PUML and PGML: Device-independent UI and Logic Markup Languages on Small and Mobile Appliances

Tzu-Han Kao, Yung-Yu Chen, Tsung-Han Tsai, Hung-Jen Chou,
Wei-Hsuan Lin, and Shyan-Ming Yuan

Department of Computer and Information Science, National Chiao Tung University,
1001 Ta Hsueh Rd., Hsinchu 300, Taiwan.
{gis89539, gis93533, is91027, is91085,
is91087, smyuan}@cis.nctu.edu.tw

Abstract. To accomplish developing mobile web applications on variety of mobile execution environments, we propose Pervasive User interface Markup Language (PUML) describing user interfaces for applications on the small devices, and Pervasive loGic Markup Language (PGML) representing the computational logic of the applications. Furthermore, we exploit the XSLT/XPath transformation mechanism to transform documents of PUML/PGML into the target languages, and evaluate them by implementing toolkit and runtime service applications. In Brief, our paper contributes an XML-based application model to assist mobile application developers.

1 Introduction

Currently, there are several mobile execution environments on plentiful mobile appliances. On these environments, XHTML MP (Mobile Profile), Compact HTML (CHTML), etc can be used to develop the mobile web-based applications. To accelerate the development on mobile platform, more some mediator languages are designed, such as XUL [9], UIML [8], AUIML [10], and XIML [11]. XUL (XML User-interface Language) is designed as a cross-platform language to describe the graphical user interfaces of the applications on desktop computers. It works to make the user interfaces of the applications portable.

In [12], we discussed the use of combination of XUL and LGML. In describing user interfaces, we abstracted useful elements from the original XUL elements to describe the widgets on the screens of the display-limited devices. On the logic side, we applied LGML, which is the XML-based language we design to represent the logic. Pervasive User Interface Markup Language (PUML) is the new markup language we have designed to describe user interfaces for applications on the small devices. Moreover, we have revised and improved LGML, and rename it as Pervasive loGic Markup Language (PGML) to state our objective definitely. We exploit the XSLT to transform them into the languages on the mobile environments.

This paper is organized as follows. Section 2 and Section 3, we describe the design of PUML and PGML, respectively. Section 4 presents the transformation mechanism

and the simulating result of generating codes. Finally, Section 5 shows the applications; Section 6 discusses the conclusion and the future work.

2 Pervasive User-interface Markup Language (PUML)

User interfaces of applications can be divided into two classes roughly. One is plentiful category; the other is fundamental category. The former has abundant widgets to display UI controls. For example, displaying HTML documents on the desktop of PCs has a variety of modules, involving Frames module, Applet module, and etc. The later merely consists of basic presenting modules, such as Form module, Image module. User interfaces of applications on the small and mobile device belong to this kind. A PUML documents is designed to comprises several containers. Each of the containers contains the basic widgets including label, text filed, single-choice listing, multi-choice listing, picture, and action. The following figure shows the conceptual view of PUML. The principles of designing PUML including: (1) user interface abstracting; (2) intermediate language fashioning, and (3) OO (Object-oriented) conceptualizing.

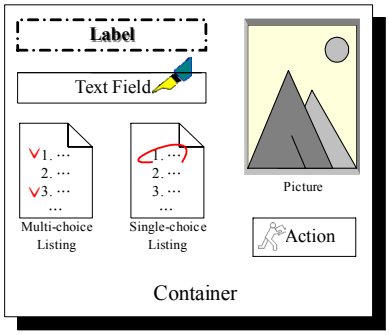

Fig. 1. The conceptual view of PUML.

In PUML in the `<puml:user-interface>` element has three types of child elements, which are `<puml:logic-objects>`, `<puml:board>`, and `<puml:layout>`. The `<puml:logic-objects>` element is used to declare the logic object, e.g. .pgml file, and the name used in a PUML document. A `<puml:board>` element can contain `<puml:label>`, `<puml:textnote>`, `<puml:listpaper>`, `<puml:picture>`, and `<puml:action>`, which stand for label, text field, list, image, and action, respectively. These elements are the widgets which are involved in a `<puml:board>` container. Listing 1 illustrates the document structure of a user interface description using PUML.

