The Reverse Engineering of a Web Application Struts Based in the ADM Approach

https://doi.org/10.3991/ijoe.v16i02.11213

Fouad Elotmani (✉), Redouane Esbai, Mohamed Atounti
Mohammed First University, Oujda, Morocco
f.elotmani@ump.ac.ma

Abstract—Since web technologies are constantly evolving, the adaptation of legacy web applications (WAs) to new paradigms such as rich internet applications (RIAs) has become a necessity. In such tendencies, it has been noticed that several web leaders have already migrated there WAs to RIAs. However, they face many challenges due to the variety of frameworks. Nevertheless, in order to facilitate the process of migration, it would be ideal to use tools that could automatically help generate or ease the generation of UML (Unified Modeling Language) models from legacy WAs. In this context, novel methodology used for migration process such as Architecture-Driven Modernization (ADM), were transformed to describe specifications and promote industry accord on the modernization of existing applications. In this paper, we propose a process to migrating a Struts WA into UML model using the toolkit MODISCO. Then, we present a case study as an example that illustrates the different steps of the proposed process. Finally, we validate the proposition within EMF (Eclipse Modeling Framework) since a number of its tools and run-time environments are conforming to ADM standards.

Keywords—Architecture-driven Modernization, Reverse Engineering, Model Transformation, UML, KDM, MODISCO, EMF, Eclipse

1 Introduction

The business needs and technology evolution make WAs more difficult in terms of the development, interactivity and usability of their user interfaces. The complexity and variety of these applications require flexibility and mixture of operations with existing models to create other new. As more advanced models are used, the importance of transformations between models increases. The Applications development aligned with current paradigms requires adapting WAs to new technologies due to the diversity of frameworks and platforms. The high cost and technical complexity of targeting development on a large number of platforms have forced developers to create applications adapted to each framework. Novel programming languages are therefore emerging to integrate the native behavior of different targeted platforms into development projects. In this direction, the UML modeling language allows, as simple

as possible, modeling applications that focus on all major infrastructure platforms such as NET, JEE and PHP in an extremely basic means.

Novel technical for information assimilation and tools like the Model-Driven Development (MDD) will facilitate to handle a large variety of internet innovations. [1] MDD provides principles and also methods to represent code through models at completely different abstraction levels. A particular concept of MDD is the Model-Driven architecture (MDA) that is created by the object Management group (OMG) [2] the outstanding ideas behind MDA are separation of the specification of the system needs from its implementation, managing the code evolution from abstract models to implementations [3] [4].

The Architecture-Driven Modernization (ADM) approach has actually established a group of services for information system innovation. ADM is defined as "Architecture-Driven Modernization which is the process of understanding and evolving existing software assets for the purpose of software improvement, modifications, interoperability, refactoring, restructuring, reuse, porting, data migration[5], language translation, enterprise application integration, SOA (Service-oriented architecture), and MDA migration."[6] The OMG ADM Task Force (ADMTF) is devlopmofing a collection of criteria (metamodels) to facilitate ability between innovation tools. To date, ADMTF has actually published many standards, but in this work, we will just focus on two of them: KDM [7] (Knowledge Discovery Metamodel) and ASTM (Abstract Syntax Tree Metamodel) [8].

The success of approaches like ADM and MDA rely on the presence of INSTANCE tools that create a major impact on software application processes like forward and reverse engineering processes. The EMF was produced for helping with system modeling and for that reason the automated generation of Java code [9] EMF started as an associate execution of MOF (Meta-Object Facility) resulting Ecore (meta model in the EMF context), the EMF metamodel or the Ecore is similar to EMOF. EMF has actually advanced starting from the knowledge of the Eclipse community to execute a spread of devices that are exceptionally related to MDD. [10] Throughout this context, the subproject Model to Model Transformation (MMT) hosts model-to-model transformation languages. Transformations are performed by modernization tools that are linked into the Eclipse Modeling framework. For example, Atlas Transformation Language (ATL) [11] could be a model for transformation language as well as toolkit that paves the way that to generate a group of target models from a group of source models.

Today, one of the most total modern technology that sustain ADM is MODISCO [12], that provides a common as well as extensible structure to promote the advancement of tools to extract models from legacy systems and use them on usage instances of modernization. As an Eclipse part, MODISCO will certainly integrate with plugins or technologies offered within the Eclipse setting [13].

In this work, we suggest a reverse engineering process from Struts WA to UML class's diagram [14] that incorporates ADM criteria with MODISCO. The procedure adheres to reverse engineering principles; the initial transformation extracts an AST model corresponding to a Java model from Struts application code using MODISCO, the second transformation gets KDM model from AST model. This transformation is

applied in QVTo [15] and the last transformation produces an UML Model from KDM model, this transformation is implemented in ATL.

This paper consists of a simple study, the reverse engineering of a Struts application CRUD (Create, Read, Update, and Delete) operations that enable us to exhibit the various steps of the process.

This paper is organized as follows. Section II offers related works that emphasize our contribution. Section III, Background, soon explains OMG requirements for model's transformation and modernization. In Section IV, a MODISCO project overview is shown. In section V, we have a tendency to provide the reverse engineering process from Struts application to UML Model. Section VI shows a simple case study in the section. Lastly, in Sections VII and VIII we give a result of our technique and also a conclusion respectively.

