An MDA Based Modeling and Implementation for Web App

Rongliang Luo
Department of Computer Science and Engineering, Zhejiang University City College, Hangzhou, P.R. China
Email: luorl@zucc.edu.cn

Xiao Peng
College of Computer Science and Technology, Zhejiang University, Hangzhou, P.R. China
Email: xiaopeng8754@zju.edu.cn

Qianxi Lv
College of International Studies, Zhejiang University, Hangzhou, P.R. China
Email: vera_lv52e@126.com

Minghui Wu*, Bin Peng, Shuoping Wang and Ming Guo
Department of Computer Science and Engineering, Zhejiang University City College, Hangzhou, P.R. China
Corresponding author email: mhwu@zucc.edu.cn

***Abstract*—Web App surges recently as the HTML technology comes to be well-developed. The development framework of existing platform demands the users' attention directed towards technique details and duplicated efforts. In order to save the developers' efforts to the design of App functions and its ease of use, the idea of Model Driven Architecture (MDA) will be extended in the development of Web App. This paper proposes a Web App development framework based on MDA—Model Driven Web App Development Framework (MDWAF). Based on the MDWAF, the development of Web App will focus on the software models. The comprehensive data management of development and release process will be achieved by Cloud Services, and the cross-platform local resources access of Web App will be provided by mobile middleware.**

***Index Terms*—Model Driven Development, Web App, Platform Independent Model, Platform Dependent Model, Model Transformation**

I. INTRODUCTION

With Mobile internet being in full swings, App development has become popular in various platforms as iOS, Android, Windows Phone, Symbian. HTML5 technology has been optimized and adopted by major investors, and quickly comes to be compatible with each browser. As a result, the development of Web App is in good momentum. It is possible that the function of Web App will approach or even surpass Native App with the standardization of HTML 5 technology and improvement of hardware quality of the platforms [1]. The coming years will certainly see the quick development of Web App with the surge of mobile equipments, cloud computing and the popularization of 4G networks. The existing development platform of Web App, such as jQTouch, Sencha Touch and jQuery Mobile are plug-ins applied for mobile web, compatible with systems like iOS, Android, and some other systems based on webkit. Based on development framework of jquerycore, jquery UI and Ext, they are expanded to provide developers with the related component library and functions required for commonly used mobile Web App. The development framework of existing platform demands the users' attention towards technique details and duplicated efforts. In order to facilitate the design and development of App, the idea of Model Driven Architecture (MDA) will be extended in the development of Web App. An abstract model framework without the involvement of hardware platform, operating system or implementation language of Web App will shift the code-centric software development paradigm to the new model-centric software development paradigm.

Model Driven Architecture (MDA) [2] is a framework specification proposed by the OMG (Object Management Group) in 2001. The main idea of MDA is to first establish PIM (Platform Independent Model) according to business logic independently of implementation technology, which will be transformed into PSM (Platform Specific Model) by mapping and finally generate executable code for target platform [3]. MDA is based on various standards proposed by OMG, separating business logic and technology depending on certain platform. Platform independent application established by MDA and its related standards can be implemented in CORBA, J2EE, .Net, Web services and other Web-based platforms [4]. The transformation from PIM to PSM through mapping and the transformation from PSM to code is the key technology in MDA [5].

A better solution of problems arising during Web App development will be achieved by adopting the MDA

doi:10.4304/jsw.8.8.1881-1888

development framework. More specifically, developers should first accomplish the business requirements by building PIM, and then transform PIM into PSM and code with model transformation engine of the development framework and specific platform configuration files [6]. Cloud service can provide component model and related templates during this process and publish the encapsulated executable files afterwards. When the Apps are released, users can download them in Web App Store which installed on the mobile terminals. After being installed, the Web App can be interpretively executed by Web App engines.

The rest of this paper is organized as follows: Section 2 introduces the architecture of MDWAF. Then the development process of Web App based on the architecture will be introduced in the order of architectural establishment, architecture implementation and transformation and generation of Web App in section 3. Section 4 describes the implementation of model driven technology within the architecture. Finally, We conclude in Section 5.

II. MDWAF ARCHITECTURE

MDWAF can be divided into three parts. The First part refers to the local development environment in MDWAF, which provides such supports as project management, Model Driven Development, program preview and program debugging, etc. The second part is provided by the Cloud service, which includes developer account management, version control, service management for developers and user terminals, and services for Web App Store. The third part is the Web App engine installed on terminal devices, which serves as a local runtime environment for users and provides accessibility to services through Web App Store. The MDWAF architecture is shown in Fig. 1.

