Getting Model of MVVM Pattern from UML Profile

https://doi.org/10.3991/ijes.v8i1.13037

El Omari Mouad (✉), Erramdani Mohammed, Rhouati Abdelkader
University Mohameed First, Oujda, Morocco
elomari.mouad@gmail.com

Abstract—The rejuvenation of applications to harmonize with technological watch is the major challenge for all computer boxes, frameworks and languages are constantly proliferating by offering a range of improvements in terms of security and performance, which pushes all applications to invest in order to align oneself, to orient oneself towards another perspective of application implementation has become a primacy. MVW is considered the new concept of application models where the developer can choose according to his needs, which component, for example, it can be a controller, a directive or a unit test for applications where we use the AngularJS framework, modeling an application is one of the basic steps to reach it , the emergence of new patterns press IT companies to think to renew their application architecture for more security and performance, moving from an old to a new model meets this need. AngularJS is one of the widely used frameworks for modern single-page web application development which is designed to support dynamic views in the applications.

We propose an UML profile for AngularJS for building a model of an AngularJS web application, and a set of transformations that transform the model into a code template.

Keywords—AngularJS; code template; models; single-page web application; UML profile.

1 Introduction

We can say that in the IT field, the relevance of the information provided and its adaptation to user's preferences are key factors for the success or rejection of test platforms. Therefore, the solution is to conquer users by providing them with personalized platforms adapted to their needs. A, we are looking in this. We opt for an MDA type approach, allowing semi-automatic generation of the platform. This approach respects the architecture of MDA as it has been proposed by OMG [1].

Our vision to achieve this work is to apply the principles of MDA and use the UML profile for AngularJS for this article and that will be extended to other JS libraries, we will use the XML generated to have full AngularJS application by defining transformation rules.

The paper is organized as follow: Section 2 is dedicated to related work. In section 3 we present the MDA principles. Section 4 defines the MVVM pattern. Sections 5

includes the approach and presents the running example. Finally, section 6 concludes the work and offers some perspectives.

2 Related Work

Many researches on MDA and generation of code have been conducted in recent years.

The authors of the work [2] show how to generate JSPs and JavaBeans using the UWE [3].

The authors of the work [4] generate the MVW application by using activity diagram which will be transformed to class diagram.

The authors of work [5] generate MVC application by using Business Process Model and Notation.

Based on the same concept [6] apply MDA approach for generating PSM from UML design to MVC 2 Web implementation. That is why they have developed two meta-models handling UML class diagrams and MVC 2 Web applications, then they have to set up transformation rules. These last are expressed in ATL language. To specify the transformation rules (especially CRUD methods) we used a UML profiles. To clearly illustrate the result generated by this transformation.

Unfortunately, current model transformation languages do not cover all these features, and thus, the study of languages covering all of them should be object of study, this paper aims to redirect researchers to an important and actual topic which will allow to get MVVM pattern by applying the standard MOF 2.0 QVT to develop the transformation rules with an input model based on UML.

3 Model Driven Engineering

3.1 The OMG approach

In November 2000, the OMG (The Object Management Group), a consortium of over 1,000 companies, initiated the MDE (Model Driven Engineering) approach [7] Models depend on metamodels; MDE operations depend on metamodels. Managing evolution for models requires managing the evolution of metamodels. Most solutions to model evolution and co-evolution have focused on metamodels. A different approach would be to discard metamodels entirely – take the view that they get in the way of efficiently supporting evolutionary processes. To what extent can we support MDE without metamodels [8].

