Journal of Network and Innovative Computing
ISSN 2160-2174 Volume 4 (2016) pp. 037-045

Transformation by modeling with MOF 2.0 QVT: From UML model to Model-View-Presenter pattern

Redouane Esbai[1] and Mohammed Erramdani[2]

[1] MATSI Laboratory, ESTO, Mohammed First University,
Oujda, Morocco
es.redouane@gmail.com

[2] MATSI Laboratory, ESTO, Mohammed First University,
Oujda, Morocco
mramdani69@yahoo.com

***Abstract*: The continuing evolution of business needs and technology makes Web applications more demanding in terms of development, usability and interactivity of their user interfaces. The complexity and diversity of these applications emerges the need of flexibility and combining operations with existing models to create other new, more complex models. As more complex models are used, the importance of transformations between models grows. This paper presents the application of the MDA (Model Driven Architecture) to generate, from the UML model, the code following the MVP (Model-View-Presenter), DI (Dependency Injection) and DAO (Data Access Object) patterns for a RIA (Rich Internet Application) using the standard MOF 2.0 QVT (Meta-Object Facility 2.0 Query-View-Transformation) as a transformation language. We adopt GWT (Google web Toolkit), Spring and Hibernate as a Frameworks for creating a target meta-model to generate an entire GWT-based N-tiers web application. That is why we have developed two meta-models handling UML class diagrams and N-tiers Web applications, then we have to set up transformation rules. The transformation rules defined in this paper can generate, from the class diagram, an XML file containing the Presentation, the Business, and the Data Access package. This file can be used to generate the necessary code of a RIA N-tiers web application.**

***Keywords*:** GWT, Model Transformation, Model View Presenter, dependency Injection, MOF 2.0 QVT, transformation**.**

I. Introduction

In recent years many organizations have begun to consider MDA (Model Driven Architecture) as an approach to design and implement enterprise applications. The key principle of MDA is the use of models at different phases of application development by implementing many transformations [47]. The goal of this transformation is to come to a technical solution on a chosen platform from independent business models of any platform [1]. These changes are present in MDA, and help transform a CIM (Computation Independent Model) into a PIM (Platform Independent Model) or to obtain a PSM (Platform Specific Model) from a PIM [49].

In fact, the technological advances of RIAs (Rich Internet applications) require from the developer to represent a rich user interface based on the composition of Graphical User Interface (GUI) widgets, to define an event-based choreography between these widgets and to establish a fine grained communication between the client and the server layers.

The Presentation Layer of a RIA should not include too much business logic, instead it will communicate with a Web Server to perform business operations. This is done by accessing a Business Layer which in turn will communicate with the DAO Model.

In a recent work [2], the authors have developed a source and a target meta-model. The first was a PIM meta-model specific to class diagrams. The second was a PSM meta-model for MVP (Model-View-Presenter) web applications (particularly GWT), then they have elaborated transformation rules using the approach by modeling. The purpose of our contribution is to produce and generate an N-tiers PSM model, implementing MVP, DI (Dependency Injection) and DAO (Data Access Object) patterns, from the class diagram. In this case, we elaborate a number of transformation rules using the approach by modeling and MOF 2.0 QVT, as transformation language, to permit the generation of an XML file that can be used to produce the required code of the target application. The advantage of this approach is the bidirectional execution of transformation rules.

This paper is organized as follows: related works are presented in the second section, the third section defines the MDA approach, and the fourth section presents the MVP, DI and DAO patterns and its implementation as frameworks, GWT, Spring and Hibernate in this case. The transformation language MOF 2.0 QVT is the subject of the fifth section. In the sixth section, we present the UML, GWT and N-tiers meta-models. In the seventh section, we present the transformation rules using QVT operational from UML source model to the GWT and N-tiers target model. The last section concludes this paper and presents some perspectives.

II. Related works

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

The authors of the work [3] show how to generate JSPs and JavaBeans using the UWE [4], and the ATL transformation language [5] [45]. Among future works cited, the authors considered the integration of AJAX into the engineering process of UWE.
Two other works followed the same logic and have been the subject of two works [6] [7]. A meta-model for Ajax was defined using AndroMDA[41] tool. The generation of Ajax code has been illustrated by an application CRUD (Create, Read, Update, and Delete) that manages people.
Meliá, Pérez and Díaz propose in [8] a new approach called OOH4RIA which proposes a model driven development process that extends OOH methodology. It introduces new structural and behavioral models in order to represent a complete RIA and to apply transformations that reduce the effort and accelerate its development. In another work [9] they present a tool called OIDE (OOH4RIA Integrated Development Environment) aimed at accelerating the RIAs development through the OOH4RIA approach which establishes a RIA-specific model-driven process.
The Web Modeling Language (WebML) [10] is a visual notation for specifying the structure and navigation of legacy web applications. The notation greatly resembles UML class and Entity-Relation diagrams. Presentation in WebML is mainly focused on look and feel and lacks the degree of notation needed for AJAX web user interfaces [11] [12].
Nasir, Hamid and Hassan [13] have presented an approach to generate a code for the dotNet application Student Nomination Management System. The method used is WebML and the code was generated by applying the MDA approach, but the creation was not done according to the dotNet MVC2 logic.
This paper aims to finalize the work presented in [2] [14], by applying the standard MOF 2.0 QVT to develop the transformation rules aiming at generating the N-tiers and GWT target model with UI.

