

Web-based User Interface Modeling and Automatic Mobile Web App Generation Tool

Kibong Choi¹ and Saehwa Kim^{2*}

^{1,2}Hankuk University of Foreign Studies, Yongin-si, Gyeonggi-do, 449-791 Korea

*Corresponding author

Abstract—This paper proposes a web-based logical user interface (UI) modeling and automatic mobile web App generation tool. Logical UI modeling is based on PELUM (Pattern and Event based Logical User Interface Modeling). PELUM is an effective embedded SW development methodology for UI centric embedded systems. Our automatic web App generation tool consists of a model editor and a code generator. The model editor provides a web environment for editing a Logical UI model (LUM) and a Programming Interface Model (PIM). The code generator generates a mobile web App from LUM and PIM. LUM abstracts screen configurations and events while PIM abstracts local database schema. Generated mobile web Apps conform to MVC (Model-View-Controller) architecture so that users can easily optimize them. By exploiting our tool, users can rapidly implement mobile web Apps that can run on multiple devices.

Keywords—software engineering; modeling; user interface; automatic code generation

I. INTRODUCTION

The proliferation of various smart embedded devices, including smart phones, tablets, navigators, and smart TVs, lead to the ever increasing needs for applications that are adaptable for multiple devices. One of the important bottlenecks for increasing such reusability is that the user interfaces (UIs) for multiple devices cannot be the same. To





overcome this problem, we have proposed PELUM (Pattern and Event based Logical User Interface Modeling) in [1]. PELUM encompasses (1) a pattern-based method for deriving a UI implementation from a UI model, (2) a meta-model for modeling both abstract UI and task model, whose name is Logical User Interface Model (LUM), and (3) its supporting modeling tool, which is based on Eclipse RCP (Rich Client Platform) [2].

In this paper, we propose a web-based logical UI modeling and automatic mobile web App generation tool for PELUM. Our related work includes UI mock-up tools such as Balsamiq [3] and Invision [4] as well as UI authoring tools such as Appery [5] and mBizmaker [6]. UI mock-up tools have limitations that they cannot generate Apps that can run on real devices. UI authoring tools also have limitations that users cannot manipulate their own source code since generated source code itself is not available to users.















The proposed tool consists of a model editor and a code generator. It is web-based and thus users do not need to install them in their computers. The model editor provides a web environment for editing a Logical UI model (LUM) and a Programming Interface Model (PIM). These models are in the xml format. The code generator configure jQuery Mobile by parsing these two model files and generate an operable App.

TABLE I.

THE 4-LAYER AND META-MODEL OF PELUM

PELUM Hierarchical Architecture	Example (Alarm)
Graphical Resource Model (GRM)	
UI Controls and Layout Model (CLM)	
Logical UI Model (LUM)	
Programming Interface Model (PIM)	

