MVC Frameworks Modernization Approach

Adding MVC Concepts to KDM Metamodel

Amine Moutaouakkil[1], Samir Mbarki[2]
MISC Laboratory. Faculty of Science
Ibn Tofail University
Kenitra, Morocco

***Abstract*—The use of web development frameworks has grown significantly, specially the Model-View-Controller (MVC) based frameworks. The ability to immigrate web applications between different frameworks available becomes more and more relevant. The automation of the migration through transformations avoid the necessity to rewrite the code entirely. Architecture Driven Modernization (ADM) is the most successful approach that standardizes and automates the reengineering process. In this paper, we define an ADM approach to generate MVC web applications models in the highest level of abstraction from Struts 2 and Codeignitter Models. To do this, we add the MVC concepts to the KDM metamodel and then we specify a set of transformations to generate MVC KDM models. This proposal is validated through the use of our approach to transform CRUD (Create, Read, Update and Delete) applications models from MVC frameworks to MVC KDM.**

Keywords*—Framework; Architecture-Driven Modernization (ADM); Knowledge Discovery Model (KDM); Model-View-Controller (MVC)

I. Introduction

Web technology systems are the most used IT solutions in Business management.

More web applications are made with the use of MVC frameworks, these frameworks are constantly evolving, and new frameworks are available. The frameworks: CodeIgniter for PHP language and Struts 2 for Java language are very used. The need to immigrate both from and to these frameworks is increasing.

The existence of a standardized and automatic process of reengineering will minimize time and costs.

Object Management Group (OMG) [1] has proposed the Architecture-driven Modernization (ADM) [2] initiative to enhance the classical reverse engineering processes by introducing the Model-driven Architecture (MDA) [3] concepts. Like the MDA approach which gives the models leading role, the ADM approach formalizes the RE [4] processes by introducing models based concepts.

It is necessary to realize methods for migrating MVC Frameworks based Web applications and define a way to represent the MVC information at a higher abstraction level. But currently there are no relied ADM based approach making it.

We defined an ADM based approach to represent MVC web systems in form of KDM models. This approach takes advantage of the potential of the Architecture Driven Modernization (ADM) to modeling the knowledge which will be extracted from the source code.

In this paper, we describe the generation of models to represent MVC web systems at the highest level of abstraction. The rest of this paper is organized as follows: Section 2 describes the process based on the ADM approach and describes their different phases. In Section 3, we illustrate our proposed approach by a case study and make the analysis of the process result. Then, we list some interesting related works. Finally, Section 4 concludes the work and presents the perspectives.

II. Research Method

A. MDRE

MDRE [5] is the application of Model Driven Engineering (MDE) principles and techniques to RE in order to get model based views from legacy systems. The MDRE is based on two main phases: Model Discovery which is extracting information from source code by using parsers, and then represents this information in form of models. And Model Understanding which is applying Model to Model transformations on extracted information to get a higher abstraction level presentation of the information.

B. ADM

Architecture Driven Modernization (ADM) is an initiative proposed by OMG to standardize and automate the reengineering process. ADM is based on three standards meta-models to represent the information involved in a software reengineering process. In the current study, only Knowledge Discovery Meta-Model (KDM) is useful for the purpose. KDM [6] allows defining models at a higher abstraction level representing semantic information about a software system.

C. Model Understanding

Model understanding consists in the transformation of models to get higher abstract models. In our study, we need to make two transformations:

Apply model to model transformation on the PSM Struts 2 model to get PIM MVC KDM model.

Apply model to model transformation on the PSM CodeIgniter model to get PIM MVC KDM model.

D. QVT Transformation Standard

QVT (Query/View/Transformation) [7] is a standard set of languages for model transformation defined by the OMG.

The QVT standard defines three model transformation languages. All of them operate on models which conform to Meta-Object Facility (MOF) 2.0 metamodels; the transformation states which metamodels are used.

QVT-Operational which we use, it is an imperative language designed for writing unidirectional transformations.

Model Extraction Process is shown in "Fig. 1".

E. Related Works

More and more research projects use the mechanisms offered by the MDA, in among these projects include, e.g.:

- ADM-Based Hybrid Model Transformation for Obtaining UML Models from PHP Code [8].
- Validation of ATL Transformation to Generate a Reliable MVC2 Web Models [9].
- A Model Driven Approach for Modeling and Generating PHP CodeIgniter based Applications [10].
- MoDisco Project [11].
- Reverse Engineering Applied to CMS-Based Web Applications Coded in PHP: A Proposal of Migration [12].

