Transformation from Web PSM to Code

Yen-Chieh Huang[1,2], Chih-Ping Chu[1], Zhu-An Lin[1], Michael Matuschek[3]
[1]*Department of Computer Science and Information Engineering,*
National Cheng-Kung University, Tainan, Taiwan
[2]*Department of Information Management, Meiho Institute of Technology, Pingtung, Taiwan*
[3]*Department of Computer Science, University of Duesseldorf, Germany*
E-mail :p7894121@mail.ncku.edu.tw

Abstract

This research proposes how class diagrams that use the Unified Modeling Language ***(UML)*** *can be converted to a user interface of a Web page using the Model Driven Architecture* ***(MDA).*** *From the Platform Independent Model* ***(PIM)*** *we go to the Web Platform Specific Model* ***(PSM)****, and then to the direct generation of code templates for Web page applications. In this research the class diagrams are drawn with the Rational Rose, then, using our self-developed program, these diagrams can be transformed into code templates with Servlets, JSP, and JAVA. We implement a case study for verification, and then calculate the transformation rate with lines of code* ***(LOC)*** *coverage rate by measuring the LOC after transforming and after the system is finished. The results show the transformation rate is about thirty-six to fifty percent, which represents that this research can help the programmers to greatly reduce the developing period.*

Keywords: Model Driven Architecture, Platform Independent Model, Platform Specific Model

1、Introduction

Software is largely intangible [1]. Software development gradually transforms from structure analysis and design to object-oriented analysis and design, but the software industry is labor intensive, even after finishing system analysis, the programmers still start from scratch and write the code. Especially in the application software development for Web pages, in the last few years, there are many researches have been proposed to reduce code and development time. This research focuses on how class diagrams can be transformed into Web pages, the results could reduce the development time for Web pages programmers. The common Web pages developing tools include JSP, PHP, and ASP etc.. The platform used in this research is JAVA, the Web pages developing tool is JSP, relevant technology are JSP, Servlets and Ajax. This research uses IBM Rational Rose as the CASE tool for class diagram object modeling, and the user interface code templates are then created via the conversion program written by ourselves.

2、Literature Review

The object-oriented paradigm has gained popularity in various guises not only in programming languages, but also in user interfaces, operating systems, databases, and other areas [2]. Classification, object identity, inheritance, encapsulation, and polymorphism and overload are the most prominent concepts of object-oriented systems [3]. The UML is a modeling language that helps describing and designing software systems, particularly software systems built using the object-oriented approach. This research uses Robustness diagrams [4] for describing the application environment of Web pages.

The MDA is a framework for software development defined by the Object Management Group (OMG). It is the importance of models in the software development process [5, 6]. The MDA development life cycle included four kinds of models. **Computation Independent Models** (CIM) describe the requirements for the system and represent the highest-level business model. It is sometimes called "domain model" or "business model". A **PIM** describes a system without any knowledge of the final implementation platform, and this PIM is transformed into one or more PSMs. A **PSM** is tailored to specify a system in terms of the implementation constructs that are available in one specific implementation technology. The final step in the development is the transformation of each PSM to **code**. The **CIM, PIM, PSM**, and **code** are shown as artifacts of different steps in the software development life cycle, which is shown in Figure 1.

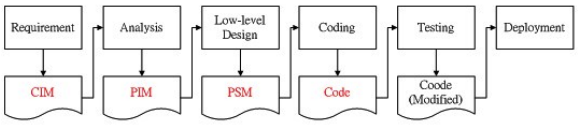

Figure 1. MDA software development life cycle and output artifacts

The most widely used architecture in the environment of Web applications is Browser/Server (B/S) approach, an example for a specific Client/Server (C/S) structure [7]. The basic architecture of Web systems includes a client browser, a Web server, and a connecting network. The

principal protocol for communication is the **Hypertext Transfer Protocol (HTTP)**. The principal language for expressing the context between the client and the server is **Hypertext Markup Language (HTML)** [8].

Relevant technologies for today's Web applications include CGI, Applets, ActiveX controls, plug-ins and Ajax etc. To explain the general structure of such a Client/Server system, a Web page can be modeled into a class, and a client page can be modeled into another class, which must be drawn by the method of extending UML [9].

3、Transformation from Class Diagrams to Web Applications

In the concept of MDA we must first create the PSM design for a specific Web application. A Web page can be expressed by class diagrams where every stereotype (including stereotype classes and associations) is defined in order to describe the situation of every Web page, then the Web class diagrams can be drawn and, in the final step, it can be transformed into a code template.

3.1 Web Pages Components Mapping Methods

3.1.1 Stereotypes

In order to extend its function of use in UML, we can use stereotypes to strengthen and define the class model. Stereotypes allow us to get a more proper description to the class objects, they can be used for describing and limiting the characteristics of the module components, and they exist in standard UML components [10]. In this paper, we use Rational Rose to define control classes and strengthen the classes that describe the Web pages. This research proposes stereotype class mapping methods as described in Table 1.

