A Model Driven Methodology Approach for e-Learning Platform Development

Rachid Dehbi, *IACSIT Member*, Mohamed Talea, and Abderahim Tragha

***Abstract*—Nowadays, new technologies and platforms are emerging and changing constantly, which implies a high effort for developing of complex systems such as E-learning platform.**

This situation generates different problems related to portability, reusability, adaptability, integration and interoperability. The Object Management Group (OMG) proposes the Model Driven Architecture (MDA), which improves portability of applications by allowing the same model to be realized on multiple platforms. Such MDA defines an architecture based on platform independent models (PIM) and platform specific models (PSM). The component approach aims to design and develop systems from prefabricated components, predesigned and pretested, to be reused in other applications, which would facilitate application's maintenance and evolution, would promote adaptability and configurability in order to produce new features. In this work we present LMSGENERATOR, a multi-target Learning management system generator with a model-driven methodology based on MDA approach coupled with component approach. Based on generative programming, from user specifications (abstract models) and the desired technologies, software bricks will be generated and assembled to produce a complete solution adapted to area and users' needs. This paper focuses on the transformation rules implemented in the LMSGENERATOR cores .Also; it presents a case study to illustrate this proposal.

***Index Terms*—E-learning, learning management system (LMS), model driven engineering (MDE), model driven Architecture (MDA), transformation approach, XML technologies.**

I. Introduction

The major drawback of information systems is their increasing complexity and rapid scalability. Dealing with this situation, researchers and industries have agreed on the fact that the solution of this problem should result in a rise of models, and a clearer separation between business and technology. This decision has helped to pass the models from their contemplative phase, which was reduced to the representation and documentation of computer systems, to a more productive phase which envisages the use of models in the heart of the systems development cycle. That's why the Model Driven Engineering (MDE) has been birth.

The Model Driven Engineering (MDE) is a recent discipline of software engineering that promotes models in first-class entities in software development [1]. It is the subject of great interest from the academic research teams (IDM-Action [2]) and industrial laboratories (Compuware [3], Softeam [4], AndroMDA [5], Xactium [6], etc…).

It is a form of generative engineering, by which all or part of a computer application is generated from templates [7]. In this new perspective, models occupy a prominent place among the artifacts of systems development and in exchange must be sufficiently precise and rich, so they can be interpreted or transformed by machines. The process of system development can then be seen as a sequence of transformations [7] of partially ordered models, each transformation taking one or more models as input and producing one or more models as output, until executable artifacts. However, the design and programming of complex applications, such as E-learning platform, Faced with the growing and changing needs of the area, has to use the standards, and has to adopt the new practice of software engineering: Model driven Engineering, Generative Programming and Component Engineering.

This article introduces the basics of a new approach [8] to program virtual learning environments. Based on generative programming, from user specifications (abstract models) and the desired technologies, software bricks will be generated and then assembled to produce a complete solution adapted to area and users' needs. This idea is implanted at different levels in the design and in the process of a Learning management system generator called LMSGENERATOR. This work shows how the model driven engineering (MDE) in particular the OMG vision of MDE (MDA) and the Generative programing can be combined to implement this system.

This article is organized as follows. In section II, we define key concepts and bases of Model-Driven Architecture. Section III gives an overview of our approach and its automatic implementation: LMSGENERATOR. In section IV, a part of case study will be presented. Finally, section V sums up the main conclusions and future works.

II. MDA

Convinced that the model has become the major paradigm by which the software industry can lift the latch automation development, the OMG (Object Management Group) released in 2000 his vision of MDE: MDA (Model-Driven Architecture) [9]. This is both a proposed architecture and a development approach.

The basic idea of MDA is to separate the functional specification of a system from details of its implementation on a specific platform. For this reason, MDA defines an architecture specification structured in several types of

Manuscript received September 15, 2012; revised December 12, 2012.

The authors are with Faculty of Science Ben M'sik, Hassan II university, Casablanca, Morocco (e-mail: dehbirac@ yahoo.fr, taleamohamed@yahoo.fr, a.tragha@univh2m.ac.ma).

models [10]: Computational Independent Model (CIM), Platform Independent Model (PIM), and Platform Specific Model (PSM), generated from the PIM based on the PDM (Platform Description Model). The CIM models are the requirements of a system, its purpose is to assist in understanding the problem and to establish a common vocabulary for a domain. In UML, the use case diagram is a good candidate to represent a CIM. The PIM, also known as the analysis and design model, is an abstract model independent from any running platform.

