Model-Driven Generation of MVC2 Web Applications: From Models to Code

M’hamed RAHMOUNI
Department of Computer Science, Faculty of Science
Ibn Tofail University
Kenitra, BP 133, Morocco

Samir MBARKI
Department of Computer Science, Faculty of Science
Ibn Tofail University
Kenitra, BP 133, Morocco

Abstract—Computer systems engineering is based, increasingly, on models. These models permit to describe the systems under development and their environment at different abstraction levels. These abstractions allow us to conceive applications independently of target platforms. For a long time, models have only constituted a help for human users, allow to manually develop the final code of computer applications. The Model-Driven Engineering approach (MDE) consists of programming at the level of models, represented as an instance of a meta-model, and using them for generating the end code of applications. The MDA (Model-Driven Architecture) is a typical model-driven engineering approach to application design. MDA is based on the UML standard to define models and on the meta-modeling environment (MOF) [1] for model-level programming and code generation. The code generation operation is the subject of this paper. Thus, in this work, we explain the code generation of MVC2 Web application by using the M2M transformation (ATL transformation language) then the M2T transformation. To implement this latter we use the Acceleo generator which is a generator language. In the M2T transformation, we use the PSM model of Struts2 already generated by M2M transformation as an input model of Acceleo generator. This transformation is validated by a case study. The main goal of this paper is to achieve the end-to-end code generation.

Keywords- MDA; PSM; Code Generation; Acceleo; Struts; ATL transformation; MVC2 architecture

I. INTRODUCTION

In recent years, development of Internet, distributed, embedded or self-managed applications has considerably complicated the development of computer systems. These are used in increasingly varied and constrained environments (mobile terminals, cars, robots) and in increasingly critical contexts (aerospace, medical, military, nuclear). In addition, during their lifetime, software is subject to constant evolution to satisfy user needs and adapt to new platforms. Academic and industrial world are thus led to rethink the software production processes in order to adapt them to these new stakes.

A pragmatic vision to adapt to these changes is to allow the software production in automatic way in order to take into account the inevitable evolutions which these are subjected to. The models have always been used in computer science as a basis for reflection. However, they have often been relegated to the rank of simple documentation. Model-driven engineering aims to place models at the heart of the development process to make them the main elements through which applications are generated. Taking model change into account and automatically generating systems by generation from these models seems to be a promising outcome to meet changing constraints.

At the heart of model-engineering are placed the model transformations. They realize the automation of essential stages of software production: refinement and models composition, reverse engineering, code generation and documentation, etc. These transformations are intended to be used repeatedly by development teams, for example in software workshops. Therefore, it is crucial that they be validated and tested so that the developer can have full confidence in them.

The presented work is the continuation of the work presented in [39]. This work allows generate automatically a MVC2 web application that is a source code of an application based on Struts2 framework [1]. To realize this operation of code generation, we used ACCELEO [37] as a code generation language. To achieve this code generation, we begin by implementing the different templates corresponding to each meta-class which is an element of the PSM model already generated in [39]. After establishing the different templates (Action classes and Jsp pages), we validate the automatic code generation with a case study.

The remaining part of this paper is structured as follows: section 2 explains the process and methodology of this work. Section 3 is devoted to the architecture of UML and Struts2 meta-models. Section 4 is devoted to the transformation rules implementation. Section 5 is dedicated to the transformation rules execution and the result of execution process. Section 6 presents the Model-to-Text transformation implementation. The Model-to-Text transformation result is dedicated to the section 7. Section 8 presents the evaluation of this work. Section 9 discusses the main related work, while section 10 wraps up the conclusions and future works.
II. Process and Methodology