III. Model Driven Architecture

In November 2000, OMG, a consortium of over 1 000 companies, initiated the MDA approach. The key principle of MDA is the use of models at different phases of application development. Specifically, MDA advocates the development of requirements models (CIM), analysis and design (PIM) and code (PSM).
The major objective of MDA [1] is to develop sustainable models; those models are independent from the technical details of platforms implementation (JavaEE, .Net, PHP or other), in order to enable the automatic generation of all codes and applications leading to a significant gain in productivity. MDA includes the definition of several standards, including UML [17], MOF [18] and XMI [19].

IV. N-Tiers architecture

In this paper, we are using the following layers:

1. Presentation Layer
2. Business Layer
3. Data Access Layer
4. Database/Data store

The presentation layer contains the components that implement and display the user interface and manage user interaction. This layer includes controls for user input and display, in addition to components that organize user interaction [42].
The business layer makes all the application decisions. This is where the "business logic" is located. The application logic knows what is possible, and what is allowed. The application logic reads and stores data in the data access tier.
The data layer stores the data used in the application. The data layer can typically store data safely, perform transactions, search through large amounts of data quickly.
Each Layer can be developed independently of the other provided that it adheres to the standards and communicates with the other layers.

A. Title and The presentation Layer with MVP pattern

The Model View Presenter is a derivative of the Model View Controller Pattern. Its aim is to provide a cleaner implementation of the Observer connection between Application Model and View.
Figure 1 shows the architecture of the MVP pattern. The main feature of this pattern is to be composed of:

- The model is an interface defining the data to be displayed or otherwise acted upon in the user interface.
- The view is a passive interface that displays data (the model) and routes user commands (events) to the presenter to act upon that data.
- The presenter acts upon the model and the view. It retrieves data from repositories (the model), and formats it for display in the view.

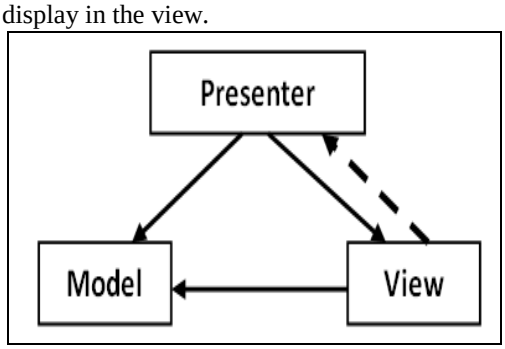

Figure 1. MVP Architecture

Based on this model many frameworks are designed to help developers build the presentation layer of their user interfaces. In the Java community, many frameworks that implements MVP pattern have emerged, among them: Mvp4g [20], GWT [21], Echo2 [22], JFace [23], Vaadin [24], ZK [25], Nucleo .NET [26]. The GWT project is one of the best examples. Implementing MVP in Google Web Toolkit requires only that some component implement the view interface.

B. The GWT framework

Google Web Toolkit (GWT) [27] is an open source web development framework that allows developers to easily create high-performance AJAX applications using Java. With GWT, you are able to write your front end in Java, and it compiles your source code into highly optimized, browser-compliant JavaScript and HTML.
However, GWT is not the only framework for managing the user interfaces. Indeed, other frameworks have been designed

for the same goal, but GWT is the most mature. The main advantage of GWT is the reduced complexity compared to other frameworks of the same degree of power, for instance, JFace, Flex and Vaadin.

C. The Business layer with Data Transfer Object and Dependency Injection patterns

Business logic layer is the Layer of abstraction between the presentation layer and persistence layer to avoid a strong coupling between these two layers and hide the complexity of the implementation of business processing to presentation layer. All business treatments will be implemented by this layer. The implementation of this layer is produced by the DTO pattern to render the result of running the service and the DI pattern to ensure a decoupling between objects [43].

In an article written in early 2004, Martin Fowler asked what aspect of control is being inverted. He concluded that it is the acquisition of dependent objects that is being inverted. Based on that revelation, he coined a better name for inversion of control: dependency injection [28].

In other words, Dependency Injection is a worthwhile concept used within applications that we develop. Not only can it reduce coupling between components, but it also saves us from writing boilerplate factory creation code over and over again. Many frameworks that implements DI pattern have emerged, among them: Spring [29], Symfony dependency injection [31], Spring.NET [30], EJB [32], PicoContainer [33]. (We have used some Spring classes in our source meta-model).

D. The persistence Layer with DAO pattern

This layer is the entry point to the database. All operations required to create, retrieve, update, and delete data in the database are implemented in the components of this layer.

The Data Access Object (DAO) pattern is now a widely accepted mechanism to abstract the details of persistence in an application. In practice, it is not always easy to make our DAO's fully hidden in the underlying persistence layer.

