

## Developing Web Applications Based on Model Driven Architecture

Yen-Chieh Huang\*

*Department of Information Management, Meiho University  
No. 23, Pingguang Rd., Neipu, Pingtung, 912, Taiwan, R.O.C.  
p7894121@mail.ncku.edu.tw*

Chih-Ping Chu

*Department of Computer Science and Information Engineering  
National Cheng-Kung University  
No. 1, University Road, Tainan, 701, Taiwan, R.O.C.  
chucp@csie.ncku.edu.tw*

Received 12 October 2011

Revised 16 October 2013

Accepted 31 October 2013

Model Driven Architecture (MDA) is a new software development framework. This paper presents a model-driven approach to the development of Web applications by combining Conallen's web applications design concept and Kleppe's MDA process. We use the UML extension mechanism, i.e. stereotypes, to define the various web elements, and use the Robustness diagram to represent MVC 2 structure for Web application. After required analysis, we start by using a use case diagram as CIM, and then transform CIM to PIM, and PIM to PSM. We propose mapping rules for model-to-model transformation. Finally, we develop a tool named WebPSM2Code, which can automatically transform PSM diagram to Web application code, such as Java, JSP, HTML, Servlet, Javascript, as well as deployment descriptor file. All the files can automatically address to the correct directory structure for JSP Web application, and the transformation rate is about 39% of the whole system. Using this methodology, systems can be analyzed, designed, and generated more easily and systematically. Thereby, the time that Web programmers spend on coding can be reduced.

*Keywords:* Model driven architecture; platform independent model; platform specific model.

### 1. Introduction

The practice of software development spans more than 50 years, and it is largely intangible [1]. Software development technology has gradually changed from structure analysis and design into object-oriented analysis and design, but the software

\*Corresponding author.

industry is still labor intensive. According to the latest Standish Group CHAOS report (2009), software project success rates are descending, with only 32% of all projects succeeding, 44% coming in late or over budget, or lacking required features and functions, and 24% failing and being cancelled prior to completion or delivered and never used [2]. Web engineering is a relatively new direction of Software Engineering with focus on the development of Web-based systems [3]. Several approaches for the development of Web applications have been proposed in the last years. Model driven Web engineering (MDWE) is the application of model driven engineering to the domain of Web application development where it might be particularly helpful because of the continuous evolution of Web technologies and platforms.

Web applications are complex systems based on a variety of hardware and software components, protocols, languages, interfaces, and standards [4], which have evolved into sophisticated computing tools that not only provide stand-alone functions to the end user, but are also integrated with corporate databases and business applications [5]. In the MDA framework [3], the database, application and web user interface are logically separated, so its architecture is appropriate for Web applications.

In recent years, many client-server applications (i.e. browser–web server applications) have been developed. Web applications are widely designed using the Model-View-Controller (MVC) process, which is also the most time-consuming and error-prone process. In MVC 2 architecture, a controller handles system navigation, a model stores a set of data, and a view (or multiple views) presents the data stored in the model. Web applications are implemented using an MVC 2 pattern using a Servlet as the controller, a Java Bean (or EJB) as the model, and JSP pages as visualization elements.

The MDA is a framework for software development that is defined by the Object Management Group (OMG) [6], which defines the model-based development process, as well as the model that is automatically mapped to the implementation approach. The MDA framework is increasingly used by software developers to develop various kinds of domain applications. Paloma Cáceres *et al.* (2003) used MDA to develop an information system [7]. Beigbeder *et al.* (2004), Fujikawa *et al.* (2004), and Tai *et al.* (2004) used MDA to develop Web applications [8–10]. Fong (2006) used MDA to develop a bank system.

In this study, the MDA development process and MVC/Model 2 pattern were used to design JSP Web applications. UML and CASE tools were used to create model diagrams for the Web applications, which included creating a PIM diagram, transforming the PIM into a PSM, and transforming the PSM into Code. Therefore, this study focused on to the step-by-step transformation of MDA models using an efficient system development approach and generating code templates, to increase system development productivity.

The rest of this paper is organized as follows: Section 2 introduces the MDA development process. Section 3 discusses the methodology for model transformation. PIM transformation is presented in Sec. 3.1, the mapping rules for the PIM to PSM

transformation are presented in Sec. 3.2, and the transformation of the PSM into code is discussed in Sec. 3.3. Section 4 presents a case study that is used for the validation of the methodology and calculates the transformation rate of the code templates. Finally, Sec. 5 draws some conclusions and outlines areas for future research.

## 2. The Model Driven Development Process

### 2.1. Model Driven Architecture (MDA)

From the perspective of MDA, software development is driven by the activity of the modeling software system. Kleppe *et al.* proposed the MDA framework [3], which includes three kinds of models as shown in Fig. 1: the platform independent model (PIM), platform specific model (PSM), and code model, and defines the *Communication Bridge*, which connects the models to each other [10]. They proposed that a PIM should be created, then transformed into one or more PSMs, and then transformed into code.

**Platform Independent Model.** A PIM is a model with a high level of abstraction that is independent of any implementation technology. A PIM is a software system that supports a business. In a PIM, the system is modeled from the viewpoint of how it best supports the business [10].

**Platform Specific Model.** A PSM is tailored to specify a system in terms of the implementation constructs that are available in one kind of implementation technology. The PIM is transformed into one or more PSMs.

**Code.** The final step in the development is the transformation of each PSM into a code model. The transformation from PSM into code models is straightforward.

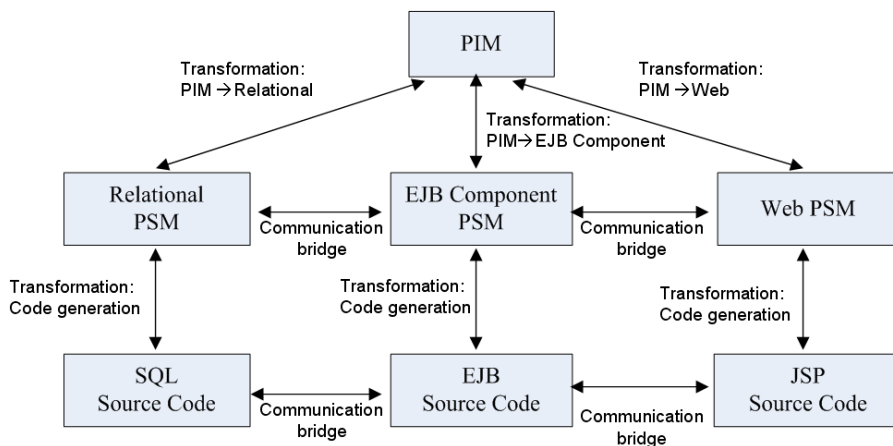


Fig. 1. The three levels of MDA (Kleppe, 2003 Rosa’s MDA).

