UI Design Patterns Generator for Pervasive Device[1]

Deng-Jyi Chen*, Ming-Jyh Tsai*, Chung-Yuan Huang**
*) Department of Computer Science
National Chiao Tung University
1001 Ta Hsueh Road, Hsinchu, Taiwan, 300, ROC
**) Department of Computer Science and Information Engineering
Chang Gung University
259 Wen-Hwa 1st Road, Tao-Yuan, Taiwan, 333, ROC
djchen@csie.nctu.edu.tw, mjtsai@csie.nctu.edu.tw, gis89802@gmail.com

Abstract- It has been shown that the major effort spent on the design and implementation of the system software for pervasive devices (or handset device such as cellular phone) is the user interfaces (UI). If UI can be developed in a short time, it can be a great help to reduce development time for application software system. Therefore, many researchers in software engineering area have been seeking better solutions to aid UI designers to crate UI.

In this paper, we propose a UI design patterns generator to generate UI for pervasive device. Specifically, a UI design patterns generator is proposed for UI designers to easily and quickly create the UI patterns for cellular phone. Furthermore, the developed UI patterns can be fine tuned with a visual UI authoring tool to generate the UI prototype of the target cellular phone system under consideration. The generated UI prototype is then as a guider for the program generator to glue the software design framework and associated functions together to produce the target application system code. Finally, In order to demonstrate the feasibility and applicability of the proposed UI design patterns generator, a simulator is designed and implemented for carrying out the software simulation.

The benefit of the proposed approach is that it enables UI designers to generate UI prototype easily and quickly, and produces automatically the target UI program without writing any textual code. Thus the proposed approach is very suitable for the UI designers (nonprogrammers). In addition, the developed UI patterns can be reused by UI designers to generate target UI prototype. Therefore, it can reduce development time.

Key-words: User interface, UI design patterns generator, Generic UI template, UI prototype, Visual UI authoring tool, Program generator, and Simulator.

1. Introduction

Developing application software with sophisticated and elegance user interface is a complex and time-consuming task. Studies have shown that near 80 percent of the code of applications is devoted to the user interface (UI) [8], and that about 50 percent of the implementation time is devoted to implementing the UI portion [14, 15]. Thus, UI plays a significant role in the development of application software. For years, researches in software engineering area have been seeking better solutions to aid system developers to build UI. In general, UI designers and UI programmers have much iteration process to go in order to meet the target UI requirement specification.

To reduce the UI designer's heavy workload in this long iterative process, we propose a UI design patterns generator for UI designers to easily design and author the UI pattern. These created UI patterns can be reused easily to create the new UI prototype for a new pervasive device. To avoid UI programmer's tedious workload in this long iterative process, the UI design patterns generator can automatically generate the UI program according to the UI pattern. Based on this innovative approach, the UI designer alone can complete the UI design and implementation without bothering the UI programmers since the UI program will be generated automatically by using the UI design patterns generator. Thus, the UI requirement and functional requirement can be separated.

In [6], a visual based software construction model has been proposed for supporting this software construction model. Here, we propose a UI design patterns generator to generate the UI prototype of mobile phone. The proposed UI design patterns generator has integrated into the visual based software construction model. In order to demonstrate the feasibility and applicability of the proposed UI design patterns generator, a simulator is designed and implemented for carrying out the software simulation.

The benefit of the proposed approach is that it enables UI designers to generate UI prototype easily and quickly, and generate the target UI program without writing any textual code. Thus the proposed approach is very suitable for UI designers. In addition, the created UI pattern can be reused by UI designer to generate target UI prototype. Therefore, it can reduce UI application system's development time.

[1] This research was supported by the National Science Council of Taiwan, the CAISER of National Chiao-Tung University, and the Bestwise International computing Company of Taiwan.

2. Related Work

There are a few specific tools available for creating the UI prototype for pervasive devices based on the Visual authoring approach. The most common tools found in industrial sectors for the UI development of pervasive devices are eMbedded Visual C++, Rapid, and Symbian's Eclipse tools.

Microsoft Windows Mobile 2003 Second Edition (mobile phone operating system) not only provides a complete developing tool on this platform such as GUI framework and Visual developing environment, eMbedded Visual C++ [12, 18], but also a simulator to verify the executing result from the machine. On creating the UI, it offers an authoring environment for limited functions such as designing pull down menus. If users want to create a more complicated design such as inserting a picture on the screen, they need to write the script program.