In this work, we begin our process by the presentation of the different meta-models (CIM, PIM, and PSM). CIM model is represented by an UML class diagram of a case study of an Employee management. This UML class diagram is represented by an Ecore model. The PIM meta-model is an extract of UML class diagram meta-model. The PSM meta-model refers to the Struts2 meta-model. After models and meta-models, we define the different ATL transformation rules. Then, we implement the KM3 models corresponding to each PIM and PSM meta-model, then the different Ecore models corresponding to these KM3. The next step is devoted to establish the traceability links between different elements of source and target meta-models and thereafter we define the different transformation rules in ATL transformation language. The ATL transformation result is a MVC2 web model represented in EMF model. In the end step, we use the generated PSM model as an input model of Acceleo generator then we generate the application code of the presented case study by applied a M2T transformation based on the cited ACCELEO generator. In the case of M2T transformation, we begin by the implementation of different templates corresponding to the Action classes and the JSP pages. The work of this transformation is validated and exemplified by a case study.

The tools support of this work is the UML [14], ATL transformation language [22]-[23]-[24], MOF [15], XMI [16], KM3 [6], OCL [17], EMF Project [15] and Acceleo [37].

The following section is dedicated to the M2M transformation based on MDA approach and ATL transformation language [25]. In this section, we begin by presenting the different meta-models.

III. UML and Struts2 meta-models

In this section, we explain the different meta-classes that form the UML source and the target meta-models.

A. UML Source Meta-model

Figure 1 presents the UML source meta-model. This meta-model is a simplified UML model composed essentially of packages containing the data types and classes. The main elements of this metamodel are:

- UmlPackage: This is an UML package concept.
- Classify: Represents the generalization concept of meta-class. This latter represents both the concept of UML class as well as the data type concept.
- Class: Shows the UML class concept.
- DataType: Explains the concept of UML data types.
- Operation: Represents the concept of UML class methods.
- Parameter: Expresses the method parameters concept.
- Property: Represents the concept of UML class properties.

B. Struts 2 Target Meta-model

In this section, we present the Struts2 target meta-model. This meta-model is explained in first time in [39]. The different elements that form the Struts 2 meta-model are as follows:

- ModelPackage: Presents the concept of UML package. This package designates the notion of Model in the MVC2 architecture.
- ControllerPackage: Expresses the Controller concept in the MVC2 architecture.
- ViewPackage: Indicates the concept of Views package.
- ActionMapper: Represents the ActionMapper class concept.
- ActionProxy: Expresses the ActionProxy class concept.
- ActionInvocation: Expresses the ActionInvocation class concept.
- Action: Indicates the concept of action in the controller package.
- JspPages: Represents the Jsp package concept.
- Result: Expresses the generated element through an Action class.
- The Interceptors: Represents the concept of an Interceptor package.
- Interceptor: Indicates the interceptor class concept.
- HttpServletRequest: Expresses the concept of HttpServletRequest class.
- HttpServletResponse: Designates the concept of HttpServletResponse class.
Figure 2. PSM Struts2 Meta-model.

IV. TRANSFORMATION RULES IMPLEMENTATION

This section is dedicated to the transformation rules implementation. It explains the different steps from implementation to execution of different transformation rules. In first step, we implement the different meta-models like: struts2.km3, struts2.ecore, UML.km3 and UML.ecore. In second step, we establish the rules specification. Then, we define the different transformation rules based on these specification rules by ATL language in a file named UML2Struts2.atl. Finally, we implement the source model. This latter is an UML class diagram of Employee management represented in XMI language.

The different tools permits to achieve this work are: ATL plug-in integrated in Eclipse, XMI, OCL, KM3, UML, MOF and EMF project.

The next section is dedicated to the implementation of rules specification and then the definition and the execution of ATL transformation rules. Ecore and km3 meta-models are not presented in this paper for letting it quite understandable and clear.

A. Rules Specification

This section is devoted to the presentation of the main rules to transform an UML Class Diagram into Struts2 Web model. The different specification rules are:
An UML package generates a Struts2 package.
The Struts package is composed of a Controller package and a View package.
Each Controller Package is composed of a set of Action classes.
An Action class is composed of a set of Result classes.
An Operation generates an Action and a JSP pages.