A PIM should be created for a business. Then it can be transformed into one or more PSMs such as Web PSM, EJB PSM, or Relational PSM (shown in Fig. 1). The Web PSM represents the User Interface (UI), the EJB PSM represents the Application, and the Relational PSM represents the Database. Each PSM can then be transformed into code. The concept of model transformation is central to the realization of the benefits of MDA. The most complex step in the MDA development process is the one in which a PIM is transformed into one or more PSMs. Its advantages include productivity, portability, interoperability, maintenance, and documentation [3].

The MDA offers a way to use models instead of traditional source code. Nowadays, designers often use models for planning and explaining projects. The most commonly used models are UML models.

## **2.2. Model-driven Web applications**

Many tools can generate code from models, but the generated code is a template that can't be executed immediately, and many additions still need to be made by hand. UML commercial CASE tools, such as Rational Rose, Magic Draw, Enterprise Architect, and Power Designer etc., can support UML transformation from class diagrams into data definition language (DDL) and development languages (e.g. Java), but the transformation from class diagrams into the source code of Web applications has been discussed little.

In 2003 OMG proposed a methodology for transforming PIM into PSM [6], and distinguished the following types of methods: marking, metamodel transformation, model transformation, pattern application, model merging, and additional information. Many studies and tools focus on MDA transformation automation but only discuss Relational PSM and Java PSM. Harrison *et al.* (2000) proposed UML diagrams mapping to Java code [11]. Kleppe *et al.* (2003), Meservy (2005), and Zhao (2007) had proposed MDA transformation theory [3, 12, 13], but they didn't practice a real case.

In the last few years Web engineers have proposed several languages, architectures, methods, and processes for developing Web applications. In particular, methods for modeling systems were developed, including OOHDM [14], OO-H [15], UWE [16, 17], WebML [18], and WebTE [19]. They focus on the specification of analysis and design models for Web systems, such as the construction of navigation and adaptation models.

WebML [4] used metamodel, UWE [16, 17, 20] used requirements and analysis, OOHDM [14] used navigability and metamodel, WebSA [19] and MDWEnet [21] used metamodel etc. Some of these models developed CASE tools, such as WebML, which developed WebRatio, UWE, which developed ArgoUWE, HDM-lite, which developed the AutoWeb System, WebSA, which developed WebTE (WebSA Transformation Engine). Some of the similarities and differences among these approaches are as follow:

First, they identified the necessity of a brand new model or diagram. Second, they clearly separated content, navigation and presentation space by means of different

models or diagrams. Third, they aimed at defining a precise and systematic for the development of the Web application, in which some steps are automatic. Fourth, they presented an exhaustive authoring process (a method and a notation). Fifth, they avowed the necessity of any kind of constraint language to augment the precision of the Web application models. Sixth, they avowed the necessity of a CASE tool that supports the whole process. Seventh, they did not follow MDA develop processes, basically, from PIM, then PSM, then to code. Finally, most of these studies didn't describe about how to generate the source code of the Web applications automatically or write the code from scratch.

Web technologies have evolved in recent years. For example, the MVC Web framework has changed into the MVC/Model 2-based Web framework, and it tends to apply the Rich Internet Applications (RIAs) to the Web content, such as using Ajax technology for connecting with the database. This author by Huang had ever proposed the "transformation from class diagram to relational table and application template" in 2004 [22], which figured out the transformation from PSM to database of DDL SQL statement and Java application code, but not included the UI solution, and this study extended previous research to open a new approach for transforming MDA frameworks to Web applications. In this paper, we focused on transforming between MDA's models and Web PSM to code for Web application, including the MVC/Model 2-based Web framework and Ajax applications. An automatic code generation program was developed based on mapping rules, but the developers still needed to manually enhance after the transformed PSM models and codes.

### 2.3. MDA tools for Web applications

Conallen [23] proposed a Web application extension for UML, which extends UML with additional semantics and constraints to model Web specific architectural elements. He used stereotypes, constrains, and tags to define the relationship between a form and a Web page. In his research, his UML extension diagrams can map to the Java, JSP, and HTML code by the defined stereotype, but not automatically, and that is not related to the components of the MVC pattern.

WebML [4] method developed the tool named WebRatio, which claimed that it can be transformed to Java, JSP, web.xml template file. It is also one of the methods UWE, which can be model to MVC 2 structure, but it does not follow the MDA process. In the book [4], there are some code templates, but there is no automatically code generating instructions.

Tai *et al.* [9] proposed the WAD (Web Application Descriptor) method and WAST (Web application development support tool) tool which is a Plug-in in the WebSphere from IBM (2004). This tool can generate three kinds of skeleton code for Struts applications. However, their study did not follow the MDA process step by step because they did not construct the PSM, and instead focused on the named WAD model for generating code, according to the paper description, the tool can be transformed to the JSP files and mapped to the configuration file.

MagicUWE [24] is a plug-in in the MagicDraw UML tool. It can enhance the tool to draw UWE diagrams, including Content diagram, Presentation diagram, Navigation diagram, Process structure diagram and Process flow diagram. However, it is only a part of drawing tool, and it has no real help for code generation.

Another UML extension for describing Web applications (UML-based Web Engineering, UWE) was proposed by Koch *et al.* (2000–2008). In their paper they used the UML extension mechanisms to define stereotypes for representing Web constructs, such as nodes and links. They developed a UML CASE tool called ArgoUWE [17], defined use case and activity diagrams as the CIM, designed the class diagram as PIM, archived the automatic model and generated code, basing on transformation rules that were defined in the QVT (Query/View/Transformation) language and metamodel. The code generation was defined using a QVT language called the Atlas Transformation Language (ATL), and then generated J2EE and JSP code template. In their paper [20], they implemented a simple project management system, and generated some Java and JSP code. They developed a CASE tool named UWE4JSF [25] in 2009, which is a plug-in in the Eclipse IDE. This tool claimed it can automatically generate code, such as Java, JSP, and supported JSF web application, but it did not show the example case for code generation.

