# Getting Model of MVVM pattern from UML Models

**Mouad El Omari**
MATSI Laboratory, Department of Computer Science,
Mohammed First University
Oujda, 60000, Morocco
elomari.mouad@gmail.com

**Mohammed Erramdani**
MATSI Laboratory, Department of Computer Science,
Mohammed First University
Oujda, 60000, Morocco
m.erramdani@gmail.com

**Saida Filali**
MATSI Laboratory, Department of Management,
Mohammed First University
Oujda, 60000, Morocco
s.filali6@gmail.com

## Abstract

All scientific research are aimed at studying how to describe and achieve what is best. One of the main problems is how to develop existing models to create other new, more complex. The continuity of the attachment of the human to web applications to respond to his needs has led researchers to think to implement mechanisms to ensure the continuity of knowledge , 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. This paper presents the application of the MDA (Model Driven Architecture) to generate, from the UML model, the Model following the MVW (Model-View-Whatever using the standard MOF 2.0 QVT (Meta-Object Facility 2.0 Query-View-Transformation) as a transformation language. We adopt AngularJS as a Frameworks for creating a target meta-model to generate an entire MVW web application. That is why we have developed two meta-models handling UML activity and class diagrams and MVVM (Model-View-View-Model) Web applications, then we have to set up transformation rules.

**Keywords(12 font)**
meta-model, MVW, transformation, Model Driven Architecture (MDA), MOF 2.0 QVT.

## 1.Introduction

Nowadays, conceiving high level layer applications allowed to software to evolve their solutions in flexible ways. This should allow reuse of existing effort and taking into account that the target infrastructure is itself evolving. At this end, we can say that modeling approach is an efficient way to master complexity and ensure consistency. These ideas, among others, were considered the base by the Object Management Group (OMG), an organization of software that develops and supports specifications to improve the practice of enterprise software development and

deployment, to address this challenge. It has devised a number of standards for software development under its Model Driven Architecture (MDA) approach (MDA,2003). The MDA uses models as first class entities, enabling the definition and automatic execution of transformations between models and from models to code. Thus, MDA encourages efficient use of system models in the software development process. As defined by the Object Management Group (OMG), MDA is a way to organize and manage enterprise architectures supported by automated tools and services for both defining the models and facilitating transformations between different model types. These transformation rules, together, describe how source models are transformed into target models (Mens,2004).The transformations between two models, are called model-to-model transformations, and model to code, called model-to-text (M2T) transformations. The strength of this approach is that there's several MDA tools that automates this transformation.
The Whatever Layer is considered as the new concept of applications patterns where the developer can choose according to his need which component , for example it can be controller , directive or unit test for applications where we use the Framework AngularJS.
In this work the main objective is to unify the design and implementation of MVW patterns around models by applying the MDA principles while introducing design patterns' practices, and also take the UML models as basis. Our vision to achieve this work is to apply the principles of MDA and use the UML models to first define a source meta-model describing the MVVM and its operations, which led us to define a new meta model apart from class and activity diagrams. Second we defined the target meta-model to represent a whatever layer independently from any platform. Then, we developed our transformation rules that will connect the two models instantiated from source and target meta-models. Last but not least.
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 relationship between the HCI and Design Pattern used to develop the meta models. Sections 5 and 6 include the approach and present the running example. Finally, section 7 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 (Koch,2006) show how to generate JSPs and JavaBeans using the UWE (Jouault,2008).The authors of the work (S.Roubi,2015) generate graphical user interfaces for Rich Internet Applications using Interaction Flow Modeling Language.
Authors of (S. Mbarki,2008) examines the application of MDA approach in the engineering of web applications. Two meta-models were designed: The first one for managing UML source models, the second web application models. The transformation rules and mapping algorithm were developed to generate an xml file containing all actions, forms, and forward JSP pages from class diagram that can be used to generate the required code of web application.
Based on the same concept (M'hamed Rahmouni,2011) 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 .
This concept had been applied on real framework by (Oualid Betari,2017) where they implement MDA approach to model the CodeIgniter PHP framework. They developed the models used for transforming Platform Independent Model (PIM) to Platform Specific Model (PSM), using a UML class diagram as a source model to generate an XML file containing the core components of a CodeIgniter PHP framework.
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 (MDA,2003).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 (F. Paige,2015).