Rapid is a crossed platform system, developed by e-SIM, for embedded system design, implementation, and simulating [18]. It provides object layout during UI development for adding new objects onto the screen or defining the position of objects. It also provides an object editor to modify screen of object, and a simulator to verify executing result.

When one uses Rapid tool to develop the UI of mobile phone, he needs to clearly define the object on screen, all of the states in the entire system, and conditions for transferring in each state. In order to effectively use the Rapid tool to design and implement the target UI prototype for the system, users need to understand all the details of machine state and operations in addition to designing the UI screen. Thus, it is probably only at programming level that users can sort out these details. The Rapid tool is therefore suitable for UI programmers to use (not for UI designers).

Eclipse tools appear for Symbian C++ developers using the open source integrated development environment from Eclipse foundation. Symbian C++ runs under Symbian OS [16, 17, 18], its operation is similar as eMbedded Visual C++.

Consequently, the following problems will be faced when using the above mentioned UI developing tools:

1) Writing textual program is still inevitable.
2) UI Programmers have to work with UI designer in order to modify the changes of UI. (UI designer alone cannot complete the UI task)
3) A long iterative process between UI designers and UI programmers cannot be avoided while using the current approaches to design and implement the UI of the pervasive devices.
4) No UI patterns generator supported in current UI developing tools. Thus, a UI designer could not use it to generate various UI patterns for future reuse

3. Framework of UI Design Patterns Generator and Visual Software Construction Model

The framework (Figure 1) includes the following major parts: (1) *The UI design patterns generator,* which is used to create the UI patterns; (2) *The visual UI authoring tool,* which is used to modify or fine tune UI patterns (created by UI design patterns generator) to generate UI prototype; (3) *The Program Generator,* which is used to produce the target application system code according to the UI prototype generated; and (4) *The simulator,* which is used for software simulation.

UI designer: A person who is responsible for the UI design of the software system. He or she can use UI design patterns generator to create a UI pattern and use the visual UI authoring tool to modify or fine tune the UI pattern.

Generic UI Template: It is consisted of UI structure template, UI layout template, and UI style template.

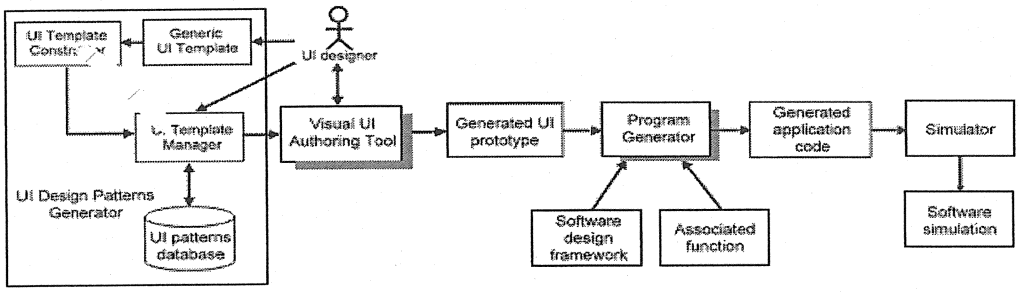

Figure 1 Framework of UI design patterns generator and VBSC model

UI Template constructor: A tool for making new UI Templates. It instantiates the Generic UI template to construct the UI pattern and then stores it into UI patterns database through the UI template manager.

UI Template manager: A database management system for managing UI patterns; it provides an interface for adding or deleting UI patterns as well as for retrieving an existing UI pattern.

UI patterns database: A database for storing UI patterns.

Visual UI Authoring Tool: It can be considered as a multimedia authoring tool.

Target UI Prototype: The target UI prototype which is generated by UI design patterns generator and fine tune by the visual UI authoring tool.

Visual Program Generator: Binding software design framework and associated function with each UI component (defined in the generated target UI prototype) to produce the targeted application system code.

Software design framework: The architecture framework for the target application system. This framework is generated by generic software framework for pervasive devices which is not emphasized in this paper.

Associated function: Associated function developed based on application program interface (API) library function developed by the functional programmers

according to the hardware specification.

Simulator: It is used to simulate the functionalities of the produced target application system on the target cellular phone.

UI designers use the *UI design patterns generator* to construct an initial UI pattern, and then use the *visual UI authoring tool* to modify or fine tune the initial UI pattern to generate the UI prototype of target cellular phone system. The generated UI prototype is then as a guider for the *program generator*, the function binding system, to glue the software design framework and associated functions together to produce the target application system code. Finally, one uses the *simulator* to do software simulation.