(a) 4-layer architecture of PELUM

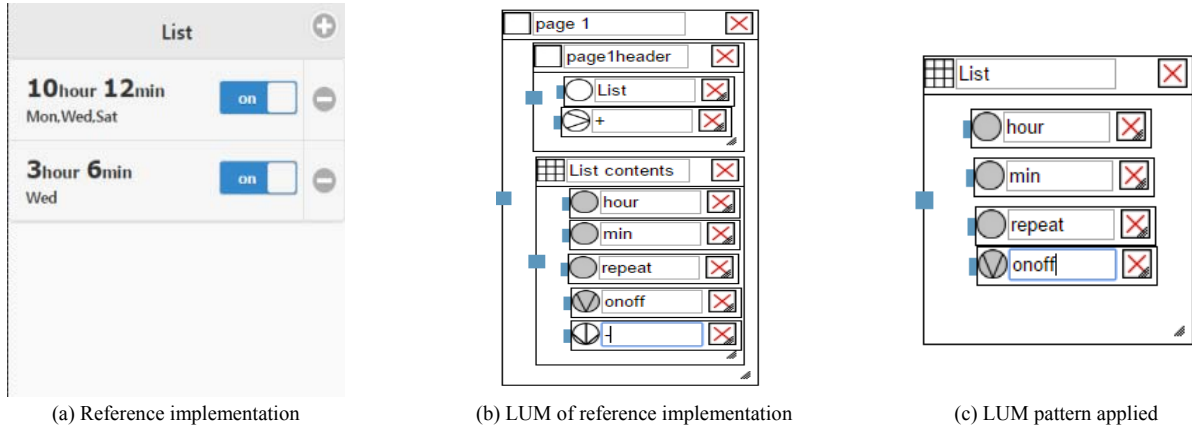
Modeling elements	Description
<input type="checkbox"/> Container	A view that contains any LUM components
 Population	<input type="checkbox"/> that represents a collection of instances of a class in PIM
 Options	<input type="checkbox"/> that contains optional services for its container
 Settings	<input type="checkbox"/> for setting certain configuration
 Presentation	A view that presents constant data or an image
 Navigator	<input type="checkbox"/> for activating another <input type="checkbox"/> or <input type="checkbox"/>
 Internal Service	<input type="checkbox"/> for providing an internal service
 Instance	<input type="checkbox"/> for representing instance variable of an object in PIM
 Editable	An Editable 
 Toggleable	 that is turned on or off instantly
 Selectable	 that is selectable from some predefined set of values
 Event	An association between components (<input type="checkbox"/> or <input type="checkbox"/>). The arrival of an event to a source component activates the target component.

(b) PELUM meta-model components

The generated App is in the MVC architecture for users to easily optimize to multiple devices. The main advantage of the proposed tool is that UI models can be flexibly applied to

multiple platforms since users can check events and logical UI components visually and the local database is automatically created.

TABLE II. LUM PATTERN APPLICATION EXAMPLE



The remainder of the paper is as follows. In Section II, we present PELUM, which our tool is based on. In Section III, we present our proposed tool. Section IV concludes the paper

II. PELUM (PATTERN AND EVENT-BASED LOGICAL UI MODELING)

PELUM proposes 4 hierarchical layer models as shown in TABLE I (a). First, PIM (Programming Interface Model) is generally given as an API layer, depending on the specific platform like Android, iOS, etc. The proposed tool in this paper exploits this PIM as its data model. Second, LUM (Logical UI Model) is a layer that models abstract UI and components with

events. Our tool in this paper exploits this LUM as the UI model. Third, CLM (UI Control and Layout Model) is a layer that determines a concrete UI controls (widgets such as buttons, check box, etc.) and a specific layout of UI elements. Finally, GRM is a layer that provides graphic resources of UI elements. Our code generator tool automatically synthesizes CLM and GRM based on jQuery mobile [7] based on patterned templates. Note that our model editor tool models PIM and LUM and our code generator tool synthesizes a mobile App with PIM and LUM modeled in our model editor along with patterned templates of CLM and GRM.