The another MDA approach, named OOHDMDA [26, 27], was proposed by Schmid and Donnerhak in 2005, which can generate Servlet-based Web applications. This method was derived from Object-Oriented Hypermedia Design Method (OOHDM), which described Web applications by an object model on three levels: the conceptual level, the navigational level, and the interface level. This method can generate some java and Servlet code, but not automatically.

RUX-Method [28, 29] integrated the concept of WebML and UWE, which focused on User Interface and RIAs, announced that it can be transformed to Java, JSP, etc. by ATL language. However, in the article [30], there is only ATL syntax, no actual code in JAVA or JSP.

In this study, we followed the Kleppe's MDA development process, from the CIM, PIM, PSM, to generate code, combined the developed concept of web application proposed by Conallen, added and redefined some stereotypes for classes, and then make them transform to web application code templates automatically, finally modified some of code by hand to complete the whole system.

### 3. Methodology for Model Transformation

Model-to-model transformation is a key aspect of model-driven development (MDD). Model-to-model transformation is the process of converting one model to another model of the same system. A transformation definition consists of a collection of transformation rules, which are unambiguous specifications of the way that one model can be used to create another model. MDA suggests building computational independent model (CIM), platform independent model (PIM), and platform specific

model (PSM). This study starts with the business model (CIM) level, which is defined as requirements model, by using the use case diagram as CIM. A PIM represents the design viewpoint, which is derived from the requirements model. Therefore, a use case can be mapped into a PIM based on additional information. A PSM represents the implementation viewpoint. According to the PIM diagram, a Web PSM diagram can be designed by the transformation rules, and finally it can be automatically transformed into the JSP Web application source codes by a self-developing program. These steps are described as follows.

### 3.1. PIM transformation

A PIM can be represented by different kinds of UML diagrams, but the class diagram is the best choice. Class diagrams are a logical way of looking at classes and their relationships in a system. This study adopted the Robustness elements in the analysis model for developing the Web application. The class object was divided into three stereotyped classes: <<Boundary>>, <<Control>>, and <<Entity>>, which were suggested by Jacobson [30].

An actor specifies a role that an external entity adopts when interacting with the system directly. It touches the boundary of a system. An actor interacts with more than one use case, and each use case must have a user interface for operating, just like the boundary class in the PIM. A use case diagram describes the relationships between use cases and actors, and among use cases themselves.

A use case is a description of the behavior of a part of the system in terms of its interaction with human and computerized actors. Each use case must have a boundary class for actor interaction. A use case may act as a control class, and the entity class is selected according to the functions of the control class. Then the entity class is implemented for processes, logical code objects, or application objects. Therefore, this study proposes that a use case can be mapped to a set of Robustness elements, as shown in Fig. 2.

In Web applications, the boundary classes are either Web pages or monitor screens. The control classes map to the processes that deliver the functionality of the system. The entity classes usually represent the persistent objects in the system, such as the tables in the database.

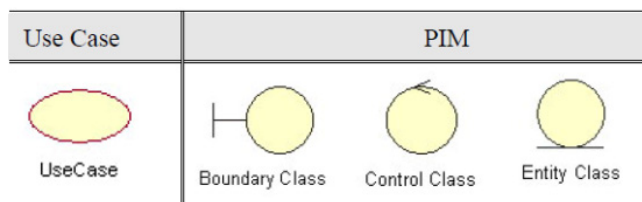


Fig. 2. A use case mapped into a set of Robustness elements.

### 3.2. Mapping rules from PIM into PSM

According to the MDA development life cycle, a PIM diagram can be transformed into one or more PSMs. For each specific technology platform a separate PSM is generated, which includes relational PSM, application PSM (Java PSM), and Web PSM. Many studies and CASE tools have supported the transformation of relational PSM and Java PSM, but this study focuses on the transformation from PIM into Web PSM, because the Web applications include many languages and different kinds of technology, such as user interfaces, applications, and database. In this study the Java PSM and database connection were synchronously generated while the Web PSM was transformed.

This section gives insight into the PIM to PSM transformation based on the mapping rules, which are divided into three classes: boundary, control, and entity.

#### 3.2.1. Boundary class transformation

A boundary class in PIM can be transformed into a combination of Server Page, Client Page, and Form in PSM. This Server Page is different than that defined by Conallen. The functions of a boundary class in PIM can be transformed into a Server Page in PSM, which is a JSP language used by servers to send messages, parameters, and operations. The displayed boundary class data in PIM can be transformed into a Client Page in PSM, and the Client Page is only an HTML-based method for displaying data. The PIM boundary class input fields can be transformed into a Form in PSM. The Form aggregates all input field elements for a Web page, such as input boxes, text areas, radio buttons, check boxes, hidden fields, etc.

There are three kinds of mapping rules for transforming a boundary class in a PIM into a Web PSM. Examples are shown in Fig. 3. Some icons are automatically transferred using the CASE tool when the stereotypes are defined. The first rule refers to the combination of a Server Page, a Client Page, and a Form. When a Web page has to operate server functions or display the parameters of a server, a Server

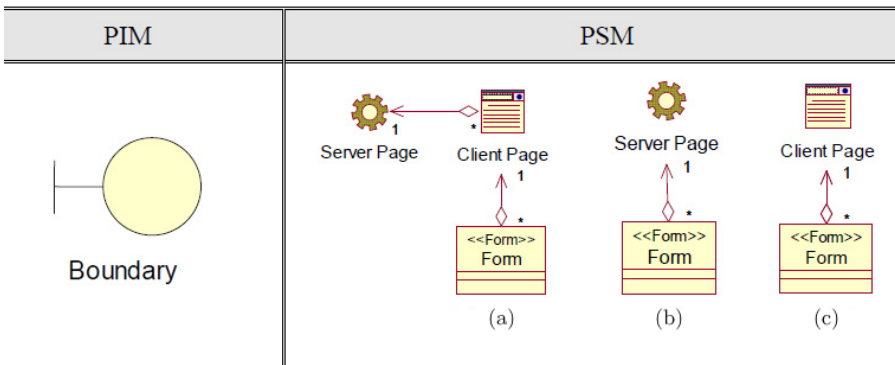


Fig. 3. A PIM boundary class transforms into three types of PSMs.