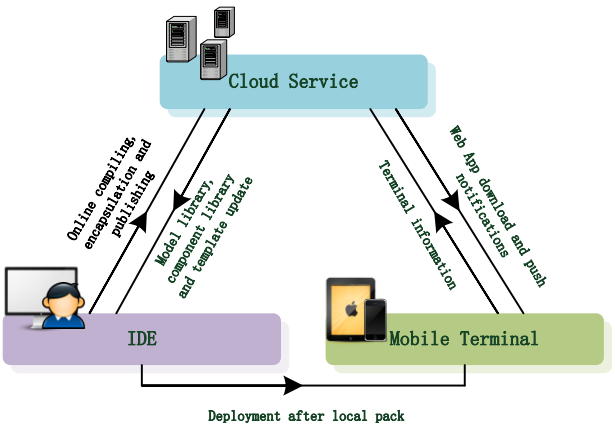


Figure 1. MDWAF architecture

Development platform provides the Web App Developer with an integrated development environment which makes the rapid development of Web App possible with model-driven architecture. Functions such as online compiling, encapsulation and publishing of Web App are achieved with cloud service. Besides, model library, component library and template library files can be downloaded and updated from the development framework through cloud service to enrich the development environment. Web App can be stored in the cloud, from which the mobile terminal users can download and deploy it in the middleware platform. And push notifications are also provided by the cloud service.

A. Cloud Service Architecture

Fig. 2 illustrates the cloud service architecture in MDWAF. The cloud service architecture mainly provides the cloud services based interface for the Web App developers and mobile terminal users, which facilitate the Web App release and update as well as download and use for end users. The cloud service architecture consists of four layers: basic software support platform layer, basic service layer, the integration and encapsulation service layer and external interface layer.

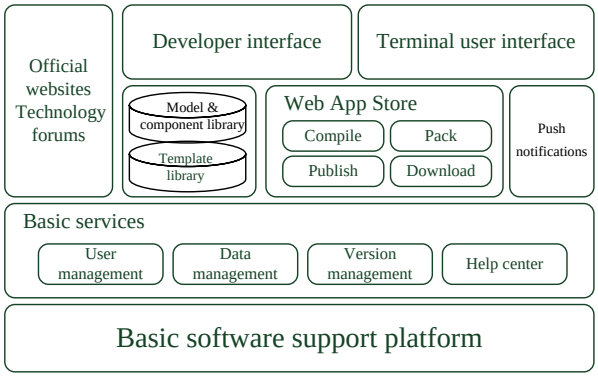


Figure 2. Cloud service architecture

The basic software support platform integrates technologies and frameworks such as, SQL Server, MongoDB, Hadoop, J2EE, ASP.NET, etc, which provide technical support and security for the basic data processing and encapsulation. Technologies like Xfire are used to publish data and business in the form of Web Service based on SOAP protocol to facilitate the invocation of upper layers.

Layer of basic services is built on the foundation of basic software support platform layer. Small-grained services provided by basic software support platform layer was combined and encapsulated into large-grained services here so as to provide the upper layers with access to the data and business. Services provided by basic services layer are user management services, data management services, version management services and help center. User management services refer to the information management for developers to publish projects in the Web App Store and users to download them, including user login authentication, user rights verification, user registration, recording of users' distributing and downloading Web App. Data management services are primarily responsible for the integration of large-scale data and indexing structure to achieve fast retrieval of existing data. Web App project released version management service is responsible for managing and publishing of multi-version Web App. Help center services provides developers and terminal users with development technology and instructions of

Web App, as well as online consulting on official websites and forums.

The integration and encapsulation layer, official websites and technology forums are built on the layer if basic services. Visitors can have access to MDWAF introduction and download support software from official websites and technology forums which also provides the existing developer or mobile terminal user with Web App development or download information. The integration service layer mainly consists of three parts. The first is the service of model library, component library and platform template library. This service is connected with component model and template engine of the development environment, which facilitates Web App development by reuse of component model and templates for different platforms. The second is the service of Web App Store, which integrates the user management, version management and data management services, and also provides interface for publishing and downloading respectively. The third is the service of Push notification, which sends notification including version updates, important news, weather alerts, and application released information etc. to the Web App Engine of mobile terminals.

The external interface layer is the top level of the cloud services, which integrates the services of lower layers, and provides interface to access cloud services for the integrated development environment and the Web App Engine of mobile terminals.