ADM-Based Hybrid Model Transformation for Obtaining UML Models from PHP Code (2019): This paper defines a model transformation process which performs reverse engineering of PHP web-based applications. The model transformation is expressed in ATL [13] (Atlas Transformation Language). The obtained models are expressed in UML.

Validation of ATL Transformation to Generate a Reliable MVC2 Web Models (2017): This paper defines an ADM-based method to generate automatically an MVC2 web model at PSM level which respects the architecture of MVC2 pattern. In This method, a mapping between PSM and PIM metamodels is defined then the transformation script from Struts2 to UML is written In ATL transformation language.

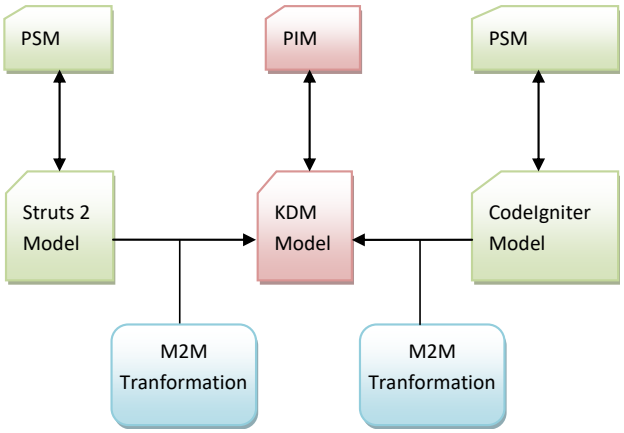

Fig. 1. Models Extraction Process.

A Model Driven Approach for Modeling and Generating PHP CodeIgniter based Applications (2017): This paper defines a Model Driven approach to model the CodeIgniter PHP framework and generate CRUD applications based on this framework. This method uses model transformations.

MoDisco Project (2014): Modisco provides the capability of extracting information from Java software artifacts, The model resulting will conform to meta-model included in Modisco. KDM Models can be extracted using Modisco. KDM allows representing the entire software system and all its entities at both structural and behavioral levels. Modisco is one of rare tools that apply the ADM principles in real. Unfortunately, the current Modisco version does not include any specific support for MVC architecture.

Reverse Engineering Applied to CMS-Based Web Applications Coded in PHP: A Proposal of Migration (2013): This paper defines an ADM-based method for migrating open-source PHP CMS-based Web applications. In the reverse engineering phase, ASTM models are extracted from the PHP code by text-to-model (T2M) transformation made by a source code parser, then KDM models are obtained from ASTM models by model-to-model (M2M) transformation. Finaly CMS model is obtained by using M2M transformations. All M2M transformations are implemented using ATL Transformation Language.

According to the related works we can conclude that ADM approaches that handle MVC structure are inexistent. "Modisco" approach does not offer any support for MVC. The approach "Reverse Engineering Applied to CMS-Based Web Applications Coded in PHP: A Proposal of Migration" uses a CMS metamodel to represent the CMS concepts of a CMS based web system. This logic is avoided in our approach because creating a specific metamodel to represent CMS based web systems or MVC based web systems in our case will contradict the abstraction logic that we want to set, instead of this the use of a metamodel such as KDM that is standardized by OMG is more convenient. In "Modisco Project", "Reverse Engineering Applied to CMS-Based Web Applications Coded in PHP: A Proposal of Migration" as well as "Validation of ATL Transformation to Generate a Reliable MVC2 Web Models" approaches ATL language is used to realize M2M transformations, instead of ATL, our approach uses the QVT language, which is newer and OMG standardized language.

F. The Method Principle

The main idea of our approach is the use of the QVT transformation language to perform Model to Model transformation from MVC web frameworks models to MVC KDM models. Two tasks that the approach will make: The Adaptation of the KDM metamodel by adding MVC concepts, then mapping MVC web frameworks metamodels elements to MVC KDM metamodel elements and then use the mapping to write the QVT transformation script.

G. Struts 2 Framework Metamodel

Apache Struts 2 [14] is an open-source, MVC framework for creating Java web applications. Struts 2 extend the Java Servlet API.