Page must implement it. In this situation the Client Page and the Form must be aggregated with the Server Page and become a JSP file, as shown in Fig. 3(a).

The second rule refers to the combination of a Server Page and a Form. If a boundary class does not need to operate server functions or display the parameters of server, and does not include the action of the client site, a form can be aggregated with a server page, as shown in Fig. 3(b).

The last rule refers to the combination of a Client Page and a Form. If a boundary class only includes the action of the client site, a Client Page can implement it, and the Form has to be aggregated with the Client Page, as shown in Fig. 3(c).

### 3.2.2. Control class transformation

In a Web application environment, a control class in the PIM can be transformed into a Servlet in PSM, because the control class's job is to send the parameters of the getting or posting method of. The control class has almost no attributes in PIM analysis, but when it is implemented the Servlet may have temporary attributes in the PSM design. (ex: initial parameters) The mapping rule from a control class in PIM to a class in PSM is shown in Fig. 4.

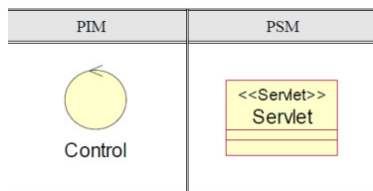


Fig. 4. A control class in the PIM is mapping to a Servlet in PSM.

### 3.2.3. Entity class transformation

An entity class in PIM can be transformed into a normal class in PSM, which represents logical and object that implement Servlet for a specific platform.

The attributes and operations in entity classes can be obtained from requirement documents, drawings, data glossaries, use case descriptions, etc. Nouns may become attributes and verbs may become operations. Figure 5 shows the mapping rules from an entity in PIM to a class in PSM.

## 3.3. Transform PSM into code templates

As mentioned previously, Web applications include many technologies and languages. In this study, we developed a program named WebPSM2Code, and a Web PSM uses WebPSM2Code to generate many Web components, such as server pages, client pages, forms, Servlets, JavaBeans, and configuration files. Each component must reflect a specific stereotype to create the specific language. This section will

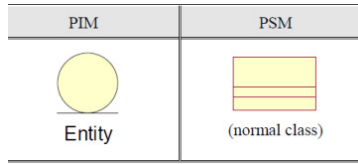


Fig. 5. An entity class in the PIM is mapping to a normal class in PSM.

introduce the transformations of components into various codes in the Web PSM as follow:

### 3.3.1. Server Page class

A Server Page class is a logical abstraction of a Web page as seen by server, which is typically implemented by a kind of scripted page, such as JSP, ASP, or PHP. In this study, the Server Page class was mapped into JSP code, which included static attributes, non-static attributes, and session attributes. The static attributes were realized by **JSP declaration**, the non-static attributes were realized by **JSP Scriptlet**, and the session attributes were obtained from the parameters of the **session**. Operations defined in the page, which are typically implemented as functions, were realized through **JSP declaration**.

Figure 6 is an example of Server Page mapping to JSP code. The *count* attribute is a static attribute realized through **JSP declaration**, *rank* is a non-static attribute realized through **JSP Scriptlet**, and *account* is a parameter of a session attribute from the *session.getAttribute* declaration.


Server Page (PSM)	JSP (Code)
 Server Page count : Integer rank : Integer <<Session>> account : String	<pre> 1 &lt;%-- JSP Declaration --%&gt; 2 &lt;%! 3 Integer count; 4 %&gt; 5 &lt;%-- JSP Scriptlet --%&gt; 6 &lt;% 7 Integer rank; 8 String account = (String)session.getAttribute(""); 9 %&gt;                     </pre>

Fig. 6. The Server Page class is mapped into the JSP code template.

### 3.3.2. Client Page class

A Client Page class can be mapped to *JSP*, *html*, or *JavaScript* code. The attributes and operations in a Client Page class are realized using *JavaScript*. The attributes were mapped to *var* declarations, and the operations were mapped to functions in *JavaScript*. When the Client Page class aggregated with a server page, it was mapped to become *JSP* code, if it aggregated with a form it was mapped to become the *html*, and if it existed alone was mapped into *JavaScript*. Figure 7 is an example

Client Page (PSM)	JSP - html - JS (Code)
 <pre> classDiagram     class RegisterClient {         +Boolean isValid         +createXMLHttpRequest()         +asynEvent()         +handleStateChange()         +register()     }         </pre>	<pre> 1 &lt;script language=JavaScript type="text/javascript"&gt; 2 var isValid = new Boolean(); 3 4 function register(){ 5 } 6 7 var xmlHttp; 8 function createXMLHttpRequest() { 9     if (window.ActiveXObject){ 10         xmlHttp = new ActiveXObject("Microsoft.XMLHTTP"); 11     } 12     else if (window.XMLHttpRequest) { 13         xmlHttp = new XMLHttpRequest(); 14     } 15 } 16 function asynEvent(){ 17     /****Insert your code****/ 18     createXMLHttpRequest(); 19     xmlHttp.onreadystatechange = handleStateChange; 20     xmlHttp.open("GET", url, true); 21     xmlHttp.send(null); 22 } 23 24 function handleStateChange() { 25     if(xmlHttp.readyState == 4) { 26         if(xmlHttp.status == 200) { 27             /****Insert your asynchronous function****/ 28         } 29     } 30 } 31 32 &lt;/script&gt;         </pre>

Fig. 7. The Client Page is mapped to a JSP, html, or JavaScript code template.

of an *asynchronous* client page mapped into JavaScript code. The attribute of *isValid* is declared by a *var* variable, and the operations of *createXMLHttpRequest*, *asynEvent*, *handleStateChange*, and *register* are mapped to the reflective functions.