MDA is a system design and development methodology intended to facilitate development in a technology-independent approach. MDA was first described by OMG in 2001 [9]. One of the main objectives of the OMG is to establish an open interface, independent of software platforms, and therefore interoperable. In this sense, MDA embodies the vision presenting a global framework to support interoperability with specifications throughout a complete system life cycle. MDA's design philosophy

must include a description of business logic, modularization, construction and systems' integration, as well as deployment, management and evolution [9]. A key concept of MDA is the separation of the functionality specification from the implementation, integration and deployment specification [9]. This is accomplished by applying an abstraction to the design process. Abstraction is understood as the removal of irrelevant details, as motivated by the reference model for open distributed systems processing. In the field of MDA, meta-modeling plays a very important role considered to be a common technique to become the abstract syntax of Models and interrelationships between elements of the model. If the model is an abstraction of elements from the real world, the meta-model represents yet another abstraction, denying the properties of the model itself. A model is said to conform to its metamodel [10].

The modeling languages that are used are designed so as to be supported by tools that software engineers are familiar with and expect to be able to use – e.g., editors, syntax highlighters, debuggers, etc. Standard frameworks, such as EMF [11], exist to help define modeling languages in such a way so as to support this. In contrast, formal specification languages are designed to support mathematical reasoning, and as such the priority is to have a sound and complete mathematical semantics, which thereafter be supported by tools.

The key principle of MDE is automating repetitive and error prone tasks. The decisions that we make, with respect to use of particular technologies and theories, the implementation of particular tasks, and the deployment of workbenches to users, should always aim to support that principle.

The transition from one level to another is provided by transformations; that can be defined as the operation of taking elements of one or more models (source) and to match them with other elements of the model (target). There are two types: Model to Model (M2M) and Model to Text transformation (M2T). The first lets us go from CIM to PIM and PIM to PSM. As for the second, it allows the generation of platform-specific code chosen. Fig. bellow shows how the transformations are done. So we can say that this relationship, introduced in [12] and [13] connects two models and is the first step to automation and code generating [14].

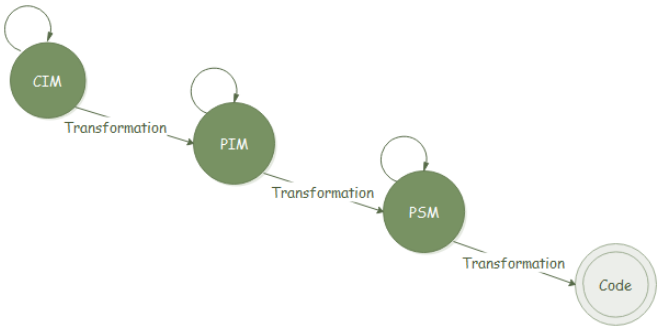

Fig. 1. Model Driven Architecture layers

CIM: These models describe the system to be designed from an IT-independent perspective. The CIM allows a vision of the system and its environment, while hiding the details of structure and implementation. CIM-level Models reduce the gap between experts in the field and between designers. Therefore, a CIM model is sometimes called a domain model. The technical independence of this model allows it to keep its full interest over time and it is modified only if knowledge or business needs change. The know-how is refocused on the CIM specification instead of the implementation technology.

PIM: It is independent of any technical platform (JEE, EJB, CORBA, .NET ,...) and does not contain information on the technologies that will be used to deploy the application. It is a computer model that represents a partial view of a CIM. The PIM represents the business logic specific to the system or the design model. It represents the functioning of entities and services. It must be lasting and last over time. It describes the system, but does not show the details of its use on the platform. At this level, the formalism used to express a PIM is a class diagram in UML which can be coupled with a constraint language like OCL * (Object Constraint Language). There are several levels of PIM. The PIM may contain information on persistence, transactions, security. These concepts allow the PIM model to be transformed more precisely into the PSM model.

PSM: It is dependent on the technical platform specified by the architect. PSM is essentially used as a basis for the generation of executable code towards the technical platform. PSM describes how the system will use this platform. There are several levels of PSM. The first, resulting from the transformation of a PIM, is represented by a UML scheme specific to a platform. The other PSMs are obtained by successive transformations until the code is obtained in a specific language (Java, C ++, C, etc.). An implementation PSM will contain, for example, information such as program code, types for, related programs, deployment descriptors.