The PIM is designed to describe the know-how (skills, expertise) or business knowledge of an organization. Having isolated the business expertise in PIMs, we need either to transform these models into other PIMs for interoperability needs, or to produce PSM models for a specific running platform based on PDMs to improve portability and increase productivity. The PDM focus on modeling the platform on which the system will be executed (component models at different abstraction levels: PHP, EJB, .NET, etc…). Specifically, it defines the various features of the platform and specifies how to use them.

III. LMSGENERATOR: A MODEL-DRIVEN METHODOLOGY

A. Architecture and Functionality

LMSGENERATOR is the automatic implementation of new learning management system programming approach. Based on new development approach of software engineering, it allows, from a business model repository, to generate business components, and then proceeds by assembling these components to generate E-learning platforms. It proposes a multi-programming approach [8] that merges the model-driven engineering, generative engineering and component engineering. This compilation of methods and techniques makes our approach more appropriate to the evolving needs of the domain and technology towards more classical and general approaches.

LMSGeneratoris [11] is an environment used for integration of business components and generation of other components from model. It aims the generation of E-learning, executable on multiple runtime environments and representing criteria of adaptability to different user categories. Based on multi-layer and interconnected software components (Fig. 1), this generator provides these users the ability to:

- Maintains a repository of conceptual components (business model repository) (PIM) durable, customizable, flexible and reusable as well as an extensible repository of platform execution model (PDM).
- Applies transformation, generation and refining rules into models (element of business model repository) in order to generate a business components runtime on the target platforms, using PDM.
- Integrates existing components in its business components repository.
- Describes and maintains models (structure) of E-learning platform and stores them in a descriptive repository for use as needed.
- Generates distance learning environments based on predefined models in the platform descriptive repository or by generating directly a new platform with the ability to save its structure in the dedicated repository.
- Deploys the environments in target execution platforms (on specific application servers and database servers).

LMSGenerator based on an infrastructure using a business model repository and a technological model repository, whose alimentation is currently the subject of our research [12].

This infrastructure is composed of two component categories. The first one involves in the generation process based on model driven architecture approach and permitting, through mechanisms of transformation or fusion, to define and generate specific business components (brick) adapted to the construction of a learning management system. While the second use the art of reusable components engineering and proceeds by assembling, adapting and refining, the generated component to produce each time a new device for learning adapted to the needs of training organizations.

The following figure identifies the software components that support the two generation phases of LMSGENERATOR which will be detailed in the following section, in particular MDA generation phase.

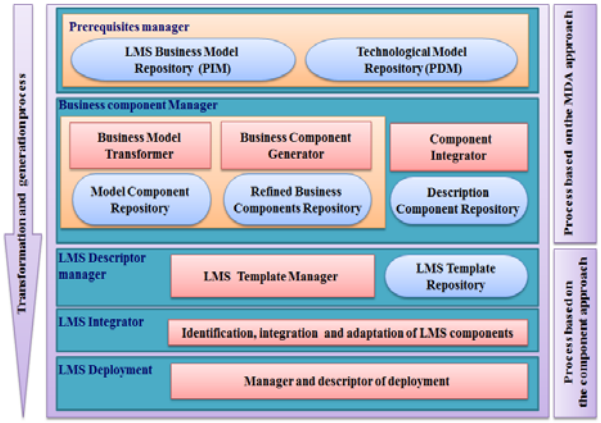

Fig. 1. Lmsgenerator software architecture

B. Generating Process Based on MDA

This phase aims to build LMS business components representing criteria of adaptability and dependent on a target technology (target runtime). The starting point of this phase is the elements of our business repository. This repository is composed of business model designed to be used in the web. For this reason we opted for models that must respect the MVC paradigm (Model-View-Controller) [13].

This programming schema proposes the separation of the application into three parts:

- The model, which contains the logic and the state of the application;
- The view, which represents the user interface;
- The controller, which handles the synchronization between view and model;

The essential point of this paradigm is to separate the graphical objects of business objects, so they can evolve independently and reused. It is in this sense that we proposed in our approach a necessary and sufficient modeling process to effectively build our business components. For this, we used a subset of UML models to describe our business.

Therefore our business model consists of three sub models:

- The business logic model to describe the business logic component.
- The application logic model to structure and synchronize between the business and the presentation of the component.
- The presentation model to define the user interface, regardless of the display means.