3.1.2 Association Stereotypes

In order to implement Web modules, it is vital to control user-site and server-site requests and responses via HTML in the network. Using association stereotypes between classes is an optional way to model HTTP parameters, and it is useful when parameters are relatively complex or have special semantics and extra documentation is necessary. Therefore, this research proposes the mapping methods of association stereotypes between classes as shown in Table 2.

3.2 PSM to Code Template Transformation

Every stereotype class has different transformation model, in here; we describe a Servlet transformation rule as an example. The attributes and methods in Servlet are implemented by traditional JAVA, but the difference lies in the association between classes. Generally speaking, a Servlet must

Table 1. Stereotypes Mapping in Class

Stereotypes	Description
<<Servlet>>	Responsible for showing the request of client site, and communicating with back end module The methods of this class contain at least Get() or Post().
<<Server Page>>	A server page represents the server site information, the attributes and methods in this class are implemented by Scripting Element.
<<Client Page>>	A client page represents the <HTML> element, which has two principal child elements: <HEAD> and <BODY>. The <HEAD> represents structural information about the Webpage; the <BODY> element represents the majority of the displayed content [8].
<<Form>>	The HTML <<Form>> stereotype class represents some attributes, such as input boxes, text areas, radio buttons, check boxes, and hidden fields, these classes map directly to a <Form> element [8].
<<Model>>	A <<model>> stereotype class represents the logical operation of business processes, which is implemented by JAVA. Its meaning is the same as traditional class diagrams, therefore a class diagram notation can ignore the <<Model>> stereotype in this research.

Table 2. Association Stereotypes

Association	Description
<<Build>>	This is an action of a Servlet or a Server Page creates a Client Page or a Form.
<<Link>> [8]	A relationship between a client page and a server-side resource or Web page.
<<Include>>[8]	A directional association from a Web page to another Web page.
<<Redirect>>	The client page should be automatically replaced with another client page, where Post and Get are two methods to achieve this, among others.
<<Object>> [8]	This represents many types of embedded objects, such as Applet, ActiveX controls. The parameters for the object are defined in the parameterized class.
<<Asynchronous>>	The client page sends an asynchronous request to Servlet.
<<Submit>>	A relationship between a <<Form>> and a server page. Post or Get are used for submitting, among other methods.

accept a Form request, and then a redirection to another Webpage occurs. Its transformation steps are as follow:

1. <<Form>> request- According to Form request the association names (Get or Post), then declare the method of doGet or doPost.
2. <<Client Page>> asynchronous- In Servlet, implement the asynchronous pattern and then declare the method doAsynWork.
3. << Redirect >>- Generate the code as follow:

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

4、Measurement

For the experimental evaluation we adopt "code coverage" to calculate the result. Code coverage is a measure used in software testing. It describes the degree to which the source code of a program has been tested. In this research, code coverage represents the ratio of information in class diagrams to the information in the full implemented system. Talking about information, we define the way of measurement and standard of quantification analysis as follows:

4.1 The Way of Measurement

In a software development project, software measurement can be achieved in a lot of ways, such as lines of code (LOC), function point (FP), object point, COCOMO, and Function requirement etc. We choose LOC, and the reasons are:

1. The value is easily measured.
2. There is a direct relationship to the measurement of person-months (effort).
3. Effort is also a size-oriented software metric [11].

For a class diagram, it expresses static information as well as the relation between classes, and the resulting LOC can be easily counted automatically after transformation.

4.2 Counting Standard

LOC counters can be designed to count physical lines, logical lines, or source lines by using a coding standard and a physical LOC counter. For different kinds of Coding Style, the LOC turns out differently, so we need to define the Coding Standard and Counting Standard which we use for our measurement.

In this research, line counters are defined as follows:

1. XML has defined and self-defined tags in Web pages, a set of tag counts as one line.
2. If the web pages are not XML, (e.g. Scripts, Scriptlets, and Expressions), every line of code counts as one line.

5、 Case Study

5.1 Experiment Steps

The CASE Tool selected for this experiment is the Rational Rose from IBM which transforms class diagrams into code templates. First, Rational Rose is used to draw the class diagrams, then the labels of the stereotypes are added in the class diagrams, and lastly we utilize the program developed by ourselves to transform the class diagrams into code templates.

5.2 Case Description

To verify the theoretical structure proposed by this research we use the practical example of a Login/Register System. It has three main functions in the Use Case Diagram. There are "Account registration", "User login", and "Display Home page".

Figure 3 is a class diagram of PIM of a user Login/Register System which reflects the Use Case diagrams. In the preliminary design, which uses Robustness diagrams for description, we include the entity classes, boundary classes and control classes. Boundary classes represent the shown Web page content, i.e. the information in the system, such as the account and password fields that LoginClient offers for the user login. Control classes deal with the parameter request by the boundary classes, such as login request to LoginServlet of LoginClient, and they are determined to call out Register of the Entity class to deal with the request.