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 (Dave Steinberg,2011), 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, τ, introduced in (J.M. Favre,2005), and (Dagstuhl Seminar,2004), connects two models and is the first step to automation and code generating.

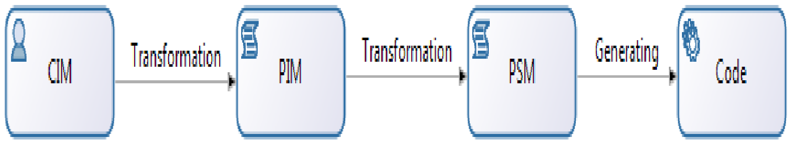


Figure 1. Model Driven Architecture layers

CIM : create a target model containing all of the business rules defined in the core business model.

PIM: create a target model containing only the data elements define in the conceptual model.

PSM: : is a more detailed version of a PIM. Platform specific elements are added. When defining a PSM a target Platform Model has to be available.

In addition to that, three key concepts are the basis of MDA: models, meta-models and transformations. Each of these concepts has a different relationship:

In the MDA, models are first-class artifacts, integrated into the development process through the chain of transformations from PIM through PSM to coded application. To enable this, the MDA requires models to be expressed in a MOF-based language. This guarantees that the models can be stored in a MOF-compliant repository, parsed and transformed by MOF-compliant tools, and rendered into XMI for transport over a network. This does not constrain the types of models you can use - MOF-based languages today model application structure, behavior (in many different ways), and data; OMG's UML and CWM are good examples of MOF-based modeling languages but are not the only ones(OMG,2003).

## 3.2 Transformation of MDA models

Once the meta models developed, MDA provides the passage between the CIM, PIM and PSM models through the execution of models' transformation. Transformation allow to convert models with a particular perspective from one level of abstraction to another, usually from a more abstract to less abstract view, by adding more detail supplied by the transformation rules.

These transformations can be written according to three approaches: The approach by Programming, the approach by Template and the approach by Modeling.

The general pattern is:

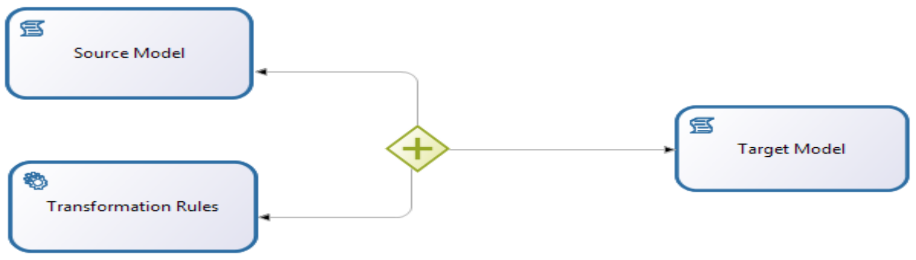


Figure 2. Transformation Process

Type mappings are generally insufficient to specify a complete transformation: additional rules are required to specify that certain types in the PIM must be annotated (marked) a specific way in order to produce the desired output in the PSM. This extra information cannot be determined from the PIM itself (OMG,2006).

Using the modeling approach is designed to have a sustainable and productive models' transformation, independently of any execution platform. This is why the OMG has developed a standard for this transformation language which is the MOF 2.0 QVT (OMG,2009), standing for Query-View-Transformation.

QVT is hybrid character (declarative / imperative) consisting of three languages: QVT-Relation, QVT-Operational and QVT-Core. Fig. 2 demonstrate this repartition.

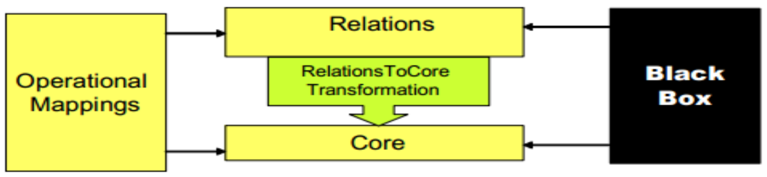


Figure 3. Relationship between QVT metamodels (Dagstuhl Seminar ,2004)

The declarative part is defined by the Relation and Core languages with different levels of abstraction, while the Imperative part is defined by the operational language. Finally, a black box is defined in the MOF 2.0 QVT standard that enables escaping the whole transformation/library or its parts that are difficult or impossible to implement in pure QVT.

