RUCID: Rapid Usable Consistent Interaction Design Patterns-Based Mobile Phone UI Design Library, Process and Tool

Avinash Raj[1] and Vihari Komaragiri[2]

[1] Toronto, Canada
avinash.raj@hotmail.com
[2] Bangalore, India
vihari@gmail.com

Abstract. This paper is based on a research effort at Kyocera Wireless, India that aimed to overcome the limitations in the mobile phone design process, by giving designers an improved design and specification tool and helping them deal routinely with some of the more rooted constraints of phone design. The tool extends the idea of templates from simple visual elements, to more abstract design components. It adds further value to this modularization of design, by taking an approach of extensive and ever-growing library of patterns to define and refine these components. The components cover most of the low- to medium-level building blocks of design. They are specified in the library as a tuple(patterns) of <design problem, design solution, context, constraints> each at the different level of hierarchy. The components are visually represented using standardized shapes with placeholder and help text and are made available as part of the design work surface of a visual prototyping tool such as MS Visio or Adobe Fireworks.

Keywords: Mobile phone UI design, patterns, architecture, design process, lib.

1 Introduction

Modern mobile phones give its Users many features: voice and data calls, text messaging, personal information management (phonebook and calendar), WAP browsing, games, etc. All these features are packaged into a handset with a small screen and 12/16 key special purpose keypad. Each new release of the phone builds additional features on top of the existing ones and there is a constant race among phone vendors to put attractive phones out into the market faster and cheaper. Among the newer mobile phones deluging the markets most have their own separate interaction paradigms. It is important to find out a uniform set of interaction paradigms to avoid a time-consuming and expensive approach, to avoid re-writing each application for each mobile phone. The Users should not be forced to undergo the grueling process of learning and unlearning these interaction styles when using a

J.A. Jacko (Ed.): Human-Computer Interaction, Part I, HCII 2009, LNCS 5610, pp. 677–686, 2009.

same set of application over multiple mobile phones of either the same vendor or, in fact, even of different vendors.

1.1 Current Mobile Phone Interaction Design Process

Though design guideline documents deliver coherence among User Interfaces, they cannot be used effectively to communicate how different components of the design will work together and how Users will interact with them. In addition, guidelines can get obsolete or ignored very soon in the fast developing world of mobile phones. Furthermore, the interaction designer is limited to the task of specifying the design that a software engineer then implements it in an embedded software development environment that is notorious for its lack of sophisticated APIs for UI creation. This indirection and the limitations inherent in the development environment, also mean that a lot of design intent and time could be lost in translation. There needs to be a way to put design implementation in the hands of interaction designers, and this needs to be done in a "backward" compatible manner. Even when new technologies like FlashLite, uiOne become the platforms of UI development, there will still be some phones that require UI development in native code. So the solution will have to support both styles of interaction design and implementation.

While phone vendors have started adopting a platform approach to software development, adding incremental features to existing code bases and builds, this has the bug/feature of perpetuating design from older phones, whether good or bad. There is also no easy way of upgrading a design element for greater usability because it is difficult to trace a design element across various features where it is used. A solution to this problem could be ensure that interaction design is modular to the extent possible and utilizes design elements in a consistent, traceable manner.

1.2 Proposal to Solve Usability, Consistency and Time-to-Market Constraints

The aim of this research is to provide a tool where interaction designers can choose from the pre-packaged design element library and use the appropriate element by mapping the usability constraints and context of the pattern with the needs and context of the feature being designed. Presented in the form of a Microsoft Visio template-based prototype tool, these patterns can be easily used by the designers.

After the usual steps of analyzing the design problem, identifying the User goals and then breaking them down into tasks, the paper proposes a change to the design process. Instead of trying to sketch the design from scratch from that point onwards, the designer simply uses the design tool and its template library to look for and use design modules that already exist. For the part of the design that does not yet exists, the designer then builds newer tasks and flows from existing building-block objects. The designer then adds these newer creations as potential candidates to the pattern template library to be verified for usability, to be incorporated and then used by other designers in creating other features. The designer achieves speed and design consistency with this approach. The modular pattern library allows for reuse across designers and design teams, and for usability refinement, design evolution and for backward compatibility with changes as the product evolves.

