A Multi User-Interface Generation Plug-in for Visual Studio .NET

Chi-Han Kao Shyan-Ming Yuan

Department of Computer and Information Science, National Chiao Tung University,
1001 Ta Hsueh Rd., Hsinchu 300, Taiwan.
Email: gis92585@cis.nctu.edu.tw

Abstract

With the variety of the mobile devices, the specification between devices has become more and more different. From the point view of programmers, in order to execute the application with the same function on the different platform, programmers have to rewrite the program in another language. The repeated action of rewriting is meaning less and unnecessary for programmers. Therefore, we want to provide a toolkit for programmers. The finished program can be translated into the target language through the toolkit.

Keywords : plug-in; user interface; transformation；

摘要

隨著手機的多樣化，各種設備間的規格也漸趨差異。就程式設計師而言，相同功能的程式，為了在不同平台上執行，便必須以不同的程式語言重新撰寫。這樣的行為對於程式開發者而言，是重覆而不必要的。因此，我們希望提供一個開發環境，具備有轉換語言的功能，將撰寫好的程式，轉換成特定語言的版本。

關鍵字：外掛程式，使用者介面，語言轉換

I. Introduction

As the mobile devices are developed vigorously in recent years, there are thousands of devices have been published in the world. Therefore, many mobile applications are coded in many different programming languages. There is a problem for the people who develop the mobile application. If they want to operate mobile applications with same function in various mobile execution environments, they have to edit various versions of program by using different programming languages.

To save the problem, the member of our lab – Shen, had proposed a toolkit named “XML-based Mobile Application Development Kit” [7]. The toolkit is designed in the concept “Write Once, Run Anywhere”, the function of this toolkit is translating the document to the target language users assigned, and the translated document can be executed in the mobile environment [3]. The document is written in the language we defined, and the translated document is written in the physical language known by public. From now on, programmers only have to write the program once, and our toolkit can translate the program to various

languages in the public by using our toolkit.

Although the toolkit proposed by shen saves the problem of various mobile execution environments, the environment for users to edit the document in the language we defined is in the text mode. It is not convenient for users to edit the document. We want users can edit the document in the WYSIWYG [6] development environment. When users develop the User Interface (UI) of the mobile application, they can get the layout of the User Interface immediately from the development environment.

At first, we want to develop the development environment ourselves own. But there is a problem about this development environment. Users who first operate the development environment have to learn the skills about the environment. It is not convenient for users.

In the next step, we find that there are many integrated development environment (IDE) have been published in the world. In normal, programmers usually use the IDE to develop the program. This situation excites our thought to embed our toolkit into the IDE known by public. In this way, programmers can use the IDE they are familiar to edit the document, and our toolkit can translate the document to the target language they assigned.

The rest of this thesis is organized as follows. In the next chapter, we review a few background and related works. The characteristics of our implementation are provided in II. Part III, we will explain how to implement our development in detail. After that, we will compare our development with the other programs which have the function are similar to us in part IV. Final is the conclusions and future works.

II. Ideas of Design

The toolkit that we developed is used to translate the program from one language to another. For example, according to Figure 1, we can edit the program in J2ME, and we can get the program written in WML[9] through our toolkit[1].

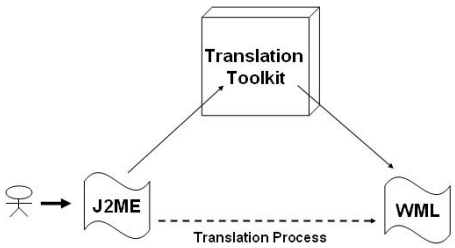

Figure 1: example of translating program from J2ME to WML

This is the usage of our toolkit. In the toolkit, we define a language, which was named Pervasive User-interface Markup Language (PUML). We will introduce the language “PUML” roughly in this chapter, and we will detail the language in Chapter Four. PUML is the intermediary language for translating; we can save the developing time of writing the program with the same function but in the different language. It’s convenient to get the program in the language we want through PUML. And if we edit the program by PUML directly, we can save the translating time.

