DEVELOPING WEB APPLICATIONS USING THE MDA PARADIGM

Lecturer Ph.D. Mihai-Constantin AVORNICULUI
"Babeş–Bolyai"University Cluj-Napoca, Romania
mihai.avornicului@econ.ubbcluj.ro

Abstract:
Data intensive web based systems, that in the most of their uptime do data processes have become a part of our everyday life. Several systems of this type appeared on the internet: webshops, online ticket shops, banks, etc. These are used everyday by users around the worls. The use of model drive approach brings several advantages that ease up the development process. Working prototypes that simplify client relationship and serve as the base of model tests can be easily made from models describing the system. These systems make possible for the banks clients to make their desired actions from anywhere. The user has the possibility of accessing information or making transactions.

Keywords: Web application, MDA, MVC, Banking services

JEL Classification: C 88

INTRODUCTION

In recent years Web applications have become a part of our everydays. On Internet have appeared many applications like these.

Modeling such a system is a very complex task, because to be able to develop a felxible and upgradeble system we have to take in account the existing and the future technologies too.

The model-driven approximation has many advantages even when is used to create Web applications, because we can quickly generate a working prototype from the models which describe the plan, and on the basis of these models we can make model-revisions.

In a short time this paradigm has become very popular and so is involved in many areas of life bringing a new line of sight in application development too.

Because of the Web applications' diversity it is becoming more and more difficult to choose the required technologies and platforms.

It is possible that at the end of the development, after installation, the application outgrow the proposed framework. The problem in this case is that maybe we have to change the platform and this may mean that the whole application must be reimplemented.

In better case changing the platform can be easily achieved, but the used technics may become out-of-date and so they will not be compatible with new technologies. This fact may imply several problems, for example after a while the solution's support will leave off. On the other hand new developers may get into the team, who don't know the older technics and so the improvemnet will be more difficult.

Practice shows that the application's logic should be formed irrespectively of technological details.Even if we use traditional application development methods, it is familiar that after the model was finished and once implemented, in the next steps will not be used but simply set aside. So, between the model and the code it can't be developed a relationship, so the change within the model doesn't appear into the code and vice versa. Because of this the model will be unable to perform his original function, to provide determinative informations about the system.

THE MVC ARCHITECTURE

To eliminate the aforementioned problems, during the design process we must place particular emphasis on the separation of the content and the visualization, thus we can reuse these components.

During the development of a Web-based application the most frequently used design pattern is the Modell-View-Controller. This one efficiently increases the application's usability and helps us to understand the accurate operation of the program.

This pattern is simple and efficient and the resulting code is reusable.

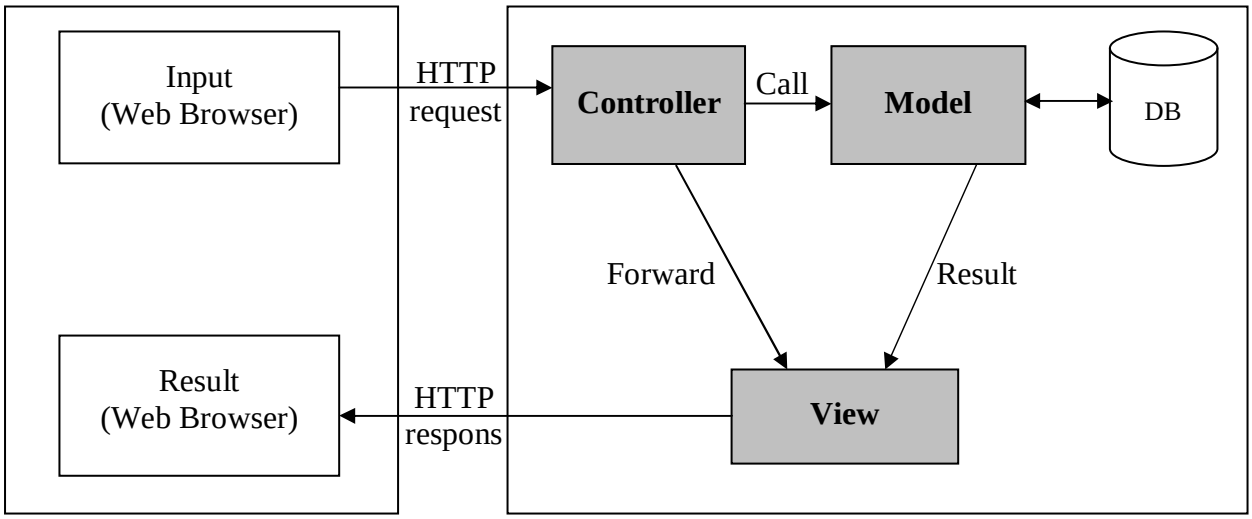

Figure 1. The MVC architecture

On server side the Controller receives the requests, forwards them to the model which executes the requested task.