2 RUCID Basics

In this section, we present a novel formulation of mobile phone interaction design architecture and build on this framework using the patterns based approach inspired by Christopher Alexander [2]. Some samples of mobile phone interaction design patterns are presented here to illustrate the concept.

We draw from the work of Alan Cooper [1] to derive interaction design architecture for mobile phones. The mobile phone has many input Triggers (typically the 12-key keypad, plus five-way navigation keys and so on). The context of use of the mobile phone is much different from the mouse and hence its interaction design structure is quite different from that of a computer mouse. There is a need for having a Primitive action for example a key press achieve not just generic input/output or application specific commands but directly address User goals. We address this in our model as follows: The User's goals in using an application can be broken down into some generic tasks common across applications. These generic tasks precede and succeed specific tasks called into existence by the needs catered to by the feature being designed. For example, "starting an application or closing it" are typically generic tasks; "playing a music track" or "composing an SMS message" are feature specific tasks.

Generic tasks are made up of Flows of input interaction in conjunction with the output – actions, symbols, graphics, and other feedback information – expressed on the screen (and speakers, vibrations etc.) of a mobile phone. The Flows are represented as Idioms to left hand side of the Alan Cooper inverted pyramid, while the output is represented on the right hand side of the inverted pyramid. The output shown on screen can be further divided into information, widgets and graphics.

A Flow can be thought of as a sub-task or a sequence of Primitive and/or Compound actions that results in an application specific function to be executed. A Compound action in turn constitutes a sequence of Primitive User actions and phone reactions that achieve a User's sub-objective. In a typical mobile phone design, a Flow that achieves a User objective, or a Compound action that achieves a sub-objective, may simply consist of one Primitive action for example; press and hold of hash key can actually achieve a User goal of locking the phone. This User goal can also be achieved by accessing settings from the menu, choosing the keypad lock menu option and then enabling the keypad lock option. The architecture (Fig.1) of interaction design in mobile phones is at the heart of our pattern exploration. It anchors our search for interaction design patterns in mobile phones and also provides means to organize, link and document them. Since this model also articulates the typical top down process of design, it lends itself to very practical application as evidenced by the prototype tool that we created.

In the following pages we look at sample patterns generated at the Widget, Primitive, Compound and Flow level, one sample each. The Compound, Primitive and Flow patterns that follow are tailored to the tool rather than the Alexandrian pattern. There were altogether 23 Primitive patterns, 3 Compound patterns, 15 Flow and 29 Widget patterns that were captured during the course of this research.

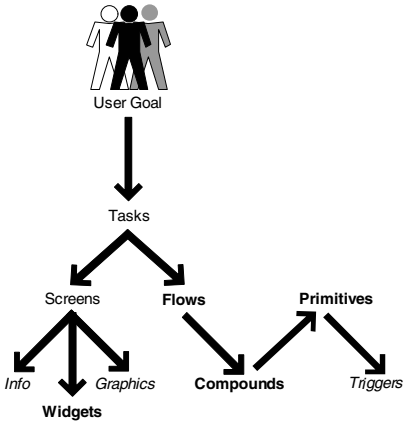


Fig. 1. "A New Mobile UI Design Architecture Model" that details out the hierarchy of design levels, starting from "User goals" all the way to " Action-Triggers"

Table 1. Sample Widget pattern (Soft key Window)