4. Generating UI patterns Using UI Design Patterns Generator

We have discussed the framework of UI design patterns generator for pervasive devices in section 3. In this section, we discuss how to use the UI design patterns generator to generate the UI patterns for mobile phone. Specifically, a generic UI template is introduced and its corresponding UI template constructor is implemented to construct the UI patterns.

4.1 UI of Mobile Phone

When we operate a mobile phone, we will see the stand-by screen after power on the mobile phone. Then, there will be several functional buttons ready for pressing to initiate the desired function. To press a specific functional button, it takes us to the corresponding screen associated with the function. These functional buttons are usually organized into a tree style structure as shown in Figure 2 and consists of two basic elements 1) Node that represents a screen or a function and 2) Link that defines the relationship between screen and screen or screen and function.

A screen (or node), which is not a leaf, can be considered as a container (or scene) that may contains many actors (or UI components) including, text actor which provides function of text representation, icon actor which provides function of drawing representation, input box actor which allows user to input data during execution of a specific function such as pressing telephone number, and list box actor which represents function of multiple data. A link provides a binding among nodes and functions, and control information for a screen to another screen (node to node) navigation.

A screen or node at leaf level will be considered as a function. There are many common functions in most of the current mobile phones including the essential functions and value-added functions. These functions usually are implemented by API programmers.

Thus, the UI of mobile phone can be quite different if the UI navigation structure is different, the layout of actors in a scene is different, and the style of actors (leave node) is different.

4.2 Generic UI Template

After comparing the UI of different mobile phones, we find that there are some similarities of appearance of user interface when these mobile phones are made by the same manufacturer. Nevertheless, even though the mobile phone comes from different manufacturers, the structure of the UI also has some similarity. Therefore, we factored out the common parts from various UI structures, screen layouts, and actor styles to define the generic UI template. These will be defined as structure template, layout template, and style template.

4.2.1 Structure Template

The UI prototype is very dependent on the UI structure for screens to screens navigation. The UI structure, based on topological point of views, can be a ring (or circular list), tree, and ring of tree as shown in Figure 2.

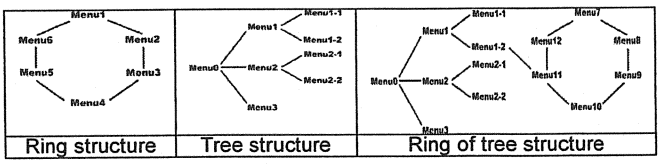

Figure 2 Various UI structures

The complete UI structure data is stored in the structure template, which includes actors in each scene, layout, and style. The layout and style are defined in layout template and style template. Thus, the structure template consisted of 1) the structure of all scenes and 2) actors information in scene.

4.2.2 Layout Template

The UI prototype is also sensitive to each actor's position in a scene. This positioning is called Layout. We could define layout template according to the layout information and then change the actor's position according to a different layout template. For example, in Layout1, the icons are at the top and text is at the bottom; in Layout2 the icons are placed at the bottom and text is placed at the top as is shown in Figure 3:

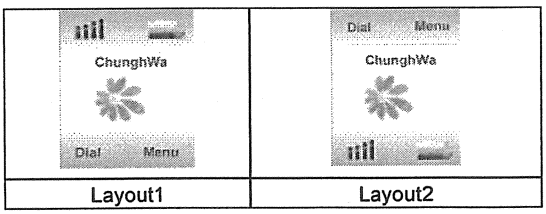

Figure 3 Two different Layouts

In general, the UI positioning arrangement is regular. Taking an example of tree structure of UI, the layout is almost the same in the same level of scene. We only needed to define few of them then we could describe layout of each scene in the entire UI completely. Therefore, we could use a layout template to create the same Layout for each scene under consideration in UI design of a mobile phone. The layout template consisted of layout data of each scene. Layout of scene is made of 1) name of Layout and 2) each actor's information in the scene.

4.2.3 Style Template

The other factor that affects the UI prototype of a mobile phone is the UI style. The UI style considers the style of actor's appearance. An actor's appearance is decided by its attributes in a scene. Even though it is the same actor, different attributes may produce different appearances.

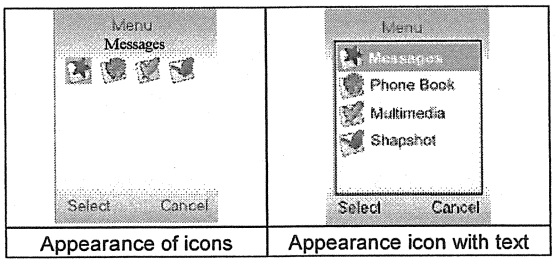

Figure 4 Two different styles

As shown in Figure 4, appearance based on icons (left hand side) could be changed to be appearance based on icon with text (right hand side) by changing its attributes. Therefore, style template is used to change each actor's appearance in the scene. The style template consisted of each actor's style data which is the relevant attributes of actor's appearance.