Listing 1. An overview of the document structure of PUML

```
1   <?xml version="1.0"?>
2
3   <puml:user-interface … >
5    <puml:logic-objects> … </puml:logic-objects>
7    <puml:board … >
8     <puml:logic-objects> … </puml:logic-objects>
10     <puml:textnote … />
12     <puml:picture … />
14     <puml:label … />
16     <puml:listpaper … />
18     <puml:action …> … </puml:action>
19   </puml:board>
21  </puml:user-interface>
```

A <puml:picture> element, in a <puml:board> … </puml:board> block, can be used to show an image. It has four attributes: name, source, altText, and align attributes. A <puml:label> is similar to the label of the windows on the PCs. It can display a read-only text string on the screen. Its main attribute is showText, which can be assigned a string value to be displayed on the screen. We can use this element to show some title or prompt information. Oppositely, <puml:textnote> can get a string from the user's input. The <puml:listpaper> element serves as a choice group for picking a single item of a listing, or selecting a group of options among the items of the listing. The are two elements (<puml:action> and <puml:buttom>) designed for event-binding in PUML v1.4. <puml:action> elements can be rendered to the submission items of a form. While users press a submission button of a form, the request parameters inputed in the form are sent to the server side by get method in HTTP protocol. <puml:buttom> is used to show the button in the interaction UIs. This element has three attributes including name, showText, and action. The two formers provide the same meaning mentioned above. Users can specify the address in the attribute action field.

3 Pervasive loGic Markup Language (PGML)

In PUML, the design principles of the language we consider are: (1) computation generalizing, (2) intermediate language fashioning, and (3) OO conceptualizing. The first means that, logic languages, such as C, Java, or WML Script, have the primary statements to declare variables and function blocks, and flow-control and condition-control mechanisms to complete basic computation. Therefore, we abstract the primary expressions and statements to define PGML. From the three points, we make PGML own the following capabilities: the object-oriented concept, variables declaration, flow-control, condition-control statements, and method declarations. Peripherally the features of PGML and PUML are different, but essentially the two languages have the same intention. Figure 2 demonstrates a PGML object in a tree

view of the XML Schema [13]. An object, e.g. a PGML file, is composed of a variable declaration <declaration> and a least one method <method>.

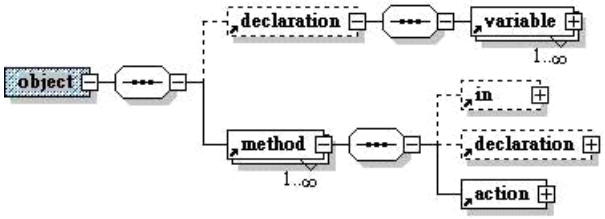

Fig. 2. A tree view of the XML Schema of PGML

Programmers can use PGML elements to write the computational logic. For instance, the <pgml:method> element expresses a method declaration block. The <pgml:in> element contains some child elements, which are the arguments needed passing into this method. In PGML, there is a element <pgml:init> similar to the <pgml:in> element. Differently, it is used to declare and initiate the local variables in a flow-control element <pgml:for>…</pgml:for>, or a method declaration block <pgml:method>…</pgml:method>. The following listing demonstrates a PGML example, and Listing 3 shows the Java source program transformed from the PGML code.

Listing 2. A method declaration in PGML

```
1 <?xml version="1.0"?>
2 <pgml:object name="addTwoNum" version="1.0"
3  xmlns:pgml="http://dcsw3.cis.nctu.edu.tw/Project/
4  pervasive/PGML/" >
5
6 <pgml:method name="sum"  visibility="public" returntype="int">
8       <pgml:in>
9         <pgml:variable name="num1" type="int" />
10        <pgml:variable name="num2" type="int" />
11      </pgml:in>
13      <pgml:action>
14        <pgml:return>
15          <pgml:add>
16            <pgml:operand select="num1" />
17            <pgml:operand select="num2" />
18          </pgml:add>
19        </pgml:return>
20      </pgml:action>
22</pgml:method>
24</pgml:object>
```

Listing 3. The Java code transformed from the PGML document in Listing 4

```
1   package SumExample;
2   import SumExample.*;
3   import java.lang.Integer;
4   import java.lang.String;
5
6   public class addTwoNum {
8     public addTwoNum(){ }
9     public int sum( String  sys_num1, String sys_num2 ){
10      int  num1= Integer.parseInt(   sys_num1);
11      int  num2= Integer.parseInt(   sys_num2);
12      return ( num1 + num2 );
13    }
14 }
```