2 Related Work

- A systematic WA-to-RIA model driven modernization has been suggested by authors in the reference [16], it aims at removing navigational models from Struts-based WAs, using a set of several model driven artifacts to define a specific and reusable process.
- The process that has been provided in [17] consists of creating an RIA client from the existing WA (WA) model, and also the navigating layer and service-oriented connection layer corresponding with the company logic at server side. Authors have particularly focused on the RIA pattern recognition activity. Master/Detail Screen pattern and also Quick look pattern have actually been recommended as possible solutions for multipage master/detail partnerships on the existing WA.
- Authors of the paper [18] have actually concentrated on the reverse engineering process which is defined to represent the pertinent information from the legacy web system. They have outlined their activities into three main actions: the very first one is the removal of technology-dependent details from the existing system, the second one is the generation of a model satisfied our MVC metamodel and lastly the transformation of this model towards a MDWE method.
- The work proposed in [19] recommends an approach in the direction of migration from a legacy system into a modern one; by using the reverse engineering method, in the first step, writers have actually defined a modeling language that permits the meaning of platform-specific models representing the syntax of the source code in PHP (ASTM PHP designs). This modeling language is defined as using a Domain name Details Language (DSL) that they have called ASTM PHP DSL, in the second step they have actually defined and also executed as a collection of automated Model-to-Model transformations that allow to produce platform-independent models (KDM models) from the ASTM PHP models.
- The approach which is recommended by Cosentino et al [20] purposes to extract business rules of Java source code by isolating the code sections worrying the business procedures. To accomplish this goal, the method makes use of model-driven reverse engineering concepts applying the adhering to primary actions:

model discovery from Java code, variable category and domain name variables model production, and business rules model extraction. The model discovery is executed with MODISCO; while the following steps represent a model-to, model change chain applied making use of the Atlas transformation Language (ATL).

3 Background

In this section, we briefly describe OMG standards for model transformation and modernization, as well as the toolkit MODISCO used in this work.

3.1 Standards for model transformation

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

Approach by modeling is used in the present paper. It consists of applying concepts from model engineering to models' transformations themselves.

MOF 2.0 QVT: The objective is modeling a transformation to reach perennial and productive transformation models, and to express their independence towards the platforms of execution. Consequently, OMG elaborated a standard transformation language called MOF 2 .0 QVT (Query/View/Transformation. 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.

This work uses the QVT-Operational mappings language implemented by Eclipse modeling to transform a Java model to KDM.

ATL: ATL (ATL Transformation Language) is a model transformation language and toolkit. In the field of Model-Driven Engineering (MDE), ATL provides ways to produce a set of target models from a set of source models. Developed on top of the Eclipse platform, the ATL Integrated Environment (IDE) provides a number of standard development tools (syntax highlighting, debugger, etc.) that aims to ease development of ATL transformations which is a part of the Eclipse M2M (Model-to-Model) and this work uses the ATL language to transform a KDM model to UML model.

3.2 Standards for modernization

From a standardization point of view, concrete implementations of three OMG ADM standard metamodels are obtainable, namely KDM, SMM [21] and ASTM as introduced before. KDM and ASTM are two complementary modeling specifications. KDM establishes a specification that permits representing semantic information about a software system, whereas ASTM establishes a specification for representing the source code syntax by suggests that of AST. ASTM acts because the lowest level foundation for modeling software inside the OMG ecosystem of standards, whereas

KDM is an entry to the higher-level OMG models. SMM is employed to each specify any kind of measure/metric on legacy software and express the obtained results (measurements). All of them include an EMF Ecore version of the metamodel additionally to the generated model handling API . Moreover, there is a more advanced support for KDM. A corresponding model discoverer permits the automated analysis and illustration of the file hierarchy of applications as so-called KDM "Source" models (using a subset of KDM), and a predefined transformation permits obtaining UML2 models (class diagrams in this case) from KDM "Code" models (using another subset of KDM). Also, a small framework has been developed based on the KDM metamodel to facilitate the future building of recent model discoverers mixing each physical resource metadata (KDM models) and code content information (e.g., Java models).

3.3 MODISCO project overview

MODISCO is an open source project that forms a part of the Eclipse Foundation [22] as well as it is incorporated within the Designing top-level project promoting MDE (Model-Driven Engineering principles) as well as their growth amongst the Eclipse community. It is additionally nowadays formally identified as well as priced quote by the OMG as supplying the recommendation applications and tooling of many of the ADM task force standards:

- The Knowledge Discovery Metamodel (KDM)
- The Software Measurement Metamodel (SMM)
- The Generic Abstract Syntax Tree Metamodel (GASTM)

As shown in Figure 1, from every kind of prospective existing artefacts (e.g. source code, databases, configuration files, paperwork, and so on) MODISCO focuses on giving the necessary capabilities for producing the matching model representations and permitting their handling, analysis and usage. Depending on the Eclipse Modeling Task as well as notably the Eclipse Modeling Framework (EMF), it uses various parts like discoverers, transformations, and also generators to apply this assistance. As a result, the structure targets the production of different views/artifacts on/from the thought-about heritage systems, relying on the expected use of the reverse engineering outcomes (e.g., software innovation, refactoring, retro-documentation, top quality analysis, etc.). One in all the main goals of MODISCO is to remain adaptable to several different situations, so promoting its adoption by a possibly bigger user base.

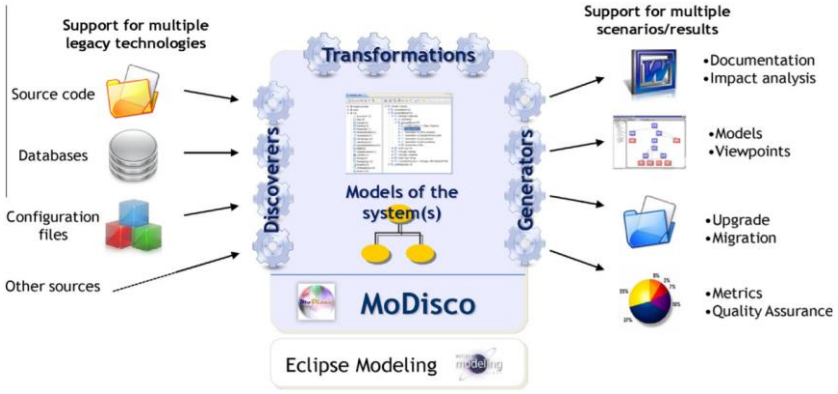

Fig. 1. Overview of the Eclipse-MDT MODISCO project (Formatted from [9]).