4.3 Generic UI Template for Pervasive Devices.

In previous subsections, we have defined three UI templates (structure template, layout template, and style template). After combination of these three templates, we obtain a generic UI Template. Furthermore, we use MVC models (model, view, control) [10] to implement a generic UI template for the UI pattern generation of the mobile phone device. The structure template produces model, layout template, and style template provide view. It could generate different UI prototype by different combination of these three templates. Structure template generates UI structure first, layout template offers layout data of actor on the scene, and then style template provides each actor's style. Let these three templates be denoted by a, b, and c, then one can produce a * b * c combinations of UI instantiations.

4.4 UI Patterns Generation Procedures Using UI Design Patterns Generator

How to use the UI design patterns generator to create different UI patterns is summarized below.

Step 1: Select a Generic UI Template: The UI designer needs to select the required type of structure template, layout template, and style template.

Step 2: Generate the complete UI structure from the chosen structure template

Step 3: Generate the desired layout based on the information in layout template for each scene (or node in the tree structure)

Step 3: Generate the desired style based on the information in style template for each actor (or UI button) on each screen (or scene)

Step 5: Repeat step 3 and step 4 till all the scenes and actors chosen from the structure template are defined.

The system continuously repeats the actions in step 3 and step 4 till the entire UI prototype required layout and style information has been filled into the chosen structure template. The system continuously repeats step 3 and step 4 till all scenes of the tree structure is completed.

After executing the above steps, we can construct the UI pattern for the handset device under consideration. Next, we can fine tune the created UI pattern by visual UI authoring tool to generate target UI prototype. Then, we use visual program generator to binding the associated function with UI component for producing the application system code. After complied, the produced object code can be executed on the simulator.

5. The UI and Application System Design and Construction Process of Pervasive Devices

According to framework of UI design patterns generator and VBSC model, we organized the system design and development process into following four stages:

5.1 UI Patterns Generation

The first stage is UI patterns generation. UI template constructor instantiates the generic UI template to construct the UI pattern, and then stores it into UI patterns database through UI template manager. By the end of this stage, we will obtain an initial version of the UI pattern for the pervasive device under consideration.

5.2 Visual UI Authoring

The stage is visual UI authoring. This stage consists of the following two steps:

1) UI editing: the UI designer uses the visual UI authoring tool to modify and fine tune the UI pattern to generate the target UI prototype.

2) The UI prototype preview: If the UI designer wants to view the results of the actual operation after the UI editing, the UI designer can use the *preview system* in the visual UI authoring tool to check whether the target UI prototype meets the user's UI requirements.

Eventually, the target UI prototype will be finalized by the end of this stage.

5.3 Code Generation

After the target UI prototype is finalized, the UI program will be generated and the binding of associated function code and the UI program will be performed by the code generator in this stage. The major components in this stage are consisted of the following three subsystems:

1) AP function and UI binder: If the target UI prototype meets the UI requirements, we use the *function binding system in the generator* to bind the associated functions to the target UI program generated by the visual UI authoring tool.

2) Application system code generator: When UI components (in the target UI program) have been bound

with the associated functions; it enters into this step to produce the target application system code by *program generator.*

3) Code optimizer: If the efficiency is not good during program execution, it enters into this step to adjust application system code until the efficiency is met.

Eventually, the target application system code for the pervasive device is generated by the end of this stage.

5.4 Simulation

After the above stages are completed, we receive the application program code of the target platform and enter the last simulation stage. In this stage, we have software and hardware simulation two phases:

1) Software simulation: We combine device module and application system code to do software simulation on laptop through the *simulator,* which verifies whether the generated application system works normally and meets the expectations.

2) Hardware emulation: When the software simulation is completed, we could download program to EPROM (Erasable Programmable Read-Only Memory) directly, and execute it on hardware (the target pervasive device).

6. Assessment of Using the Created UI Pattern to Generate Target UI Prototype

6.1 Purpose

The purpose of this experiment is to explore the efforts spent with the use and without the use of created UI patterns to produce the UI prototype for pervasive devices. Three application examples which contain six, twelve, and eighteen screens UI prototype (listed in the Appendix A) were used to make such a comparison. On average, there are about 7 UI actors for each screen.