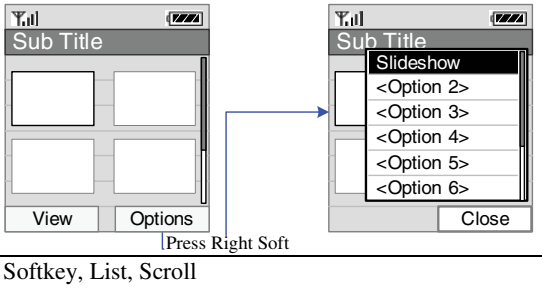
Problem	User needs to access additional functions that can be performed on a screen.
Context	For a given Screen, a User has more number of actions possible than the maximum number of Soft keys.
Solution	One of the options that can be accessed through a softkey can provide gateway to multiple options. The User can move to these options and select a desired one.
Rationale	A limited number of Softkeys can be displayed at any one time. A dedicated key cannot be assigned to each option because: 1) The options and their number keep changing depending on the screen. 2) The surface area of mobile is small and limited. Using a single key to access a variable sized list allows for any number of items to be accommodated.
Examples	
Related Patterns	Softkey, List, Scroll

Table 2. Sample Primitive Pattern (Type a Digit)

Problem	**Solution**	**Constraint**
User wants to type a digit	[1]Press a Number key [2]Press & Hold a Number key until the digit appears on the screen [3]Speak the digit [4]Press Navigation key (up/down) to change digits	[1]User is in Numbers only mode. There is a clear mapping between what User presses on the keypad and what appears on the screen. [2]User is in Normal Alpha or Rapid Entry mode. Typing a digit in this mode may not be intuitive but it is easy to learn. [3]In current Phones it's only used as feedback. [4]This can be used when input allows a small Numeric range (date, year, month etc.), otherwise its time consuming.
Examples	•Press a Number key to type digit in the Numbers only mode • Press & Hold a Number key until the digit appears on the screen in Normal Alpha or Rapid Entry mode • In Voice Dial the digits are typed as one speaks • In Set Alarm Press Navigation key (up/down) to change hours, minutes	
Related Patterns	c1. Type a string	

Table 3. Sample Compound Pattern (Type a String)

Problem	**Solution**	**Constraint**	**Child Patterns**	**Example**
[1]User wants to type a Numeric string	[1] Press Number keys, Navigate, Delete	[1]User is in Numbers only mode	p1.Type a Digit, p3.Delete Character to the left of cursor, p10. Navigate left/right	p1.1,p3.1,p10.1
[2],[3]User wants to type a Alphabetic string	[2]Press a key (2-9) once for the first alphabet, twice for the second alphabet, and so on, Navigate, Delete; [3]Press0Next key	[2]User is in Normal Alpha mode; [3]User is in Rapid Entry mode. Rapid Entry mode checks its dictionary of common words and guesses at the word User is trying to spell.	p2. Type an Alphabet, p3.Delete Character to the left of cursor, p10. Navigate left/right; p2.Type an Alphabet	p2.1, p3.1, p10.1; p2.2
[4]User wants to type a Alphanumeric string	[4]Press & Release keys for alphabets & Press & Hold keys for Numbers, Navigate, Delete	[4]User is in Normal Alpha mode.	p1. Type a Digit, p2.Type an Alphabet, p3.Delete Character to the left of cursor, p10. Navigate left/right	p1.2, p2.1, p3.1, p10.1

Table 4. Sample Flow Pattern (Initiate)

Problem	User needs to start an Application		
Solution	**Constraint**	**Child Patterns**	**Example**
[1]Go through the Menu hierarchy	[1]This is most simple & generic, though most time consuming way.	p24. Access additional screen functionality/ Options, c2.Scroll & Select/p7.2 Select menu items by pressing digits on Num keys	p24.2, c2/P7.2
[2]Use Shortcuts	[2]User is in the Standby mode. Using keypad Shortcuts is faster than using Menus. But it's not possible to have shortcut for every Application.	p18.Quick access (in Standby)	p18.1
[3]Accept Alert	[3]When the User accepts an alert it takes her to the respective Application. This is applicable to very few Applications & is not a direct User action.	p24.Access additional screen functionality/ Options	p24.2
[4]Initiate an Application through another	[4]User is already in an Application. Through Softkey Window the User initiates another Application. Such Applications generally compliment each other.	p24.Access additional screen functionality/ Options, c2.Scroll & Select	p24.1, c2.1
[5]Type a Contact num, then choose to initiate Call or Send message	[5]User is in standby. Only applicable to Messaging and Calling.	c1, (p12.Create outgoing call)/ (p24.Access additional screen functionality/ Options, c2.Scroll & Select)	c1, (p12.1) /(p24.1, c2.1)