### 3.3.3. Form

A form stereotyped class is a collection of input fields that are aggregated to a client page. This class maps directly to the HTML `<form></form>` tag. Its attributes include the form’s input fields: text, password, textarea, checkbox, file, image, button, option, radio, reset, hidden, submit, etc. The tagged value named method is used to set the method attribute: *GET* or *POST* (default). All the input fields (attributes) must have a stereotype to reflect, and only attributes that map to predefined form attribute types are forward engineered. Some examples are as follows:

- `<<text>>id:String`           →   `<input type="text" name="id" value="" >`
- `<<text>>password:String`   →   `<input type="password" name="password" value="" >`
- `<<submit>>button=Login`   →   `<input type="submit" name="button" value="Login" >`

Form (PSM)	JSP or HTML Code
<pre> &lt;&lt;Form&gt;&gt; RegisterForm &lt;&lt;text&gt;&gt; id : String &lt;&lt;text&gt;&gt; password : String &lt;&lt;checkbox&gt;&gt; validBox : Boolean &lt;&lt;submit&gt;&gt; button = Register                     </pre>	<pre> 1 &lt;form name = "RegisterForm" method="Post" action="RegisterServlet.do" &gt; 2   &lt;input type="text" name="id" value="" &gt; 3   &lt;input type="text" name="password" value=""&gt; 4   &lt;input type="checkbox" name="validBox" value=""&gt; 5   &lt;input type="submit" name="button" value="Register"&gt; 6 &lt;/form&gt;                     </pre>

Fig. 8. A Form in PSM is mapped to an html code template.

A <<submit>> stereotyped association in-form linked file is allowed to send all attributes to a server page. This association is used to determine the *action* attribute, which is set to the linked file or Servlet as a server page. Figure 8 shows an example of the Form in Web PSM mapping to a code. Each property is mapped to an <input> tag and the action is a request to a *RegisterServlet.do* Servlet.

### 3.3.4. Servlet

The user interfaces will send data or call entity classes through control classes. A control class in the PIM will be transformed into a Servlet in the PSM. The attributes and methods in the Servlet are implemented using JAVA, and must consider the association relationships between classes. Generally speaking, a Servlet must accept a Form request, and then redirect to another Web page. Figure 9 is an example of an asynchronous Servlet mapping to Java code. Its transformation steps

Servlet (PSM)	Java Code
<pre> &lt;&lt;Servlet&gt;&gt; RegisterServlet doPost() register() doAsynWork()                     </pre>	<pre> 1 public class RegisterServlet extends HttpServlet { 2   RegisterCheck bean0 = new RegisterCheck(); 3   public boolean register(String argname, int pw){ 4     boolean tempValue = true; 5     /*****Insert your code*****/ 6     return tempValue; 7   } 8 9 public void doPost(HttpServletRequest request, HttpServletResponse 10 response) throws IOException, ServletException{ 11   /*****Insert your code*****/ 12   //for Request with Page 13   RequestDispatcher view = request.getRequestDispatcher("/**** 14 Redirect Page ***/"); 15   view.forward(request, response); 16 } 17 18 //for Ajax check 19 public void doAsynWork(HttpServletRequest request, HttpServletResponse 20 response) throws IOException, ServletException { 21   /*****Insert your code*****/ 22   response.setContentType("text/xml"); 23 } 24 }                     </pre>

Fig. 9. An example of code for an asynchronous Servlet.

are as follow:

1. <<Form>> request  
According to the Form request the association names (Get or Post), then declare the method of *doGet* or *doPost*.
2. <<Client Page>> asynchronous  
In Servlet, the asynchronous pattern is implemented with Ajax pattern, and then declares the method named *doAsynWork*.
3. <<Redirect>> or <<Forward>> – Generated the code as follows:

```
RequestDispatcher view = request.getRequestDispatcher("/*.*.*Redirect Page
***/"); view.forward(request, response);
```

### 3.3.5. Deployment descriptor (DD file)

The *web.xml* file, commonly known as the deployment descriptor, contains configuration and deployment information about Web applications, which contains the information about the definitions and mappings to both JSP pages and Servlets. Another type of information that can be contained within the deployment descriptor is Servlet context initialization parameters.

In this study, when the PSM was transformed to code, all Servlets were transformed to Java codes and automatically generated servlet-mapping path records in the *web.xml* file. Notice that there is a `<servlet-name>` parameter tag here that corresponds to a Servlet defined above in the *web.xml* file. The other tag contained in this section is the `<url-pattern>` tag, which signifies the URL pattern with which the Servlet is mapped.

Figure 10 shows an example of partial code in *web.xml* that the *RegisterServlet* in `<servlet-name>` is mapping to the *RegisterServlet.do* in `<url-pattern>`. It is clear that the real Servlet file is *Servlet.RegisterServlet* class.

```

1  <servlet>
2    <servlet-name>RegisterServlet</servlet-name>
3    <servlet-class>Servlet.RegisterServlet</servlet-class>
4  </servlet>
5  <servlet-mapping>
6    <servlet-name>RegisterServlet</servlet-name>
7    <url-pattern>/RegisterServlet.do</url-pattern>
8  </servlet-mapping>
```

Fig. 10. An example Servlet mapping in the *web.xml* configuration file.

### 3.3.6. Database connecting

The *Communication Bridge* in MDA is the function for connecting with the database. In the design phase we added a *DBManager* class in the PSM diagram for connecting with database. Figure 11 shows an example connection with a MySQL database that will be automatically transformed into Java code.

Database connection	Java Code
<pre> classDiagram     class DBManager {         dbDriver : String = "com.mysql.jdbc.Driver";         url : String = "jdbc:mysql://localhost:3306/dbname";         username : String = "root";         password : String = "your_password";         con : Connection = DriverManager.getConnection(url,use...         stmt : Statement = con.createStatement();         sqlStr : String         rs : ResultSet          setSqlStr()         executeQuery()         executeUpdate()     </pre>	<pre> 1 public class DBManager { 2     private String dbDriver = "com.mysql.jdbc.Driver"; 3     private String url = "jdbc:mysql://localhost:3306/dbname"; 4     private String username = "root"; 5     private String password = "your_password"; 6     private Connection con = DriverManager.getConnection(url,username,password); 7     private Statement stmt = con.createStatement(); 8     private String sqlStr = ; 9     private ResultSet rs = ; 10 11     public void setSqlStr(String str){ 12         /****Insert your code****/ 13         this.sqlStr = str; 14     } 15     &lt;omit&gt;---- 16 } </pre>

Fig. 11. The example code of MySQL database connection.