The advantage of this abstraction is that we can change the persistence mechanism without affecting the logic domain. All we need to change is the DAO layer which, if designed properly, is a lot easier to do than changing the entire logic domain. In fact we might be able to cleanly swap in a new data access layer for our new database or alternate persistence mechanism. Many frameworks that implements DAO pattern have emerged, among them: SpringDao [29], JPA [34], Hibernate [35], iBatis [36], NHibernate [37], EJB [32]. We have used Hibernate in our work because it is the most used solution within the java community.

V. Approach by modeling

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.

The advantage of the approach by modeling is the bidirectional execution of transformation rules. This aspect is useful for the synchronization, the consistency and the models reverse engineering [39].

A. MOF 2.0 QVT

Transformations models are at the heart of MDA, a standard known as MOF 2.0 QVT being established to model these changes. This standard defines the meta-model for the development of transformation model.

The QVT standard has a hybrid character (declarative / imperative) in the sense that it is composed of three different transformation languages (see Figure 2).

The declarative part of QVT is defined by Relations and Core languages, with different levels of abstraction. Relations are a user-oriented language for defining transformations in a high level of abstraction. It has a syntax text and graphics. Core language forms the basic infrastructure for the declaration part; this is a technical language of lower level determined by textual syntax. It is used to specify the semantics of Relations language in the form of a Relations2Core transformation. The declarative vision comes through a combination of patterns, source and target side to express the transformation.

The imperative QVT component is supported by Operational Mappings language. The vision requires an explicit imperative navigation as well as an explicit creation of target model elements. The Operational Mappings language extends the two declarative languages of QVT, adding imperative constructs (sequence, selection, repetition) and constructs in OCL edge effect.

The imperative style languages are better suited for complex transformations including a significant algorithm component. Compared to the declarative style, they have the advantage of optional case management in a transformation. For this reason, we chose to use an imperative style language in this paper.

Finally, QVT suggests a second extension mechanism for specifying transformations invoking the functionality of transformations implemented in an external language Black Box.

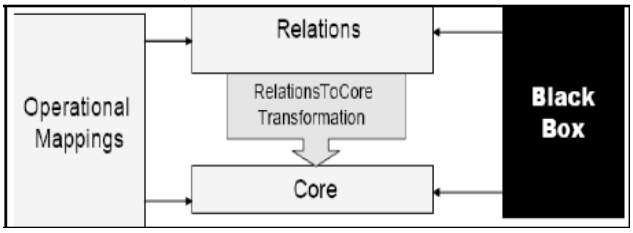

Figure2. The QVT Structure

This work uses the QVT-Operational mappings language implemented by Eclipse modeling [40].

B. OCL (Object Constraint Language)

Object Constraint Language (OCL) is a formal language used to describe expressions on UML models.

These expressions typically specify invariant conditions that must hold for the system being modeled or queries over objects described in a model. Note that when the OCL expressions are evaluated, they do not have side effects. OCL expressions can be used to specify operations / actions that, when executed, do alter the state of the system. UML modelers can use OCL to specify application-specific constraints in their models.

In MOF 2.0 QVT, OCL is extended to Imperative OCL as part

of QVT Operational Mappings.
Imperative OCL added services to manipulate the system states (for example, to create and edit objects, links and variables) and some constructions of imperative programming languages (for example, loops and conditional execution). It is used in QVT Operational Mappings to specify the transformations.
QVT defines two ways of expressing model transformations: declarative and operational approaches.
The declarative approach is the Relations language where transformations between models are specified as a set of relationships that must hold for successful transformation.
The operational approach allows either defining transformations using a complete imperative approach or complementing the relational transformations with imperative operations, by implementing relationships.
Imperative OCL adds imperative elements of OCL, which are commonly found in programming languages like Java. Its semantics are defined in [38] by a model of abstract syntax.
The most important aspect of the abstract syntax is that all expression classes must inherit OclExpression.
OclExpression is the base class for all the conventional expressions of OCL. Therefore, Imperative Expressions can be used wherever there is OclExpressions.

VI. UML, N-tiers and GWT meta-models

To develop the algorithm of transformation between the source and target model, we present in this section, the different meta-classes forming the UML source meta-model and the N-tiers target meta-model. The meta-model source structure simplified UML model based on a package containing the data types and classes. These classes contain properties typed and characterized by multiplicities (upper and lower). The classes contain operations with typed parameters. Figure 3 shows the source meta-model:

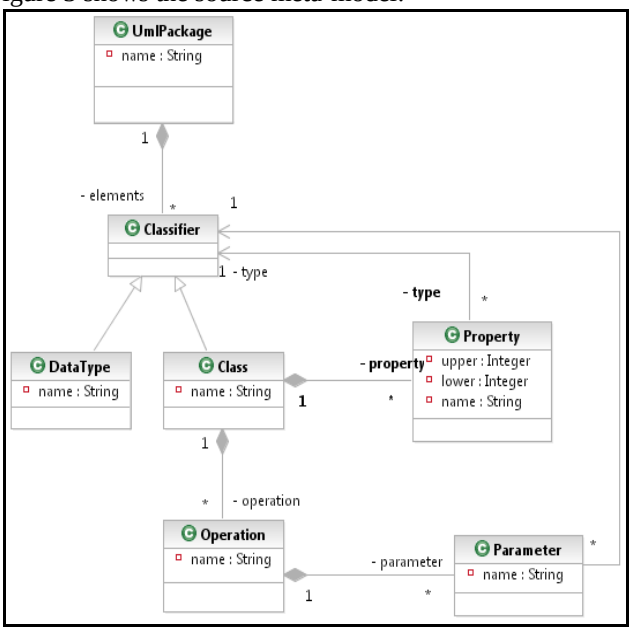

Figure 3. Simplified UML meta-model

Figure 4 illustrates the first part of the target meta-model. This meta-model represents a simplified version of the DAO pattern. It presents the different meta-classes to express the concept of DAO contained in the DaoPackage:

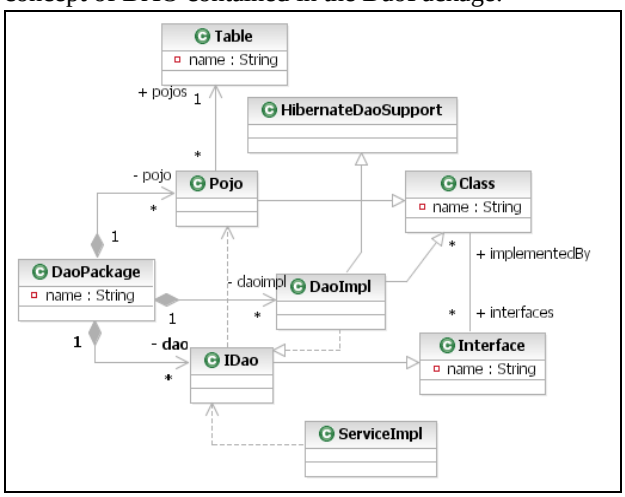

Figure 4. Simplified meta-model of DaoPackage

Figure 5 illustrates the second part of the target meta-model. This meta-model is the business model of the application to be processed. In our case, we opted for components such as DTO and DI pattern. Here, we present the different meta-classes to express the concept of DI contained in the Business Package.

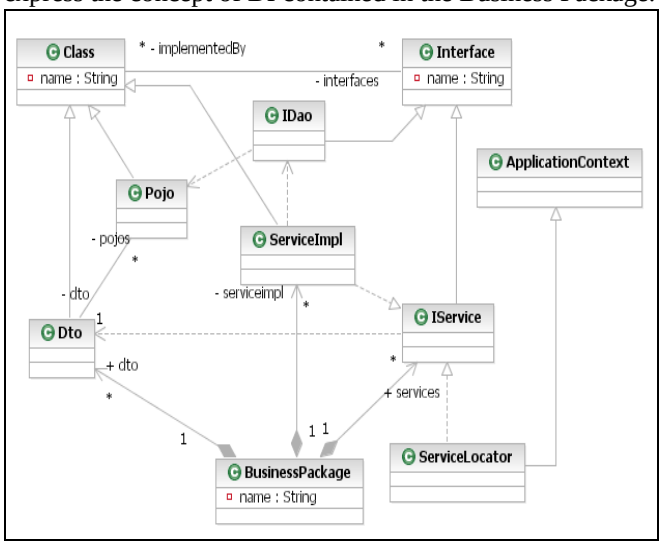

Figure 5. Simplified meta-model of a BusinessPackage

Figure 6 shows the third part of the target meta-model. This meta-model represents a simplified version of the MVP pattern. It presents the different meta-classes to express the concept of MVP implementation.
Like the Abstract Window Toolkit (AWT) and Swing, GWT is based on widgets. To create a user interface, you instantiate widgets, add them to panels, and then add your panels to the application's root panel, which is a top-level container that contains all of the widgets in a particular view. GWT contains many widgets whose classes are described by an inheritance hierarchy. An illustration of some of those widgets is shown in Figure 7.