3 The Prototype

In this section, the part of the problem statement for this research paper is addressed - the design and development of a tool that interaction designers can use to perform rapid, usable, consistent interaction design.

The design, details and use of the tool for mobile phone feature design is outlined. The tool is based on the pattern-based UI Design Library sampled in the previous section. In order to make efficient use of the patterns library however a platform is needed. Visio is a diagramming program that can help to create business and technical diagrams that document and organize complex ideas, processes, and systems. Visio seemed like a suitable platform for using the captured pattern for various reasons:

1. Shape libraries can be easily created with Visio tools.
2. New template can be created from a drawing file or from an existing template by assembling diagrams by dragging predefined shapes.

3. Reviewers' comments and changes to shapes can be tracked using the review mode.
4. Visio can be integrated with Microsoft Office Excel, Microsoft Office Word, Microsoft, and other formats.

Once we identified patterns at each level (Primitive, Compound, Flow, Widget), Visio templates were created for each level of the mobile phone architecture. We built a design vocabulary. Higher levels templates for example Screen could be constructed from lower levels like Widgets. The application will facilitate in effective and efficient designing of Screens and task flows of newer applications by just dragging and dropping existing UIs and Interactions. Revision and iterations are also easier, whenever a shape or a whole template is added to the tool.

The tool follows hierarchical method of design on which our research is based on. If we take an application, application has some goals. Theses goals can be broken down into tasks. Task can be broken down into UIs (Widgets, Screen) and Interactions (Flows, Compound, and Primitives). Thus the whole application can be constructed using this tool.

3.1 Sample Application (Slideshow) to Explain the Prototype

User Goal: View Camera Picture In slideshow
Tasks:

1. Launching Slideshow from image browser
2. Pausing/Continue the Slideshow
3. Skip to "Next/Previous Image"
4. Set Transition duration
5. Enable/Disable Looping
6. Exit Slideshow

We take 1st Task i.e. "Launching Slideshow from image browser" and try to construct it through our prototype. This is an instance of Initiate Flow. If it was available we would just need to drag the shape from Intimate template and put customized labels for the Screens & Flows. However let's assume that the Flow was not available and therefore try to construct it from scratch from the available Widgets and Actions.