## 4. A Case Study

This study implemented a Project Management System to check the validity of the methodology proposed in Sec. 3. The used platforms were JVM 1.6.0, MySQL 5.0 for the database and Apache Tomcat 6.0 for the Web container. The UML CASE tool was the IBM's Rational Rose. First, Rational Rose was used to design the Use Case diagram. Second, the PIM diagram was constructed for each use case based on the mapping rules. Third, the PSM diagram was designed according to the mapping rules and the stereotypes added to the PSM classes. Finally, through the WebPSM2Code tool, the PSM diagram could be automatically transformed into the Web application code templates. Each file counted the lines of code as a measurement of a programmer's productivity, and then counted again when the system was completed.

### 4.1. PIM diagram

Our approach includes three functions: user login and registration, project management (adding a new project, modifying a project, and deleting a project), and project query. Figure 12 shows the six use cases and an example mapping PIM diagram for login and registration. The preliminary analysis phase uses three kinds of objects in the Robustness diagram as depicted in the PIM diagram. These include boundary classes, control classes, and entity classes. Boundary classes represent Web pages contents, i.e. information in the system, such as the account and password fields in the *LoginClient* that allow users to login. Control classes deal with the parameter requests for boundary classes, such as *LoginClient* logins and requests for the *LoginServlet*. They then call the *Register* Entity class to handle the request.

### 4.2. Web PSM diagram

Based on the defined stereotypes in the previous section, the PSM is created by the mapping rules. The PSM diagram for *Login/Register* is shown in Fig. 13. The PSM

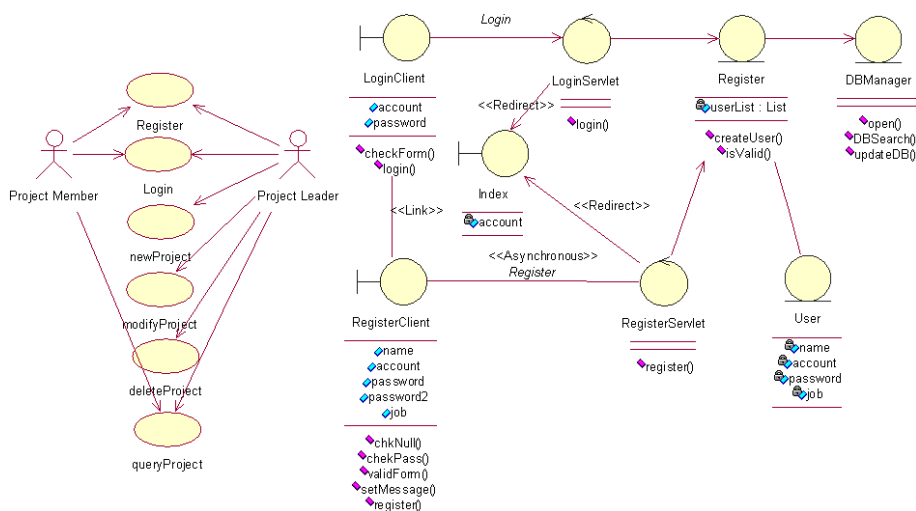


Fig. 12. The Use Case diagram and a PIM diagram mapping example for Login/Register use cases.

diagram reflects the specific platform and has more detailed information in it, including stereotypes, data types, initial values, and associations between classes. The results of the PIM to PSM mapping are shown in Table 1. Here we only demo the transformation of *Login/Register* as follows:

Use Case 1: Register

This case included the boundary classes *RegisterClient*, *RegisterForm*, and *RegisterBackForm*, the control class *RegisterServlet*, and the entity class *Register* as the back end. There were asynchronous relations between the *RegisterClient* class and *RegisterServlet* class, which are shown in Fig. 13. Therefore, the Ajax pattern was used for the code transformation. When users successfully registered, the *RegisterServlet* class redirected them to the Index page.

Use Case 2: Login

This use case included the boundary classes *LoginClient*, *LoginForm*, and *LoginToRegister*, and the control class *LoginServlet*. When users inputted their account and password, the *LoginForm* class sent a request to the *LoginServlet* class using the *Post* method, and then the *LoginServlet* class decided if the user was redirected to the Index page or the *LoginClient* page.

4.3. Transforming Web PSM to code

The summarized Web PSM diagram was transformed into Web application codes. We developed the transformation program named WebPSM2Code using Microsoft Studio 2008 c#. It requires a runtime environment upper version than NET framework 3.5.