The MDA approach seems very attractive at first. However, do not be mistaken, if it has its fervent followers, it also has its detractors whom advance cautiously in his direction. We will first see the advantages of this approach, then the drawbacks that may result, to finally highlight some feedback on experience.

3.2 Transformation of MDA models

A transformation is an automatic generation of one or more target models from one or more source models, respecting a definition of transformation. A transformation's definition is a set of transformation rules that describe how a model in the source language can be transformed into a model in the target language. A transformation's rule is a description of how one or more constructions in the source language can be transformed into one or more constructions in the target language.

To implement this transformation's process, a transformation engine takes as input one or more model (s) conforming to one (s) metamodel (s) source (s) and produces one or more other output (s) model (s) conforming to a target metamodel (s). The transformation engine, composed of a set of rules, must itself be considered as being a model. Consequently, it is based on a corresponding metamodel, which is an abstract

definition of the transformation language used. [15] Propose a taxonomy of model transformations where they define two orthogonal dimensions: a horizontal transformation versus a vertical transformation and an endogenous transformation over an exogenous transformation.

- Vertical transformations of the models are used to refine or abstract a model and, in this case, the models are located in different levels of abstraction. The horizontal transformations do not affect the abstraction of the models and they mainly serve to restructure them given that these models belong to the same level of abstraction.
- Endogenous transformations are transformations between models that are exprimed in the same language while exogenous transformations are transformations between models of defined with the help of different languages. (Mens and Gorp, 2006) qualify endogenous transformations by the term reformulation and sexogenic transformations by the term translation [15].

4 MVVM Design Pattern

The Model-View-View Model (abbreviated MVVM, from the English Model View ViewModel) is an architecture and a design method used in software engineering. MVVM is from Microsoft and suitable for the development of applications based on Windows Presentation Foundation and Silverlight technologies via the MVVM Light tool for example. This method allows, like the MVC model (Model-View-Controller), to separate the view from the logic and from the data access by emphasizing the principles of binding and event.

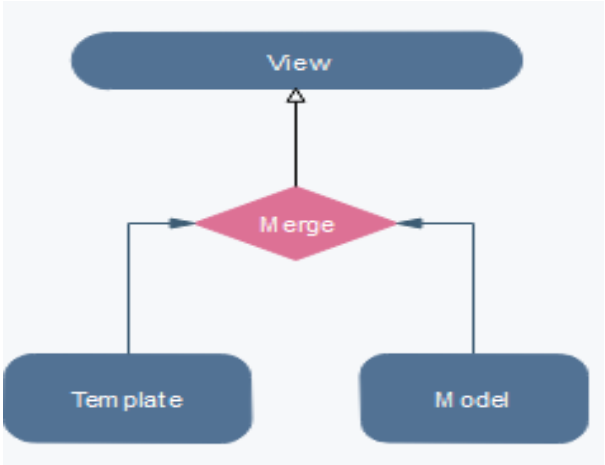

Fig. 2. MVC data-binding

- The model: represents the data received from the server.
- The view: contains all the views displayed to the user.
- The Template: contains the application logic.

MVVM provides powerful bidirectional data binding between model and view. This eliminates the need for wrappers, getters/setters or class declarations.

It conserves the concept of data applications which are separated into multiple tiers.

The Model defines the data structure and communicates with the server.

The View displays Model information and receives user actions.

The Controller manages the events and the update of the View and the Model.

We have a first breakdown of the application which already allows us to answer some of our problems. By clearly identifying the logical parts, we can more easily maintain and test our application.

The View therefore has no connection with the Model. Thus the ViewModel takes care entirely of the modification cycle of the latter. It both receives and sends data to the View. We then speak of "data binding". The information displayed is linked between two entities and updated in real time.

This latter mechanism is the key to the MVVM pattern. It allows us to decouple the different parts of our application by being able to develop it in a modular way.