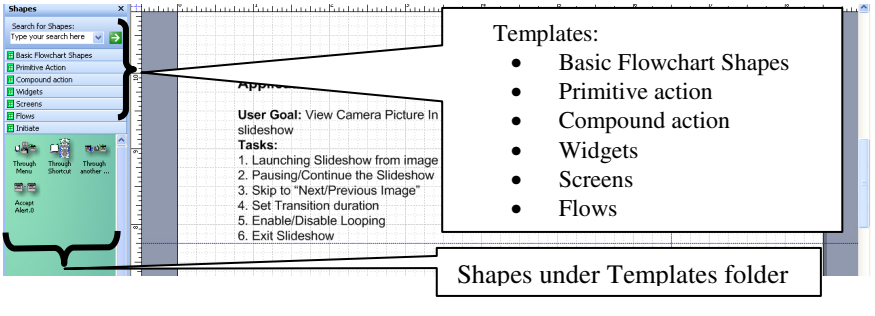
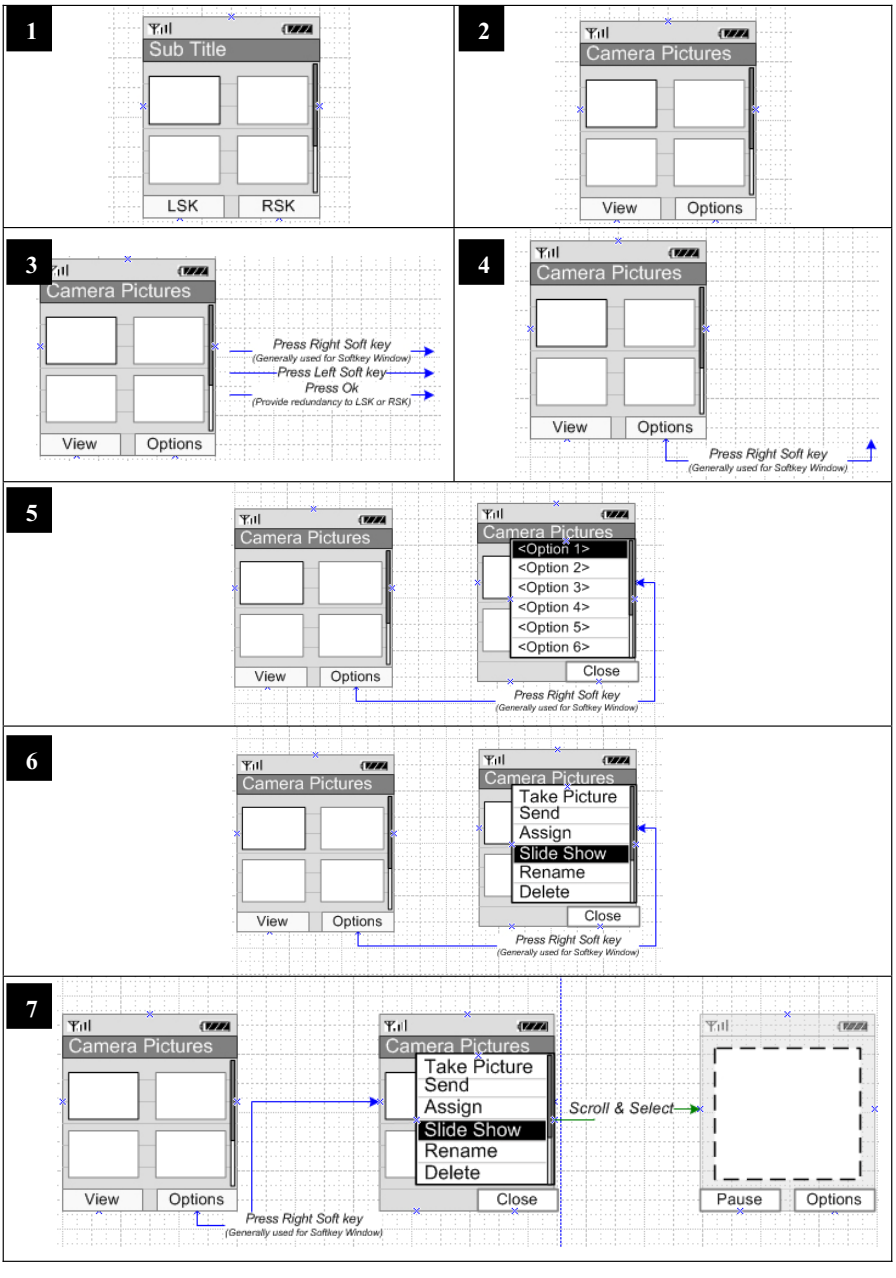


Fig. 2. Starting screen

Table 5. Task1 (Launching Slideshow from image browser)



Steps:

1. Drag Image browser shape from the Widget template
2. Put appropriate labels to the Image Browser
3. Drag the Access Additional Screen Functionality/Options shape from the Primitive action Template
4. Choose the most appropriate action in the given Constraint
5. Drag the Softkey window from the Widget. The new component gets attached to an appropriate place through the connectors
6. Put appropriate labels to the Softkey Window
7. Drag the Image shape from the Screen Template and make customization to the labels and the layout

In the same way rest of the tasks can be constructed. Later this Flow can be added as a shape to the Initiate template after getting it reviewed by the design team.