4 Migrating Legacy Code in the ADM Context

In this paper, a reverse engineering process from Struts WA to UML model within the ADM context is forecasted. The process adheres the reverse engineering principles; the first transformation extracts a GAST model particular to Java model from Struts application code. The second transformation obtains a KDM model from the GAST model; this transformation is executed in QVT. The last transformation produces an UML model from KDM model; these transformations are executed in ATL.

For each and every transformation, source as well as target metamodels are specified. A source metamodel specifies the family members of resource models to that transformation will be used. A target metamodel characterizes the generated models. Figure 2 summarizes the suggested process.

The primary step is the reverse engineering of source code to get the abstract syntax structure of the code and includes 2 phases:

- Getting the very first model of the code by utilizing a model injector. This model adapts source code metamodel like Struts framework and Java. The acquired model could be refactored to restructure and also customize the syntactical elements. The refactoring is implemented as a model-to-model transformation whose source and also target models are circumstances of source code metamodel.
- Getting the abstract syntax structure tree model, instance of the ASTM (Abstract Syntax Tree Metamodel), from the obtained model within the previous phase by an ATL model-to-model transformation.

In this initial step of the process, an injector and a transformation to get the ASTM model ought to be applied for each language, whereas the series of concerned transformations in the followings steps of the migration process are independent of the language of the existing code.

The second action creates the KDM model. This process is guided by an QVTo model-to-model transformation that takes as input a model conforming to the ASTM metamodel and produces a model that conform with the KDM metamodel.

The next action is connected to an ATL model-to-model transformation that generates a diagram class from a KDM model. After that, it is possible to create other kind of frameworks from the class representation model by using model-to-text transformation by the uses of Acceleo [23].

The work was valid in the open resource application platform Eclipse taking into consideration that some of its tools and run-time environments are straightened with ADM. Eclipse gives executions of several metamodels like Java, ASTM as well as KDM adapting Ecore metamodel.

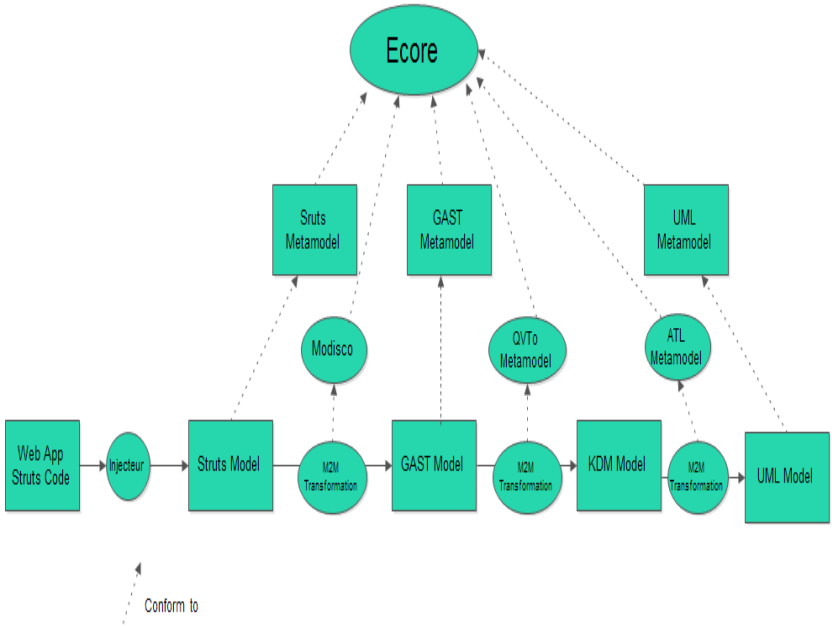

Fig. 2. The Reverse Engineering Process.

5 Case Study: Reverse Engineering Struts Project Code Model

In this section, we define a reverse engineering process from Struts project code to a high level of abstraction in UML class diagrams. This process begins extracting Java models from the Struts code. Next, these models are transformed into KDM models that allow generating UML diagrams class which can be utilized to produce a various target Frameworks such as RIA, in a simple way.

To illustrate the reverse engineering process, we define a straightforward case study, how to reverse the Struts code of a CRUD operations to UML model.

CRUD operations are most commonly implemented in all systems. That is why we have actually taken into consideration in our migration rules these sorts of operations.

The subsections describe the steps of the migration process.

5.1 Struts and UML metamodel

In our ADM approach, we have opted for the modeling and template approaches to generate the UML Model. As mentioned above, these approaches require a source meta-model and a target meta-model. In this section, we present the various meta-classes forming the Struts WA source meta-model and the UML class diagram target meta-model.

Struts metamodel: The Struts Metamodel conforms to Ecore and is composed of four essential parts.

Figure 3 illustrates the first part of the Struts metamodel. This meta-model is a simplified diagram of relational databases. It consists of several tables, themselves composed of typed columns.