4 Leveraging XSLT/XPath Transformation

4.1 An overview of language transformation mechanism

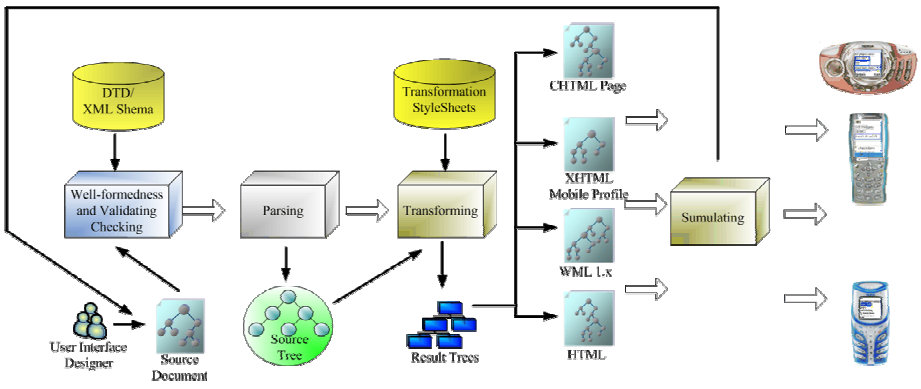

Fig. 3. the well-form checking, parsing, transforming, and simulating sequence

In the implementation, we use an XSLT engine to transform parsed PUML trees into the result trees. Next, stream results will be serialized into the source programs of the target languages (e.g. WML, ASPX, etc). The process contains the following four main steps:

PUML well-formedness checking and validating: When finishing writing a PUML document, a programmer can use the development toolkit to check the well-formedness and validity for the PUML/PGML document.

PUML parsing: With no error found in the document, the document will be parsed into a DOM tree.

PUML transforming: The PUML/PGML transformer, deriving from the transformation engine Xalan-Java 2.5.1 [12], is capable of transforming the tree into the result trees through each transformation stylesheet for the target languages. Each of stream results (`javax.xml.transform.stream.StreamResult`) a result tree is generated. Besides, the stream results are serialized into source programs. The transformation procedure will be explained in the next section.

PUML compiling: MExE Language Compiler (mexe-compiler) can compile each of the source programs generated into its specific executable code. For example, for the J2ME and PJava platforms, a Java source program (a .java file) generated is compiled into Java bytecodes (.class files). Nonetheless, not all the generated source programs need to be compiled. .wml (WML) and .wmls (WML Script) codes need no compilation, since they can be interpreted by user agents of client devices. From this point, if the source programs of Java were generated, they would be compiled into executable codes by the mexe-compiler. Otherwise, when the programs of WML and WML Script are generated, they are not compiled.

Table 1. The mappings from PUML tags into WML expressions in the PUML-to-WML transformation stylesheet

	PUML Tag	**MIDP Expression**
Root element	`<user-interface>`	`javax.microedition.lcdui.MIDlet`
Container	`<board>`	`javax.microedition.lcdui.Form`
Text String	`<label>`	`javax.microedition.lcdui.StringItem`
Text Field	`<textnote>`	`javax.microedition.lcdui.TextField`
Selection List	`<listapper>` `<item>` … `</listpaper>`	`javax.microedition.lcdui.ChoiceGroup`
Image Display	`<picture>`	`javax.microedition.lcdui.ImageItem`
Action Trigger	`<action>`	`javax.microedition.lcdui.Command`

5 Applications

5.1 An example

In the section, we will demonstrate an example written in PUML and PGML documents, and the simulating results of the generated J2ME and WML codes from apply-

ing the transformation mechanism, mentioned above. Listing 4 is a PUML document (UIExample.puml). There are three <puml:textnote> elements in the listing. The first two can get two input numbers, and the last can display the result by summing the two numbers up. The <puml:action> … </puml:action> block describes that the two inputted values are passed into the sum method of object1, e.g. addTwoNum.pgml (declared in Line 12-14). Furthermore, the value attribute of the <textnote> widget, sum in board2, will be updated by the retuned value after the action is triggered. Line 27 describes the code to accomplish that. The section of addTwoNum.pgml can sum the two inputted number.