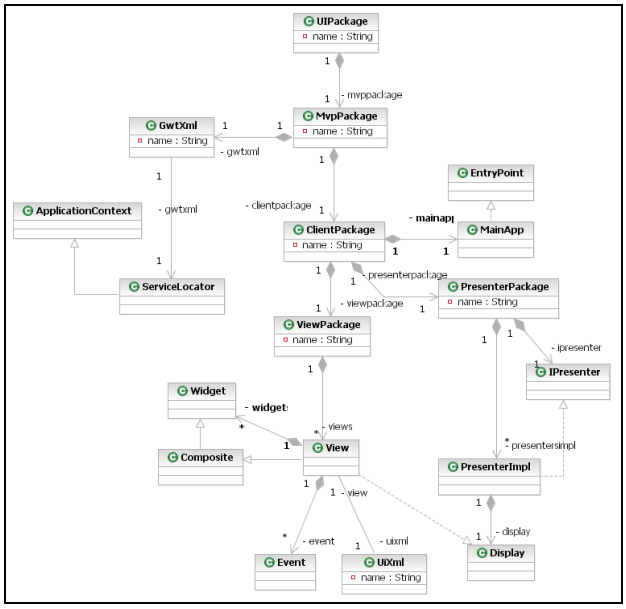

Figure 6. The proposed MVP meta-model

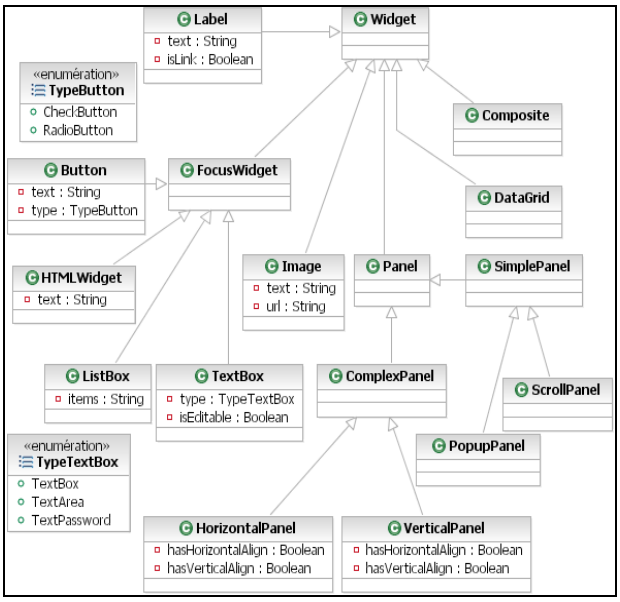

Figure 7. Simplified meta-model of a GWT widgets

VII. The process of transforming UML source model to N-tiers GWT target model

CRUD operations (Create, Read, Update, and Delete) are most commonly implemented in all systems. That is why we have taken into account in our transformation rules these types of transactions [46].

We first developed ECORE models corresponding to our source and target meta-models, and then we implemented the algorithm using the transformation language QVT Operational Mappings [47].

To validate our transformation rules, we conducted several tests. For example, we considered the class diagram (see Figure 8). After applying the transformation on the UML model, composed by the class Employee, we generated the target model (see Figure 10).

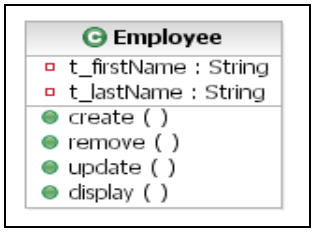

Figure 8. UML instance model

A. *The transformation rules*

By source model, we mean model containing the various classes of our business model. The elements of this model are primarily classes.

Main algorithm:

```
input umlModel:UmlPackage
output crudModel:CrudProjectPackage

begin
create CrudProjectPackage crud
create DaoPackage daoPackage
for each e ∈ source model
  x = transformationRuleOnePojo(e)
  link x to dp
  x = transformationRuleOneIDao(e)
  link x to dp
  x = transformationRuleOneDaoImpl(e)
  link x dp
end for
create BusinessPackage bp
for each pojo ∈ target model
  x = transformationRuleTwoDto(pojo)
  link x to bp
end for
for each e ∈ source model
x = transformationRuleTwoIService(e)
  link x to bp
  x = transformationRuleTwoSrviceImpl(e)
  link x to bp
end for

create UIPackage uip
create MvpPackage mvpPackage
create ClientPackage clientPackage
create MainApp mainapp
link mainapp to clientPackage
create PresenterPackage presenterPackage
create IPresenter ipresenter
ipresenter.name = 'IPresenter'
ipresenter.methods = declaration of {do,bind}
link ipresenter to presenterPackage
for each e ∈ source model
   x = transformationRuleThreePresenter(e)
   link x to presenterPackage
end for
create ViewPackage viewPackage;
for each e ∈ source model
x=transformationRuleThreeView(e)
   link x to viewPackage
end for

create GwtXml gwtxml;

link presenterPackage to clientPackage
link viewPackage to clientPackage
link clientPackage to mvpPackage
link mvpPackage to uip
link gwtxml to uip

link dp to crud
link bp to crud
link uip to crud
create object GWTXML gwtxml;
return crud
end
```

```
function
transformationRuleOnePojo(e:Class):Pojo
begin
create Pojo pj
pj.name = e.name
pj.attributes = e.properties
return pj
end
function
transformationRuleOneIDao(e:Class):IDao
begin
create IDao idao
idao.name = 'I'+e.name+ 'Dao'
idao.methods = declaration of e.methods
return idao
end
function
transformationRuleOneDaoImpl(e:Class):DaoImpl
begin
create DaoImpl daoImpl
daoImpl.name = e.name+ 'DaoImpl'
for each e1 ∈ DaoPackage
  if e1.name = 'I'+e.name+ 'Dao'
     put e1 in interfaces
  end if
end for
link interfaces to daoImpl
return daoImpl
end
function
transformationRuleTwoDto(p:pojo):Dto
begin
create Dto dto
dto.name = p.name
dto.attributes = p.attributes
return dto
end
function
transformationRuleTwoIService(e:Class):IService
begin
create IService iservice
iservice.name = 'I'+e.name+ 'Service'
iservice.methods= declaration of e.methods
return iservice
end
function
transformationRuleTwoServiceImpl(e:Class):Servi
ceImpl
begin
create ServiceImpl serviceImpl
serviceImpl.name = e.name+ 'ServiceImpl'
for each e1 ∈ BusinessPackage
   if e1.name = 'I'+e.name+ 'Service'
      put e1 in interfaces
   end if
end for
link interfaces to ServiceImpl
return ServiceImpl
end

function
transformationRuleThreePresenter(e:Class):Prese
nterImpl
begin
create PresenterImpl presenterImpl
presenterImpl.name= e.name+'PresenterImpl'
for each e1 ∈ PresenterPackage
  if e1.name = 'I'+e.name+ 'Presenter'
     put e1 in interfaces
  end if
end for
link interfaces to presenterImpl
return presenterImpl
end

function
transformationRuleThreeView(e:Class):ViewPackag
begin
create ViewPackage vp
for each e ∈ source model
  if e.methods.name ≠ 'remove'
     create View page
     link  page to vp
  end if
end for
return vp
end
```