Each of these three models is structured under package format covering the various UML diagrams of the analysis and design stage, and using a good presentation of the business model needs.

Once the model is presented, it undergoes a generation process based on MDA approach supported by two software components of LMSGENERATOR: Business Model Transformer and Business Component Generator. Starting from a conceptual business model(element of Business Model Repository), we can apply a set of transformation, derivations rules and successive enrichments to produce a refined LMS business component(element of Refined Business Component Repository) , ready to be used as a fundamental building block in the construction of our learning platform.

This generation phase consists of two basic steps. The first step aims to generate from a business model a component model independently of the technology expressed in XML file format, while the second uses the XML file to generate a business component depending on a target technology. The first step is based on a generation process based on model transformations adopting the transformation by programmation approach using APIs manipulating UML models provided in development environments.

Thus, with minimal effort, a component model independent from technology can be quickly obtained by a series of transformations, mergers or refinements of the three sub models mentioned previously (Fig. 2). This transformation by programming approach is the charge of Business Model transformer (Fig. 1). This component model, as close as possible to the needs of the application, specifying the services provided and required, enter, in its turn, in a generation process supported by Business Component Generator (Fig. 1), adopting the transformation by template approach. This second step is based on a set of conceptual models for technology development defined in our technological model repository (Fig. 1).

Starting from the description under XML file format of the component independently of technology, Business Component Generator identifies the XML element and replaces its abstract description with a concretized description and dependent technology, until obtaining finished artifacts of the business component for a specific platform execution and the XML file description of the component. This XML file contains a detailed description of the internal components of the business component (web pages, business classes, control classes and the SQL schema). It is also used to feed the component description repository used at the time of platform construction. Once business components are generated, they are refined and ready for use by the process of LMS generation of the second phase. The figure below, present a partial view of the generation process based on MDA approach.

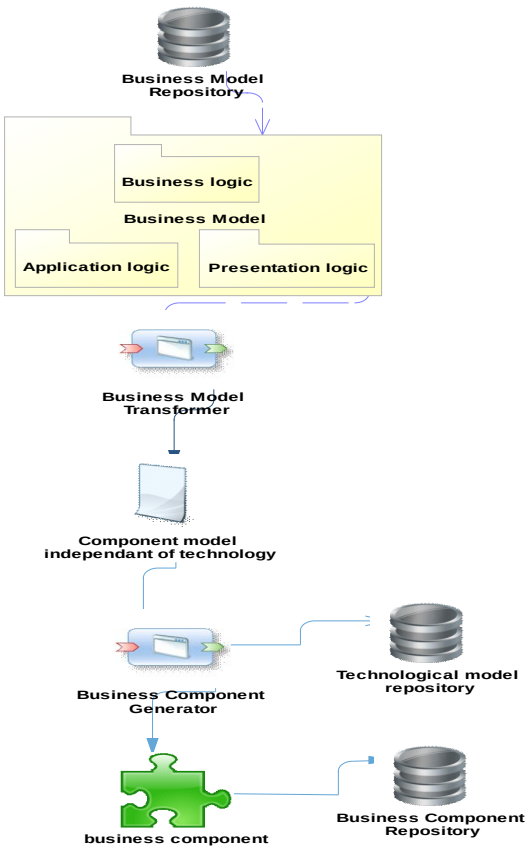

Fig. 2. Generation process based on MDA approach of lmsgenerator

IV. Case Study

A. The Model Specification

In this section we have chosen as business model, a portion of the communication model presented in [12] that would serves as the forum model. This asynchronous communication mean is considered as a stand-alone model and plays at the same time an important role in E-learning platform. After processing, we will be given a reusable business component that can be easily integrated into any platform. The main requirement of e-learning platforms, which is a part of collaborative tools, is the discussion forum. The same tool is used in web applications based on collaborative work.

Discussion forums - or news groups - can create thematic exchanges places where everyone can express themselves. They can be federated or not by a moderator who is responsible for "filtering" the questions or the answers and also manages subjects. These forums are a continuous source enriched by the contributions of each. They allow anyone learning or teaching to ask a question and get the answer or answers. Each question or answer is published by an authenticated user for privacy concerns. The questions belong to a particular topic and can have multiple answers. Whereas answers are associated with one and only one question.