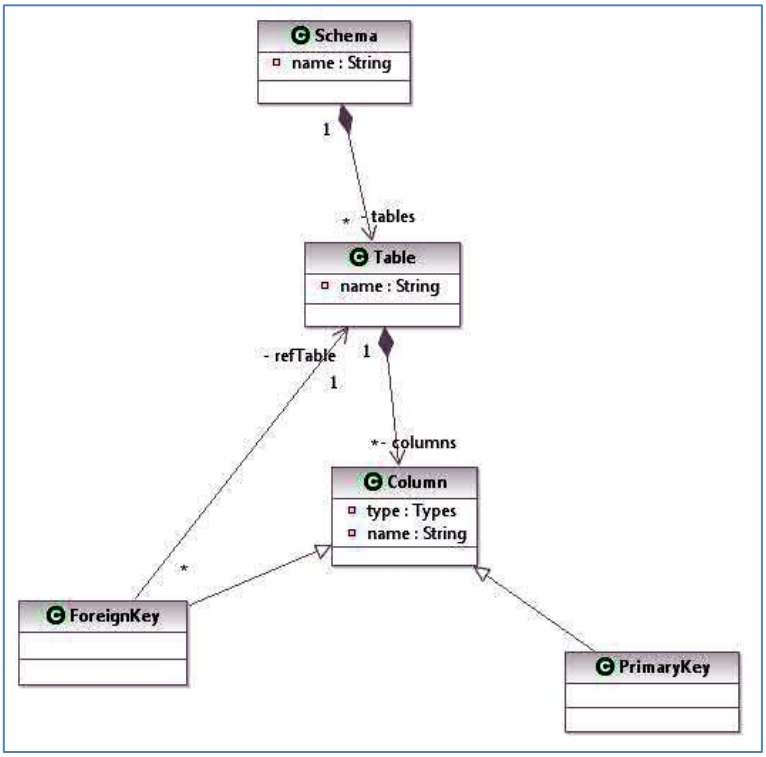

Fig. 3. Simplified Meta-Model of a Relational Database

Figure 4 illustrates the second part of the target meta-model. This is the business model of the application to be processed. In our case, we have opted for components such as Beans. We recall that Struts which does not provide specific classes.

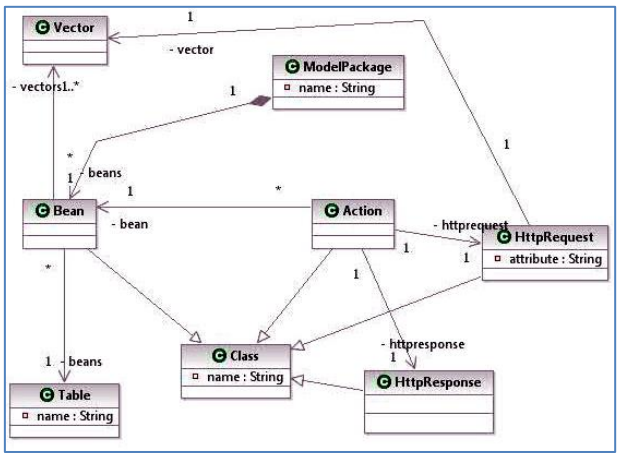

Fig. 4. Simplified Meta-Model of a Model Package

Figure 5 illustrates the third part of the target metamodel. This meta-model illustrates the models that represent the display of the application. In this model, the servlet calls the execute () method on the instance of the class action. It performs its processing and then calls the mapping.findForward() method with a return to the JSP page specified.

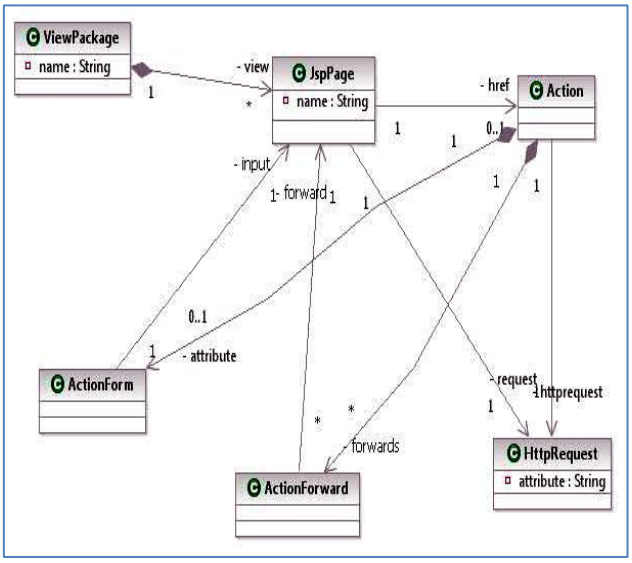

Fig. 5. Simplified meta-model of a View Package

Figure 6 shows the fourth part of the target metamodel. This meta-model is the package controller. This meta-model illustrates models that represent the controller application. The controller is responsible for receiving applications sent by the client, with the invocation of the class action. It, thus, interacts with the business model and coordinates with the display by sending it to the client.