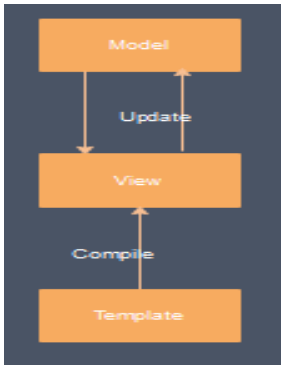

Fig. 3. MVVM data-binding

5 AngularJS as Study Case

AngularJS enables the creation of a single page applications and allows some of the logic (such as validation) to be included on the client side [16].

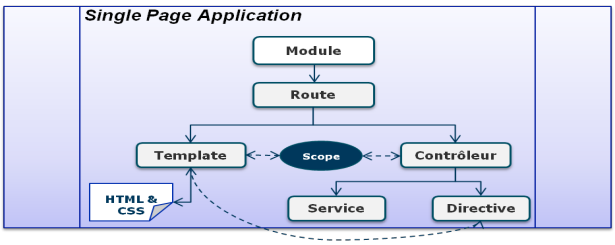

Fig. 4. AngularJs Architecture

A module can be considered as a container of different parts of the application like controllers, directives , services.

AngularJS applications must be developed as a hierarchy of Components. Each Component is an isolated part of the application for maintenance reasons.

Initially, AngularJS is a framework for developing One-Page applications. And therefore by definition, there is no need to change pages (since there is only one) and the routing is not useful. However, as the growing framework and its uses became more diverse, it soon became possible to integrate a routing system. In versions prior to 1.2, which were beta versions, developers natively integrated this routing system to the framework for the sake of simplicity (they had many other priorities). However, as of version 1.2 which is stable and which made the renown of AngularJS, it was necessary to refocus it on these primary objectives: these famous applications One-Page. As the project progressed, many functionalities were born and the team decided to separate those that were not essential, a certain page load in AJAX and the result will be added in a portion of the DOM.

Each view works pairwise with a controller. The view consumes data in data-binding and calls the controller's methods. It can also include directives and use filters that are declared in the application.

The logic of the views is organized in the controller, so the code in the controller must be simple, that is to say, the controller couldn't interact with the DOM and manipulate data. The role of the controller is to define variables, so called, $scope variables, and to encapsulate views related to logic. When executing AngularJS, the latter will create the different working contexts called scopes. They are organized as a tree of objects, with everything at the top of $rootScope. The Scope allows the join between the Controller and the View by allowing the binding of the variables in both directions. To ensure code capitalization, components that manipulate the DOM can be created as a directive. The directives are the modules used to manipulate DOM, to bind events and define their actions. They translate into HTML components. The business part of code must be in services, they are singletons, that is, single instances of objects. The role of a service is to provide a set of tasks necessary for the operation of the application.

5.1 UML profile

A UML profile allows you to adapt the UML language to a domain that it could not properly cover. Profiles are not only used to generate PIM or PSM but also to switch from PIM to PSM. The specificities of each platform can be modeled using UML extension mechanisms defined by UML profiles. For example, stereotypes allow the addition of new elements to the meta-model, tagged values allow the addition of properties to a meta-class and constraints allow the addition or modification of rules. It is possible to associate stereotypes, marked values and constraints with any UML concept (class, attribute, association, use case). These elements make it possible to establish a correspondence between UML concepts and domain concepts.

The OMG has defined several profiles, each of which has a specific role in the transformations. For example:

- The EDOC profile (Enterprise Distributed Object Computing, version 1) aims to facilitate the development of business models, systems or organizations.
- The EAI (Enterprise Application Integration, version 1) profile simplifies the integration of applications by standardizing the exchanges and translation of metadata.
- The SPEM profile (Software Process Engineering Metamodel, version 1) is defined both as a UML profile and as a MOF meta-model. SPEM defines the ways to use UML in software projects and allows the creation of process models (for PIM only).
- The Test profile (version 1 adopted) allows the specification of tests for the structural (static) aspects as well as for the behavioral (dynamic) aspects of UML models.
- The real-time modeling profile (Schedulability Performance and Time, version 1) for modeling real-time applications.