Figure 9 illustrates the first part of the transformation code of UML source model to the N-tiers GWT target model.

Figure 9. First part of the transformation code UML2Ntiers-GWT

The transformation uses as input a UML type model, named umlModel, and as output a N-Tiers type model named crudModel. The entry point of the transformation is the main method. This method makes the correspondence between all elements of type UmlPackage of the input model and the elements of type crudProjectPackage output model.

The objective of the second part of this code is to transform a UML package into N-Tiers gwt package, by creating the elements of type package 'Dao', 'Business' and 'UI. It is a question of transforming each class of package UML, to IPresenter and PresenterImpl in the Presenter package, to Dispaly contains widgets in the View Package, to DTO, IService and ServiceImpl in the Business package, and to Pojo, IDao and DaoImpl in the Dao package, without forgetting to give names to the different packages.

B. Result

Figure 10 shows the result after applying the transformation rules.

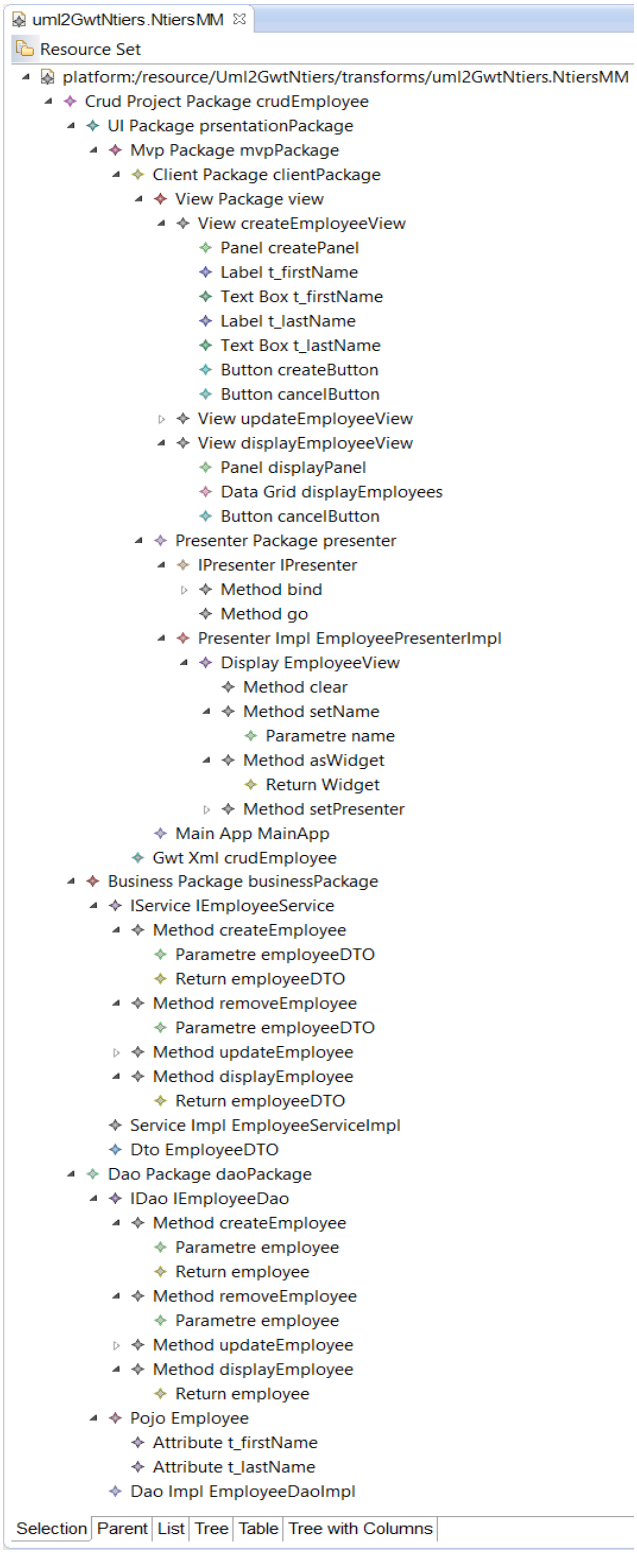

Figure 10. Generated PSM Ntiers-GWT model

The first element in the generated PSM model is UIPackage which includes MvpPackage that contains gwt.xml file and Client Package. The Client Package contains the main application, the Presenter Package and the View Package that contains the Three Views, namely CreateEmployeeView, DisplayEmployeeView and UpdateEmployeeView. Since the operation of the removal requires any view, we'll go to every view element, which contains a multiple element widget like Panel, firstNameTextBox, lastNameTextBox, actionButton and cancelButton. Since the view Display contains the DataGrid widget that contains removal button. The Presenter Package includes one presenter's interface, one presenter's implementation that contains methods with their parameters and their implementations.