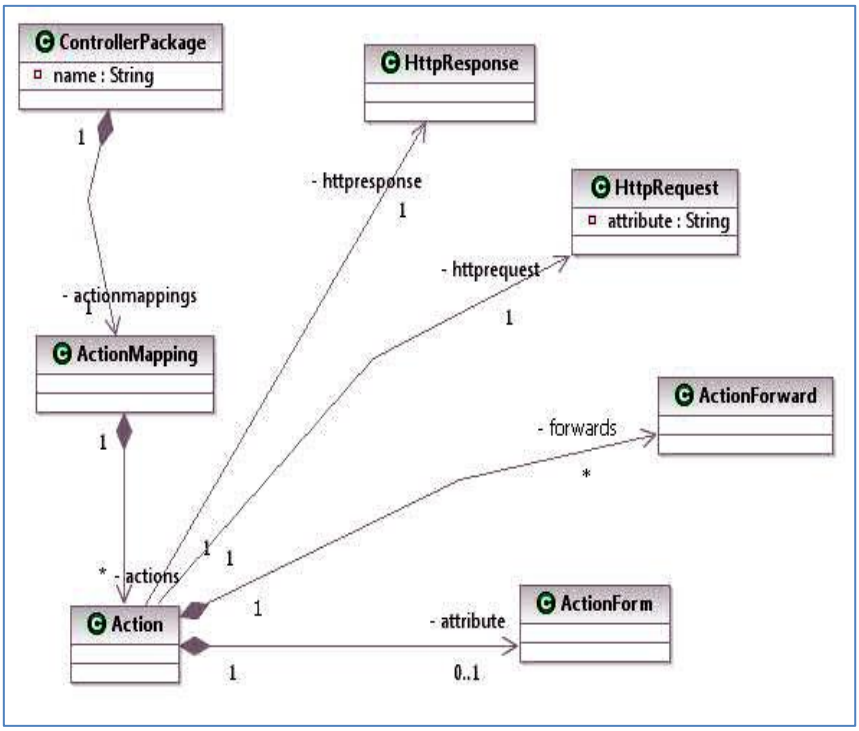

Fig. 6. Simplified Meta-Model of a Controller Package

UML metamodel: Figure 7 illustrates the simplified UML source meta-model based on packages including data types and classes. Those classes contain typed properties and they are characterized by multiplicities (lower and upper). The classes are composed of operations with typed parameters.

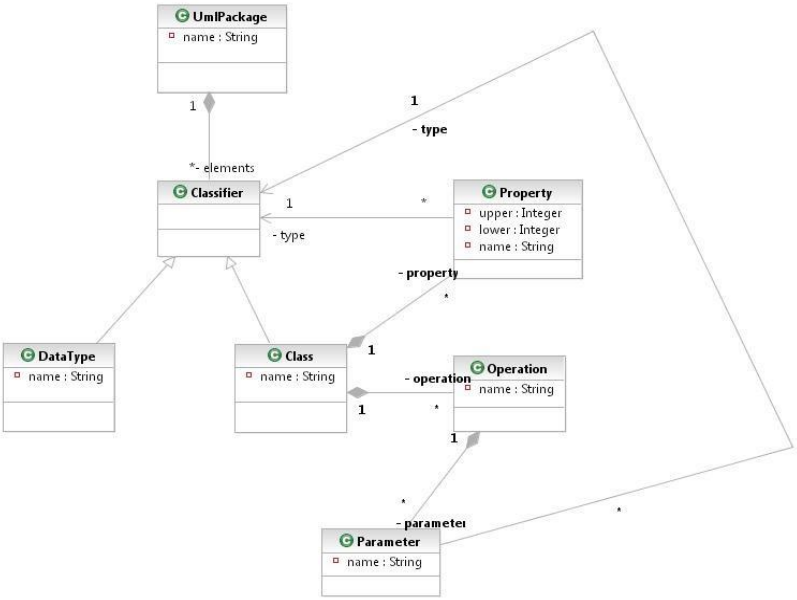

Fig. 7. Simplified UML Meta-Model

- **UmlPackage:** Is the concept of UML package. This meta-class is connected to the Meta-class Classifier
- **Classifier:** This is an abstract meta-class representing both the concept of UML

  Class and the concept of data type:

- **Class:** Is the concept of UML class
- **DataType:** Represents UML data type
- **Operation:** Is used to express the concept of operations of a UML class.
- **Parameter:** Expresses the concept of parameters of an operation. There are two-Types: Class or DataType. It explains the link between Parameter meta-class and Classifier meta-class.
- **Property:** Expresses the concept of properties of a UML class. These properties are represented by the multiplicity and meta-attributes upper and lower.

6 Obtaining Java Model from Struts Project Code

The figure 8 shows the UML presentation of our example that illustrates the Struts model, which conforms to the simplified meta-model of a relational database above.

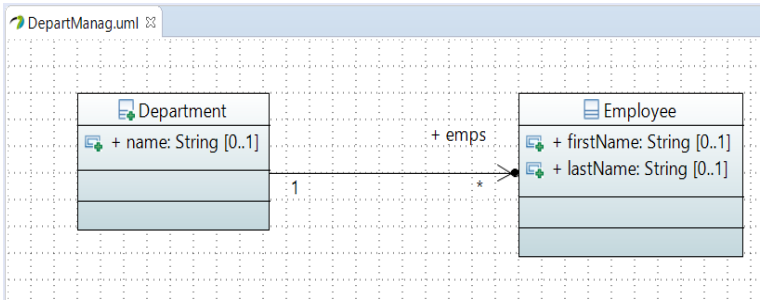

Fig. 8. The UML Presentation of the Struts Model.

The figure 9 shows the WA of our case study, which implements the Struts project, in fact it is the entry point of our reverse engineering, from which, we will extract the UML model.

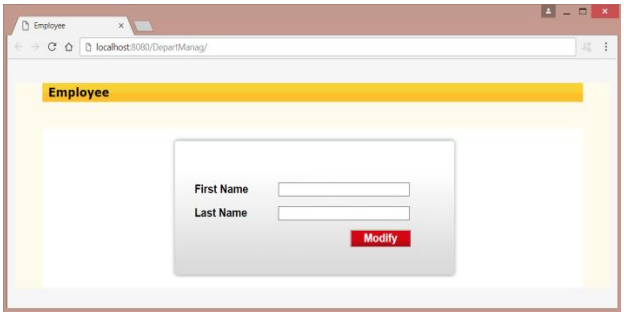

Fig. 9. The WA implementing Struts Framework.

We initially established ECORE models representing our source and also target meta-models.