This work uses the QVT-Operational mappings language implemented by Eclipse modeling (Eclipse,2016).

## 4.N-Tiers and MVW architecture

Most existing web applications are based on some form of the well-known model-view-controller (MVC) pattern. But the problem with the MVC is that it is not a very precise pattern, but rather a high-level, architectural one. Worse yet, there are many existing variations and derivatives of the original pattern (MVP and MVVM seem to be the most popular ones). To add to the confusion, different frameworks and developers tend to interpret the mentioned patterns differently.

Building apps based on various architectures seems to be the hot topic off-late in Android development. In this talk we will look at the various architectures available out there and try to find the best architecture to build your app.

N-tier data applications are data applications that are separated into multiple tiers. Also called "distributed applications" and "multitier applications," n-tier applications separate processing into discrete tiers that are distributed between the client and the server. When you develop applications that access data, you should have a clear separation between the various tiers that make up the application. For this purpose the Model-View-Controller architecture makes this design with 3 classes associated with each component:

- The model: stores content and contains methods to change this content.
- The view: displays the content is responsible for drawing on the screen the data stored in the model.
- The controller: manages the interaction with the user.

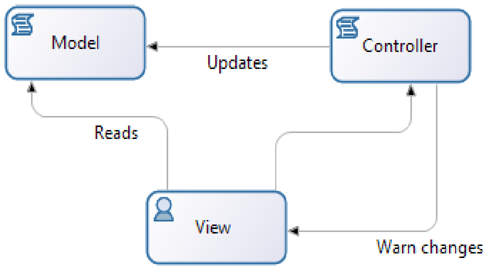
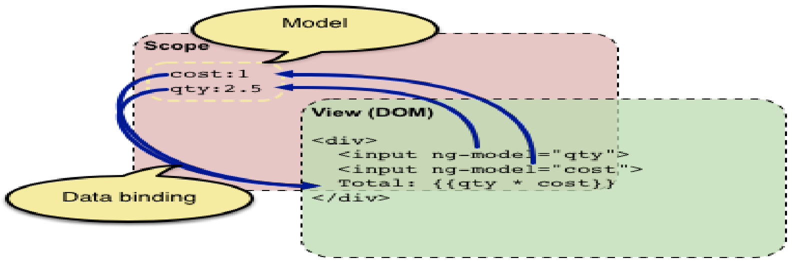


Figure 4. The MVC pattern

MVVM Provides powerful bidirectional data binding between model and view. This eliminates the need for wrappers, getters/setters or class declarations.It conserve the concept of data applications which are separated into multiple tiers.

## 4.1MVVM real example

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Angular's data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology[19].



```
cost:1
qty:2.5
```

```
<div>
  <input ng-model="qty">
  <input ng-model="cost">
  Total: {{qty * cost}}
</div>
```

Figure 5 .AngularJS biderctionnel binding(Angular,2014)

Rendering a template is straightforward with AngularJS; the framework shines when used to build dynamic web applications. In order to appreciate the real power of AngularJS.