B. Rules-Based transformation written in ATL

This section expresses the different rules which transform the UML class diagram into MVC2 web model. The different rules are:

Rule 1: From Operation to Struts 2 Action

This rule represents the main rule of this transformation. Thus, this rule allows generate the different action classes and Jsp pages of each Action. The name of each jsp page is composed from the name of the operation concatenated with the name of the class and followed by the extension “.jsp”. The different rules that compose this main rule are as follow:

- Rule 1: An Operation can generate an Action.
- Rule 2: An Operation can generate a Result.
- Rule 3: Each Result class is composed of a set of Jsp pages.

The main rule is shown in figure 3. These rules are implemented by ATL language.

```
rule Operation2Action{
    from
    c : UML::Operation
to
    js : Ntiers::JspPage {
        name<-'if c.name='Delete'then
            c.name+c.class.name+'.jsp'
        else 'Retrieve'+c.class.name+'.jsp'
        endif
    },
    frm : Ntiers::Action{
        name<-'c.name+c.class.name+Action',
        method<-'c.name+c.class.name',
        class<-'com.web.struts2'+c.class.name+Action',
        result<-'Sequence(fr)'
    },
    fr : Ntiers::Result{
        name<-'Success',
        type<-'redirect',
        jsp<-'js'
    }
}
```

Figure 3. Main Rule : From Operation to Action.

V. TRANSFORMATION RULES EXECUTION

In this section, we present a case study to demonstrate and exemplify our proposition. The UML class diagram of this case study represents the source model of our ATL transformation. The execution algorithm of ATL transformation allows browsing all transformation rules and thereafter generates the MVC2 web model. This latter is represented in figure 6.

A. Case study

The current case study is a system of a three classes. This system makes it possible to manage an employee of a given department. The different classes that compose this system are: City class, Department class and Employee class. In this case study, we elaborate only the CRUD operations (Create, Retrieve, Update, and Delete). These operations are most often implemented in all systems. Figure 4 presents the Ecore model equivalent to the UML class diagram of a department’s employees.

```
platform/resource/accelo.test/model/Employee.ecore
```

Figure 4. Ecore model of a department's employees.

B. ATL Transformation Result

Figure 5 shows the generated MVC2 Web model of Struts2. This model contains the different ingredients for implementing a presentation layer respecting the architecture of MVC2 pattern.

The generated model is composed of a Struts package that is composed of a set of Action classes and the View package that is composed of a set of Jsp pages.
VI. MODEL-TO-TEXT (M2T) TRANSFORMATION IMPLEMENTATION

In this M2T transformation, we use the Acceleo [37] code generator which is a code generator integrated into Eclipse. The following section gives an idea of this generator then a brief history of this latter.

A. ACCELEO Generator

Acceleo [37] is a source code generator of Eclipse foundation that enables the implementation of MDA (Model driven architecture) approach to realize applications from models based on EMF. This is an implementation of the Object Management Group (OMG) standard for model-to-text (M2T) transformations.

The famous Acceleo project was born in 2006 around the Acceleo.org website by Obeo. In its first versions, Acceleo 1.0 and 1.1 were at the time under GPL licence and compatible with Eclipse 3.2 and many modelers based on EMF or UML. In 2009, for his passage to version 3, the project was admitted to the Eclipse Foundation. From this transition, Acceleo changed the language used to define generators to use the standard OMG language for model-to-text transformation. This language is an implementation of the MOF2M2T standard. ACCELEO is based on the concept of template approach.

B. Structure of ACCELEO project

According to the model already generated shown in figure 6, the ACCELEO structure project is composed of the following principles elements:

- The “GenerateStruts2” package presents the different generated classes (Bean and Action classes).
- The “org.eclipse.acceleo.ecore2java.files” package contains the main template explained in figure 8.
- Represents the main template named “genStruts2.mtl”.
- Represents the main template named “genStruts2.mtl”.
- Represents the template of Jsp pages.
- The “GenerateJsp” folder contains the different JSP pages generated by code generation.