6.2 Participants

Study participants were 39 students enrolled in a graduate software engineering course. The students' majors were computer science (29), electrical engineering (6), information management (2) and other computer-related departments (2). Those students who participated in the experiment all have rich programming experience (at least three years working experience in software programming related area).

6.3 Design

All the students were asked to be familiar with visual UI authoring tool, and use it to generate target UI prototype for a handset device. Then two weeks later, those students were asked to use the created UI pattern to do the same task. Appendix B shows the results of those two different approaches.

6.3.1 without the Use of Created UI Pattern to Generate Target UI Prototype

If there is no suitable UI pattern in the UI patterns database can be used directly, we have to use a visual UI authoring tool to generate target UI prototype.

6.3.2 Using the Created UI Pattern to Generate Target UI Prototype

If there is suitable created UI pattern, we can modify it into target UI prototype by a visual UI authoring tool. This modification includes (1) Replacing icon and text from one of the screens; (2) Deleting some screens; (3) Creating one screen to the created UI pattern.

6.4 Data and Analysis

6.4.1 Case 1: Development Time of Generating a Six-screen UI Prototype

Figure 5 presents the time spent to generate a six-screen UI prototype with the use of created UI pattern and without the use of created UI pattern. Based on the collected data, we found that the average time spent to generate this UI prototype with the use of created UI pattern is 9.92 minutes and the average time spent without the use of created UI pattern is 36.38 minutes. The development time without the use of created UI pattern is about 3.7 times [(36.38 / 9.92)] longer than with the use of created UI pattern. The 27-minute difference represents a time savings of 73%.

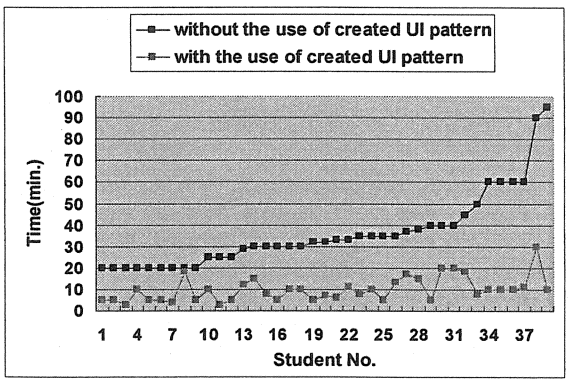

Figure 5 Time spent in generating a six-screen UI prototype

6.4.2 Case 2: Development Time of Generating a Twelve-screen UI Prototype

Figure 6 presents the time spent to generate a twelve-screen UI prototype with the use of created UI pattern and without the use of created UI pattern. Based on the collected data, we found that the average time spent to generate this UI prototype with the use of created UI pattern is 10.38 minutes and the average time spent without the use of created UI pattern is 54.10 minutes. The development time without the use of created UI pattern is about 5.2 times [(54.10 / 10.38)] longer than with the use of created UI pattern. The 44-minute difference represents a time savings of 81%.

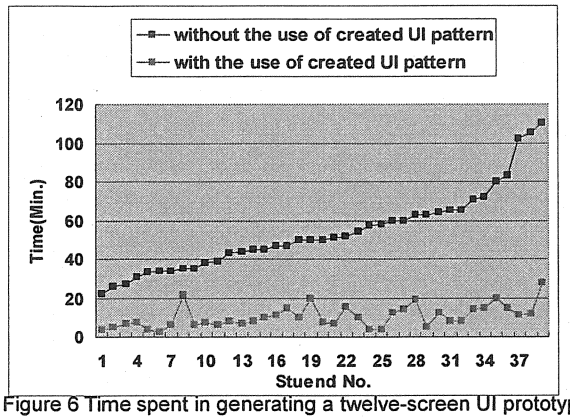

Figure 6 Time spent in generating a twelve-screen UI prototype

6.4.3 Case 3: Development Time of Generating an Eighteen-screen UI prototype

Figure 7 presents the time spent to generate an eighteen-screen UI prototype with the use of created UI pattern and without the use of created UI pattern. Based on the collected data, we found that the average time spent to generate this UI prototype with the use of created UI pattern is 10.85 minutes and the average time spent without the use of created UI pattern is 80.05 minutes. The development time without the use of created UI pattern is about 7.4 times [(80.05 / 10.85)] longer than with the use of created UI pattern. The 69-minute difference represents a time savings of 86%.