A. Ideas of Design

As we mentioned above, in order to make user convenient use PUML to edit the program, there are some characteristic about our toolkit

(embed PUML in IDE) described as follow:

- Easy Install Easy Uninstall
- Tight Combination
- FamiliarDevelop Environment
- Click and Generation
- Easy Drag-and-Drop
- Visional Set Property

B. The Process of Embedding

The embedding process of our toolkit into Visual Studio .NET can separate into two parts. One is the front-end environment combination. The other is binding the document that front-end environment generated with the back-end translation engine. In the front-end combination with Visual Studio .NET[5], we have to detect all the UI controls that users put on the design form. The action "detect UI controls" contains several things, we describe as follow:

The number of controls:

When users use the IDE (in our implementation, the target IDE is Visual Studio .NET) to edit the User Interface of the application, they often modify the User Interface layout of the application. At the moment user triggers the IDE, ask to generate PUML code according to the controls they design, our toolkit have to know the modification about the controls. For example, user may add a button on the design form of the IDE, when generating the PUML code, toolkit have to detect that there is new control added to the form, and generate the PUML code of the button in the PUML document.

The modification of the controls:

When design the application layout, user not only modifies the controls, but also the attributes of the controls. After detecting the modification of the controls, toolkit has to generate the attributes of each control.

After users finish the UI design of the PUML document, the next step is translating the PUML file to the language they wants. In the process of translation, the machine that users operate must to be connected with the Internet. Nowadays, it is not difficult for computers to be connected with Internet; therefore; we design our translation engine as Web Services, and the machine can send the document to the translation engine through Internet.

The reason we decide to make our translation engine as Web Services is that we can simplify the process of embedding our toolkit into the IDE. When users need the service of translation, they send the document they edit to the engine through the internet. After translation engine gets the document sanded by users, the engine translates the document to the language that the user wants. Finally, the engine sends the document been translated to the machine that the user operate, and the IDE displays the document sanded back though the emulator. Users can see the result from the emulator, decide whether the User Interface is they want or not by the result displayed by the emulator. Figure 2 is an example about the translation process.

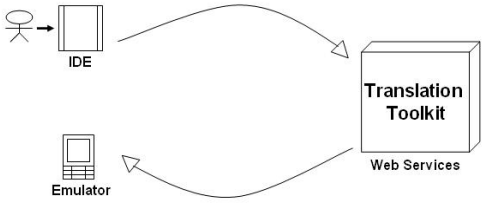

Figure 2: an example of the translation process

There are five steps of the translation; we describe each action of the steps that users take as follow:

Users edit the document with the IDE - Visual Studio .NET. The IDE that users use has been embedded the translation toolkit we developed, and the way to use the IDE to edit the User Interface of the application is the same as usual.

After users design the User Interface of their program by putting controls on the design form of the IDE, our toolkit can generate the code of the PUML language according to the controls that users put automatically. In this step, users can edit what action of the control is when it is triggered.

When users finish their program, they can click the button we build in the IDE to translate the program. After users click the button, our toolkit will transmit the PUML document to the translation engine through Internet.

Translation engine gets the document form the toolkit embedded in the IDE, it translate the document to the language users want, and send the translated document to the toolkit back.

When the toolkit receives the translated document, it will display the result to users through the emulator.

In this part, we have described the combination process of our toolkit. First of all, we need to know what controls users have dragged on the design form, our toolkit have to generate the document corresponding to the controls on the design form of the IDE. The generated document includes the tags of the controls and the attributes of the controls that users set in the properties window.

In the following, we will introduce the language “PUML” at first. After the part of “PUML”, we will explain the implementation how we embed our toolkit into Visual Studio .NET [4], and transmit the document to the back-end translation engine.

III. Implementation

In this part, we will introduce the core implementation – Pervasive User-interface Markup Language. After PUML is the section of the combination of our toolkit and the IDE.

A. Introduction of Pervasive User-interface Markup Language

PUML is a language for describing the User Interface of the application. When we divide mobile application into two parts: User Interface and Logic Computing, the responsibility of PUML is to describe the User Interface.