B. The Modeling Process

Since the business model is the heart of our generation process [16]. Its representation is a crucial task. For this we

will follow the best suited approach to develop a dynamic web application presented in [17]. At first, the needs will be modeled using the UML use case (Fig. 3). They will be represented by a more concrete HMI model (Human Machine Interface) designed to react to future users. In our case we used only create and delete operations to avoid complicating our case study.

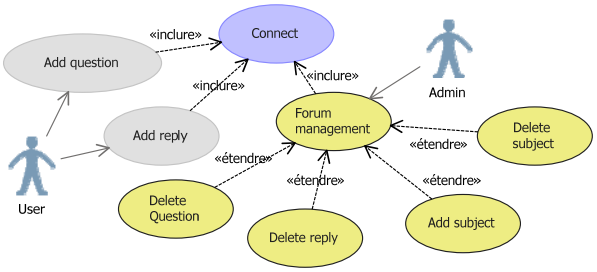

Fig. 3. Use case diagram of forum discusion model

In the context of object-oriented systems, the structure of the code is defined by the software classes and their groupings into groups called packages. So we need diagrams representing software classes and showing the data contained in them (called attributes), the services provided by those classes (called operations) including the relationships between the operations . UML proposes the class diagrams to convey this information. The allocation of good responsibilities to good class is one of the most delicate problems of object-oriented design. For each service or feature, you must decide which the class that will contain it. We must therefore divide the integral system behavior between design classes, and we must describe the induced interactions using interaction diagrams. Each interaction diagram will then represents a set of objects of different classes working under a system execution scenario (use cases). These diagrams also help to write the code inside the operations, especially operations nested calls.

To present our case study model, we will use a notation that has been proposed by I. Jacobson [18] and then popularized by RUP [19]. The Analysis classes they advocate are divided into three categories: Dialogues classes, controls classes, and the entity classes. The Classes that allow interaction between the website and its users are called "dialogues". This is typically the screens proposed to the user: entry forms, search results, etc. They come directly from the analysis of the model. Those containing the kinematics of the application will be called "controls". They make the transition between dialogue and business classes, allowing screens to manipulate information held by one or more business object. Those representing business rules, are termed "entity" They come directly from the domain model, though they are each confirmed and complemented by a use case.

The following diagram (Fig. 4) shows the forum detailed design class diagram used as input to the generation process of our approach. In this diagram the dialogue classes begins with a << V >>, the control classes starts with a <<Ctr>> and the business classes start with a << M >>, this notation meets the MVC paradigm ;a paradigm that organizes our design model into three categories of class : view category, control category, and entity category.

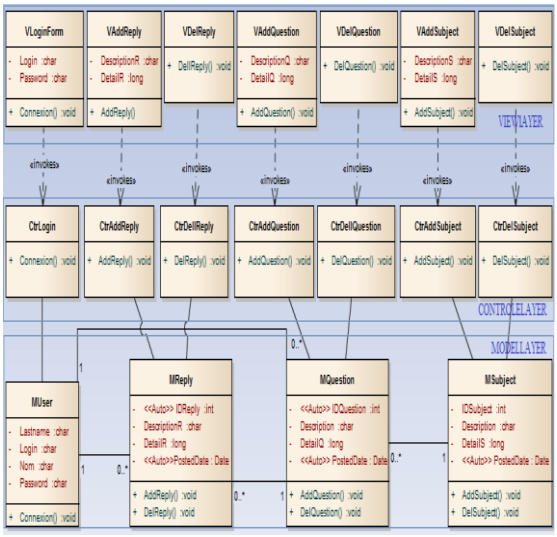

Fig. 4. Class diagram of forum discussion model

In this model, each use case is represented by a dialogue class, a control class and a business class. For example in the "Connect" use case, we will need the following classes: VLoginForm, CtrLogin and Muser. The user via the "VLoginForm" class specifies a login and password, and then he sends the form to the "CtrLogin" class via airworthiness links. This control class recovers the content form and uses the "connection" method of the "Muser" business class for verification of the login and the password. The same for all classes of our model, each one of them collaborates for the elaboration of a use case.

C. PIM to PIM Transformation

To convert the model in Figure 4 in a XML file, we will use the programming transformation approach. Using the API manipulating of UML diagrams, especially those manipulating class diagrams, we will generate a file describing our business model in an XML file format, divided into four section in which we respect a meta-model transformation target that we have developed and whose overview is provided in figure (Fig. 5).