Figure 7 presents the created ACCELEO project and the different packages and templates that constitute this project.

In the following section, we present the different templates necessary to implement the automatic code generation by ACCELEO and then to respond to our case study.

C. Code Generation Implementation

In this paper, we present the different templates permits to generate the source code from PSM model (Figure 5) already generated by applying M2M transformation in [39].
The generated PSM model implements only a CRUD application that performs the operations of listing, adding, modifying and deleting on a given entity. This use case is so common in software development that it is rare to find an application that does not do CRUD.

The aim of this paper is to implement the automatic code generation of a Java web application allowing to make CRUD on an entity model (UML class diagram) using Struts2 as a presentation framework.

The automatic code generation by ACCELEO is based necessary on the implementation of templates. In this paper, we implement the templates correspond to the Action classes and Jsp pages. The different templates needed to realize this project are listed in the following UML class diagram shown in figure 7.
According to the figure 7, the principle templates are as follow: “Template Action” and “Template View”. In this section, we begin by implementing the “Template Action” then in second hand we implement the “Template View”.

1) Template of Actions

The different templates which constitute the templates of Actions, according to the figure 7, are as follow:

a) Template of Create Action class

This template permits to generate a Create class Action. This latter allows insert or create a new employee, a new city or a new department. The figure 8 shown below presents this template.

```java
public class CreateAction extends ActionSupport {

    public String execute() {
        String name = request.getParameter("name");
        Employee newEmp = new Employee(name);
        EMPLOYEE.addClass(newEmp);
        return SUCCESS;
    }
}
```

Figure 8. Template of Create Action Class.

b) Template of Retrieve Action Class

This template allows generate a Retrieve class Action. This latter allows display all employees, departments and cities existing in the BD. The figure 9 shown below presents this template.

```java
public class RetrieveAction extends ActionSupport {

    public String execute() {
        List<Employee> employees = EMPLOYEE.getClasses();
        request.setAttribute("employees", employees);
        return SUCCESS;
    }
}
```

Figure 9. Template of Retrieve Action Class.

c) Template of Update Action Class.

This template allows generate an Update class Action. This latter allows modify an existing employee, an existing department or an existing city. The figure 10 shown below presents this template.

```java
public class UpdateAction extends ActionSupport {

    public String execute() {
        int id = Integer.parseInt(request.getParameter("id"));
        Employee emp = EMPLOYEE.getClass(id);
        request.setAttribute("emp", emp);
        return SUCCESS;
    }
}
```

Figure 10. Template of Update Action Class.
d) Template of Delete Action Class.

This template permits to generate a Delete class Action. This latter allows delete a selected employee from the existing list of employees, departments or cities in database. The figure 11 shown below presents this template.

```
Figure 11. Template of Delete Action Class.
```

2) Template of JSP pages

According to the figure 8, the different templates which constitute the super template “Template View” are as follow: “Template of Create Jsp page”, “Template of Retrieve Jsp page”, “Template of Update Jsp page” and “Template of Delete Jsp page”.

a) Template of Create Jsp page

This template permits to generate a “Create JSP page”. This page allows insert or create a new employee, a new city or a new department in the database. It represents the layer presentation of the application. The figure 12 shown below presents this template.
b) **Template of Retrieve JSP page**

This template permits to generate a Retrieve JSP page. This page allows display all employees, all departments or cities existing in database. The figure 13 shown below presents this template.

```java
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="jdbc" uri="http://java.sun.com/jsp/jstl/jdbc" %>
<%= pageContext.setAttribute("dept", new EmployeeDAO().getDepartmentNames()) %>
<%= pageContext.setAttribute("city", new EmployeeDAO().getCityNames()) %>
<%= pageContext.setAttribute("employee", new EmployeeDAO().getEmployeeNames()) %>

<% if (dept != null) { %>
<%@ taglib prefix="jdbc" uri="http://java.sun.com/jsp/jstl/jdbc" %>
<%= pageContext.setAttribute("dept", new EmployeeDAO().getDepartmentNames()) %>
<%= pageContext.setAttribute("city", new EmployeeDAO().getCityNames()) %>
<%= pageContext.setAttribute("employee", new EmployeeDAO().getEmployeeNames()) %>
<% } %>
</body>
</html>
```