Listing 4. A user interface described in PUML

```
1  <?xml version="1.0"?>
2  <puml:user-interface name="UIExample" version="1.2"
3   xmlns:puml="http://dcsw3.cis.nctu.edu.tw/Project/
4   pervasive/PUML/">
5
6    <puml:board name="board1" title="FirstPage" >
7      …
8    </puml:board>
9
10   <puml:board name="board2" title="SecondPage">
11
12     <puml:logic-objects>
13       <puml:object name="object1"  source="addTwoNum.pgml" />
14     </puml:logic-objects>
15
16     <puml:label name="mainTitle"
17      showText="Input two numbers:" />
18     <puml:label name="num1Title" showText="Number 1:" />
19     <puml:textnote name="num1" value="0" />
20
21     <puml:label name="num2Title" showText="Number 2:" />
22     <puml:textnote name="num2" value="0" />
23
24     <puml:label name="sumTitle" showText="Sum:" />
25     <puml:textnote name="sum" value="0" />
26
27     <puml:action name="action2" showText="action2">
28       <puml:change container="board2"
29        component="sum" update="value">
30         <puml:use-object name="object1" method="sum">
31           <puml:param select="num1" />
32           <puml:param select="num2" />
33          </puml:use-object>
34       </puml:change>
35     </puml:action>
36   </puml:board>
37</puml:user-interdace>
```

The following code (Listing 5) is the WML code which generated from UIExample.puml. In the code, there are three <input> elements which are converted from the three <textnote> elements in Listing 6. The top three of Figure 4 demonstrate

simulating the WML code generated. For example, a user inputs two number, 1 and 2, in Step 1, and selects action 2 in Step 2 subsequently. Then, the `<go>` would be performed. However, there is a problem—how to accomplish passing the values of the numbers into the WMLS function, and updating the value attribute of the `<input name="sum" … />` element—must be coped with. The trick we used is exploiting `WMLBrowser.getVal()` and `WMLBrowser.setVal()`. `WMLBrowser.getVal()` can be used to get the value of the variable specified from the WMLBrowser environment. Specifically, the values of the variables, standing for the two `<input>`, can be obtained by invoking `WMLBrowser.getVal()` in the WML Script. Returning the computed result can be completed through `WMLBrowser.setVal()`. addTwoNumBroker.wmls is the code generated in transformation to manipulate this event-handling. The related usage of the two methods can be referred in [14].

Listing 5. The WML code transformed from the PUML code in Listing 6

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
3     "http://www.wapforum.org/DTD/wml_1.1.xml">
4   <wml>
5
6     <card title="FirstPage" id="board1">
7       …
8     </card>
9
10    <card title="SecondPage" id="board2">
11      <p>Input two numbers:</p>
12      <p>Number 1:</p>
13      <p><input name="num1" value="0"/></p>
14      <p>Number 2:</p>
15      <p><input name="num2" value="0"/></p>
16      <p>Sum:</p>
17      <p><input name="sum" value="0"/></p>
18      <do name="action2" type="accept" label="action2">
19        <go href="addTwoNumBroker.
20          wmls#start('board2action2')"/></do>
21    </card>
22
23  </wml>
```

Fig. 4. The WML code generated from the code of PUML/PGML

5.2 A web service by using XML web services

The following (Figure 5) demonstrates the architecture which provides the transformation mechanism as the web services. This architecture has two advantages: (1) diverse front-end side and extensibility, and (2) version control. The former is that we can design not only win-based toolkit, but also construct a web-based toolkit for programmers to design UIs through the Internet. The program of the front-end side is not restricted. You can develop your applications and then use the interfaces of the transformation functions to generate WML, CHTML, and XHTML MP pages by using XML Web Services invocation. The latter indicates that the original components can be replaced easily with new ones.

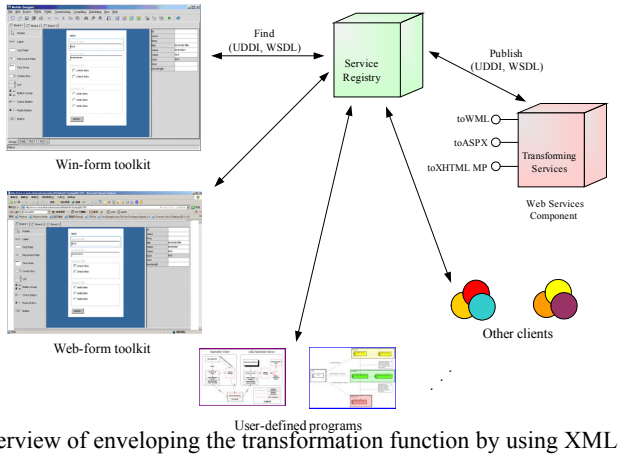

Fig. 5. the overview of enveloping the transformation function by using XML web services

Listing 6. An overview of the document structure of PUML:

```
Server side
[WebMethod]
public XmlDocument  TransformToLang(String lang, XmlDocument  inPUML){
   XmlDocument resultNode = transformers.Transform(lang, inPUML);
   return resultNode;
}
Client side
...
WindowsApplication1.TransForm.AdaptationServer tfcl=new WindowsApplica-
tion1.TransForm.AdaptationServer();
tfcl.Url=(string)HMtrans[name];
((TextBox)page.Controls[0]).Text=tfcl.TransformTest(page.Text,
myXmlDocument);
...
```

6 Conclusions and Future work

In this paper, we introduced the design of PUML/PGML and the applications for the small and mobile devices. Programming by using XML shows the following advantages: (1) transforming a PUML source into others in the target languages easily, and (2) neglecting which target environment to run applications when writing the programs of the applications. They can create form interaction block of the web pages by using PUML. However, it is not user-friendly for programmers to write PUML document directly. To improve that, we are planning to design a toolkit module which enables programmers to construct PUML from existing HTML pages in drag & drop way. Besides, toward the pervasiveness, we will explore the mechanisms of transforming PUML not only into CHTML [4], XHTML [5], JavaSwing [6], but also into .NET Mobile Pages [1], XForm [7], and the representation of the small IA devices.

References

1. Microsoft Mobile Web Forms. http://samples.gotdotnet.com/mobilequickstart/(mgk4rd2jnyo1zm55tgnot02p)/Default.aspx.
2. Ricardo Devis, The Object-Oriented Page. June 1997. http://www.well.com/user/ritchie/oo.html
3. Xalan-Java 2.5.1. http://xml.apache.org/xalan-j/.
4. Tomihisa Kamada. Compact HTML for Small Information Appliances. Feb 1998. http://www.w3.org/TR/1998/NOTE-compactHTML-19980209.
5. Steven Pemberton, Daniel Austin, Jonny Axelsson, et al., XHTML(TM) 1.0 The Extensible HyperText Markup Language (Second Edition). W3C Recommendation. 1 August 2002. http://www.w3.org/TR/xhtml1.
6. David M. Geary. Graphic Java 2: mastering the JFC 3rd ed. Sun Microsystems. 1999.
7. Micah Dubinko, Leigh L. Klotz, Jr., Roland Merrick, et al. XForms 1.0. W3C Recommendation. October 2003. http://www.w3.org/TR/xforms/.
8. Marc Abrams, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams, Jonathan E. Shuster. UIML: An Appliance-Independent XML User Interface Language. http://www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html.
9. Neil Deakin. XUL Tutorial , July 2002. http://www.xulplanet.com/tutorials/xultu/.
10. Roland A. Merrick and et. al. AUIML: An XML Vocabulary for Describing User Interfaces. May 2001. http://www.belchi.be/download/merrick.pdf.
11. Angel Puerta and Jacob Eisenstein. XIML: A Common Representation for Interaction Data. Proceedings of the Sixth Intelligent User Interfaces Conference (IUI 2002). San Francisco, California, USA. January 2002.
12. Tzu-Han Kao, Sheng-Po Shen, Shyan-Ming Yuan, and Po-Wen Cheng. An XML-based Context-Aware Transformation Framework for Mobile Execution Environments. Lecture Notes in Computer Science (LNCS) of Springer-Verlag (APWeb 2003, Xian, China), Vol. 2642 / 2003, pp. 132 - 143.
13. David C. Fallside. XML Schema Part 0: Primer. W3C Recommendation. May 2001. http://www.w3.org/TR/xmlschema-0/.
14. WAP Forum. WMLScript Specification Version 25-Oct-2000, Sept 2001. http://www.wapforum.org/.