The root element of our XML file is the element "BusinessComponent", this complex element has one attribute "Name" and four sub-elements: ViewLayer, ControleLayer, and ModelLayerDAOLAyer. ViewLayer is the section representing the view portion of our component. It includes all dialogue classes transformed into presentation pages "ViewPage" regardless of the technology. These pages contain input fields (Input), display fields or hyperlinks (Link). The item "ControleLayer" includes control classes transformed into control pages "ConrolePage." These control pages communicate with business classes "BusinessClass" contained in "ModelLAyer" to reformulate an answer "Response" under a presentation page format. Finally, the section "DAOLayer" represents the logical data model of our component.

The algorithm used in this transformation is very simple: we traverses the class diagram of our model class by class, if the class name begins with a "V", its description will be added in ViewLayer section as ViewPage. If the class name

begins with a "Ctr" it is added in "ControleLayer" as "ControlePage", and if the class name begins with an "M"; in this case; it is a class entity that will be converted first into business class and second into entity of a logical data model. Without forgetting the associations between different classes which ensure airworthiness link between all of these elements, the portion of our transformation result is shown in Figure (Fig. 6).

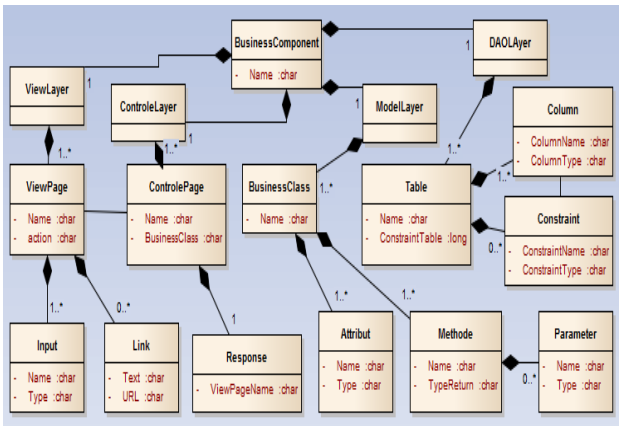

Fig. 5. Meta-model transformation target

```
<?xml version="1.0" encoding="utf-8" ?>
<BusinessComponent name="ForumComponent">
<ViewLayer>
<ViewPage name="ViewLoginForm" action="CtrLogin">
<Input name="Login" type="Text"></Input>
<Input name="Password" type="Text"></Input>
<Input name="Connexion" type="Button"></Input>
</ViewPage>
</ViewLayer>
<ControleLayer>
<ControlePage name="CtrLogin"BusinessClass="MUser">
<MethodeControle name="connexion">
<param name="Log" source="ViewLoginForm"></param>
<param name="Pass" source="ViewLoginForm"></param>
</MethodeControle>
<RequestViewPageName="Bienvenue"valid="OK">
</Request>
<Request ViewPageName="ErrorPage" valid="NO">
</Request>
</ControlePage>
</ControleLayer>
<ModelLayer>
<BusinessClass name="MUser">
<Atrribute name="Lastname"Type="String">
</Atrribute>
<Atrribute name="Firstname" Type="String">
</Atrribute>
<Atrribute name="Login" Type="String"></Atrribute>
<Atrribute name="Password" Type="String">
</Atrribute>
<Methode name="Connexion">
    <param name="Log" Type="String"></param>
    <param name="Pass" Type="String"></param>
    <Return type="Void"></Return>
</Methode>
</BusinessClass>
</ModelLayer>
<DAOLAyer>
<Table name="USER">
<Column name="Login" type="String"></Column>
<Column name="Password" type="String"></Column>
<Column name="Lastname" type="String"></Column>
<Column name="Firstname" type="String"></Column>
    </Table>
  </DAOLAyer>
</BusinessComponent>
```

Fig. 6. Partial component model under XML format

The XML file above contains a part of the description in XML format of our business model (Fig. 4), which represents only the authentication function. The root element of our XML file is the "BusinessComponent" element. The value "ForumComponent" of its "name" attribute is the name of the final business generated component. This component will include a presentation page named "ViewLoginForm" containing two text boxes and a button. When you click on the button, the form is sent to the control class specified in the action attribute of "ViewPage". Once the form is recovered by the control class, it is processed using the method specified in "MethodeControle" (Connexion) of the class specified in "BusinessClass" (Muser). Then a response is formulated according to the result of method execution. In this functional requirement we can associate a table of the same description as the business class "Muser". At this table we can add constraints at the table or column level.