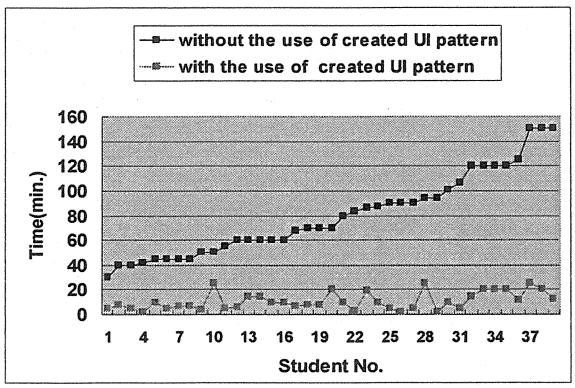

Figure 7 Time spent in generating an eighteen-screen UI prototype

Table 1 Average time spent in generating six, twelve, and eighteen screens UI prototype

Time spent / *UI prototype*	Average time spent with the use of created UI pattern (Minutes)	Average time spent without the use of created UI pattern (Minutes)	Time ratio between average time spent without the use of created UI pattern and average time spent with the use of created UI pattern
six screens	9.92	36.38	3.7
twelve screens	10.38	54.10	5.2
Eighteen screens	10.85	80.05	7.4

From Table 1 we can note that (refer to column 3) without the use of created UI pattern to generate UI prototype development time increases tremendously as the number of screens increased. But using the created UI pattern to generate UI prototype even as the number of screens increased (refer to column 2) the time increment never exceeds 1 minutes.

From Table 1, one can see the time ratio increments approximately 1.5 times each time as we add six screens.

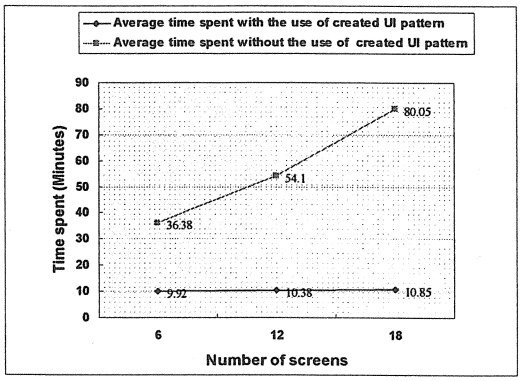

Figure 8 Average time spent with the use of created UI pattern and without the use of created UI pattern to generate target UI prototype

From Figure 8 we can note that the average time spent without the use of created UI pattern sharply raises. But using the created UI pattern remains smoothly. From the observation, we can foresee that as the number of screens increased then the time ratio between those two approaches will become wider.

7. Conclusions

The well designed UI allows user to operate the product conveniently. So, the design of UI becomes very important which can give customers different impression on the mobile phone and influence their decision to purchase it or not.

UI plays an important role during software development. However, it is also time-consuming for creating a good UI. If UI can be developed in a short time, it can be a great help to reduce development time for application software system. Therefore, many researchers in software engineering area have been seeking better solutions to aid UI designers to builder UI.

By the proposed approach, UI designers use the *UI design patterns generator* to create UI patterns and store them for future usage. Furthermore, it could be modified to be a custom-designed UI prototype by the *visual UI authoring tool*. Based on this innovative approach, the UI designers alone can complete the UI design and implementation without bothering the UI programmers since the UI program will be generated automatically by using the UI design patterns generator. With the assistance of the *Program Generator*, the UI component is equipped with associated function. We only need to bind the relevant system function with the UI component to generate the target application system code. Finally, we provide a *simulator* for software simulation.

The proposed generating UI for pervasive devices using UI design patterns generator has following characteristics.

1) Ideal for UI designer (nonprogrammer) to produce the UI prototype for the target application software system. An UI designer does not need to interact with UI programmers anymore because the UI design patterns generator and visual UI authoring tool will generate the target UI program automatically.

2) Rich UI pattern can be supported by the proposed UI design patterns generator to ease a UI designer to reuse to create target UI prototype. Hence, it can reduce time spent while developing application system.

Appendix A: Generating Target UI Prototype

A.1 Generating a Six-screen UI Prototype

At the first phrase, a six-screen UI prototype (Figure A-1) was generated which includes Stand by, Menu, Dial, Message, Address Book, and Settings. The actor layout of each screen is shown in Figure A-7 - A-9, A-14 - A-15, and A-24

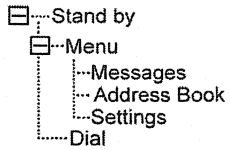

Figure A-1 A six-screen UI prototype

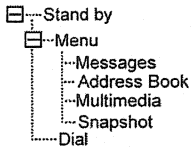

Figure A-2 The created UI pattern

A.1.1 Generating a Six-screen UI Prototype without the Use of Created UI Pattern

If there are no suitable created UI patterns in the UI patterns database, we have to create above mentioned screens by using visual UI authoring tool.