Then, we extract an AST model that conforms to Java model from the Java code, we can do this with the tool MODISCO very fast.

Figure 10 shows the obtained result in XMI format.

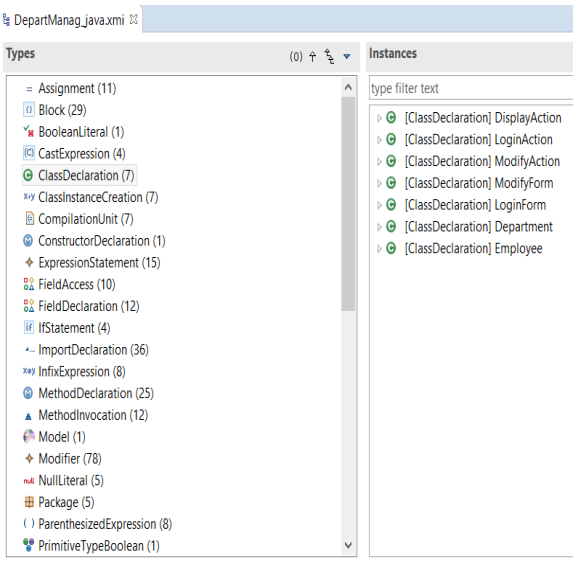

Fig. 10. The Graphical Presentation of the Java Model.

6.1 Obtaining the KDM models

The previous transformations rest of the legacy code language, that is, for every language, the model injector as well as consequently the transformation to obtain the generic AST model should be implemented.

In difference to the previous phase, the sequence of transformations from ASTM models to KDM models are independent of the language, that is, this change is common for all language of the legacy code. This change is applied in QVT as well as specified by means that of QVT components composed of the list below aspects:

- An optional import area that makes it possible for to import some existing QVT libraries.
- A header section that specifies the names of the improvement component and also the variables of the source and also target metamodels.
- A set of mapping rules that defines just how resource model components are matched and also navigated to create and also initialize the aspects of the target models.

To exhibit the QVT transformations, the Java2KDM transformation is partly shown in figure 11

java2kdmm.qvto

```
modeltype javamm uses "http://www.eclipse.org/MoDisco/Java/0.2.incubation/java";
modeltype kdmmm uses "http://www.eclipse.org/MoDisco/kdm/action";
modeltype kdmcode uses "http://www.eclipse.org/MoDisco/kdm/code";

transformation java2kdmm(in javam:javamm, out kdmm:kdmcode);


main() {

    javam.objectsOfType(Package)->select(name='model')->map StrutsPackage2KDMPackage();
}

mapping Package::StrutsPackage2KDMPackage () : kdmcode::Package {

    result.name:=self.name;
    codeElement:=self.ownedElements[ClassDeclaration]->map class2Class();
}
mapping ClassDeclaration::class2Class():ClassUnit {
    name:=self.name;
    codeElement:=self.bodyDeclarations[FieldDeclaration]->map attribut2attribut();
}
mapping FieldDeclaration::attribut2attribut(): StorableUnit{
    name:=self.fragments->first().name;
    type:= self.type.type.map type2type();
}
mapping Type::type2type():InterfaceUnit {
    name:=self.name;
}
```

Fig. 11. The QVT Code Transformation.

The mapping Java2KDM defines the method to produce KDM models (target) from Java models (resource). Some mappings that execute the change are the followings:

The mapping StrutsPackage2KdmPackage changes each JavaPackage into a KDM Package that possesses models such as Code Model containing code elements (data kinds, approaches, variables, etc.) and also Inventory Model that contains physical artefacts of the existing software program system (resource file, binary data, etc.).

The mapping Class2Class changes each Class Declaration right into a Class Unit. Code aspects are gotten from the variables and also function of the original course; source is obtained from the resource code location.

The mapping Attribute2Attribute changes each Declaration Field into a storable Unit.

The resulting model of this migration is partially received figure 12.