4 RUCID Testing, Analysis and Future Improvement

Peer reviews were conducted from time to time throughout the research and feedback was fed into the next course of the research.

Finding people to test with were not difficult because the people going to use the tool were the fellow interaction designers. However it was difficult to find time from their busy schedules. Some basic usability testing was conducted. The Visio prototype was very close to the actual envisioned tool.

The Scenario given to the User was same as in the prototype shown earlier in the paper i.e. to Create the Slideshow application using the tool. Since they were already acquainted with Visio they could easily operate it. They felt that the tool gave a lot of control and freedom in the designer's hands. They recommended on adding some more Microsoft Office Visio templates, such as Basic Flowchart Shapes, which will be helpful in making flow diagrams and adding comments. They felt only the generic design should be laid out through the Visio tool. When the task flows with the Screens are made it will have references to particular chapter in the generic UI specification, where the guideline is laid out. This way the patterns will complement the design guidelines.

However a pattern based approach to UI design can go beyond the Visio templates or even more customized specification tool. If the specification is actually done in something like Flashlite, and the patterns are codified in FlashLite, the designer can leverage a lot of the common elements of design and finally the specification output actually becomes the UI for the phone and the Flashlite version of the design can just be loaded upon a phone. There is no need for additional software development at least on the UI side.

Individual patterns and the language as a whole can be refined by tapping into the broad expertise of the Kyocera design staff by discussing and gathering their collective wisdom. The library of patterns can be converted into a workable set of standards by agreeing on an appropriate rating scale and by assembling a representative group of reviewers who rate the content according to the same criteria.

5 Conclusion

Ultimately, we expect the patterns-based UI Design library and the process and tool that it facilitates, will result in a strengthened Kyocera User Interface design and brand and a more efficient design staff.

A patterns-based approach has a better chance because of its inherent modularity which the tool is able to take advantage of.

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" (Alexander et al., 1977. p.x).

The patterns created as part of this project have still some distance to cover in terms of rigor and validation to justify Alexander's description above. But we believe they form a promising and important first step in that direction.

Future technologies like uiOne [11], Flashlite are a fertile field for the extension of the patterns based approach. The UI design library makes is convenient to constantly evolve the various elements of design and to seamlessly integrate innovative elements into the future phone features as a normal part of the design process. In fact, it should be possible to write parsers that would read a Visio file and produce Flashlite scripts or uiOne triglets that could potentially be directly loaded onto a phone as its User Interface. This removes the indirection involved in implementing a design. The interaction designer is able to control the actual look, usability of a feature on a phone directly. That would be a very good goal to achieve.

References

1. Cooper, A., Reimann, R.: Essentials of Interaction Design. About face 2.0
2. Alexander, C.: A Pattern Language
3. Implementing a Pattern Library in the Real World: A Yahoo! Case Study
4. van Welie, M., van der Veer, G.C.: Pattern Languages in Interaction Design: Structure and Organization
5. Vora, P., Castillo, J.: Using Patterns to Design Usable Interfaces for Web Applications. Alpha Cube, Inc. (June 29, 2005)
6. Todd, E., Kemp, E., Phillips, C.: What makes a good User Interface pattern language?
7. Tidwell, J.: COMMON GROUND: A Pattern Language for Human-Computer Interface Design, `http://www.mit.edu/~jtidwell/Interaction_patterns.html`
8. van Welie, M.: Patterns in Interaction Design, `http://www.welie.com/`
9. Yahoo Design Pattern Library, `http://developer.yahoo.com/ypatterns/`
10. Forum NOKIA, `http://forum.nokia.com/`
11. uiOne, `http://brew.qualcomm.com/brew/en/about/uione.html`