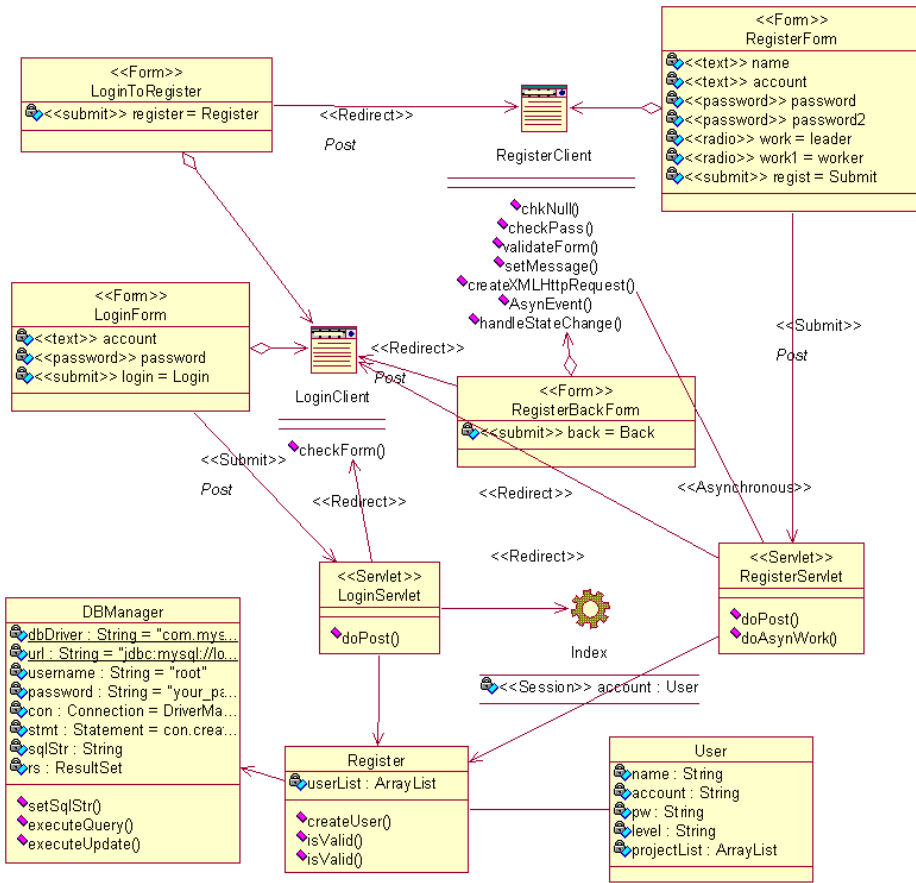


Fig. 13. The PSM Class Diagram of a Login/Register.

The codes files followed the JSP Web application framework; the Model, Controller, View, and deployment descriptor (DD) file were assigned to the Context Root folder. Table 2 shows the example of filename and directory structure for the *Login/Registration* use case. The Controller file was stored in the *WEB-INF/classes/*

Table 1. The results of PIM mapping to PSM for Login/Register use case.

Use case	PIM			PSM		
	View	Controller	Model	View	Controller	Model
Register	RegisterClient	RegisterServlet	Register	RegisterClient	RegisterServlet	Register
			User	RegisterForm		User
			DBManager	RegisterBackForm		DBManager
Login	LoginClient	LoginServlet	Register	LoginClient	LoginServlet	Register
			User	LoginForm		User
			DBManager	LoginToRegister		DBManager



Table 2. The listing of filenames after transforming from PSM into code.

Use case	PSM View	Code Context Root	PSM controller	Code WEB-INF\classes\Servlet	PSM Model	Code WEB-INF\classes\Model
Register	RegisterClient	RegisterClient.html	RegisterServlet	RegisterServlet.java	Register	Register.java
	RegisterForm				User	User.java
	RegisterBackForm				DBManager	DBManager.java
Login	LoginClient	LoginClient.html	LoginServlet	LoginServlet.java		
	LoginForm					
	LoginToRegister					

Servlet directory, all Java technology class files were stored in the *WEB-INF/classes/Model* directory, the deployment descriptor was stored in a file called *web.xml* in the *WEB-INF* directory, and all the view pages were stored in Website Context Root folder.

After the transformation, we counted the LOC for each file using counting standard, then manually completed this system using Coding Standard, and then counted again. The results of the case study are shown in Table 3.

Table 3. The files list for the Web application directory structure and comparison of code coverage results.

Type	Filename	Code template size	Completed program size	Code coverage (%)
Model	DBManager.java	14	54	26
	MGT.java	7	92	8
	Project.java	11	13	85
	Register.java	11	52	21
	User.java	8	10	80
	Subtotal	57	227	25
Controller (Servlet)	DomainModifyServlet.java	8	20	40
	GetDomainServlet.java	8	14	57
	GetProjectServlet.java	10	15	67
	LoginServlet.java	9	20	45
	ProjectDeleteServlet.java	9	20	45
	ProjectMGTServlet.java	11	31	35
	ProjectQueryServlet.java	10	14	71
	RegisterServlet.java	11	33	33
	Subtotal	76	167	46
DD file	web.xml	52	52	100
View	domainmgtpage.jsp	11	31	35
	domainpage.jsp	15	52	29
	index.jsp	10	16	63
	loginclient.html	10	22	45
	projctcmgtpage.jsp	15	47	32
	projectcpage.jsp	22	54	41
	projectprivatepage.jsp	17	45	38
	projectquerypage.jsp	9	33	27
	registerclient.html	31	83	37
	Subtotal	140	383	37
	Total	319	823	39

The results show that the average transformation rate was about thirty-nine percent. When the programs had to deal with more business logic operations, the coverage rates were lower because the functions had to be manually added by programmers. When the programs were responsible for controlling processes or sending parameters, the coverage rates were higher.

## 5. Conclusions and Future Works

This study aims to realize transformation between model diagrams, and then to make them transform into the MVC Model 2 Web application code automatically, and it includes the development of program, such as Java, Servlet, JSP, HTML, web.xml etc. We used the example case to validate the proposed transformation methodology. This case could be transformed to a total of 23 program files (see Table 3). The final completed system showed that the transformation rate was about 39%. Since no one has mentioned the results of the transformation code number and the transformation rate in the previous literature, it was clear that our code generation had much richer information than others, and the results of this study show a significant contribution to the Web application.

This study had presented a new MDA approach to transform Web applications from CIM to PIM, then to PSM, and finally to code. The Robustness diagram bridges the gap between analysis and design. Applying MDA process, this study aims to build sets of CIM, PIM, and PSM by analyses, design, and implementation phases of the model-driven process. By using the UML Web Application Extension (UML WAE) to define the stereotypes, the PSM can be automatically translated into code by the developed program. The code templates included parts of Java Web application files, such as html, Java Bean, Servlet, JSP, deployment descriptor file (*web.xml*) etc.

Comparing with other Model-Driven methodologies, such as WebML, WAD, UWE4JSF, OOHDMDA, or RUX-Method, this study not only realized automatic code generation, but also supported the integrated JSP Web application. As a result, all the files can address to the corrected Web application directory structure. Table 4 summarizes these tools of comparison to support what kinds of developed program.

This study found that this method can automatically transform 39% of the whole system from Web PSM to code templates, which shows the information is still contained in different models. On one hand, because the PIM and PSM models are represented by class diagrams, they only can express static content and association relationships. On the other hand, the other information is hidden in another model diagrams, such as dynamic information and behavior information. For this reason, we can use sequence diagrams or state diagrams to describe dynamic calls and transfers between states in the future. Therefore, our future work will focus on generating Web application codes from interaction diagrams and behavior diagrams.

The main contribution of this study is reducing costs associated with application life-cycles and application development time, improving application quality,

Table 4. Comparison of prior researches on code generation.