Based on [15], a Struts 2 Framework meta-model “Fig. 2” is defined. This metamodel “Fig. 3” will be useful in our approach.

H. Codeignitter Framework Metamodel

CodeIgniter [16] is a PHP MVC pattern based open-source framework. The MVC pattern structures the development by separating the application logic and the presentation layer. Compared to other PHP frameworks, Codeigniter is known to be fast and light.

Based on [17], a CodeIgniter Framework meta-model “Fig. 4” is defined. This metamodel “Fig. 5” will be useful in our approach.

I. KDM

KDM is a standard defined by OMG for the representation of software systems. This is a meta-model “Fig. 7” allows to represent systems artifacts in a high level of abstraction.

The KDM specification consists of 12 packages that are arranged into the following four layers “Fig. 6” [18]:

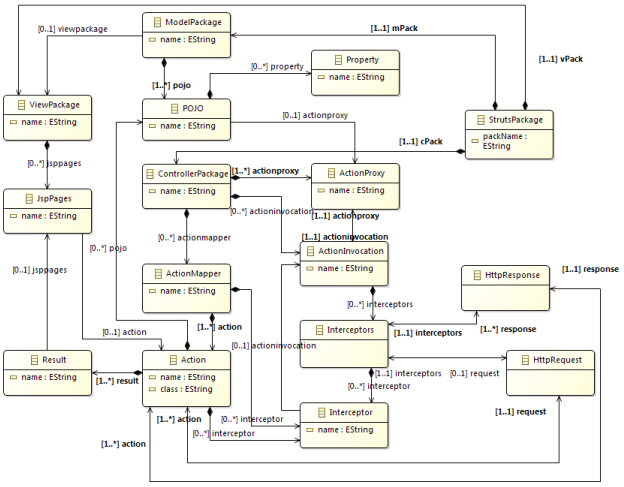

Fig. 2. Struts 2 Meta-Model.

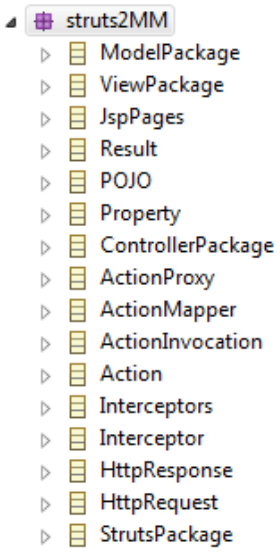

Fig. 3. Struts 2 Ecore Meta-Model.

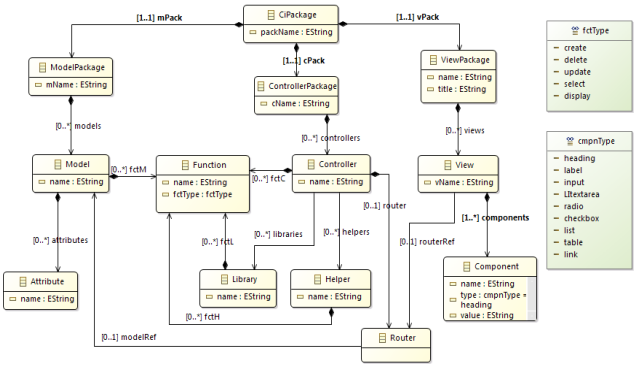

Fig. 4. CodeIgniter Meta-Model.

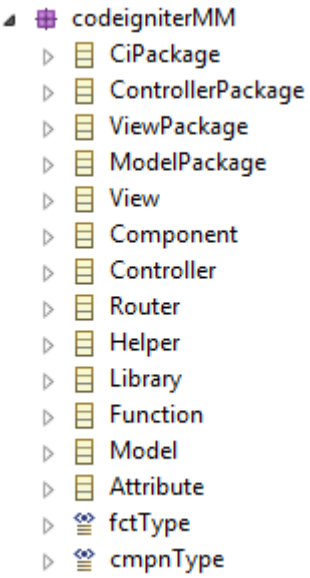

Fig. 5. CodeIgniter Ecore Meta-Model.