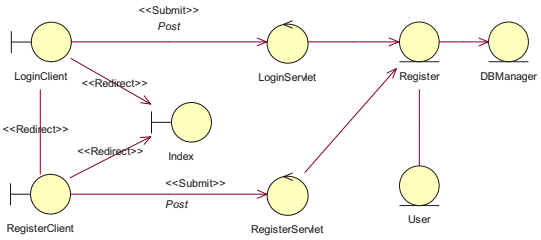

Figure 3. The PIM of a Login/Register System

Use Case 1: Account Registration

This use case includes the boundary classes RegisterClient, RegisterForm, and RegisterBackForm, the control class RegisterServlet, and the entity class Register as back end. Between the classes RegisterClient and RegisterServlet, there are asynchronous relations, so the Ajax pattern will be used for realizing the code transformation.When the user succeeds to register, the class RegisterServlet will redirect him to the Index home page.

Use Case 2: User Login

This use case includes the boundary classes LoginClient, LoginForm, and LoginToRegister and the control class LoginServlet. When the user inputs his account and password, the class LoginForm will send a request to the class LoginServlet using the Post method, and then the class LoginServlet makes the decision if the user is redirected to the Index or

the class LoginClient.

Use Case 3: Display the home page

The home page includes Index and AVLTreeApplet, and is displayed by a Java Applet. It is described how the Applet object is loaded and integrated into the Index home page via object parameter classes.

5.3 Measurement Result

We measured the LOC of the code template for each use case after transformation and the LOC of the finished system by the previously defined counting standard. The data is shown in Table 3.

Table 3. Measurement Result

	LOC of Code template after transform	LOC of finished system	Transform ratio
Use Case 1: Account Registration			
registerclient.html	42	95	44%
RegisterServlet.java	14	38	37%
Use Case 2: User Login			
loginclient.html	11	22	50%
LoginServlet.java	11	21	50%
Use Case 3: Display home page			
index.jsp	5	14	36%

The results show that the transformation rate is about thirty-six to fifty percent. When we focus on the part not responsible for the program logic in this class, this is a relatively high proportion. The transformation into the code template according to the defined Web page class diagrams represents the static structure model of the system, consisting of attributes, operations, and associations between classes. However, the system operation logic cannot be expressed in detail. This part is still up to the programmers.

6、 Conclusion

Nowadays, Web code must be programmed from scratch even if the PSM analysis is finished, but in this research we proposed a method of code template transformation. By adding stereotypes to class diagrams, they can describe Web pages, synchronous or asynchronous relations, and we can transform them into code templates with distinct logical, control, and view code blocks using JSP&Servlets or the MVC model.

Asynchronous relations can be realized using many methods. This research adopts Foundations of Ajax to express that the client site responds to the server site. Furthermore, reverse engineering is a factor to be considered, so that maybe change of the code can be reflected in the Web class diagrams afterwards.

For the case study example in this research, a class diagram transformed into code templates can only be achieved about thirty-six to fifty percent of the whole system, which expresses that it does not discover all the sufficient information we want.

Because of the definition of class diagrams and the representing models, they can only express static class content and relationships. There are also other aspects that cannot be described in design and transformation for more complicated program logic.

For this reason, we can make use of sequence diagrams, and state diagrams, in order to describe the dynamic call and transfer between the states. So, the further research will study how to create Web code templates from interaction diagrams and behavior diagrams.

Reference

[1] Lethbridge, T.C. and Laganiere, R., *Object-Oriented Software Engineering: Practical Software Development using UML and JAVA Second Edition*, Mcgraw-Hill, 2005.

[2] Nierstrasz, O., *A Survey of Object-Oriented Concepts, In Object-Oriented Concepts, Databases and Applications*, W. Kim and F. Lochovsky, , ACM Press and Addison-Wesley, 1989, pp. 3-21.

[3] Gottlob, G., Schrefl, M. and Rock, B., Extending Object-Oriented Systems with Roles, *ACM Transactions on Information Systems*, Vol. 14, No. 3, Jul. 1996, pp. 268-296.

[4] Ambler, S.W., *The Object Primer: Agile Model-Driven Development with UML 2.0*, Cambridge Univ Pro, 2004.

[5] Kleppe, A., Warmer, J. and Bast, W., *MDA Explained: The Model Driven Architecture™: Practice and Promise*, Addison Wesley, Apr. 2003.

[6] Koch, N., Classification of model transformation techniques used in UML-based Web engineering, *IET Software*, Vol. 1, Issue 3, Jun. 2007, pp. 98-111.

[7] Li, J., Chen, J. and Chen, P., Modeling Web Application Architecture with UML, *IEEE CHF*, 30 Oct. 2000, pp. 265-274.

[8] Conallen, J., *Building Web Applications with UML Second Edition*, Addison Wesley, 2002.

[9] Conallen, J., Modeling Web Application Architectures with UML, *Communications of the ACM*, Vol. 42, No. 10, Oct. 1999.

[10] Djemaa, R.B., Amous, I., and Hamadou, A.B., WA-UML: Towards a UML extension for modeling Adaptive Web Applications, *Eighth IEEE International Symposium on Web Site Evolution*, 2006, pp. 111-117.

[11] Humphrey, W.S., *PSP A Self-Improvement Process for Software Engineers*, Addison Wesley, Mar. 2005.