B. Mobile Terminal Architecture

The mobile terminal architecture is shown in Fig. 3. It consists of the operating system of mobile terminal, the browser kernel based on webkit, the platform API plug-ins and its management, integrated services of the Web App Store, Web technology framework, JavaScript plug-in extensions, and Web App.

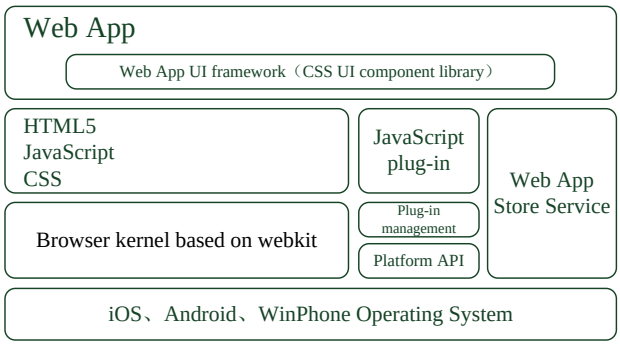


Figure 3. Mobile terminal architecture

Other modules and services are deployed on the operating system of mobile terminal. Operating systems of the mobile terminal such as iOS, Android, WinPhone offers an integrated management of underlying hardware of mobile terminals and interfaces for upper software.

Middleware engine of the mobile terminal, built on the basis of operating system, is primarily responsible for loading and running Web App. The core of the mobile terminal middleware engine is a webkit-based browser kernel, which can render the common code of web standards and interpretively execute the framework. The platform API plug-ins and plug-in management module intercept the platform API invoking events. Access to local resources and external services based on JavaScript was enabled by invoking Operating System APIs supported by the mobile terminal device. Web App Store service takes charge to manage local Web App, and the inquiry and download of Web App are achieved through the interaction with the cloud services.

Above the mobile terminal middleware engine is the common web development standards such as HTML5, JavaScript and CSS3 and JavaScript plug-in library extended for operating system of mobile terminals. HTML5, JavaScript and CSS3 are the key technology in Web App development [7]. Web App can access local resources and external services through interaction between JavaScript and middleware engine of the mobile terminal by Defining the mapping between JavaScript APIs and native APIs of mobile platform [8].

On the top of the architecture is Web App. A Web App UI framework is also included as componentized Web App are achieved though HTML5 and JavaScript components encapsulation.

C. Development Environment Architecture

Development Environment Architecture is the core of MDWAF architecture, functions of which are project management, model-driven development, code file editing, platform simulator and project release.

The developer starts a new project in the project management module which will deploy the project structure and related file automatically, followed by the establishment of the model according to the Web App in the graphical interface of the MDA module and deployment of the related platform template; and finally ended with the transformation to HTML5 and JavaScript codes with MDA module. Developer could modify generated code in the environment provided by code edit module to complete complex component and business logic. Compiled code can be performed on Platform Emulator so that developers can preview or debug the program. The project can be encapsulated and released to the cloud services of Web App Store through the project release module. The development environment should have access to cloud services where various component models and platform template models can be downloaded to implement more model functions and support variant platforms and standards.

The Integrated development environment of MDWAF is implemented based on the Eclipse Extension Mechanisms, and it integrates techniques supported by OMG such as Meta Object Facility (MOF), Unified Modeling Language (UML) and Common Warehouse Metamodel (CWM), as well as Eclipse Modeling Framework (EMF) [9] and Velocity Template Technique to implement relevant functions. Its architecture is shown in Fig. 4 as below.

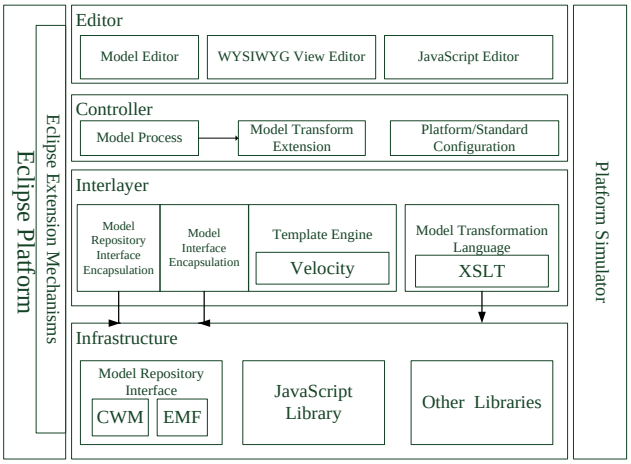


Figure 4. Development environment architecture

The Integrated development environment of MDWAF mainly contains six parts. As illustrated in the Fig. 4, MDWAF was integrated in the Eclipse platform and its extension mechanism as a plug-in to achieve the capabilities of extension and maintenance. In the four-layer architecture of MDWAF, the Infrastructure contains the specific implementation of MDA standard which enables us to access the model and modify it. Object Constraint Language (OCL) is too complex to satisfy the flexible and simple development requirement in widget development. So here we use JavaScript to describe model behavior, with involvement of some other libraries are contained in this layer [7].