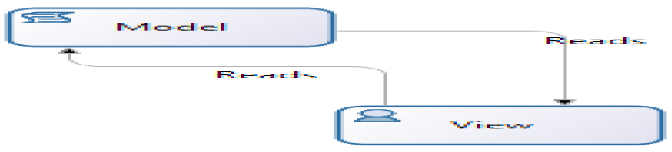
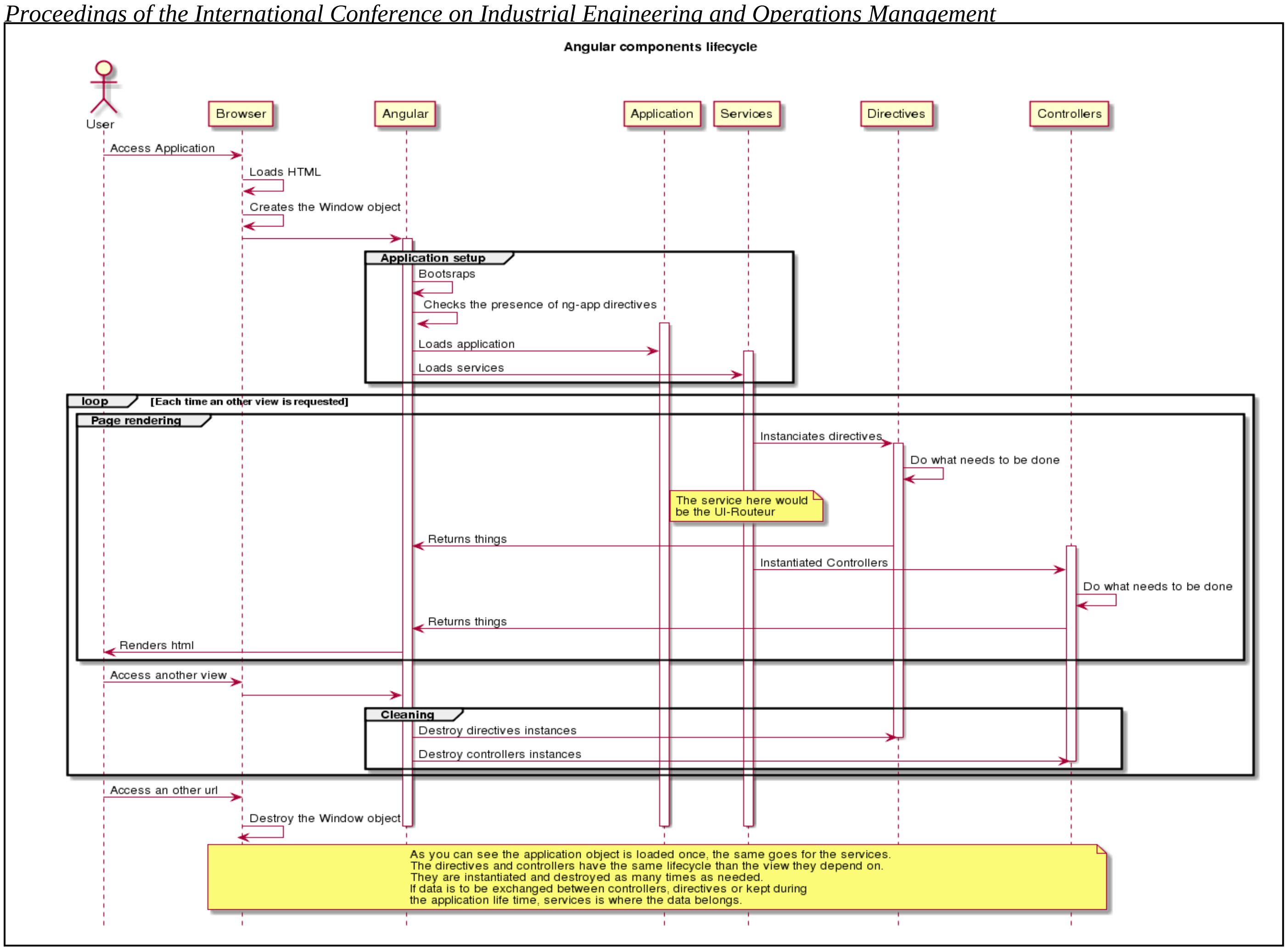


Figure 6.The MVVM pattern

In reality, AngularJS has even more intimate connections to HTML and the DOM as it depends on a browser to parse the template's text (as a browser would do with any other HTML document). After a browser is done transforming the markup's text to the DOM tree, AngularJS kicks in and traverses the parsed DOM structure. Each time it encounters a directive, AngularJS executes its logic to turn directives into dynamic parts of the screen.

Among UML models, the two models that describe in the best way the user's interaction with the system are the use case and activity diagrams. However, this representation stays week from any additional



As you can see the application object is loaded once, the same goes for the services.
The directives and controllers have the same lifecycle than the view they depend on.
They are instantiated and destroyed as many times as needed.
If data is to be exchanged between controllers, directives or kept during
the application life time, services is where the data belongs.

Figure 7.Angular lifecycle

The first phase of the AngularJS life cycle is the bootstrap phase, which occurs when the AngularJS JavaScript library is downloaded to the browser. AngularJS initializes its own necessary components and then initializes your module, which the ng-app directive points to. The module is loaded, and any dependencies are injected into your module and made available to code within the module.

The second phase of the AngularJS life cycle is the HTML compilation stage. Initially when a web page is loaded, a static form of the DOM is loaded in the browser. During the compilation phase, the static DOM is replaced with a dynamic DOM that represents the AngularJS view.