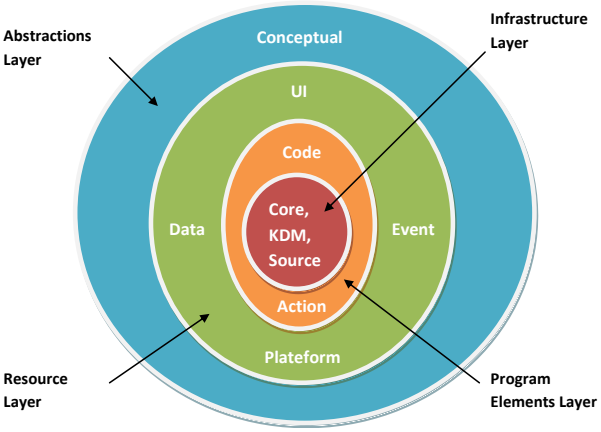

Fig. 6. KDM Meta-Model Layers

We can notice that the KDM metamodel does not give a representation of MVC architecture elements.

J. KDM MVC Package

In our approach we add a MVC package “Fig. 8”. The MVC package represents MVC architecture elements as: models, view and controller.

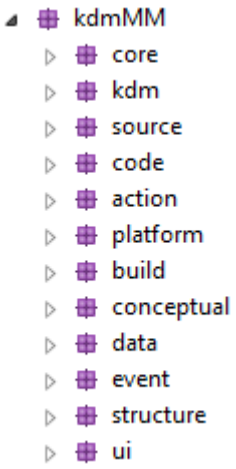

Fig. 7. KDM Ecore Meta-Model Provided by OMG.

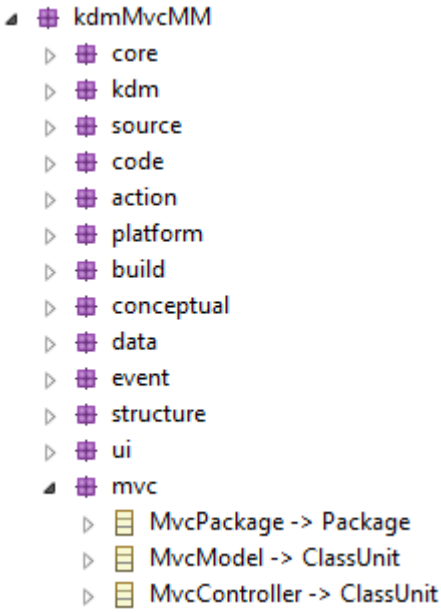

Fig. 8. MVC KDM Ecore Meta-Model.

K. Struts 2 to KDM QVT-O Transformation Script

We have defined a mapping table "Table I" between Struts 2 model elements and KDM model elements.

TABLE. I. STRUTS 2 ELEMENTS TO KDM MVC ELEMENTS MAPPING

Struts 2 element	MVC KDM element
StrutsPackage	code::CodeModel
ModelPackage	mvc::MvcPackage
POJO	mvc::MvcModel
ControllerPackage	mvc::MvcPackage
ActionMapper	mvc::MvcController
ViewPackage	mvc::MvcPackage
JspPages	code::ClassUnit
Property	code::StorableUnit
Action	code::MethodUnit

Based on the mapping table, we have written a QVT-O transformation script to map Struts 2 model elements to MVC KDM model elements.

```
modeltype STR uses 'http://struts2MM';
modeltype KDM uses 'http://kdmMvcMM';
transformation struts2kdmmvc(in str : STR, out
KDM);

main() {
 str.rootObjects()[STR::StrutsPackage]->map R00();
}

mapping STR::StrutsPackage::R00() :
KDM::kdm::Segment {
        model += self.map R0();
}

mapping STR::StrutsPackage::R0() :
KDM::code::CodeModel {
  codeElement += self.getModelPackage()->map R1();
  codeElement += self.getControllerPackage()->map
                        R2();
  codeElement += self.getViewPackage()->map R3();
               result.name := self.packName;
}

mapping STR::ModelPackage::R1() :
KDM::mvc::MvcPackage {
        name := self.name;
        type := "Model";
         codeElement += self.getModels()->map R11();
}

mapping STR::POJO::R11() : KDM::mvc::MvcModel {
 name := self.name;
 codeElement += self.getMdlProperties()->map R4();
                        }
mapping STR::Property::R4() :
KDM::code::StorableUnit {
        name := self.name;
}

query STR::StrutsPackage::getModelPackage() :
OrderedSet(STR::ModelPackage) {
        return
self.subobjects()[STR::ModelPackage]-
>asOrderedSet()
                        }
query STR::ModelPackage::getModels() :
OrderedSet(STR::POJO) {
        return self.subobjects()[STR::POJO]-
>asOrderedSet()
                        }
query STR::POJO::getMdlProperties() :
OrderedSet(STR::Property) {
        return self.subobjects()[STR::Property]-
>asOrderedSet()
}
…
```

L. Codeigneitter to KDM QVT-O Transformation Script

We have defined a mapping table "Table II" between CodeIgniter model elements and MVC KDM model elements.