The interlayer mainly serves to implement interface encapsulation for infrastructure devices, such as model repository interface encapsulation. And model transformation such as model-to-model transformation and model-to-code transformation are also involved as the former one can use the self-defined transformation rules.

The control later is responsible for controlling the development process, such as platform reconfiguration, system engine invocation for the processing of models and the defined model transformation and extension.

On the top of Fig. 4 is the developers interface, including a graphic-based model editor, WYSIWYG （What You See Is What You Get）and an open-source JavaScript editor.

As shown in the left of the MDWAF architecture, a run-time engine, or may be referred to as platform simulator was set where the generated code can be previewed. Besides, component model and template functions should be updated through cloud services to gradually enrich and improve the model and template library of development environment.

In addition to what is shown in the figure, other functions are provided in the MDWAF architecture for the ease for developers, such as the project and model files starting guide, MDWAF-specific perspective, property interface displaying the component properties, as well as a configuration wizard running the model transformation engine.

Various functional modules of the MDWAF are closely linked and provide a complete model-driven development support altogether. Taking form as an Eclipse plug-in, it can be easily installed or uninstalled, which can take full advantage of the Eclipse extension. Other MDWAF modules are also plug into the Eclipse platform, improving the development efficiency and flexibility.

III. MDWAF Development Process

MDWAF development is a process that builds PIM and then uses model transformation to generate code, which is stared with the establishment of PIM irrelevant of specific platform or specific and followed by the transformation from PIM to PSM and codes. Fig. 5 describes the MDWAF development process of Web App based on MDA.

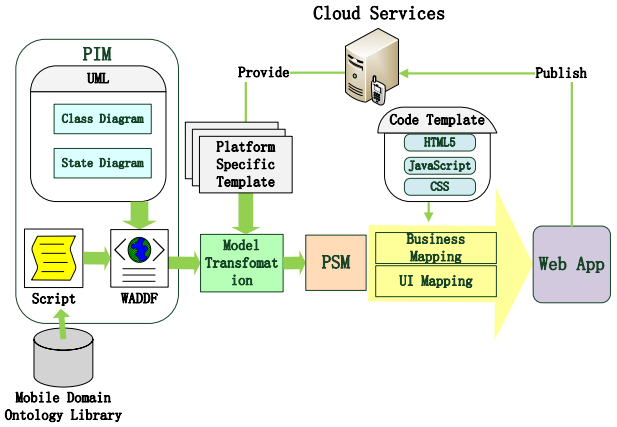


Figure 5 MDWAF development process

PIM is described in UML, which is developed by Eclipse Modeling Framework and used to illustrate the Class Diagram and State Diagram. The Class Diagram is used to present data and the State Diagram is used to present business process in PIM. UML is stored in the form of XML, because of UML model is not suitable for model transformation. In MDWAF we use XSLT to transform UML model to XML document which is suitable for model transformation in Web App development. The product is called Web App Domain Description File (WADDF). In the meantime, an operation script generated by JavaScript Editor which extends the mobile domain ontology library is added to MDWAF. WADDF can be used as PIM model file and the input of PIM to PSM model transformation. According to the chosen Platform Specific Template PSM is created. In the PIM transformation, operation scripts specified in the service ontology are integrated into JavaScript codes. And PSM can be mapped into Web App under the restriction of the Code Template. In the development process, both the Platform Specific Template and Code Template are provided by Cloud Services, so does the complete Web App.