D. PIM to PSM Transformation :

We uses XML file generated in the previous section to generate a business component dependent on a target technology. We chose the J2EE particularly MVC paradigm to finalize our case study. This step is based on generative engineering in which a PIM (XML file) is transformed into a PSM. This transformation is based on the template approach. From the specifications contained in the XML file we generate finished artifacts. This transformation is based on predefined technological templates stored in our technological model repository. Here is the partial result of this transformation.

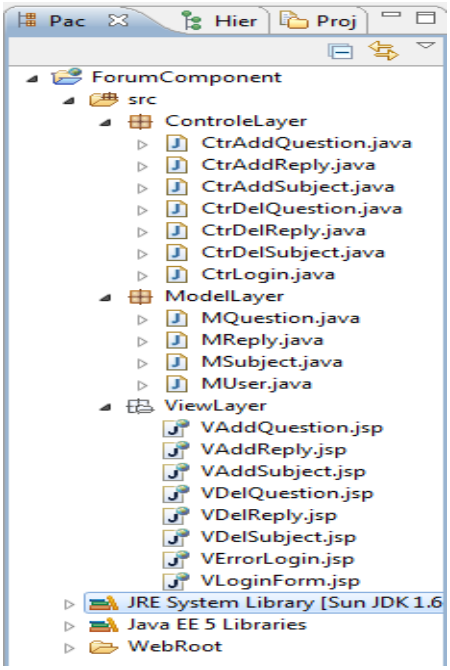

Fig. 7. J2EE web application

The first element of our generation process is the application component. It is a dynamic web application respecting the MVC2 paradigm. It consists of three packages: ViewLayer, ControlLayer, and ModelLayer. The ViewLayer package is the presentation layer of our application. It contains the following JSP pages: VAddQuestion.jsp, VAddReply.jsp, VAddSubject.jsp, VDelQuestion.jsp, VDelReply.jsp, VDelSubject.JSP, and VLoginForm.jsp VErrorLogin.jsp. The ControleLayer package includes the

following servlets controls: CtrAddQuestion.java, CtrAddReply.java,CtrAddSubject.java, CtrDelQuestion.java, CtrDelReply.java, CtrDelSubject.java, and CtrLogin.java. These servlets control support recovery form pages and make their treatment based on the methods of the following classes of ModelLayer Package: MQuestion.java, MReply.java, MSubject.java, and Muser.java.

The second element of our generation process is the script to create database generated under SQL format with the data description language. The object-relational mapping is done manually via tools such as Hibernate currently being worked on to make this process automatic.

V. Conclusion

Our approach facilitates the design of e-learning platforms, through their construction by assembling components generated from business model. It also accelerates their development and deployment by the principle of software reuse, as it facilitates their development by providing a clear separation between specifying and implementing components. The main advantage of our approach is the consideration of future developments in technology or domain. These developments are supported by modifying (or creating) generators associated with the models that will automatically propagate changes across all environments and components products.

We presented in this work, an overview of our approach and its implementation (LMSGENERATOR) and we focus on the generation phase based on MDA approach. A target meta-model was presented to facilitate understanding the transformation rules of PIM to PIM types. Future work would include the finalization of this generator, by defining generation rules implemented in these cores that are associated with PHP and .NET technologies and by feeding the technological repository currently containing until now only J2EE models used in our case study.

References

[1] J. Bézivin, "Sur les principes de base de l'ingénierie des modèles," *RTSI-L'OBJET*, vol.10, no. 4, 2004, pp. 145-157.

[2] Action IDM. [Online]. Available: http://www.actionidm.org.

[3] OptimalJ - Model-driven development for Java. Electronic Source: Compuware. [Online]. Available: http://www.compuware.com/products/optimalj/, 2003.

[4] Softeam, *Support de formation Objecteering 6/MDA*, Modeler version 2.0, Janvier, 2008.

[5] AndroMDA. (2008). [Online]. Available: http://www.andromda.org/.

[6] XMF-Mosaic. Electronic Source. [Online]. Available: http://www.xactium.com.