According to user commands respectively to the Model's result, it is practical to let the Controller choose the fitting View.

Using the MVC we have the following benefits (Adamkó and Kollár, 2008):

- **Modularity**: allows to replace either component
- **Reusability**: we can reuse the already existing code
- **Easy expandability**: the Controller and the View can be expanded along with the Model
- **Centralized controller**: with the help of the Controller the manageability becomes easier

In case of Web applications we can't only lean on MVC design patterns, because we may forget the apllication's several other features. As a result the application may be difficult to develop and to maintain. Practice has confirmed, that MVC patterns can successfully used for developing web applications combined with other well known tools.

THE MODEL-DRIVEN ARCHITECTURE

The Model-driven architecture (MDA) is a new application development approximation which in recent years has become a paradigm too. The most important assumption of this paradigm (Schmidt, 2006) is that the designing and the development in fact mean creating models with different level of abstraction.

An MDA specification is composed of a platform independent (PIM) model and one or more platform specific models (PSM).

According to MDA paradigm, first of all will be finished the examined system's computational independet model (CIM) which illustrates the business logic without reference to the machine.

During the design of Web applications, after finishing the CIM we need to create the models below (Valverde, 2007; Adamkó and Kollár, 2008):

- **Structural model**: which describes the objects of speciality concept and the structural relations between them
- **Navigational model**: which presents the Web application's reticulated structure

- **Component model**: wherein is written those classes' organization into bigger logical units, which represent the functionally cohesive concepts
- **Presentational model**: which is an abtsraction of the user interface and it is in elementary relation with diplayed components

In fact a PIM is a full specification which is platform independent. A PIM model must be transformed to a PSM model which insures the infrastructure of system architecture. This transformation means the implementation of PIM, i.e. the executable application's generation.

The PIM allows the solution's visual modeling on a higher level of abstraction. Thus when a new technology appears isn't necesary to rewrite the application, we only have to regenerate it.

The figure 2. demonstrates the concept of the MDA:

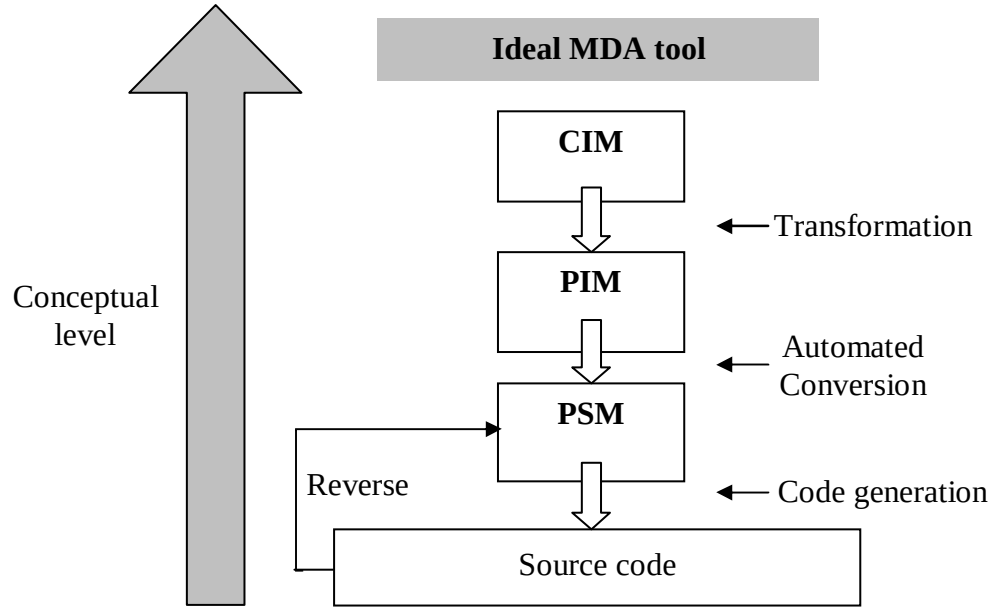

Figure 2. The concept of the MDA

The most difficult is to convert the PIM model to PSM model, because several tools don't completely support the conversion (Avornicului, 2010).