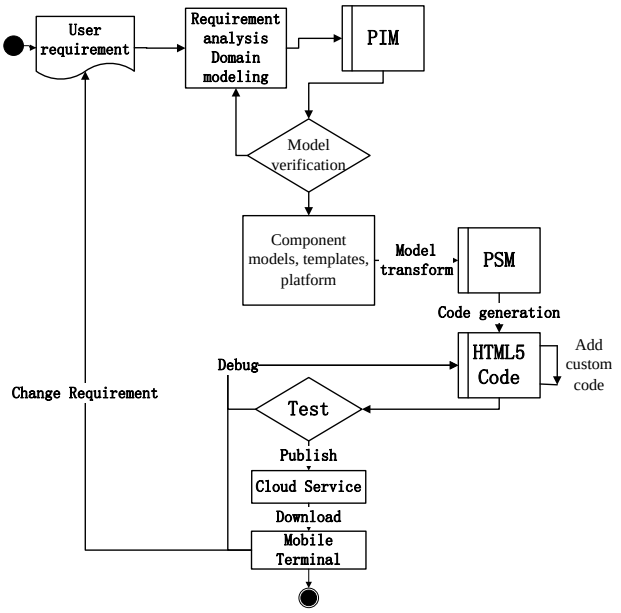


Figure 6 Life cycle of Web App developed in MDWAF

The life cycle of Web App developed in MDWAF can be concluded according to the above mentioned development process as Fig. 6. The Web App development in MDWAF starts with user demands. Based on the requirements documents upon user requirements, requirements analysis and domain modeling are done to achieve the PIM in MDWAF. After that, the PIM will undergo the model verification. If the PIM meets the requirement and MDWAF specifics, the developers will select component models, templates, and the terminal platform. Otherwise domain modeling should be done again. PIM will be transformed into PSM after the selection of component models, templates and terminal platform. HTML5 code framework and operation script based on the service ontology library in mobile domain can be generated based on the selected template.

But the degree of automation of code generation is not necessarily perfect, which requires developers to add custom code to generate code based on demand. In modifying the process, a WYSIWYG view editor and a JavaScript script editor was provided to facilitate development. The code being modified, developers can encapsulate and test the code, and modify the bugs. Web App will be released to the cloud services after test. Cloud services provide mobile terminal users with a service interface with Web App Store to download Web App. Users who download Web App can make comments, and developers can modify the existing bugs or establish new user requirements document and release new versions of the Web App.

IV. KEY TECHNOLOGIES

A. The Establishment of Business Model

Demand models in MDWAF are built based on EMF modeling technology. EMF implements the OMG MOF specification, by extracting the core element set of MOF achieve meta-model concept called Ecore which belongs to the M3 layer, describing UML modeling language meta-model [10]. In the integrated development environment supported by EMF, the model persistence is achieved by XMI. XMI uses the standardization of the XML document format and DTD provides a MOF model and other model defines an xml-based data interchange format. By using XSLT technology can transform ecore files based on UML described in XMI into easy to be understood and implementation platform related model XML document [11].

Fig. 7 illustrates the transformation of business model described in UML from XMI definition to XML.

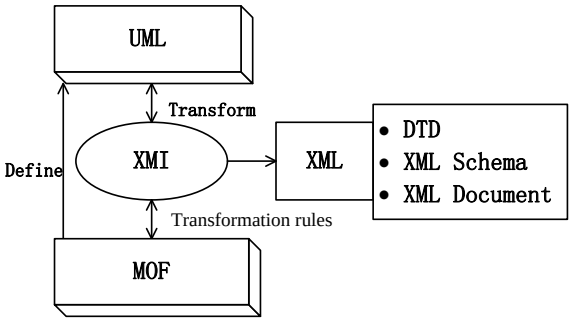


Figure 7 Transformation of business model

Simplified UML class diagram meta-model and state diagram meta-model based on the functional requirements in MDWAF Web App development is shown in Fig. 8.

According to the meta-model definition of MDWAF, the concept of MDWAF requirements modeling is simplified and adapted compared to OMG's UML meta model in order to meet the needs of ease of use and quick development of Web App Development. Annotation was added to UML class diagram meta-model to bind each Class or Attribute into a corresponding component, which facilitate the generation of views in model transformation [12]. Analysis of these two meta models shows the concepts involved in the use of UML class diagrams and UML state diagram are class, association, association class, attribute and operation in class diagram model, as well as state, transition, action and trigger events in the state diagram. Based on these concepts extracted from the class diagram model and the state diagram model, description document WADDF, i.e. the PIM of MDWAF can be established. WADDF will be defined in the following subsection.

B. The Establishment of PIM

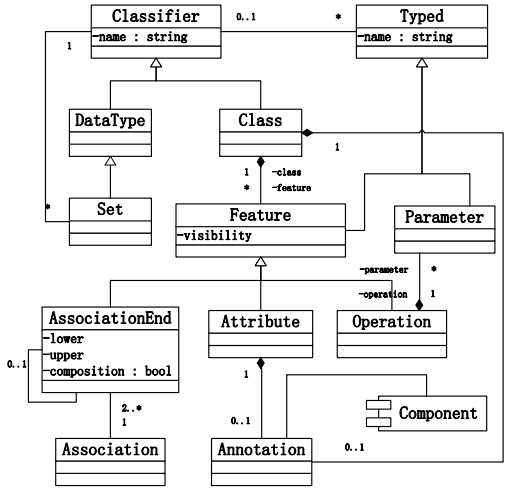
In the field of Web App development, every screen interface of mobile terminal can achieve MDWAF PIM in accordance with the MOVE (Model Operation View Event) mode, and the definitions are shown as follows.