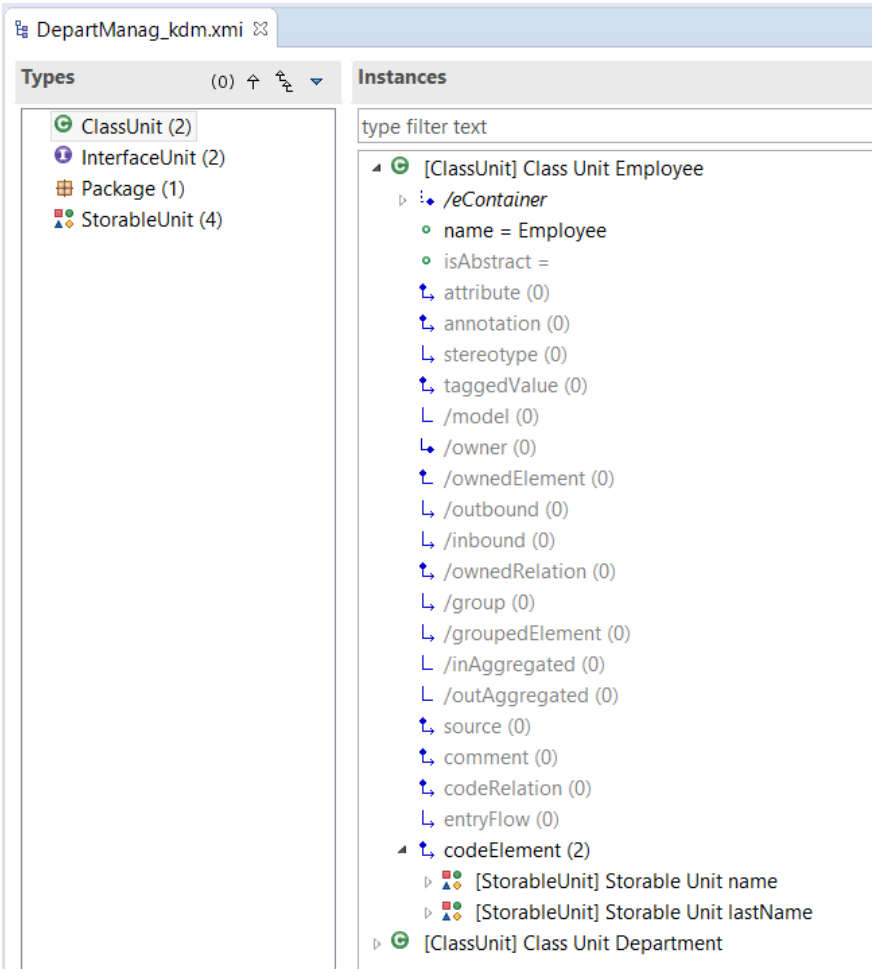

Fig. 12. The Graphical Presentation of the KDM.

6.2 Generating the UML Model

From a KDM model, it is possible to create a class Representations in UML by using ATL.

The KDM-to-UML transformation is partially received figure 13.

*kdm2uml.atl

```
-- @atlcompiler atl2006
-- @nsURI uml=http://www.eclipse.org/uml2/2.1.0/UML
-- @nsURI kdm=http://www.eclipse.org/MoDisco/kdm
--authors: Gabriel Barbier, Mia-Software, gbarbier@mia-software.com
--Transform KDM Models to UML 2.1 models

module KDMtoUML;
create umlOutput : uml from kdmInput : kdm;

rule SegmentToRootModel {
    from src :kdm!Segment
    to tgt :uml!Model(
        name<- if (src.name.oclIsUndefined()) then 'root model' else src.name endif
        ,packagedElement<-src.segment
        ,packagedElement<-src.model
        ,packagedElement<-src.extension
    )
}
rule ExtensionFamilyToProfile {
    from src :kdm!ExtensionFamily
    to tgt :uml!Profile (
        name <- src.name
        ,packagedElement <- src.annotation
        ,packagedElement <- src.attribute
        ,packagedElement <- src.stereotype
    )
}
rule KDMModelToModel {
    from src :kdm!KDMModel
    to tgt :uml!Model (
        name <- src.name
        ,packagedElement <- src.extension
        ,packagedElement <- src.ownedElement
    )
}
```

Fig. 13. The ATL Code Transformation.

The component KDM2UML defines the way to generate UML models (target) from KDM designs (resource). Some guidelines that execute the change are the followings:

The guideline Segment To Root Model changes each KDM sector into UML Bundle.

The Guideline Class2Class transforms each Class Unit into a Class Element. Code aspects are obtained from the variables and also function of the initial course; source is acquired from the resource code area.

The Regulation Attribute2Attribute transforms each storable Unit into an Owned Element.

7 Results

The resulting design of this transformation is displayed in figure 14

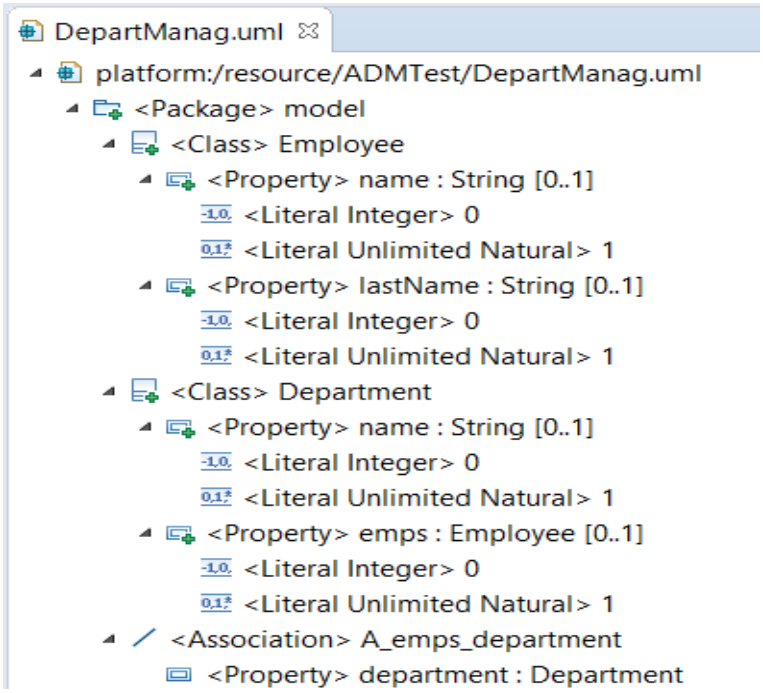

Fig. 14. The Graphical Presentation of the UML Model Obtained

8 Conclusions and Future Work

In this work, we used an approach dedicated to systematic WA-to-RIA model driven modernization. In our study, we have especially focused on models used for extraction from Struts-based WAs. The main activities (locating, representing, transforming) and artifacts (code, metamodel, model, transformation rules), related to the process of extraction have been detailed by means of a running example, and the process itself is led by a variety of model driven artifacts which allow to define a systematic and reusable process. We have also specified our own Struts metamodel in order to define intermediate models that may be eventually projected to the selected approach using model transformations.

The proposal was validated in the open source application platform Eclipse since some of its tools and run-time environments are aligned with ADM standards as mentioned above.

As future works, we are planning to work on extraction models by considering the following steps. The first step is applying our approach to a set of case studies in the purpose to refine it. Subsequently, the second step is targeting other MVC-based Web frameworks to support our approach. The third step is defining a tool chain to support the whole migration process. The fourth step is automatically generating the code of the target WA based on RIA platform as GWT (Google Web Toolkit).