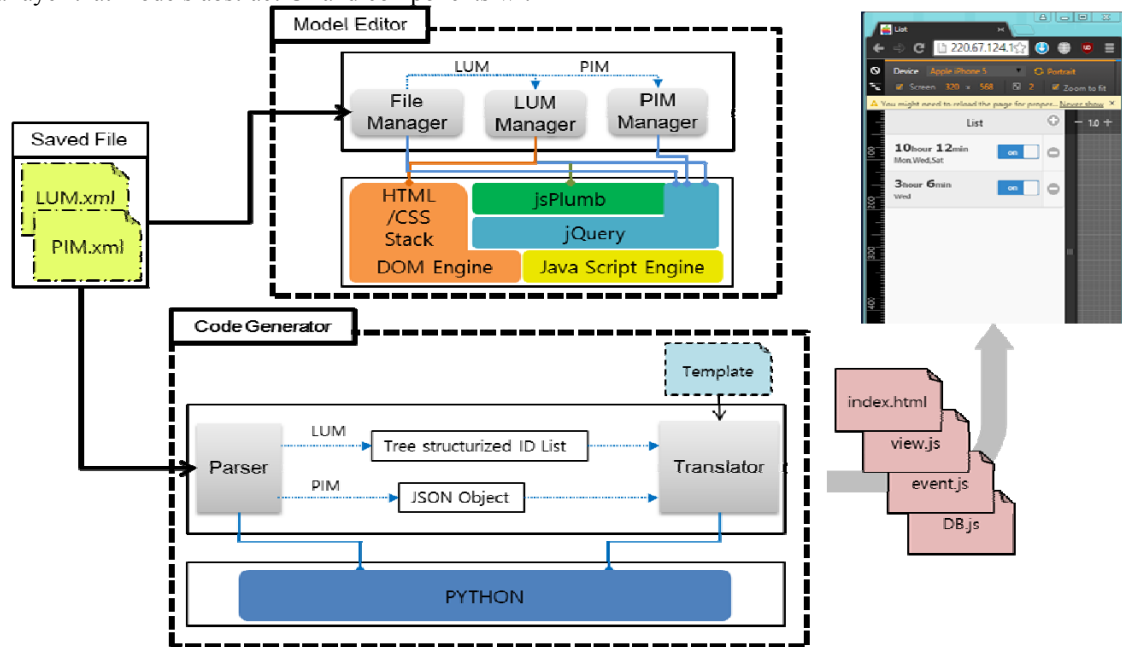


FIGURE I. WEB-BASED UI MODELING AND AUTOMATIC MOBILE WEB APP GENERATION TOOL ARCHITECTURE

TALBE I (b) shows the brief explanation for each LUM meta-model. Container is a general view that contains any

LUM components. Population is a container that represents a collection of instances of a class or classes in PIM.

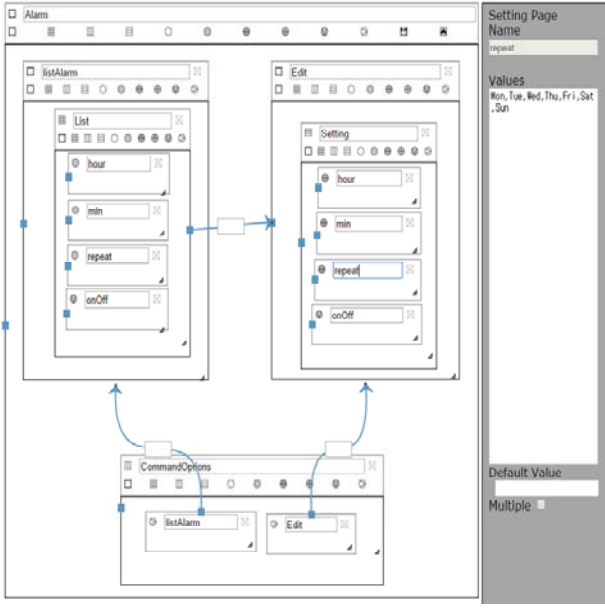


FIGURE II. OUR PROPOSED LOGICAL MODEL EDITOR: THE LEFT IS FOR LUM (LOGICAL UI MODEL) AND THE RIGHT IS FOR PIM (PROGRAMMING INTERFACE MODEL)

Option is a container that contains optional services for its container. Setting is a container for setting certain configurations. Presentation is a view for constant data or images. Navigator is a presentation that activates other presentations. InternalService is a presentation that provides internal service that is connected with APIs of PIM. Instance is a presentation for representing an instance variable of an object in PIM. Editable is an editable presentation. Toggable is an instance that is turned on or off instantly. Selectable is an instance that is selectable from some predefined set of values. Finally, Event is an directional association between components (container or presentation). If the source component of an arrow gets the event, the target component is activated

TABLE II shows a pattern example for population whose detailed explanations are available in [8]. TABLE II (a) shows a reference implementation, Table II (b) shows an LUM model that models all UI elements. Table II (c) shows a patterned LUM model for this which omits the title of header, the add button, and, the delete-button of list as TABLE 2 (a).