Definition 1：A screen of MDWAF can be defined as a quintuple:

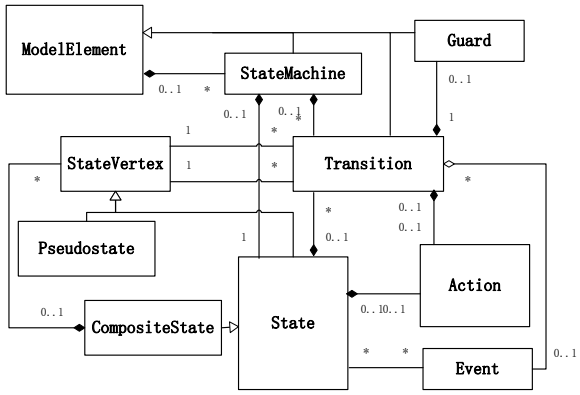
SP = (Id, Models, Operations, Views, Events) (1)

Id is used to identify a mobile terminal screen. Models here refer to the model of the current status of a mobile

terminal within the screen. Operations are the collection of operations of the mobile terminal screen. Each Op in Operations will be described in detail in Definition 2. Views are on behalf of the all the view elements in the mobile terminal screen, including the component elements and page layout. Events are the collection of events triggered by the operation in the mobile terminal screen. Each event in Events will be described in detail in Definition 3.



(a). Class diagram meta-model



(b). State diagram meta-model

Figure 8 meta-model in MDWAF

Definition 2: Op involved in a mobile terminal screen can be defined as a quintuple:

Op = (Id, TargetPage, Parameters, Return, Action) (2)

Each Op has a unique Id to identify the present operation. TargetPage shows transitions of the mobile terminal screen. The transition from current page to targetPage is equivalent to SP->SP ∪ SP', but can different operations can be edited for different transitions. Parameters are the collection of input parameters in the mobile terminal screen. Return refers to the result of the execution of the mobile terminal screen. Action is the specific operation;

Definition 3: the event triggered by operations in a mobile terminal screen interface can be defined as a triplet:

Event = (Dispatcher, EventDescription, EventHandler) (3)

Dispatcher means the event trigger, and composes a complete description of an event with EventDescription. The EventHandler is the process and operation of the current event, generally corresponding to an Op.