The second element in the generated PSM model is businessPackage which includes one service' interface, one service' implementation and one Dto' object correspond to the object 'employee'.

The last element in the generated PSM model is DaoPackage which contains one Pojo' object that contains their attributes, one Dao' interface that contains methods with their parameters and their implementations.

VIII. Conclusion and perspectives

In this paper, we applied the MDA to generate, from the UML class diagram, the code following the MVP, DI and DAO patterns for a RIA web application.

The purpose of our contribution is to finalize the works presented in [2] [14]. This involves developing all meta-classes needed to be able to generate an N-tiers application respecting a MVP, DI and DAO patterns and then we applied the approach by modeling and used the MOF 2.0 QVT standard as a transformation language. The transformation rules defined allow browsing the source model instance class diagram, and generating, through these rules, an XML file containing layers of N-tiers architecture according to our target model. This file can be used to produce the necessary code of the target application. The algorithm of transformation manages all CRUD operations. Moreover, it can be re-used with any kind of methods represented in the UML class diagram.

In the future, this work should be extended to allow the generation of other components of Web application besides the configuration files. Afterward we can consider integrating other frameworks like Flex, JSF and JFace or other execution platforms like PHP and DotNET.

References

[1] Pastor, O.,Molina J.C, *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling* (New York: Springer-Verlag, 2007).

[2] Esbai, R., Erramdani, M., Mbarki, S., Model-Driven Transformation for GWT with approach by Modeling:From UML model to MVP web applications, *International Review on Computers and Software (I.RE.CO.S.),* Vol. 9. n. 9, pp. 1612-1620, September 2014.

[3] Andreas K., Alexander K., and Nora K.., Model-Driven Generation of Web Applications in UWE, *in Proceedings of the 3rd International Workshop on Model-Driven Web Engineering MDWE 2015,* Como, Italy, July 17, 2015

[4] M'hamed Rahmouni, Samir Mbarki, MDA-Based Modeling and Transformation to Generate N-Tiers Web Models. *Journal of Software (JSW),* Volume 10, Number 3, March 2015, pp: 222-238, (2015).

[5] Pawar et al., Model–Driven Content Management of Web Applications Using Atlas Transformation Language, *International Journal of Advanced Research in*

Computer Science and Software Engineering 5(1), Volume 5, Issue 1,January - 2015, pp. 403-410.

[6] Distante, D., Rossi, G., Canfora, G., Modeling Business Processes in Web Applications: An Analysis Framework. *In Proceedings of the The 22nd Annual ACM Symposium on Applied Computing* (Page: 1677, Year of publication: 2007).

[7] 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)

[8] Meliá S., Gómez J., Pérez P., Díaz O., A Model-Driven Development for GWT-Based Rich Internet Applications with OOH4RIA, *Proceedings of ICWE '08. Eighth International Conference on*, Yorktown Heights, NJ, (Page: 13, Year of publication: 2008).

[9] Meliá S., Gómez J., Pérez S., Diaz O. Facing Architectural and Technological Variability of Rich Internet Applications. *IEEE Internet Computing*, vol. 99, pp.30-38, 2010.

[10] S. Ceri, P. Fraternali, and A. Bongio. Web modeling language (WebML): a modeling language for designing web sites. *Computer Network*s, vol. 33(1-6), pp 137–157, 2000.

[11] Preciado J. Carlos, M. Linaje, S. Comai, and F. Sanchez-Figueroa. Designing Rich Internet Applications with Web engineering methodologies. *Proceedings of the 9th IEEE International Symposium on Web Site Evolution* (WSE'07) (Page: 23 Year of publication: 2007).

[12] Trigueros M. L., J. C. Preciado, and F. S ánchez-Figueroa. A method for model based design of Rich Internet Application interactive user interfaces. In ICWE'07: *Proceedings of the 7th International Conference Web Engineering* (page: 226 Year of publication: 2007).

[13] Nasir, M.H.N.M., Hamid, S.H., Hassan, H., WebML and .NET Architecture for Developing Students Appointment Management System, *Journal of applied science*, Vol. 9, n. 8, pp. 1432-1440, 2009

[14] Esbai. R, Erramdani, M., Mbarki, S., Arrassen. I, Meziane. A. and Moussaoui. M., Model-Driven transformation with approach by modeling: From UML to N-tiers Web Model, *International Journal of Computer Science Issues (IJCSI)* , Vol. 8, Issue 3, May 2011.

[15] Esbai. R, Erramdani, M., Mbarki, S., Arrassen. I, Meziane. A. and Moussaoui. M., Transformation by Modeling MOF 2.0 QVT: From UML to MVC2 Web model, InfoComp - *Journal of Computer Science*, vol. 10, no. 3, p. 01-11, September of 2011.