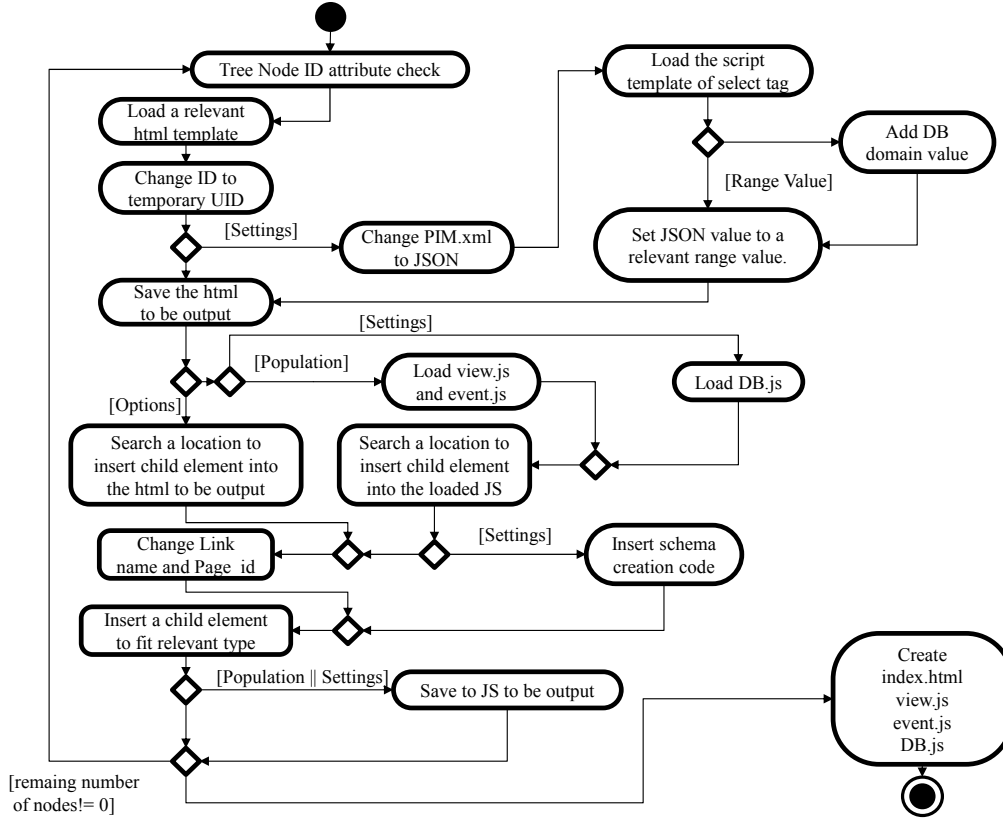


FIGURE III. AUTOMATIC MODEL TRANSLATION ALGORITHM (REPRESENTED IN UML ACTIVITY DIAGRAM).

III. WEB-BASED UI MODELING AND AUTOMATIC MOBILE WEB APP GENERATION TOOL

FIGURE I shows the architecture of our proposed web-based UI modeling and automatic mobile web app generation tool. This tool is composed of a model editor and a code generator. Figure II shows the model editor which enables for

users to edit UI model and data model, save their edited model, and load their saved model to the editor.

The model editor is composed of the file manager, the UI model manager, and data model manager as shown in FIGURE I. The file manager loads and saving files. The file manager delivers each file of a specific type to a proper manager by classifying files. We implemented the file manager based on jQuery to support multi browsers and to simplify searching DOM (Document Object Model) and binding events. Our UI model manager is based on jsPlumb as well as jQuery. We exploit jsPlumb to draw inter-model event on browsers. Finally, our data model manager manages the value of the data model that is needed to create the database schema. We also implemented the data model manager based on jQuery.

The code generator is composed of the parser and the translator as shown in FIGURE I. The parser receives the xml values from the model editor as its input. Then, the xml values are modified for the translator. In the parser, UI model is changed to tree structured ID list and the data model is modified to JSON objects. Parser is based on python for easy string manipulation. The translator gets preprocessed input value from the parser and generates a mobile web App based on the rules defined at [9].