According to the above definition, the general WADDF document structure is shown as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<WADDF
    xmlns:wmodel="http://mdwaf.zju.com/waddf/model"
    xmlns:woperation="http://mdwaf.zju.com/waddf/operation"
    xmlns:wview="http://mdwaf.zju.com/waddf/view"
    xmlns:wevent="http://mdwaf.zju.com/waddf/event">
    <ScreenPage Id="">
        <wmodel:Models/>
        <wview:Views/>
        <wcontroller:Operations>
            <wcontroller:Operation Id="">
                <wcontroller:targetPage/>
                <wcontroller:Parameters/>
                <wcontroller:Return/>
                <wcontroller:Action/>
            </wcontroller:Operation>
            ...
        </wcontroller:Operations>
        <wevent:Events>
            <wevent:Event>
                <wevent:Dispatcher/>
                <wevent:EventDescription/>
                <wevent:EventHandler/>
            </wevent:Event>
            ...
        </wevent:Events>
    </ScreenPage>
    ...
</WADDF>
```

Figure 9 WADDF Document Structure

xmlns: wmodel, the xmlns: woperation, the xmlns: wview and the xmlns: wevent in the Fig. 9 is correspondent with the four XML Schema document that defines the meta-model of data, operation, views and events. The elements of WADDF file should observe the Schema definition.

C. Model Transformation and Code Generation

The PIM of WADDF can be transformed into PSM of Web App through the model transformation engine based on XSLT [13]. javax.xml.transform.Transformer supports the XSLT transformation. Completely absorbed the XPath syntax and rules, the XSLT resolve WADDF and build XML node tree and transform the PIM element into PSM elements according to the corresponding XML Schema file, which will be put into the platform model files. These XML-based PSM files describing the structure of the final code are input files of the code generation. The transformation process is shown in Fig. 10, where WADDF document enters into the model transformation engine as the input of the entire model transformation. First, the XSLT-based model transformation engine transforms the WADDF document into a node tree. Elements of WADDF were extracted from the document through component model library supported by cloud and four XML Schema documents defined by PIM. Nodes of the source tree are used to create a result tree fragments according to XSLT template rules. In creating result document, node lists including the root node of the source tree undergo matching, processing and selection recursively and the result tree fragments will be added to the existing result tree. Each node should go through all the matching node template, and then

processed according to platform template file and rules of the transform rule base. That node will be regarded as the current node, and the source node list as the current node list. Finally, the results tree will be generated as PSM files.

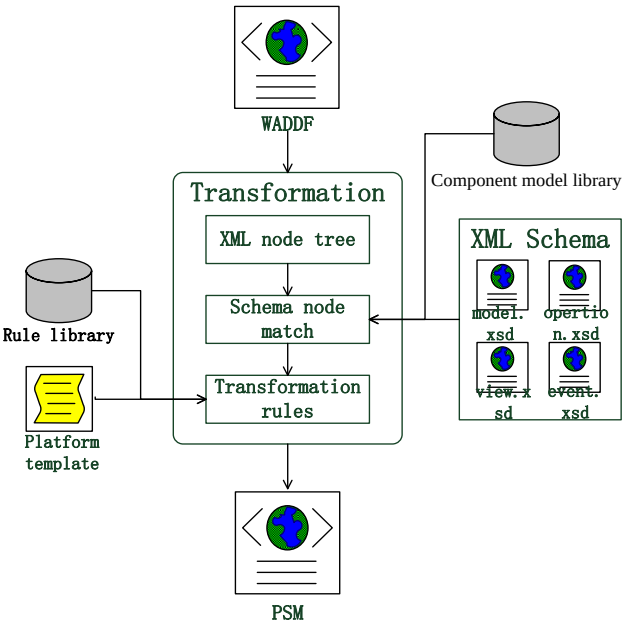


Figure 10 Transformation process

In the generation of PSM files, Model in the WADDF are transformed into data components according to its type ; View into view components according to its description and the corresponding event trigger behavior, and being bound to its corresponding data components; Operation into function of script tag according to properties as the type of parameter return value; Event is bound to corresponding component properties according to its Dispatcher and Description, script of the behavior of the components should be in accordance with the function generated in Operation corresponding to the EventHandler in Event.

In code generation process, the data component should be transformed recursively into corresponding XML description or Jason description in accordance with its data structure, which will be associated with the corresponding view components [14]. The relationship among view components of the PSM file are described in nested form. The generated interface first renders the root view tab, and then recursively render the view contained. The rendered view is an HTML document fragment, whose "hook" position is specified by the parent view. In the generated HTML document, the parent view will create the DIV element for the child view in its rendering. If the child view is defined HTML5 tags, its corresponding HTML document fragments will be hooked directly to the DIV element specified by the parent view. If the child view remains to be complex type, the entire process should be done recursively. The view components belonging to html.dtd files will be mapped to the corresponding components of HTML5 and set its properties according to the component model and existing property values [15]. The components exclude the html.dtd file will be mapped into div tag and generate corresponding JavaScript file according to the settings of its component model, which will be bound according to its Id. The structure of components should maintain unchanged in the mapping process. The transformed view components will be collected in the body labels. The transformation of components exclude the html.dtd file into JavaScript includes four steps. First, the class tag will be mapped into a closure, and then the properties of class tag and attribute tags will be injected into corresponding objects through setAttribute; after that function under method label will be extracted and put into the closure; and finally a transformation of view label similar to viewComponent will be done, where tags belonging to HTML definition will be transformed into corresponding components while others into div tags. Compared with the above mentioned components mapping, the mappings and combinations here are implemented with getElementById or appendChild function in JavaScript.