Figure 13. Template of Retrieve JSP page.

c) **Template of Update JSP page**

This template permits to generate an Update JSP page. This page allows modify or update an employee, information of a department or city existing in database. The figure 14 shown below presents this template.

```java
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="jdbc" uri="http://java.sun.com/jsp/jstl/jdbc" %>
<%= pageContext.setAttribute("dept", new EmployeeDAO().getDepartmentNames()) %>
<%= pageContext.setAttribute("city", new EmployeeDAO().getCityNames()) %>
<%= pageContext.setAttribute("employee", new EmployeeDAO().getEmployeeNames()) %>

<% if (dept != null) { %>
<%@ taglib prefix="jdbc" uri="http://java.sun.com/jsp/jstl/jdbc" %>
<%= pageContext.setAttribute("dept", new EmployeeDAO().getDepartmentNames()) %>
<%= pageContext.setAttribute("city", new EmployeeDAO().getCityNames()) %>
<%= pageContext.setAttribute("employee", new EmployeeDAO().getEmployeeNames()) %>
<% } %>
</body>
</html>
```

Figure 14. Template of Update JSP page.

d) **Template of Delete JSP page**

This template permits to generate Delete JSP page. This page allows delete or destruct an employee, a department or city existing in database. The figure 15 shown below presents this template.

```java
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="jdbc" uri="http://java.sun.com/jsp/jstl/jdbc" %>
<%= pageContext.setAttribute("dept", new EmployeeDAO().getDepartmentNames()) %>
<%= pageContext.setAttribute("city", new EmployeeDAO().getCityNames()) %>
<%= pageContext.setAttribute("employee", new EmployeeDAO().getEmployeeNames()) %>

<% if (dept != null) { %>
<%@ taglib prefix="jdbc" uri="http://java.sun.com/jsp/jstl/jdbc" %>
<%= pageContext.setAttribute("dept", new EmployeeDAO().getDepartmentNames()) %>
<%= pageContext.setAttribute("city", new EmployeeDAO().getCityNames()) %>
<%= pageContext.setAttribute("employee", new EmployeeDAO().getEmployeeNames()) %>
<% } %>
</body>
</html>
```

Figure 15. Template of Delete JSP page.
VII. MODEL-TO-TEXT TRANSFORMATION RESULT

According to the generated PSM model (Figure 5), the different classes and JSP pages generated in this paper are listed in the figure 16 for the Action classes and in the figure 21 for the JSP pages.

In the next section, we present the different Action classes that are the CRUD Action classes.

A. Action classes Result

The Action class is one of Struts 2 novelties, the developer is no longer constrained to extend a given class and frees up J2EE objects to facilitate unit tests. It leaves the possibility of creating a mother class for its controllers, in which you can write all their common code (factorization) and which will implement the "ActionSupport" interface. The different Action classes generated in this paper are shown in the following figure.

For letting this paper quite understandable and clear, we present only one example of each Action class. We note that in the application we arrived to generate all classes and all JSP pages.

1) Create Action Class

This class is the result of M2T transformation by using the ACCELEO generation language. The Create Action Class "CreateActionClass" allows insert or create a new employee, a new city or a new department. In this case, we present the "CreateEmployeeAction.java" class. Figure 17 shows this class.
2) Retrieve Action class

This class is the result of a M2T transformation by using the ACCELEO generation language. The Retrieve Action Class “RetrieveActionClass.java” allows retrieve or lists all employees, all cities or all departments. In this case, we present the “RetrieveDepartmentAction.java” class. Figure 18 presents this class.