[7] S. Diaw, R. Lbath, and B. Coulette. State of the art of software development based on model transformations. [Online]. Available: http://www.tsi.revuesonline.com/article.jsp?articleId=14833

[8] R. Dehbi, M. Talea, and A.Tragha, "The generation approach of Multi-target learning management system," presented at the International Conference on Education Technology and Computer, Cape Town, South Africa, August 18-19, 2012.

[9] Soley *et al.*, *MDA (Model-Driven Architecture)*, White Paper, Draft 3.2, OMG Staff Strategy Group, 27, November, 2000.

[10] X. Blanc, *MDA en action : Ingénierie logicielle guidée par les modèles*, Eyrolles, 2005.

[11] R. Dehbi, M. Talea and A. Tragha, "Lmsgenerator: multi-target learning management system generator based on Generative Programming and Component Engineering," presented at the IEEE International Conference on Education and E-Learning Innovations, Sousse, Tunisia, July 1-3, 2012.

[12] R. Dehbi, M. Talea, and A. Tragha, "The modeling elements of LMSGENERATOR business repository," presented at the IEEE international Colloquium in Information Science and Technology, Fes, Morocco, October 22-24, 2012.

[13] J. Galloway, *Professional ASP.NET MVC 3*, John Wiley & Sons, 2011.

[14] J. Pauli and G. Ponçon, *Zend framework*, Eyrolles, 2008.

[15] D. Alur, J. Crupi, and D. Malks, *Core J2EE Patterns: Best Practices and Design Strategies*, Prentice Hall, 2003.

[16] E. Evan, *Domain-Driven Design: tackling complexity in the heart of software*, Addison-wesley, 2003.

[17] P. Roques, *UML2: Modéliser une application web*, 4 ed. Eyrolles, 2007, vol. 1, pp. 14-21.

[18] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*, 1 ed, Addison-Wesley, February 14, 1999.

[19] *Best Practices for Software Development Teams*, Rational Software White Paper, Rational Unified Process, TP026B, Rev 11/01.

Rachid Dehbi was born in Casablanca, Morocco in 1977. In 2002 received the Graduate of Computer Science Engineer from INSEA (Nationnal Institute of Statistics and Applied Economics). In 2011 Ph-D-Student at Faculty of Science Ben M'sik, MITI laboratory, Hassan II university. In 2003 Certified Lotus professional in development of Lotus Notes and Domino application. In 2006 IBM Certified System Administrator Lotus Notes and Domino R6/6.5. He worked for five years at the Ministry of Interior in which he served as department head. In 2007 he joined office of vocational training and promotion of work where he occupies until now the post of animator trainer and trainer at the training complex for offshoring business and information technologies and he is also the head of IT development pole. In 2008 he joined the Moroccan school of engineering science where he served until now as Oracle product trainer. He has published this year nine papers in international conferences and journals. His work focuses on e-learning and model driven engineering. He is an expert on developing web application based on J2EE, PHP and .NET technologies.

Mr Rachid dehbi is an IACSIT member, a member of IEEE Moroccan section, and organization chair of ICCMA'13 (IEEE International Conference on Computer Medical Applications) that will be held in Sousse (Tunisia) on the January 20-22, 2013.

Mohamed Talea was born in Casablanca, Morocco in 1964, Professor of Higher Education at the Faculty of Sciences Ben M'Sik, UNIVERSITY HASSAN II MOROCCO CASABLANCA. He obtained his PhD in collaboration with the LMP laboratory in Poitiers University, FRANCE in 2001. He obtained a Doctorate of High Graduate Studies degree at the University Hassan II-Mohammedia in 1994. Actually he is the Director of Information Treatment Laboratory. He has published twenty papers in conferences and national and international journals. His search major field is on Systems engineering, in security of system information.

Abderrahim Tragha was born in Casablanca, Morocco in 1959. He received the B.S. degree in applied mathematics from Mohammed V University, Morocco, 1983, and Doctorate of High Graduate Studies degree in Theories of Computer Sciences from Mohammed V University, Morocco, 1988 and Doctorate of state degree (or Ph D) July 2006 in Computer sciences from Hassan II Mohammedia University, Morocco. In 1988, he joined the faculty of science of Ben M'sik, Hassan II Mohammedia University, Morocco where he is currently a Professor in Department of mathematics and computer sciences.

He directs research and supervises thesis in Hassan II Mohammedia University. Actually he is the Director of Information Treatment and Modeling Laboratory. He has published twenty papers in conferences and national and international journals. His search major field is on Systems engineering, in security of system information and in computational linguistic.