The code generation ends the model-driven Web App development of MDWAF. The existing framework cannot cover all aspects of Web App development, developers need to modify and debug the generated files and improve the interface. The refined projects will be released to the Web App Store through cloud.

V. CONCLUSION

The development framework of existing platform demands the users' attention towards technique details and duplicated efforts. In order to improve the efficiency of the developers in the development of cross-platform App, the paper proposes an MDA-based Web App development Framework MDWAF. Adopting the idea of MDA into Web App development of Mobile devices, the framework mainly focuses on models in the development. The development starts with the establishment of models based on user requirements and then make progress by model transformation and code generation. Besides, this approach ensures the Web App reusability and extension. In addition, an integrated data management of development and release through cloud services, various middleware for different mobile platforms, and JavaScript plug-ins extended and unified through different platform API, the Web App development can be deployed to multiple platforms. However, the present MDWAF lacks of the model verification function, the support for complex modeling, and some component libraries. We will improve it in the future work.

ACKNOWLEDGMENT

This work is partly supported by the Science Foundation of Zhejiang Province under Grand No. 2010R50009, People's Republic of China.

REFERENCES

[1] David M,"Building Websites With HTML5 to Work With Mobile Phones," HTML5 Mobile Websites, 2012:3-53.

[2] Miller J M J. MDA Guide Version 1.0.1[EB/OL]. http://www.omg.com/mda.
[3] Frankel D. Model driven architecture: applying MDA to enterprise computing[M].Wiley, 2003：328.
[4] Pahl C., Zhu Y., "Model-driven Connector Development for Service-based Information System Architectures," Journal of Software, Vol 4, No 3 (2009), pp. 199-209, May 2009.
[5] James H. Hill, Aniruddha Gokhale,"Model-driven Engineering for Early QoS Validation of Component-based Software Systems," Journal of Software, Vol 2, No 3 (2007), pp. 9-18, Sep 2007.
[6] Bragança A, Machado R J,"A model-driven approach for the derivation of architectural requirements of software product lines," Innovations in Systems and Software Engineering, Vol 5, No 1 (2007) , pp. 65-78
[7] David M. HTML5 JavaScript Model[J]. HTML5, 2010:209-240.
[8] Shi W, Wu M, Wang S. et al," Local resource accessing mechanism on multiple mobile platform", Proceedings of the 9th IEEE International Conference on Embedded Software and Systems, ICESS-2012, pp. 1716-1721
[9] Westfechtel B,"Merging of EMF Models ," Software & Systems Modeling, 2012.
[10] Steinberg D.,"EMF: Eclipse Modeling Framework[M]," Addison-Wesley, 2009：704.
[11] Mattsson A, Beekveld M.,"Simplifying maintenance by using XSLT to unlock UML models in a distributed development environment," Software Maintenance(ICSM), 2007.
[12] Fouad A, Phalp K, Kanyaru J M, et al,"Embedding requirements within Model-Driven Architecture ," Software Quality Journal, Vol 19, No 2 (2011) ,pp. 411-430.
[13] Groppe S, Groppe J, Böttcher S, et al," Optimizing the execution of XSLT stylesheets for querying transformed XML Data ," Knowledge and Information Systems, Vol 18, No 3 (2009) ,pp. 331-391.
[14] Kuntsche S, Barz T, Kraus R, et al. "MOSAIC a web-based modeling environment for code generation," Computers & Chemical Engineering, Vol 35, No 11 (2011) ,pp. 2257-2273.
[15] Marco Casario P E C B. "HTML5 Solutions: Essential Techniques for HTML5 Developers[M],"2011

Rongliang Luo received the BS degree in Automation from Zhejiang University Technology in 1997 and MS degree in Computer Science and Engineering from Zhejiang University in March 2003. His major interests include Software Engineering, Software Development Methodology.

Peng Xiao received the BE degree in Software Engineering from Zhejiang University in 2010 and MS degree in Computer Science and Technology from Zhejiang University in March 2013. His research interests include Software Engineering, Model Driven Architecture.

Qianxi Lv is a master degree candidate in Translation and Inter pretation from Zhejiang University, China, from 2011 to 2013. Her current research interests include Translation Studies Based on Practice.

Minghui Wu obtained the M.S. and PhD. Degrees in Computer Science and Technology from Zhejiang University, China, in 2000 and 2011, respectively. Since Dec. 2011, he is a Professor in Computer Science and Technology, Zhejiang University City College. His current research interests include Software Engineering and Network Application.

Bin Peng obtained the M.S. degree in Computer Science and Technology from Zhejiang University, China, in 2003.He is an instructor of Computer Science and Technology, Zhejiang University City College from 2005.His current research interests include Business Automation, SOA and mobile Internet.

Shuoping Wang obtained the M.S. Degree in Computer Science and Technology from Zhejiang University, China, in1996. Since Dec. 2011, she is an Associate Professor in Computer Science and Technology, Zhejiang University City College. Her current research interests include Management Information System and Network Application.

Ming Guo obtained PhD. Degree in Computer Science from Zhejiang University, China, in 2004. Since Dec. 2008, he is Associate Professor in Computer Science and Technology, Zhejiang University City College. His current research interests include Mobile Computing, Cluster Computing, Programming Language.