THE DEVELOPMENT PROCESS

As in case of any other application the first step is the requirements' analyses.

With the help of Use Case diagrams as well as Activity diagrams we define the basic functionalities of the system. The differing roles of the application's users are represented by the *actors* while their activities are represented by the *use case*s.

The second step is the conceptual modeling of the application's data structure and access paths.

After the platform is finished using an independet model, we make the platform-specific model by the help of a model transformation. This one can be used to generate code.

THE DESIGN STEPS

The planning process is composed of several complex workflows. In case of a Web application the analyses and the design are closely linked activities.

During the stpes of design, analyses and requirement definition we have to make the models below:

- Use case models
- Conceptual model

- Navigation model
- Component model
- Presentation model

In the course of an iterative process these models will increasingly include the speciality requirements. The analyses of requirement will be drawn up under the form of use cases. Our conceptual model will be trained from these, and will give the relationships which are determinant from the viwepoint of the specilaity. This can be viewed like a platform independent model. From this one will be derived the navigation model and the component model, and we will not yet take into consideration the technology and the implementation's aspects.

The purpose of the representation model is to replicate the component model's elements to other ones which bind to different well known grafical user interfaces. The platform-specific model will be derived from these. In the following we present a case study in which we follow the stages of designing the mentioned application.

CASE STUDY: ONLINE BANKING SERVICES

Using the application the clients have the possibility to open charge accounts, to pay in money, to transfer money to another account whithin the bank, as well as to abolish their own account. One client can open only one account but one account may be owned by more people.

Our first step is the requirement's analyses. This is that step during wich we define what kind of application we have to make. If we made a Web application, we may have several problems:

- A Web application may have one or more entry points
- The resources and the technologies change too fast
- There is no well developed systematic guide-line

The use cases help us to explore the system requirements. In our case the use case diagram may be like this one below:

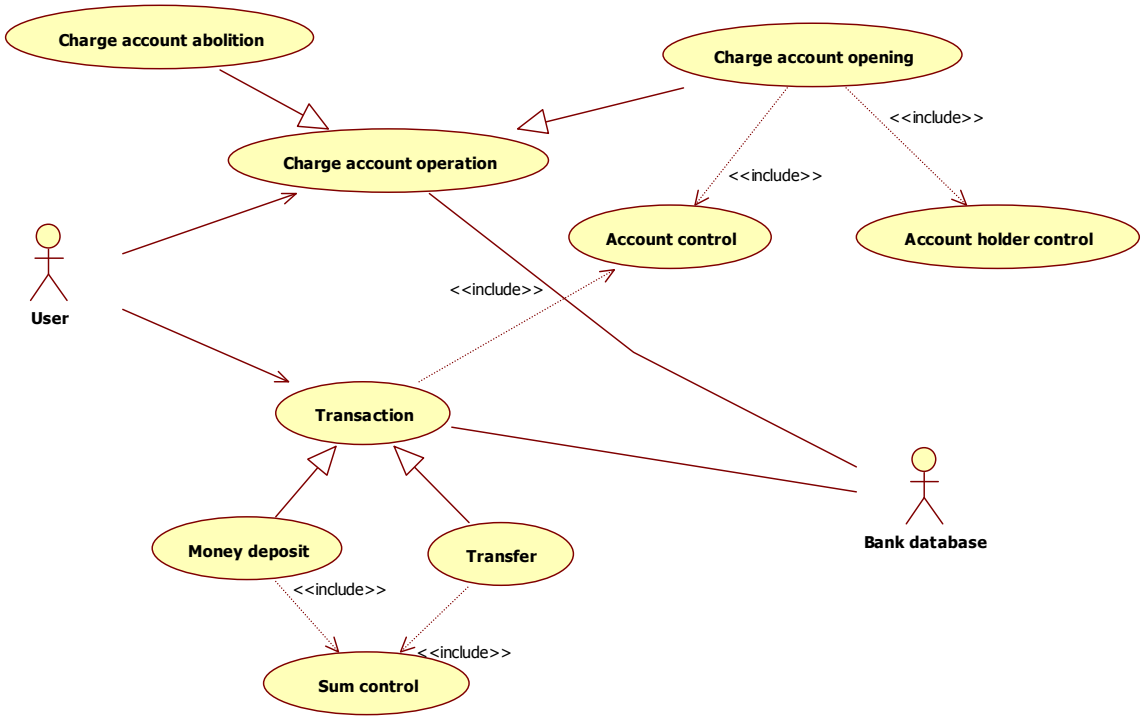

Figure 3. The user's functions

Usualy the requirements are summarized by not only one people because this is a complex task.

Making the conceptual model will be the second step. In this phase the goal is to make a model which include the relevant conceptions to the Web application.

During this phase we define the classes, the important attributes and methods. To make the speciality's structural model we usualy use classes and packages.

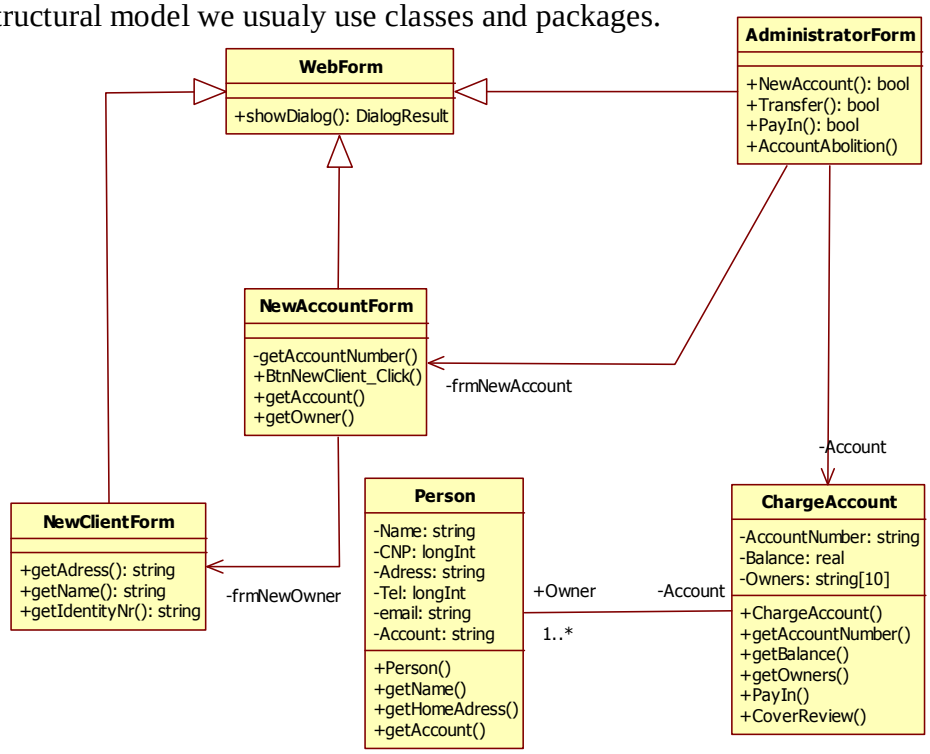

Figure 4. Structural model

The third step is designing the navigation, which is a crucial step in making a web application. In this step we produce the application's structure and the designers have an inportant role, because they have to decide which classes will be used and what sort of navigation paths will set up. These will be set up according to requirements fixed in the conceptual model as well as in use cases. The next figure demonstrates the navigation diagram in case of our system:

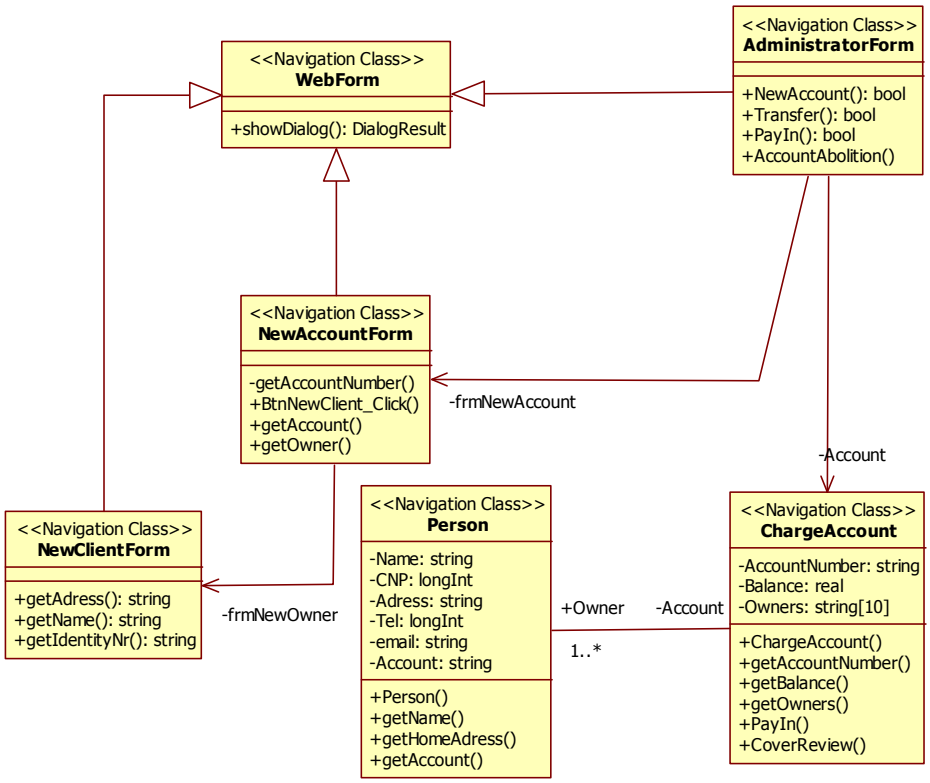

Figure 5. Navigation diagram

As shown in figure 5, in order to reflect upon the navigability within the application, class diagram was used in which classes and relations are used. Relations express navigability within the application.

During the previous steps we have made the most abstract model of our application. The next step is creating the component model. The components can be used in all that places where the classes appear.