	WebML/ WebRatio	WAD/ WAST	UWE4JSF	OOHMDA	RUX-Method	WebPSM2 Code
Proposed by	Kraus Knapp Koch 2007	IBM 2004	Kroiss Koch Knapp 2009	Schmid Donnerhak 2005	Linaje Preciado 2007 2008	This Study
Java	✓		✓	✓	✓	✓
Servlet				✓		✓
HTML						✓
JSP	✓	✓	✓		✓	✓
DD file	✓	✓			✓	✓
DBConnection						✓
Javascript (Ajax)						✓
Case Example	✓					✓
MDA process				✓		✓

✓: Support.

increasing returns on technology investments, and quickening the inclusion of emerging technology benefits into existing systems.

The advantages of using models transform to codes template can reduce the coding time, but it does have some limitations. This study only focused on dynamic interactive webpage, such as a form submission and the associated response from the server. The Ajax not only used to check a registered username, but also can be applied to other web applications. This study should provide a transformable basis for additional research. There is a continuing need for an adequate theoretical basis for the practical application of statically structure models and dynamic behavior models.

## References

1. T. C. Lethbridge and P. Laganieri, *Object-Oriented Software Engineering: Practical Software Development Using UML and JAVA*, 2nd ed. (McGraw-Hill, 2005).
2. J. Johnson, *CHAOS Summary 2009*, The Standish Group, 2009.
3. A. Kleppe, J. Warmer and W. Bast, *MDA Explained: The Model Driven Architecture™: Practice and Promise* (Addison-Wesley, 2003).
4. S. Ceri *et al.*, *Designing Data-Intensive Web Applications* (Morgan Kaufmann, 2003).
5. R. S. Pressman, *Software Engineering: A Practitioner’s Approach*, 7th ed. (McGraw-Hill, 2010, OMG, MDA Guide Version 1.0.1., 2003).
6. P. Cáceres, E. Marcos and B. Vela, An MDA-based approach for web information system development, in *Proceedings of Workshop in Software Model Engineering*, 2003.
7. S. M. Beigbeder and C. C. Castro, An MDA approach for the development of Web applications, in *Proc. of ICWE*, 2004.
8. Y. Fujikawa and T. Matsutsuka, New Web application development tool and its MDA-based support methodology, *FUJITSU Sci. Tech.* **40**(1) (2004) 94–101.
9. H. Tai *et al.*, Model-driven development of large-scale Web applications, *IBM J. Res. Dev.* **48** (2004) 797–809.

10. S. J. Mellor, K. Scott, A. Uhl and D. Weise, *MDA Distilled: Principles of Model-Driven Architecture* (Addison-Wesley, Boston, 2004).
11. W. Harrison, C. Barton and M. Raghavachari, Mapping UML designs to Java, in *15th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications, ACM SIGPLAN Notices*, Minneapolis, 2000, pp. 178–187.
12. T. O. Meservy and K. D. Fenstermacher, Transforming software development: An MDA road map, *Computer* (2005) 52–58.
13. C. Zhao and K. Zhang, Transformational approaches to model driven architecture — A review, in *Software Engineering Workshop, 2007*, pp. 67–74.
14. D. Schwabe and G. Rossi, Developing hypermedia applications using OOHDMD, in *Workshop on Hypermedia Development Process, Methods and Models*, Pittsburg, USA, 1998.
15. J. Gómez and C. Cachero, OO-H Method: Extending UML to model web interfaces, in *Information Modeling for Internet Applications* (IGI Publishing, Hershey, 2003).
16. N. Koch, Classification of model transformation techniques used in UML-based Web engineering, *IET Software* 1(3) (2007) 98–111.
17. N. Koch *et al.*, UML-based Web engineering — An approach based on standards, in *Web Engineering: Modelling and Implementing Web Applications* (Springer Verlag, 2008).
18. A. Schauerhuber *et al.*, Bridging WebML to model-driven engineering: from DTDs to MOF, *IET Software* 1(3) (2007) 81–97.
19. S. Meliá, J. Gómez and J. L. Serrano, WebTE: MDA transformation engine for web applications, in *Proc. 7th International Conference on Web Engineering* (Springer-Verlag, 2007).
20. A. Kraus, A. Knapp and N. Koch, Model-driven generation of Web applications in UWE, in *Proc. 3rd International Workshop on Model-Driven Web Engineering*, 2007.
21. A. Vallecillo *et al.*, MDWEnet: A practical approach to achieving interoperability of model-driven Web engineering methods, in *Third Int. Workshop Model-Driven Web Eng*, 2007.
22. Y. C. Huang, Transformations from Class Diagram to Relational Table and Application Template, Masters Thesis, National Sun Yat-Sen University, 2004.
23. J. Conallen, *Building Web Applications with UML*, 2nd ed. (Addison-Wesley, 2002).
24. M. Busch and N. Koch, MagicUWE — A CASE tool plugin for modeling Web applications, in *Proceedings of 9th International Conference on Web Engineering*, 2009.
25. C. Kroiss, N. Koch and A. Knapp, UWE4JSF: A model-driven generation approach for Web applications, in *Proceedings of 9th International Conference on Web Engineering*, Springer, 2009, pp. 493–496.
26. H. A. Schmid and O. Donnerhak, The PIm to servlet-based PSM transformation with OOHDMDA, in *5th International Conference on Web Engineering*, Sydney, 2005.
27. H. A. Schmid and O. Donnerhak, *OOHDMDA — An MDA Approach for OOHDMD* (Springer, New York, 2005).
28. M. Linaje, J. C. Preciado and F. Sánchez-Figueroa, Engineering rich internet application user interfaces over legacy Web models, *IEEE Computer Society* 11(6) (2007) 53–59.
29. J. C. Preciado *et al.*, Designing rich internet applications combining UWE and RUX-method, in *Proceedings of the Eighth International Conference on Web Engineering*, IEEE Computer Society, 2008.
30. I. Jacobson *et al.*, *Object-Oriented Software Engineering: A Use Case Driven Approach* (Addison-Wesley, Boston, 1992).

Copyright of International Journal of Software Engineering & Knowledge Engineering is the property of World Scientific Publishing Company and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.