Based on the mapping table, we have written a QVT-O transformation script to map CodeIgniter model elements to KDM model elements.

```
modeltype CI uses 'http://codeigniterMM';
modeltype KDM uses 'http://kdmMvcMM';
transformation codeigniter2kdmmvc(in ci : CI, out
KDM);

main() {
        ci.rootObjects()[CI::CiPackage]->map R00();
}

mapping CI::CiPackage::R00() : KDM::kdm::Segment {
        model += self.map R0();
}

mapping CI::CiPackage::R0() : KDM::code::CodeModel {
  codeElement += self.getModelPackage()->map R1();
  codeElement += self.getControllerPackage()->map
R2();
  codeElement += self.getViewPackage()->map R3();
  result.name := self.packName;
}

mapping CI::ModelPackage::R1() : KDM::mvc::MvcPackage
{
        name := self.mName;
        type := "Model";
        codeElement += self.getModels()->map R11();
}

mapping CI::Model::R11() : KDM::mvc::MvcModel {
 name := self.name;
 codeElement += self.getMdlAttributes()->map R4();
}

…

mapping CI::Attribute::R4() : KDM::code::StorableUnit
{
        name := self.name;
}

query CI::CiPackage::getModelPackage() :
OrderedSet(CI::ModelPackage) {
        return self.subobjects()[CI::ModelPackage]-
>asOrderedSet()
}

query CI::ModelPackage::getModels() :
OrderedSet(CI::Model) {
        return self.subobjects()[CI::Model]-
>asOrderedSet()
}

query CI::Model::getMdlAttributes() :
OrderedSet(CI::Attribute) {
        return self.subobjects()[CI::Attribute]-
>asOrderedSet()
}
```

TABLE. II. CODEIGNITER ELEMENTS TO KDM MVC ELEMENTS MAPPING

CodeIgniter element	MVC KDM element
CiPackage	code::CodeModel
ModelPackage	mvc::MvcPackage
Model	mvc::MvcModel
ControllerPackage	mvc::MvcPackage
Controller	mvc::MvcController
ViewPackage	mvc::MvcPackage
View	code::ClassUnit
Attribute	code::StorableUnit
Function	code::MethodUnit

III. DISCUSSION

In the ADM approach, KDM is very important. This meta-model allows representing the structural and semantic aspect of the software systems artifacts in the higher possible level of abstraction.

Being represented in form of KDM model, thanks to its high level abstraction, the immigration of an MVC web system to another platform becomes easier. Adding MVC concepts to KDM metamodel will also enlarge its domain of application and allows to more systems to be modeled in a higher level of abstraction. Obtained results correspond to our aim goal which was to represent MVC web systems in a high level abstract way.

IV. RESULTS

A. Case Study Example

Both of CodeIgniter and Struts 2 give the possibility to manage basic CRUD operations through their predefined structures. So, our approach, takes a basic CRUD example as a case study.

1) Struts 2 model "Fig. 9":

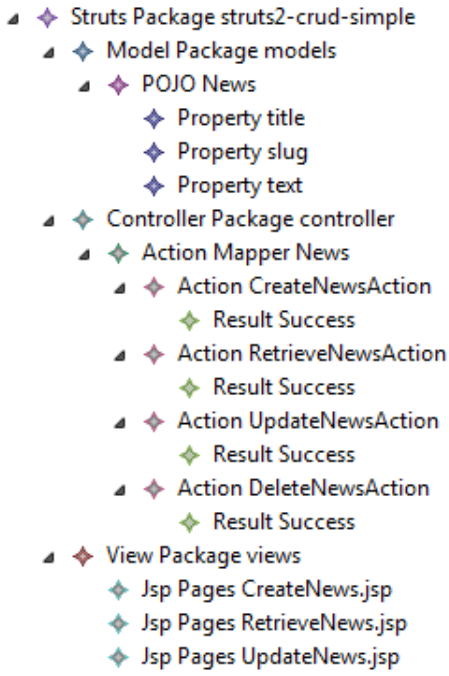

Fig. 9. Stuts2 CRUD Example.