[16] Miller, J., Mukerji, J., al. MDA *Guide Version 1.0.1* (OMG, 2003).

[17] *UML Infrastructure Final Adopted Specification*, version 2.0, September 2003.

[18] *Meta Object Facility (MOF),* version 2.0 (OMG, 2006)

[19] *XML Metadata Interchange (XMI),* version 2.1.1 (OMG, 2007),

[20] Mvp4g A framework to build a GWT application the right way (https://code.google.com/p/mvp4g/) Accessed 09 September 2014

[21] GWT source web site (https://code.google.com/p/google-web-toolkit/)

[22] Echo2 source web site (http://echopoint.sourceforge.net/) Accessed 10 September 2014

[23] Harris, Robert; Warner, Rob, The Definitive Guide to SWT and JFACE (1st ed.), (Apress, 2004).

[24] Vaadin Framework web site (https://vaadin.com/home). Accessed 25 Avril 2014

[25] ZK framework web site (http://www.zkoss.org). Accessed 25 Avril 2014

[26] Nucleo .NET framework web site (http://nucleo.codeplex.com/). Accessed 25 Avril 2014

[27] GWT project web site http://www.gwtproject.org/. Accessed 25 Avril 2014

[28] Fowler, M., Inversion of Control Containers and the Dependency Injection pattern (http://martinfowler.com/articles/injection.html)

[29] Spring Source Web Site (http://www.springsource.org/). Accessed 25 Avril 2014

[30] SpringNet Web Site(http://www.springframework.net/). Accessed 25 Avril 2014

[31] Symfony open-Source PHP Web Framework Site (http://www.symfony-project.org/. Accessed 25 Avril 2014

[32] Panda, D., Rahman, R., Lane, D., EJB3 in action (Manning co., 2007).

[33] PicoContainer. http://www.picocontainer.org/. Accessed 25 Avril 2014

[34] Schincariol, M., Keith, M., Pro JPA 2: Mastering the Java Persistence API (Apress, 2009)

[35] Hibernate Framework (http://www.hibernate.org/). Accessed 25 Avril 2014

[36] Apache Software Foundation: The Apache iBatis Framework (http://ibatis.apache.org/).

[37] NHibernate Framework home site (http://nhforge.org/). Accessed 25 Avril 2014

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

[39] Czarnecki, K., Helsen, S., Classification of Model Transformation Approaches, *Proceedings of the 2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA. Anaheim* (Year of publication: 2003).

[40] Eclipse modeling, http://www.eclipse.org/modeling/. Accessed 20 Avril 2015

[41] AndroMDA web site (http://www.andromda.org/). Accessed 25 September 2014

[42] Günter Graw, Peter Herrmann, Generation and Enactment of Controllers for Business Architectures Using MDA, *Software Architecture*, LNCS Volume 3047, 2004, pp 148-166 (Springer Berlin Heidelberg, 2004)

[43] Xuejiao Pang, Kun Ma, Bo Yang, Design pattern modeling and implementation based on MDA, *Web Information Systems and Mining, LNCS* Volume 6988, 2011, pp 11-18 (Springer Berlin Heidelberg, 2011).

[44] Philip Langer, Manuel Wimmer, Gerti Kappel, Model-to-Model Transformations By Demonstration, *Theory and Practice of Model Transformations, LNCS* Volume 6142, 2010, pp 153-167 (Springer Berlin Heidelberg, 2010)

[45] Jouault, F. Kurtev, I.: Transforming models with ATL. *In: Bruel, J.-M. (ed.) MoDELS 2005, LNCS, vol. 3844*, pp. 128-138. Springer, Heidelberg (2006)

[46] De Lara, J., Guerra, E.: Pattern-based model-to-model transformation. *In: Ehrig, H., Heckel, R., Rozenberg, G., Taentzer, G. (eds.) ICGT 2008. LNCS*, vol.5214, pp. 426-441. Springer, Heidelberg (2008)

[47] Varro, D.:Model Transformation by Example. In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) *MoDELS 2006: LNCS*, vol. 4199, pp. 410-424. Springer, Heidelberg (2009)

[48] Dennis Wagelaar,Viviane Jonckers, Explicit Platform Models for MDA, *Model Driven Engineering Languages and Systems, LNCS* Volume 3713, 2005, pp 367-381 (Springer Berlin Heidelberg, 2005).

[49] Kevin Lano,Shekoufeh Kolahdouz-Rahimi, Model-Driven Development of Model Transformations, *Theory and Practice of Model Transformations, LNCS* Volume 6707, 2011, pp 47-61, (Springer Berlin Heidelberg 2011).

Author Biographies

Redouane Esbai teaches the concept of Information System at Mohammed 1 University,. He got his thesis of national doctorate in 2012. He got a degree of an engineer in Computer Sciences from the National School of Applied Sciences at Oujda. He received his M.Sc. degree in New Information and Communication Technologies from the faculty of sciences and Techniques at Sidi Mohamed Ben Abdellah University. 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, Spring, Struts, GWT, etc.

.

Mohammed Erramdani teaches the concept of Information System at Med I 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.