A.1.2 Using the Created UI Pattern to Generate a Six-screen UI prototype

We search for the created UI patterns from UI patterns database, and found there is one created UI pattern as shown in Figure A-2. By using the visual UI authoring tool, we can fine tune the created UI pattern to generate the six-screen UI prototype. The processes are listed below:

1). First, replace the image actor by and change the text actor "ChunghWa" to be "Mobile" of "Stand by" screen from the created UI pattern.
2). Then, delete the multimedia and snapshot screen provided by the created UI pattern
3). Last, create a new screen– "Setting", and add one list box in this screen, as shown in Figure A-15.

A.2 Generating a Twelve-screen UI prototype

At the second phrase, a twelve-screen UI prototype was generated. See Figure A-3.

A.2.1 Generating a Twelve-screen UI Prototype without the Use of Created UI Pattern

If there are no suitable created UI patterns, we have to create above mentioned screens (Figure A-7 – A-16, A-23, and A-24) by using a visual UI authoring tool.

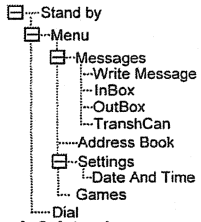

Figure A-3 A twelve-screen UI prototype

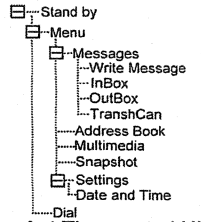

Figure A-4 The created UI pattern

A.2.2 Using the Created UI Pattern to Generate a Twelve-screen UI prototype

We search for the created UI pattern from UI patterns database, and found there is one created UI pattern for use as shown in Figure A-4. By using the visual UI authoring tool, we can fine tune the created UI pattern to generate the twelve-screen UI prototype. The processes are listed below:

1). First, replace the image actor by and change the text actor "ChunghWa" to be "Mobile" of "Stand by" screen from the created UI pattern.
2). Then, delete the multimedia and snapshot screen provided by the created UI pattern
3). Last, create a new screen – "Games", and add one list box in this screen, as shown in Figure A-23.

A.3 Generating an Eighteen-screen UI Prototype

Third, an eighteen-screen UI prototype was generated. See Figure A-5.

A.3.1 Generating an Eighteen-screen UI Prototype without the Use of Created UI Pattern

If there are no suitable created UI patterns in the UI patterns database, we have to create above mentioned screens (Figure A-7-A-24) by using the visual UI authoring tool.

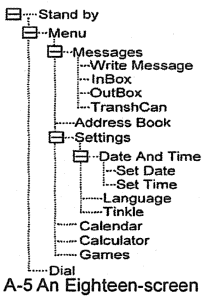

Figure A-5 An Eighteen-screen UI prototype

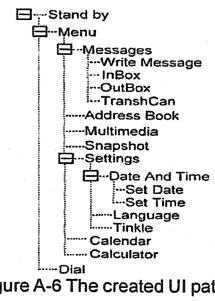

Figure A-6 The created UI pattern

A.3.2 Using the Created UI Pattern to Generate an Eighteen-screen UI prototype

We search for the created UI patterns from UI patterns database, and found there is one created UI pattern for use as shown in Figure A-6.

By using the visual UI authoring tool, we can fine tune the created UI pattern to generate the eighteen-screen UI prototype. The processes are listed below:

1). First, replace the image actor by and change the text actor "ChunghWa" to be "Mobile" of "Stand by" screen from the created UI pattern.
2). Then, delete the multimedia and snapshot screen provided by the created UI pattern.
3). Last, create a new screen – "Games", and add one

list box in this screen, as shown in Figure A-23.
The actor layout of each screen is shown below:

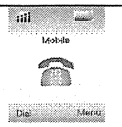			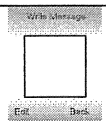	
Figure A-7 Screen of Stand by	Figure A-8 Screen of Menu	Figure A-9 Screen of Messages	Figure A-10 Screen of Write message	Figure A-11 Screen of Inbox
				