PUML is a XML-based language for describing the User Interface. PUML have to follow all the rules in XML, and inherit all the

characteristics about XML. We get some ideas from XUL, UIML, and WML to design PUML. Besides the part we learn from other languages described above, PUML also contains our own ideas of design. These ideas make PUML more suitable for rendering the User Interface in the mobile environment. We will introduce these ideas in later of this chapter.

After a PUML document has been translated by the translation engine, we can get the program written in the other language, and the program can be executed in the mobile environment. That is to say, the User Interface described in the PUML document will be translated to the target language. When the translated document is executed in the mobile environment, it will render the layout of the User Interface in the PUML document.

B. The document transmission to the back-end toolkit

Because the transmission to the target language is based on the PUML document, we have to send the PUML document to the translation toolkit. The target language we provide can be divided into two parts: One is the web-based language; the other is the local-side language.

The web-based languages supported by our translation toolkit are XHTML MP and WML. This function is reused from our colleague; we deploy the toolkit as the web service. The requirement of using our web-base translation is the internet connection. After users finish the User Interface design, they press the button to transmit the PUML document to the toolkit we deployed on the internet. The toolkit can translates the PUML document to the target language we supported, and transmits the result back to users.

The other part is the local-side languages. The term “local-side” means the entire program is executed on the machine. The target languages we support in this part is C#[5] and J2ME[8]. The requirement of this function is the XSLT[10] style sheet. Through the style sheet, users can get the User Interface in their assigned language. The translation in this part needs no network connection. Users only have to download the style sheet, put the style sheet in the machine. In the process of the translation, our toolkit will generate the target language through the style sheet automatically.

IV. Conclusion and Future Work

A. Conclusion

In the above part, we have introduced all the things about this paper. First of all, we have described the translation toolkit developed by our lab colleague. The motivation of this toolkit is similar with the concept of Java. The author wants the concept “Write Once, Run Anywhere” being realized in the mobile environment. The usage of the toolkit is to translate the document written in the language we defined, and users can assign the target language they want to translate the document. The translated document can be executed in the mobile environment according to the language users assign. This is the usage of our toolkit.

For users no matter in the process of operation or the installation. The method of

operating the IDE is the same as before. We embed our translation toolkit into the IDE, users operate the IDE as usual. The IDE embedded with our toolkit can generate the PUML code according to the dragged controls on the form. After users have installed our plug-in, they can convenience edit the document in our language, and reduce the repeat action of developing.

B. Future Work

We wish the target IDE that embedded with our toolkit could be supported more widely and the developing works can be more easily, so we propose a few future works that enhance our plug-in more complete.

- Increase the number of IDEs:
- Improve the combination of the IDE:

References

[1] Anna Maria Jankowska, and Andrzej Dabkowski, "Content Adaptation TagLibrary – An Approach for User Interface Adaptation for Different Devices", European University Viadrina, Chair of Business Informatics

[2] Extensible Markup Language, W3C http://www.w3.org/XML/

[3] Kris Luyten and Karin Coninx, "An XML-based runtime user interface description language for mobile computing devices", 2001 Springer-Verlag

[4] Les Smith, “Writing Add-Ins for Visual Studio .NET” , APress 200

[5] Microsoft Visual Studio Developer Center http://msdn.microsoft.com/vstudio/

[6] Ono, K.; Koyanagi, T.; Abe, M.; Hori, M.; Applications and the Internet, 2002. (SAINT 2002). Proceedings. 2002 Symposium on 28 Jan.-1 Feb. 2002 Page(s):150 - 159

[7] Sheng-Po Shen, Shyan-Ming Yuan, “XML-based Mobile Application Development Framework”, 國立交通大學，電資學院碩士班論文，民國 93 年 6 月

[8] Sun Microsystems Inc, Java 2 Platform Micro Edition, http://java.sun.com/j2me/

[9] WAP Forum, OMA, Open Mobile Alliance http://www.wapforum.org/index.htm

[10] XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 , November 1999 , http://www.w3.org/TR/xslt