The algorithm of the translator is outlined in Figure III. As shown, it checks the tree node attribute at ID list that is structured by the parser. Then, it loads templates that correspond to the types of node ID attributes. Next, the translator inserts templates at the proper location that is found at a pre-defined flag within the output file. For implementing the templates, we exploit jQuery Mobile and the MVC architecture. Note that these templates cover CLM and GRM in the PELUM 4-layer architecture which we presented in Section II. We classify templates according to the MVC architecture as shown in FIGURE IV. Template file types are as follows.

- Html template: it implements UIs based on jQuery.
- JavaScript template: it implements db, view, event, which map model, view, and controller of the MVC architecture, respectively.

The code generator generates index.html, db.js, view.js, event.js files from assembling these templates as shown in Figure I. The main strong point of our tool is that it enables for users to generate and initialize a local database (Indexed DB) only by modeling UIs. Moreover, our tool is very flexible because it represents UIs from the logical point of view, which dramatically increases the reusability of mobile Apps for multiple devices.

IV. CONCLUSION

In this paper, we have presented our web-based UI modeling and automatic mobile web App generation tool. This tool is composed with a model editor and a code generator. The Users can model their UIs using the model editor. Specifically, the model editor enables users to model their logical UI models (LUM) and the programming interface model (PIM). The code generator generates a mobile web app based on (LUM and PIM) models edited in the model editor. The automatically generated App can be easily configurable according to various

screen sizes, actuators, and sensors since the implementation is based on jQuery mobile and the architecture follows the MVC architectural pattern. With this, users can rapidly develop their Apps for multiple devices. As future work, we will enable for users to model the GRM (Graphical Resource Model) and CLM (Control UI and Layout Model). With this, users can more easily customize their Apps by customizing graphic, control widget and layouts.

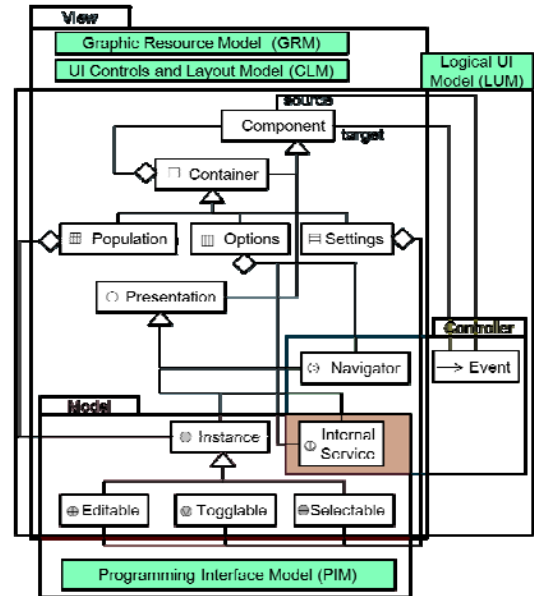


FIGURE IV. WEB-BASED UI MODELING AND AUTOMATIC MOBILE WEB APP GENERATION TOOL ARCHITECTURE

ACKNOWLEDGEMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2013R1A1A3006819).

REFERENCES

- [1] S. Kim, Pattern and Event Based Logical UI Modeling for Multi-Device Embedded Applications, Proceedings of International Conference on Convergence and Hybrid Information Technology, 2011.
- [2] S. Kim, Graphical Modeling Environment for Logical User Interfaces Based on Eclipse GMF, Journal of Information Industrial Engineering, 2011.
- [3] Balsamiq, <https://balsamiq.com/>
- [4] Invision, <http://www.invisionapp.com/>
- [5] Appery, <https://www.appery.io/>
- [6] mBizmaker, <http://http://www.mbizmaker.com/>
- [7] jQuery mobile, <https://jquerymobile.com/>
- [8] K. Choi, S. Kim, Mobile Web Based Reference Implementation for Logical UI Modeling for Embedded Applications, Journal of Information Industrial Engineering, 2014
- [9] K. Choi, S. Kim, Mapping Logical UI Models to HTML5-based Embedded Applications, Korea Computer Congress (KCC), 2014.