2) CodeIgniter model "Fig. 10"

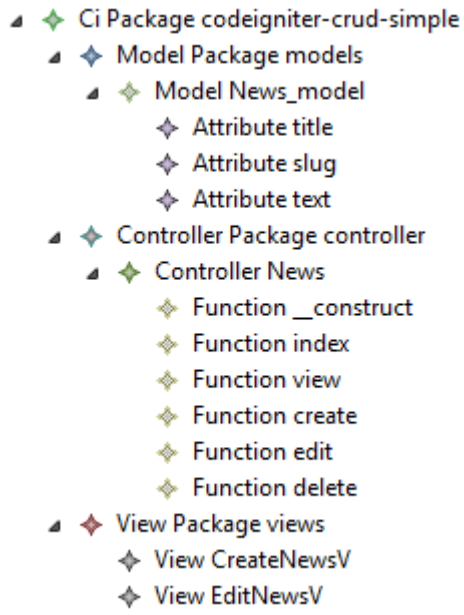

Fig. 10. Codeignitter CRUD Model Example.

B. Results

1) MVC KDM model obtained from Struts 2 models "Fig. 11".

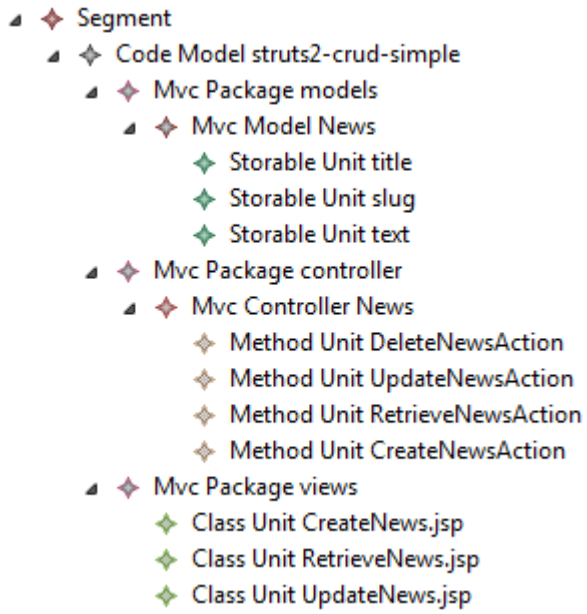

Fig. 11. MVC KDM Model Obtained from Struts2 CRUD Model.

2) MVC KDM model obtained from CodeIgniter models "Fig. 12"

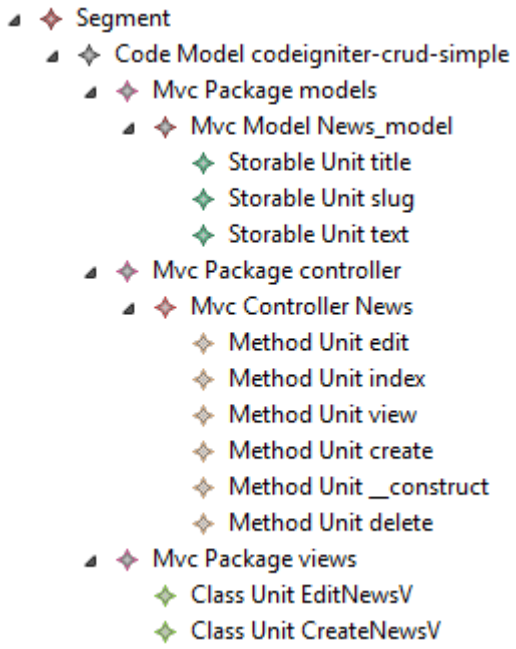

Fig. 12. MVC KDM Model Obtained from CodeIgniter CRUD Model.

V. Conclusion

This paper presented an ADM based approach that Adds MVC Concepts to KDM metamodel.