5.2 UML profile for AngularJs

As mentioned in the introduction, MDA has made it possible to reduce the duration of development of computer applications by ensuring a perennial know-how using models.

In the same way, we will use UML profiles for AngularJS and which will be executed by a tool like magic draw, output will be an application of AngularJS.

The elements of Figure. 4 present the cornerstone of our MDA profile.

A stereotype is a model element that defines additional values [17].

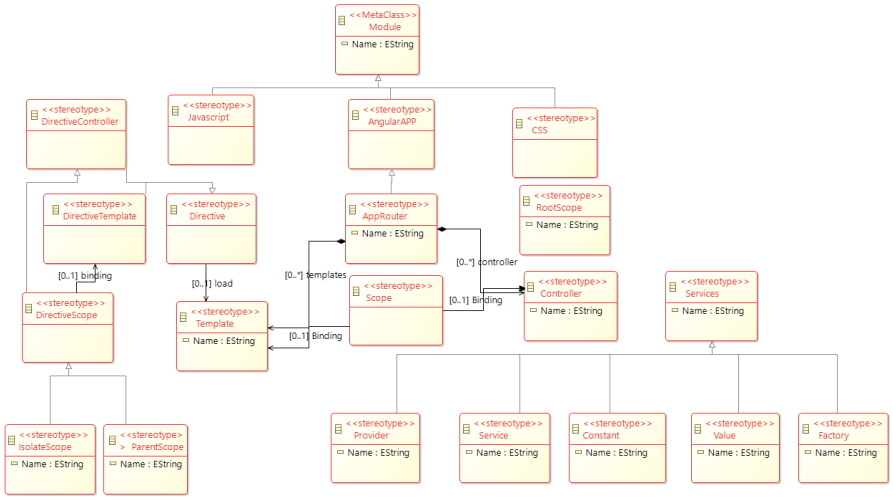

Fig. 5. Panorama of AngularJS Profile

In the diagram above, we put under the lightning different parts of our AngularJS profile:

Table 1. Description ofstereotypes

Stereotype	UML Meta Class	Description
AnguarApp	Class	Present the facade of the application
CSS	Class	CSS file to import
javaScript	Class	JavaScript file to import
AppRouter	Class	Configuration of navigation system
Scope	Artifact	Contain data exchanged between view and model
Controller	Artifact	Contain all business functions which manipulate scope
Template	Artifact	View
Directive	Artifact	Directive
DirectiveController	Artifact	Controller of this directive
DirectiveTemplate	Artifact	Template of the directive
DirectiveScope	Artifact	Scope of the directive , it can be isolate or depend to parent scope
RootScope	Artifact	Set variables when module is initialized
Services	Artifact	Return Promises and contain business, it's the part which will interact with the backend
Binding-Load	Association	Pass The data between two parts or load DOM

This UML profile divides the different parts that an AngularJs application needs to have, the module as a container encompasses all, the routing system via predefined angular services, $ route is the central service of the ngroute module, $ routeProvider Initializes the routing, pointing to the controller and template associated with this route.

The template is loaded by overloading the scope by actions described in the controller, the scope inherits the rootscope, and Each AngularJS application has exactly one root scope, but may have any number of child scopes [18].

The controller can use the service methods, where processing is to be done and calls to other APIs, 5 types of services are present and each has its own interest, but the most used is the factory, one Service returns a promised in AngularJS.

5.3 Code generation

The tool will generate an xml that will be retrieved and parsed by a JAVA API and make transformations on this file in order to generate an application.

I will give a study case of an application where well will get AngularJS application, stereotype in this case will be replaced by AngularJS , because this profile can be extended to other js libraries like React or VueJS.

