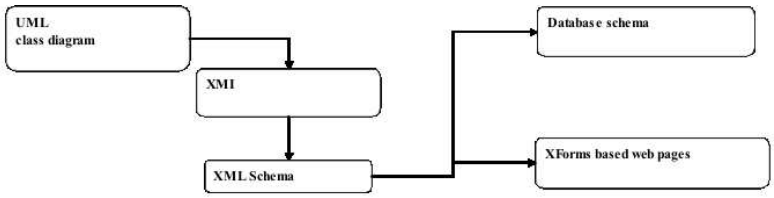

Figure 4: Using UML and XML technologies to create prototypes

5. Further possibilities

In this paper we have illustrated how data-oriented Web applications differ from traditional software, how complex and not at all systematic tasks. We have introduced a new methodology to help develop Web applications rapidly and effectively based on UML and XML technologies supporting data management task in small and medium sized projects. We have added some remarks in the implementation phase utilizing XML technologies to develop modular, scalable and expandable Web based systems. Ongoing researches can go in several interesting research directions in the design and development phase. We are going to study the additional expandability of our UML based methodology.

References

[1] Ginegi, A., Murgesan S., The Essence of Web Engineering, in IEEE Multimedia, Vol. 8 no. 3 (2003).

[2] Schwabe, D., A Conference Review System. *1st Workshop on Web-oriented Software Technology*, (2001).

[3] Holck, J., 4 Perspectives on Web Information Systems, *in Proceedings of the 36th Hawaii International Conference on system Sciences,* IEEE, (2002).

[4] Gnaho, C., Web-based Information Systems Development – A User Centered Engineering Approach, *Lecture Notes in Computer Science*, (2001).

[5] Bauer, C. et al., Matching Process Requirements with Information Technology to Assess the Efficiency of Web Information Systems, *Information Technology and Management* 2, (2001).

[6] Hennicker, R., Koch, N., A UML-based Methodology for Hypermedia Design., UML'2000, LNCS 1939, *Springer Verlag*, (2000), 410–424.

[7] W3C - World Wide Web Consortium, `http://www.w3.org/`

[8] Schattkowsky, T., Lohmann, M., Rapid development of modular dynamic web sites using UML, *In Proc. of 5th International Conference on UML 2002*, Springer, LNCS 2640, (Oct. 2002), 336–350.

Attila Adamkó
Department of Information Technology
University of Debrecen
H-4010, P.O. Box 12, Debrecen
Hungary