This approach is composed of two phases: 1) adding MVC main concepts to the KDM metamodel, 2) Generation of KDM models, KDM models are generated from Struts 2 and CodeIgniter models by means of M2M transformations. For the implementation of the Generation of KDM models phase we have implemented transformation rules using QVT-Operational language.

The major contribution is the adding of the MVC concepts to the KDM metamodel and the use of the QVT transformation language defined by the OMG to realize transformations between platforms. Using this approach reduces the necessary time to migrate the MVC web applications. As a future work, we will perform similar approach to other MVC platforms and we will realize the reverse way which is the generation of a specific MVC web system from KDM abstract models.

References

[1] Object Management Object (OMG) . [Online]. Available: http://www.omg.org/.

[2] Architecture Driven Modernization (ADM) . [Online]. Available: http://adm.omg.org/.

[3] A. Kleppe, J. Warmer, W. Bast, "MDA Explained: The Model Driven Architecture: Practice and Promise". Addison-Wesley Professional. January 2003.

[4] E. J. Chikofsky, J. H. Cross II, "Reverse engineering and design recovery: A taxonomy," IEEE Software. vol 7, issue 1 , pp. 13-17, January 1990.

[5] C. Raibulet, F. Arcelli Fontana, M. Zanoni, "Model-Driven Reverse Engineering Approaches : A Systematic Literature Review," IEEE Access. vol 5 , pp. 14516-14542, December 2017.

[6] Knowledge Discovery Meta-Model specification of the OMG. [Online]. Available: http://www.omg.org/spec/KDM/1.3.

[7] I.Arrassen, A.Meziane, R.Sbai, M.Erramdani, "QVT transformation by modeling : From UML Model to MD Model," International Journal of Advanced Computer Science and Applications (IJACSA). vol 2, issue 5, pp. 7-14, January 2011.

[8] A. Elmounadi, N. Berbiche, N. Sefiani, N. El Moukhi, "ADM-Based Hybrid Model Transformation for Obtaining UML Models from PHP Code," International Journal of Embedded Systems (IJES). vol 7, issue 1, pp. 32-41, January 2019.

[9] S. Mbarki, M. Rahmouni, "Validation of ATL Transformation to Generate a Reliable MVC2 Web Models," International Journal of Engineering and Applied Computer Science (IJEACS). vol 2, issue 3, pp. 83-91, March 2017.

[10] K. Arrhioui, S. Mbarki, O. Betari, S. Roubi and M. Erramdani, "A Model Driven Approach for Modeling and Generating PHP CodeIgniter based Applications," Transactions on Machine Learning and Artificial Intelligence (TMLAI). vol 5, issue 4, pp. 259-266, August 2017.

[11] H. Brunelière, J. Cabota, G. Dupé, F. Madiot, "MoDisco: a Model Driven Reverse Engineering Framework," Information and Software Technology, Elsevier, vol 56, issue 8, pp. 1012-1032, April 2014.

[12] FF. Trias, V. de Castro, M. López-Sanz, and E. Marcos, "Reverse Engineering Applied to CMS-Based Web Applications Coded in PHP: A Proposal of Migration. Evaluation of Novel Approaches to Software Engineering," 8th International Conference (ENASE). Angers. pp. 241-256, July 2013.

[13] M. Rahmouni, S. Mbarki, "MDA-Based ATL Transformation to Generate MVC 2 Web Models," Int. J. Comput. Sci. Inf. Technol., vol 3, issue 4, pp. 57–70, August 2011.

[14] Apache Struts Framework. [Online]. Available: https://struts.apache.org/.

[15] S. Mbarki, M. Rahmouni, "Validation of ATL Transformation to Generate a Reliable MVC2 Web Models," International Journal of Engineering and Applied Computer Science (IJEACS). vol 2, issue 3, pp. 83-91, March 2017.

[16] Codeigniter Web Framework. [Online]. Available: https://www.codeigniter.com/.

[17] K. Arrhioui, S. Mbarki, O. Betari, S. Roubi and M. Erramdani, "A Model Driven Approach for Modeling and Generating PHP CodeIgniter based Applications," Transactions on Machine Learning and Artificial Intelligence (TMLAI). vol 5, issue 4, pp. 259-266, August 2017.

[18] KDM Technical Overview from KDM Analytics. [Online]. Available: http://kdmanalytics.com/resources/standards/kdm/technical-overview/