This phase involves two parts: traversing the static DOM and collecting all the directives and then linking the directives to the appropriate JavaScript functionality in the AngularJS built-in library or custom directive code. The directives are combined with a scope to produce the dynamic or live view.

The last phase of the AngularJS application is the runtime phase, which exists until the user reloads or navigates away from a web page. At that point, any changes in the scope are reflected in the view, and any changes in the view are directly updated in the scope, making the scope the single source of data for the view.

AngularJS behaves differently from traditional methods of binding data. Traditional methods combine a template with data received from the engine and then manipulate the DOM each time the data changes. AngularJS compiles the DOM only once and then links the compiled template as necessary, making it much more efficient than traditional methods.

## 5.Modeling approach

Currently, the models' transformations can be written according to three approaches: The approach by Programming, the approach by Template and the approach by Modeling.

The approach by Modeling is the one used in the present paper. It consists of applying concepts from model engineering to models' transformations themselves.

In this section we will present meta classes forming the UML source meta-model and target meta-model.

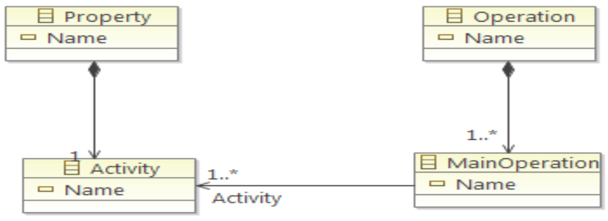


Figure 8.source metaModel

Our proposed PIM meta model contains the following :

- **Operation**: describes the global functionality offered by the system.
- **MainOperation**: Describing the operation performed by the user on the view.
- **Activity**: MainOpeation can contain lot of activities which can have type.
- **Property**: offers more information on the type of activity and action associated with that activity, for envelopes Example of changing the value of a radioBouton should alert the model the type boolean was hereby amended.

- Once we have established the source meta model for our approach, the second step while applying the MDA approach is to elaborate the target meta model that will form the PSM independently from any specific programming language. We are focused, as stated earlier in this paper, on the basic operations required in any User interface; which are: input, selection and click operations. Therefore, we used the two design patterns MVVM and Composite to establish the PSM in question.

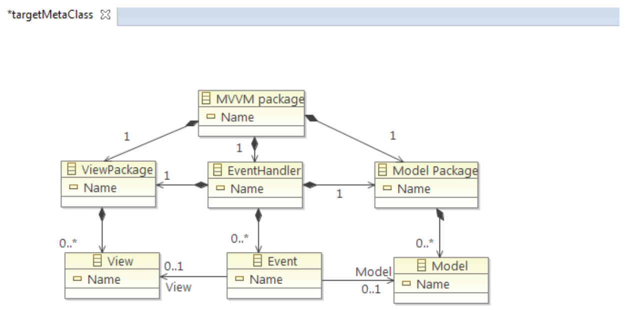


Figure 9.target metaModel

- **MVVM package**: expresses the concept of package that includes the entire elements of the model; the view and the controller, that are themselves regrouped, respectively, in ModelPackage, ViewPackage and EventHandler.
- **View:** contains the widgets of the View that will be defined hereafter.
- **EventHandler:** The responsible for the warning two sides if one had been changed .
- **Model:** express the business behind the application and is composed of methods that is triggered by a specific listener.

## 5.1Transformation rules

To make MDA model construction and transformation practical, transformation rules must be developed between models (Gharavi ,2008) .For our work, we established the mapping rules that lead to the transformation from PIM elements to PSM equivalent elements. The algorithm presented below illustrates the sequence of required transformations. Besides, the position layout defined by the user in the input model are also taken into account in the transformation process.

Main algorithm

```
input srcModel:UmlPackage
output destModel:MVVM package
begin
create MVVMpackage MVVMpackage
create ViewPackage ViewPackage
create ModelPackage ModelPackage
create EventHandler EventHandler
for all c in  srcModel.Operation
        map Operation2View(c)
        map Operation2Controller(c)
        map Operation2Model(c)
end for
end
mapping Operation 2View(act:Activity):View
begin
create View theView
theView.name = act.container.name + 'View'
for all act.type
if act.type is select
        map activityToSelect()
end if
if act.type is Select
        map activityToSelect()
end if
if act.type is click
        map activityToButton()
end if
if act.type is select
        map activityToselection()
end if
end for
end
mapping Operation2Event(c:Operation):Event
begin
create Event event
link theView to event
link theModel to event
Create Listener theListener
Link theListener to theView
end
mapping Operation2Model(c:Operation):Model
begin
create theModel Model
Create theMethod Method
Link theMethod to theEvent
end
```