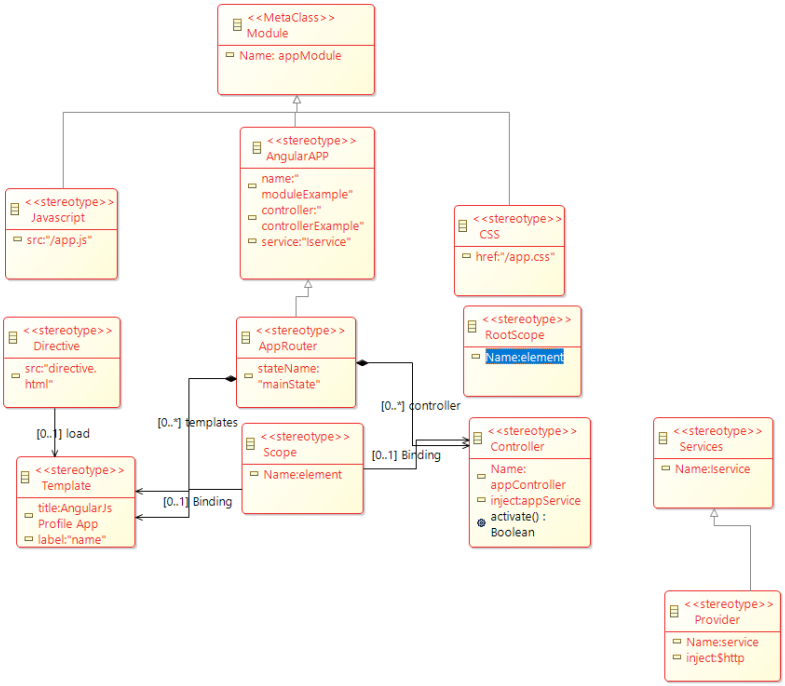

Fig. 6. Study case of AngularJS Application Profile

The diagram above illustrates an example on which we will apply our UML profile, to display a title and a label containing the name, all configuration files such as bower and the JSON package of nodeJS and the management of tasks by gulp will be generated with JSON files already defined.

While the various other parts will be configurable by the user who will add them either as src for dependencies, inputs, labels and buttons for templates, injections, methods and scope value for the controller, Methods with add, delete, updatefollowed by the property to modify, will be generated for controller and services.

An example of transformation rule:

```
<<Stereotype>> will be $stateProvider.state
(' AppRouter ', {
AppRouterurl: '/mainState',
stateName: "mainState"
controller: 'AppController'
stateUrl: "/mainState"
templateUrl: 'templates/views/mainState.html'
```

By the same concept, other transformations' rules have been implemented for the other stereotypes, the generated xml file will be parsed by DOM parser and via a modification template we will generate js, html and css files.

6 Conclusion

In this paper , UML profile concepts have been taken as the basis on which we can ensure the generation of an application, the use of an UML profile was the way in which we finally proceeded to generate our application, such a Methodology will ensure a gain on the time and durability side.

As a perspective, it is intended to extend to other functionalities by ensuring another user interface for the user to facilitate the description of the skeleton and adding the service layer interacting with the WS REST.

This UML profile can be spread out to generate other applications based on other JS libraries and this will guarantee a closer coverage of the Front Party code generation.

7 References

[1] AngularJS Documentation,(2013) ,https://angularjs.org/.

[2] Mbarki.S,Erramdani.M,(2008), Toward automatic generation of mvc2 web applications InfoComp, Journal of Computer Science, Vol.7 n.4, pp. 84-91, ISSN: 1807-4545.

[3] Kraus, Andreas & Knapp, Alexander & Koch, Nora. (2007). Model-Driven Generation of Web Applications in UWE.

[4] El Omari.M, Erramdani.M, Filali.S,(2016) Getting Model of MVVM pattern from UML Models.Proceedings of the International Conference on Industrial Engineering and Operations Management Rabat, Morocco,April 11-13, 2017.