3) Update class

This class is the result of a M2T transformation by using the ACCELEO generation language. The Update Action Class allows modify or update an employee, a city or a department. In this case, we present the “UpdateDepartmentAction” that permits to update the information of a given Department. The figure 19 shown below presents this class.

4) Delete class Action

This class is the result of a M2T transformation by using the ACCELEO generation language. The Delete Action Class “DeleteActionClass” allows delete an employee, a city or a department from database. In this case, we present the “DeleteDepartmentAction.java”. Figure 20 shows this class.
5) Create JSP page

This JSP page is the M2T transformation result by ACCELEO generation language. The Create JSP page “CreateClass” allows insert or create a new employee, a new city or a new department in database. In this case, we present the “CreateCity.jsp” page. Figure 22 presents this page.

![CreateCity.jsp](image)

Figure 22. Create JSP page: “CreateCity.jsp”.

6) Retrieve JSP Page

This JSP page is the M2T transformation result by using ACCELEO generation language. The Retrieve JSP page “RetrieveClass.jsp” allows retrieve or lists all employees, all cities or all departments. In this case, we present the “RetrieveEmployee.jsp” class. Figure 23 shows this class.

![RetrieveEmployee.jsp](image)

Figure 23. Retrieve JSP page: “RetrieveEmployee.jsp”.

B. Generated JSP pages

The different JSP pages generated by ACCELEO generator is shown in figure 21.

For letting this paper quite understandable and clear, we present only one example of each JSP page. We note that we have been able to generate all the requested JSP pages according to the generated Struts2 model presented in figure 5.
7) Update JSP page

This JSP page is the M2T transformation result by using ACCELEO generation language. The Update JSP page allows modify or update an employee, a city or a department. In this case, we present the “UpdateCity.jsp” that permits to update the information of a given Department in database. Figure 24 shows this JSP page.

![Figure 24. Update JSP page: “UpdateCity.jsp”](image)

8) Delete JSP page

This page is the result of M2T transformation by using ACCELEO generation language. The Delete JSP page “DeleteClass” allows delete an employee, a city or a department from database. In this case, we present the “DeleteDepartment.jsp”. Figure 25 shows this JSP page.

![Figure 25. Delete JSP page: “DeleteDepartment.jsp”](image)

### VIII. EVALUATION

In this paper, we were conducted a process of code generation by using Acceleo generator. After generating this code, we want to know the percentage of the generated code with respect to the total application code. In this case study, we obtained a very respectful percentage of code generation occurred. The generated code has reached into 100% of total code.

### IX. RELATED WORK

After examining the related work concerning the automatic code generation by applying MDA approach, we can cite several works that are conducted in this domain such as: [26]-[11]-[27]-[28]-[29]-[30]-[31]-[32]-[33]-[34]-[36].

The work presented in [26] permits to generate only JSP pages and JavaBeans by applying the UWE [11] and ATL as a transformation language [24].

The work cited in [27] permits to transform the PIMs models implemented by Enterprise Distributed Object into PSMs for different services platforms. The transformation rules of this work are defined by ATL language. This work generates only the PSM model.

In [28], Billing et al., explains the different transformation rules permits to transform PIM into PSM in the EJB context. The different transformation rules of this work are defined by the approach by modeling based on QVT.

The work presented in [29] considers that MDA is a software industrialization pattern (or a software factory). The idea of this work is illustrated by a real case study in an IT services company. The main objective is to create MDA tools founded on XMI, XSLT and Visitor pattern. It is a proposal to create MDA tools taking as base XMI, XSLT and the Visitor pattern.

In [30], the objective of this work is the model-driven development approach for E-Learning platform. Thereby, the authors implement the CIM model by analyzing business logic. And thereafter, they establish the system diagram and the robustness analysis. Thus, the authors define a transformation method from PIM to PSM layer by layer.

In the work [31] the objective is to integrate a new framework for secure Data Warehouses design by applying the approach by modeling based on QVT.