Figure A-12 Screen of Outbox	Figure A-13 Screen of Trashcan	Figure A-14 Screen of Address book	Figure A-15 Screen of Settings	Figure A-16 Screen of Date and Time
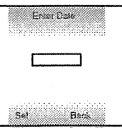	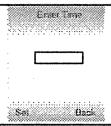	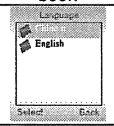	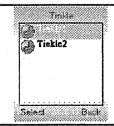	
Figure A-17 Screen of Setting date	Figure A-18 Screen of Setting time	Figure A-19 Screen of Language	Figure A-20 Screen of Tinkle	Figure A-21 Screen of Calendar
				
Figure A-22 Screen of Calculator	Figure A-23 Screen of Games	Figure A-24 Screen of Dial		

Appendix B:

Name:

Estimate items / Methods	Using the created UI pattern	without the use of created UI pattern
Time spent in generation a six-screen UI prototype (Minutes)		
Time spent in generation a twelve-screen UI prototype (Minutes)		
Time spent in generation an eighteen-screen UI prototype (Minutes)		

References:

[1] Ben Shneiderman, Creating creativity: user interfaces for supporting innovation, *ACM Transactions on Computer-Human Interaction*, Vol. 7, No. 1, March 2000, pp.114-138.

[2] Benjamin B. Bederson, Aaron Clamage, Mary P. Czerwinski, George G. Robertson, DateLens:A fisheye calendar interface for PDAs, *ACM Transactions on Computer-Human Interaction*, Vol. 11, No. 1, March, 2004, pp. 90-119.

[3] Browne, G.J. and Rogich M. B., An Empirical Investigation of User Requirements Elicitation: Comparing the Effectiveness of Prompting Techniques, *Journal of Management Information Systems*, Vol. 17, No. 4, 2001, pp. 223-249.

[4] Chen, D.J., Chen, W.C., Kavi, K.M., Visual requirement representation, *The Journal of Systems and Software 61 (2002)*, pp. 129-143.

[5] Chen, D.J., Huang, S.K., Interface of Reusable Software Components, *Journal of Object-Oriented Programming*, Vol. 5, No. 8, 1993, pp 42-53.

[6] Chen, D.J., Tsai, M. J., Dai, J.C., Chen, David TK, Visual Based Software Construction: Visual Requirement Authoring tool and Visual Program Generator," *Proceeding of the International Computer Symposium*, Dec 12-17, Taipei, Taiwan, 2004, pp.171-176.

[7] Christopher Frauenberger, Tony Stockman, Veronika Putz, Robert Holdrich, Mode Independent Interaction Pattern Design, *Proceeding of the Ninth International Conference on Information Visualization (IV'05)*, 2005, pp. 24-30.

[8] Ed Lee, User-Interface Development Tools, *IEEE Software*, Vol. 7, No. 3, May/June 1990, pp. 31-36.

[9] Eric Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Pattern*, Addison Wesley, Pearson, 1995.

[10] Glenn E. Krasner, Stephen T. Pope, A cookbook for using the Model-View Controller user interface paradigm in Smalltalk-80, *Journal of Object-Oriented Programming*, 1998, Vol. 1, No. 3, August/September 1998, pp. 26-49.

[11] Landay, J.A., Kaufmann, T.R., User Interface Issues in Mobile Computing, *Proceeding of the Fourth Workshop on Workstation Operating Systems*, October 14 – 15, Los Alamitos, CA., 1993, pp. 40-47.

[12] Michael Barr, *Programming Embedded Systems in C and C ++*, O'RELLY, 1999.

[13] Mitrovic, N. J. Royo, A. E. Mena. ADUS: Indirect Generation of User Interfaces on Wireless Devices, Database and Expert Systems Applications, *Proceeding of the 15th International Workshop on (DEXA'04)*, 2004, pp. 662-666.

[14] Myers, Brad A. User interface software tools, *ACM Transactions on Computer-Human Interaction*, Vol. 2, No. 1, March 1995, pp 64-103.

[15] Myers Brad, Hudson Scott E., Pausch Randy, Past, present, and future of user interface software tools, *ACM Transactions on Computer-Human Interaction*, Vol. 7, No. 1, March 2000, pp. 3-28.

[16] Richard Harrison, *Symbian OS C++ Mobile phones Vol. 2*, 2004, John Wiley.

[17] Steve Babin, *Developing Software for Symbian OS: An Introduction to Creating Smart phone Applications in C++*, John Wiley, 2005.

[18] Tammy Noergaard, *Embedded Systems Architecture, First Edition: A Comprehensive Guide for Engineers and Programmers*, Newnes, 2005.