[5] Rhazali, Yassine & Hadi, Youssef & Mouloudi, Abdelaaziz. (2016). Model Transformation with ATL into MDA from CIM to PIM Structured through MVC. Procedia Computer Science. 83. 1096-1101. 10.1016/j.procs.2016.04.229. https://doi.org/10.1016/j.procs.2016.04.229

[6] Moutaouakkil, Amine &Mbarki, Samir. (2019). MVC Frameworks Modernization Approach Adding MVC Concepts to KDM Metamodel. International Journal of Advanced Computer Science and Applications. 10. 304. https://doi.org/10.14569/ijacsa.2019.0101043

[7] OMG. MDA. http://www.omg.org/mda.

[8] Debnath, Narayan & Riesco, D. & Montejano, Germán & Grumelli, A. & Maccio, A. & Martellotto, P.. (2003). Definition of a new kind of UML stereotype based on OMG metamodel. 49. 10.1109/AICCSA.2003.1227482. https://doi.org/10.1109/aiccsa.2003.1227482

[9] Drozdova, Matilda &Kardos, Martin & Kurillova, Zuzana & Bucko, Boris. (2017). Transformation in Model Driven Architecture.

[10] Zhang, Yuan & Gao, Qin & Wu, Heng. (2010). Research on Model Driven Architecture. Applied Mechanics and Materials. 40-41. 10.4028/www.scientific.net/AMM.40-41.1012. https://doi.org/10.4028/www.scientific.net/amm.40-41.1012

[11] Steinberg.D, Budinsky.F, Merks. E, and Paternostro.M, (2008) EMF: Eclipse Modeling Framework. Pearson Education.

[12] Singh, Yashwant & Sood, Manu. (2009). Model Driven Architecture: A Perspective. 2009 IEEE International Advance Computing Conference, IACC 2009. 10.1109/IADCC.2009.4809264. https://doi.org/10.1109/iadcc.2009.4809264

[13] Miller, Granville & Evans, Andy & Jacobson, Ivar & Jondell, Henrik & Kennedy, Allan & Mellor, Stephen & Thomas, Dave. (2003). Model driven architecture. 273. 10.1145/949404.949409.
[14] Ross, J. (2004). Review: Model Driven Architecture. The Computer Bulletin. 46. 31-31. 10.1093/combul/46.1.31. https://doi.org/10.1093/combul/46.1.31
[15] El Omari, Mouad &Erramdani, Mohammed & Hajbi, Rachid. (2017). For Formed Entrepreneurial Culture. 10.1007/978-3-319-46568-5_47.
[16] El Omari, Mouad &Erramdani, Mohammed & SaidaFilali(2016)Model to Model Transformation by Modeling Getting Model of MVVM pattern from UML Models .Proceedings of the International Workshop on COmputing Sciences (WCOS'16), December, 21-22, 2016, Kenitra, Moroc.
[17] LisboaFilho, Jugurta&Iochpe, Cirano. (2017). Modeling with a UML Profile. 10.1007/978-3-319-17885-1_809.
[18] Ambler, Tim & Cloud, Nicholas. (2015). AngularJS. 10.1007/978-1-4842-0662-1_8.

8 Authors

Mouad El Omari is pursuing his PhD at Mohamed first University at MATSI laboratory. He graduates as a computer science engineer from ENSA (High School of Applied Science). His research activities at the MATSI Laboratory (Applied Mathematics, Signal Processing and Computer Science) are focused on MDA (Model Driven Architecture) approach applied to dynamic generation of code.

Mohammed Erramdani teaches the concept of Information System at Mohammed First University. He got his thesis of national doctorate in 2001. His activities of research in the MATSI Laboratory (Applied Mathematics, Signal Processing and Computer Science) focusing on MDA (Model Driven Architecture) integrating new technologies XML, EJB,MVC, Web Services, etc.

Rhouati Abdelkader is Phd researcher a private laboratory of Novelis Company in Paris.He got his thesis of national doctorate in 2019 from Mohamed first University Morocco. His main subject of research is about Machine learning and its application in software engineering.

Article submitted 2020-01-06. Resubmitted 2020-03-03. Final acceptance 2020-03-04. Final version published as submitted by the authors.