## 6.Conclusion and perspectives

In this paper, we have defined a new way to model MVVM pattern  based on MDA meta models. To do this, we came up with a new meta model, which is a collection of activity ,class and activity diagram. We elaborated first the PIM as a base meta model for designing the user needs, then the target meta model representing the PSM meta model for the AngularJS platform. After that, we defined the mapping rules for the model to model transformation that allow the transition between the two models. And last but not least, we defined the Model To Text transformation based on the output file resulting from the first transformation. Following these steps, we were able to generate an MVW application architecture described simply with a use case, activity and class first.

In future works, we are aiming to define Model to Text transformation , which will have activity diagram in the input and full MVW application in the target.

## References

AngularJs , https://docs.angularjs.org/guide/introduction

Dagstuhl Seminar ,*on Language Engineering for Model-Driven Software Development*, Dagsthul, Germany, February 29-March 5, 2004, DROPS proceedings, http://drops.dagstuhl.de/portals/04101/.

Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: Eclipse Modeling Framework. Pearson Education,* 2008.

Eclipse modeling, http://www.eclipse.org/modeling/.

J.M. Favre, *Foundations of the Meta-pyramids: Languages and Metamodels* - Episode II, Story of Thotus the Baboon.

Jouault, F., Allilaire, F., Bézivin, J.and Kurtev, I., *ATL: A model transformation tool*. Science of Computer Programming-Elsevier Vol. 72, n. 1-2: pp. 31-39, 2008.

Gharavi V., Mesbah, A., Deursen, A. V., *Modelling and Generating AJAX Applications: A Model-Driven Approach*. Proceeding of the7th International Workshop on Web- Oriented Software Technologies, New York, USA (Page: 38, Year of publication: 2008, ISBN: 978-80-227-2899-7).

Koch, N., *Transformations Techniques in the Model-DrivenDevelopment Process of UWE*, Proceeding of the 2nd International 6 2015 5th World Congress on Information and Communication Technologies (WICT)Workshop Model-Driven Web Engineering, Palo Alto (Page: 3 Year of publication: 2006 ISBN: 1- 59593-435-9).

M'hamed Rahmouni and Mbarki Samir, *MDA-BASED ATL TRANSFORMATION TO GENERATE MVC 2 WEB MODELS* . International Journal of Computer Science & Information Technology (IJCSIT) Vol 3, No 4, August 2011.

Mens T., Czarnecki K. and Van Gorp P. *A Taxonomy of Model Transformations Language Engineering for Model-Driven Software Development*, Dagstuhl February - March 2004.

Meta Object Facility (MOF) 2.0 Query/View/Transformation(QVT), Version 1.1 (OMG, 2009).

Object Management Group (OMG), *MDA Guide 1.0.1*. http://www.omg.org/cgi-bin/doc?omg/03-06-01.

Object Management Group (OMG), *MDA* , http://www.omg.org/mda/specs.htm,2004

Oualid Betari,Mohammed Erramdani,Sarra Roubi,Karim Arrhioui ,Samir Mbarki, *Model Transformations in the MOF Meta-Modeling Architecture: From UML to CodeIgniter PHP Framework*, Springer International Publishing AG 2017.

Richard F. Paige, Nicholas Matragkas and Louis M. Rose, *Evolving Models in Model-Driven Engineering: State-of-the-Art and Future Challenges,* The Journal of Systems & Software (2015).

S.Roubi,M.Erramdani and S.Mbarki, *A model driven approach to generate graphical user interfaces for Rich Internet Applications using Interaction Flow Modeling Language*, Intelligent Systems Design and Applications (ISDA), June 2015 15th International Conference, ISSN: 2164-7151.

S. Mbarki and M. Erramdani, *Toward automatic generation of mvc2 web applications*, Journal of Computer Science, Vol.7 n.4, pp. 84-91, December 2008, ISSN: 1807-4545.

## Biography

**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.

**Saida Filali** teaches the concept of Management at Mohammed First University. Her activities of research in the MATSI Laboratory (Applied Mathematics, Signal Processing and Computer Science) focusing on MDA application on entrepreneurship.