The figure below contains that classes which fit the *account module:*

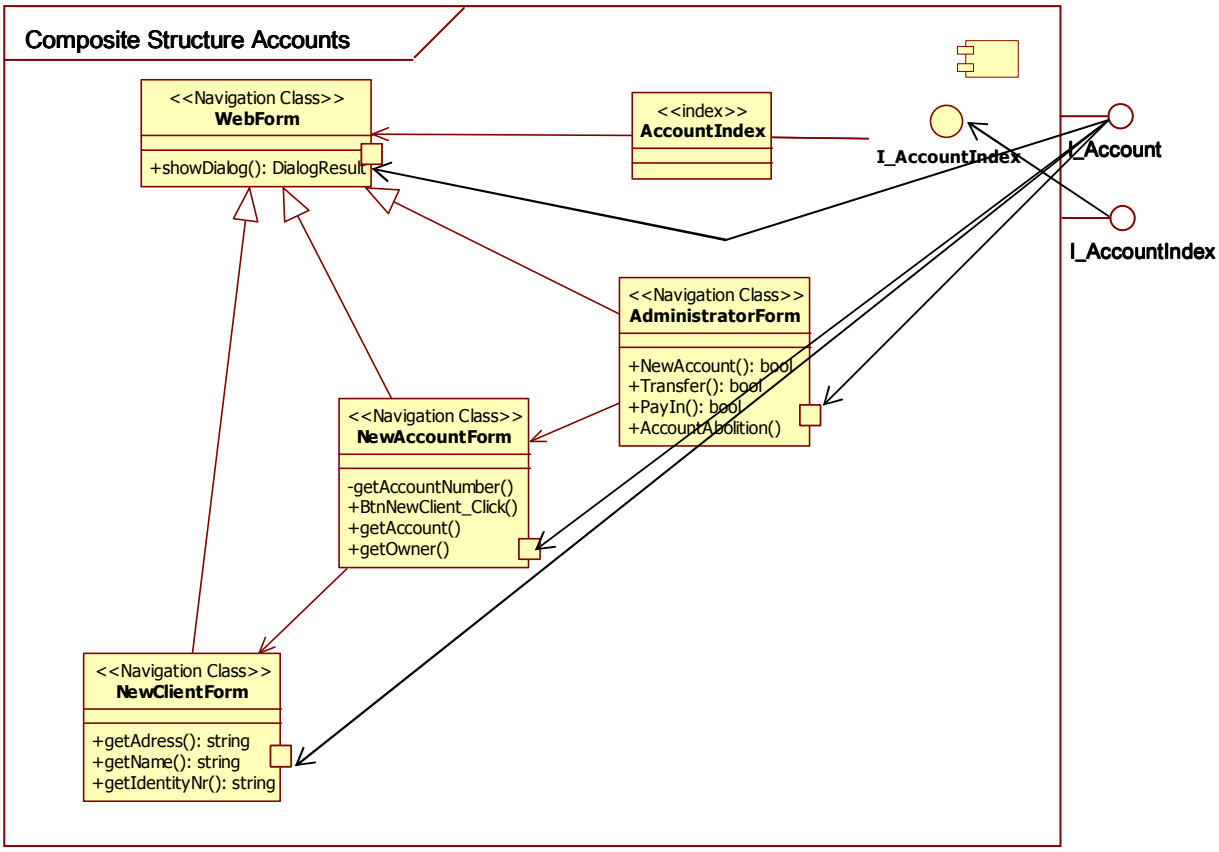

Figure 6. The account's components

One component may appear in several roles within the conceptual model.

The next step is creating the presentation model. Using a PIM-PSM transformation we replicate the components to a given platform. In this model new structures are not introduced, but ole components are reorganized.

In UML components are universal. Components can be used only in places where classes exist. Following the MDA paradigm we can use numerous transformations to achieve a PIM-PSM transformation.

According to the component, navigation and presentation models we derive XForm-type pages. The XForm is a dynamic and scripting language independent system which is integrated in another language (Valverde, 2007).

The advantages of XForm (Valverde, 2007; Adamkó and Kollár, 2008):

- Developing the MUC model
- Is a decorative language
- Is compatible with XML standards
- Is extensible

Its only disadvantage is that isn't spread enough to be natively implemented in browsers.

By the help of XForms we can make forms that allows the client to easily create XML documents, by filling and sending these forms. In this way the data input is easy to realize.

CONCLUSIONS

The model-driven application development brings the solution's specification closer to formulate the problem. The MDA paradigm is able to organize and develop data-oriented Web applications.

In case of a Web application the individual development phases are complex and not always systematic. The described steps in this article can be applyed efficiently to develop a comprehensive Web-based system.

REFERENCES

1. Adamkó A., Arató M., Fazekas G., Juhász I. (2007) 'Performance Evaluation of Large-Scale Data Processing Systems' Proc. of the 7th *International Conference on Applied Informatics*
2. Adamkó A., Kollár L.(2008) 'MDA-based development of data drive Web applications' Proc. of the 4th *International Conference on Web Information Systems and Technologies*
3. Avornicului C., Avornicului M. (2006) 'Aspects of using MDA paradigm in sistem development', *Proceeding of International Conference on Business Information Systems*, Iaşi
4. Avornicului C., Avornicului M. (2010) 'Managementul şi proiectarea sistemelor informatice de gestiune' *Editura Risoprint*, Cluj-Napoca
5. Hazra T. K.(2002) 'MDA brings standards-based developing Modeling to EAI Teams', *Application Development Trends*, May
6. Kennedy A. (2004) 'Model Driven Architecture and Executable UML – The Next Evalutiopnaryx Steps in System Development? ', *Szoftvertechnológiai Fórum*, Budapest
7. Raffai M. (2005) 'Az UML 2 modellező nyelv' *Szakkönyv Kiadás alatt, Palatia Kiadó*, Győr
8. Schimdt D. C. (2006) 'Model-Driven Engineering', *IEEE Computer*, 39(2)
9. Valverde, F., et al. (2007) 'A MDA-Based Environment for Web Applications Development: From Conceptual Models to Code' in 6th *International Workshop on Web-Oriented Software Technologies* (Pending Publication)
10. www.omg.org/docs/formal/03-03-01.pdf