9 References

[1] M. Brambilla, J. Cabot, and M. Wimmer, "Model-Driven Software Engineering in Practice: Second Edition," Synth. Lect. Softw. Eng., vol. 3, no. 1, pp. 1–207, Mar. 2017. https://doi.org/10.2200/S00441ED1V01Y201208SWE001

[2] "Model Driven Architecture (MDA) | Object Management Group." [Online]. Available: https://www.omg.org/mda/.

[3] S. Gotti and S. Mbarki, "IFVM Bridge: A Model Driven IFML Execution", International Journal of Online and Biomedical Engineering (iJOE), vol. 15, no. 04, p. 111, 2019. https://doi.org/10.3991/ijoe.v15i04.9707

[4] A. Azzaoui, O. Rabhi and A. Mani, "A Model Driven Architecture Approach to Generate Multidimensional Schemas of Data Warehouses", International Journal of Online and Biomedical Engineering (iJOE), vol. 15, no. 12, p. 18, 2019. https://doi.org/10.3991/ijoe.v15i12.10720

[5] R. Esbai, F. Elotmani and F. Belkadi, "Toward Automatic Generation of Column-Oriented NoSQL Databases in Big Data Context", International Journal of Online and Biomedical Engineering (iJOE), vol. 15, no. 09, p. 4, 2019. https://doi.org/10.3991/ijoe.v15i09.10433

[6] "ADM Platform Task Force | Object Management Group." https://www.omg.org/adm/.

[7] "About the Knowledge Discovery Metamodel Specification Version 1.4 https://www.omg.org/spec/KDM/.

[8] "About the Abstract Syntax Tree Metamodel Specification Version 1.0." [Online]. Available: https://www.omg.org/spec/ASTM/.

[9] E. Babkin and O. Radzinskaya, "Application of the Eclipse EMF technology for multifaceted organization modelling," Bus. Informatics, vol. 37, no. 3, pp. 15–29, Sep. 2016. https://doi.org/10.17323/1998-0663.2016.3.15.29

[10] "Eclipse Modeling – EMF –Documents | The Eclipse Foundation https://www.eclipse.org/modeling/emf/docs/

[11] "ATL – Documentation | The Eclipse Foundation" https://www.eclipse.org/atl/documentation/.

[12] Eclipse MoDisco | The Eclipse Foundation." https://www.eclipse.org/MoDisco/.

[13] H. Brunelière, J. Cabot, G. Dupé, and F. Madiot, "MoDisco: A model driven reverse engineering framework," Inf. Softw. Technol., vol. 56, no. 8, pp. 1012–1032, 2014. https://doi.org/10.1016/j.infsof.2014.04.007

[14] "About the Unified Modeling Language Specification Version 2.5.1." https://www.omg.org/spec/UML.

[15] "About the MOF Query/View/Transformation Specification Version 1.3." [Online]. Available: https://www.omg.org/spec/QVT.

[16] J. M. Conejero, R. Rodríguez-Echeverría, F. Sánchez-Figueroa, M. Linaje, J. C. Preciado, and P. J. Clemente, "Re-engineering legacy Web applications into RIAs by aligning modernization requirements, patterns and RIA features," J. Syst. Softw., vol. 86, no. 12, pp. 2981–2994, Dec. 2013. https://doi.org/10.1016/j.jss.2013.04.053

[17] R. R. Echeverria, F. Macías, V. M. Pavón, J. M. Conejero, and F. S. Figueroa, "Legacy Web Application Modernization by Generating a REST Service Layer," IEEE Lat. Am. Trans., vol. 13, no. 7, pp. 2379–2383, Jul. 2015. https://doi.org/10.1109/tla.2015.7273801

[18] U. Sabir, F. Azam, and M. W. Anwar, "A comprehensive investigation of Model Driven Architecture (MDA) for reverse engineering," in ACM International Conference Proceeding Series, 2017, pp. 43–48. https://doi.org/10.1145/3178212.3178215

[19] F. 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," in Communica-

tions in Computer and Information Science, 2013, vol. 417 CCIS, pp. 241–256. https://doi.org/10.1007/978-3-642-54092-9_18

[20] V. Cosentino, J. Cabot, P. Albert, P. Bauquel, and J. Perronnet, "A model driven reverse engineering framework for extracting business rules out of a java application," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2012, vol. 7438 LNCS, pp. 17–31. https://doi.org/10.1007/978-3-642-32689-9_3

[21] "About the Structured Metrics Metamodel Specification Version 1.2." [Online]. Available: https://www.omg.org/spec/SMM/.

[22] "Enabling Open Innovation & Collaboration | The Eclipse Foundation." [Online]. Available: https://www.eclipse.org/.

[23] "Acceleo | Home." [Online]. Available: https://www.eclipse.org/acceleo//.

10 Authors

Fouad Elotmani is a PhD student from Mohammed First University (Nador, Morocco). His research interests at the MASI Laboratory (Applied Mathematics and Information System) include model driven modernization, Software development, modeling JEE Frameworks, and modeling architectures.

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 MASI Laboratory (Applied Mathematics and Information System) focusing on MDA (Model Driven Architecture) integrating new technologies XML, Spring, Struts, GWT, etc

Mohamed Atounti is a professor of applied mathematics and computer science at Multidisciplinary Faculty of Nador, University Mohammed First. His research intersts are numerical analysis and computer science in the laboratory of applied mathematics and information systems. Mohamed Atounti is the director of this laboratory.

Article submitted 2019-07-06. Resubmitted 2019-10-30. Final acceptance 2019-10-30. Final version published as submitted by the authors.