In [32], the author presents the AndroMDA approach of the community of web-based MDA [32]. This work permits to transform a PIM schemes to model by integrating a wide variety of scenarios and comes with a set of plug-ins, called cartridge.

In [33], the authors arrived to generate a MVC 2 web model from Struts framework. The meta-model of Struts is realized in first time in this work. The different transformation rules of this work are defined by ATL language in view to generate the CRUD operations from three classes C1, C2 and C3.
The subject of the work [34] is to generate the MVC 2 web model from the combination of UML class diagram and the UML activity diagram. The work presented in [35] is the continuation of [34], the objective of this work is to generate the code from the MVC2 web model already generated in [34] by using the JET2 generator.

In [38], the authors arrived to generate the MVC2 web model, but in this case, from the combination of UML class diagram and UML sequence diagram. This work is considering as an end-to-end code generation by using JET2 generator and based in the MDA approach.

The objective of the work presented in [36] is to generate the N-tiers PSM model by integrating Struts2, Spring IoC and Hibernate DAO frameworks. The different transformation rules of this work are defined by ATL language.

The objective of the paper [37] is to generate the Struts2 PSM model and to validate the ATL transformation rules presented in this work which was not possible in [33]-[34]-[35]-[36]-[38]. This paper describes a new validation method of ATL transformation rules and the test of this method by a case study.

Thus, we can say that the main contributions of our approach compared to others are: the use of ACCELEO generator to generate the code of Struts2 framework which was not possible in the case of [34]-[35]-[38]. This work is realized for the first time in this paper.

X. CONCLUSION AND FUTURE WORK

Model-driven architecture (MDA) is a software development approach proposed and supported by the OMG foundation. This is a particular variant of model-driven engineering (MDE). A typical example of MDA approach is the automatic generation of source code from UML modeling, which involves combining: The UML standard, the modeling tool that implements it and the UML generation templates. The passage from the PSM to the code generation is the logical continuation of this treatment. It can be realized by generators such as these in order to produce any type of technological targets.

In this paper, we generate a MVC2 web code from Struts2 model already generated by applying an ATL transformation. This is the M2M transformation. After that, we implement the model-to-text transformation based on the MDA approach in view to obtain the code of this application from the generated model. Thereby, this transformation is started by the M2M transformation to obtain the PSM Struts2 model. The generated PSM model is an EMF model. This latter is used, in the M2T transformation, as an input model of Accelio generator to produce automatically the necessary target application code. Finally, the code generation result was demonstrated and exemplified by a case study.

Furthermore, we plan to generate an e-commerce web code from a PSM model result of the integration of struts2, Spring IoC and Hibernate DAO. In other hand, we can extend this method for considering other frameworks like: PHP, Zend and DotNet.

REFERENCES


© 2017 by the author(s); licensee Empirical Research Press Ltd. United Kingdom. This is an open access article distributed under the terms and conditions of the Creative Commons by Attribution (CC-BY) license. (http://creativecommons.org/licenses/by/4.0/).

DOI:10.24032/ijeacs/0207/04 www.ijeacs.com

M’hamed RAHMOUNI received the Diploma of Higher Approfondie Studies in Computer Science and Telecommunication from the faculty of science, Ibn Tofail University, Morocco, 2007, and Doctorat of High Graduate Studies degrees in Computer Sciences from Ibn Tofail University, Morocco, January 10, 2015 . He participated in various international congresses in MDA (Model Driven Architecture) integrating new technologies XML, EJB, MVC, Web Services, etc. and he published many papers in the MDA domain.

Samir MBARKI received the B.S. degree in Applied Mathematics from Mohammed V University, Morocco, 1992, and Doctorat of High Graduate Studies degrees in Computer Sciences from Mohammed V University, Morocco, 1997. In 1995 he joined the faculty of science Ibn Tofail University, Morocco where he is currently a Professor in Department of Computer Science. His research interests include software engineering, model driven architecture, software metrics and software tests. He obtained an HDR in Computer Science from Ibn Tofail University in 2010.