Design Patterns for User Interface for Mobile Applications

Erik G. Nilsson

SINTEF ICT, Postboks 124, Blindern, N-0314 Oslo, Norway – egn@sintef.no

Abstract In this paper we present a collection of user interface design patterns for mobile applications. The patterns in the collection are grouped into a set of problem areas that are further grouped into three main problem areas. After presenting this problem structure we present one of the patterns in some detail. Then we present some relevant findings from a validation of the patterns collection. This validation shows that both the patterns collection and the individual patterns are relevant and useful for usability professionals with a mixed background. Finally, we discuss pros and cons of using a patterns format for documenting design knowledge, related work and future research.

Introduction

In the UMBRA and FLAMINCO projects, we have developed a set of design guidelines to aid developing more user friendly applications on mobile devices (PDAs/SmartPhones), giving practical advices for how to solve various problems that arise when designing user interfaces on mobile devices. The main part of these design guidelines is a collection of user interface design patterns for mobile applications [5] (the patterns collection is available at http://www.sintef.no/flaminco). Each problem is presented on a design pattern format [4]. The "sources" for the problems addressed in the patterns collection are problems identified in the requirements elicitation phase of the UMBRA and FLAMINCO projects, and practical experience in developing and using mobile applications among the project partners.

Main problem areas

The design guidelines presented in the patterns collection follow a given structure. On the top level, they are grouped into *three main problem areas*. Within each of

these three main problem areas, a small number of *problem areas* are defined. Within each of these problem areas, a number of problems are identified. In Table #-1 below, some of the 26 identified problems (UI design patterns) with their connection to problem areas are listed. A brief version of the problem in bold font is presented in the next section.

Table #-1. UI design patterns and their connection to problem areas

Main probl. area	Problem area	Individual problems / UI design Patterns
Utilizing screen space	Screen space in general	Presenting elements in lists
		Principles and mechanisms for grouping information
		Mechanisms for packing information
	Flexible user interfaces	**Handling dialogs when SW keyboard is shown/hidden**
		Supporting switching between portrait and landscape mode
		UIs that should run on equipment with different screen size
Interaction mechanisms	Handling input	Mechanisms for entering text
		Mechanisms for entering numerical data
		Multi modal input
	Not using the stylus	Interacting with applications without using stylus
		Retrieving data from a database without using keyboard
Design at large	Guidelines	Standard features in an automatically generated prototype
		Combining branding, aesthetics, and screen space
	"Difficult to understand"	User interaction during synchronization
		User interaction during long-lasting operations

Handling dialogs when software keyboard is shown and hidden

Background. On PDAs without keyboard, a common solution for entering text is to show a software keyboard on the bottom of the screen where the user can enter text using the stylus. This area may already be used by the application, thus leaving less room for its "normal" interaction.

Problem. The main problem is how to resize the dialogs to avoid some parts of the dialogs becoming invisible. The severity of this problem depends on the type/style of the user interface. It is most challenging for forms-based UIs, while for UIs containing arbitrary text or visual presentations adding or adjusting a scrollbar is usually sufficient. Handling tab folders and buttons that are placed on the bottom of the screen is also a challenge.

Solutions. The most obvious and most simple solution to this problem is to *add or adjust scroll bars* when the keyboard appears. The other solutions presented below are solutions where the need for adding scroll bars are removed or reduced.

In some cases it is OK *letting the keyboard cover part of the UI*. How "bad" this solution is depends on what is placed on the part of the screen that will be covered by the keyboard. If this part is occupied by output fields, the solution may work fine as long as the keyboard is removed when not needed. If this part of the screen contains important input fields or tab folders the solution is useless.

Another simple, but seldom very practical solution is to *just use the part of the screen that will not be covered by the keyboard*. In practice, what this solution does is reducing the size of the screen. This solution may be OK for dialog boxes, but is seldom practical for normal windows.

A more advanced, but still fairly easily implemented solution is to *use one large UI control as a buffer*. By this we mean that when the keyboard is added, one of the controls is reduced vertically to be just as much smaller as the size of the keyboard. Controls that may be used for this are primarily list boxes and multi line text boxes.

Yet another fairly simple solution is *having two variants* of the UI, one that uses all the screen space and one that makes room for the keyboard. The main disadvantage with the solution is – in addition to added development work – that the user may be confused when the UI changes.

The buffer solution may also be used with two or more large UI controls sharing the amount of size reduction to be applied. Generalized, this solution ends up as *dynamic resizing* of the controls in the UI. This may be done using two different approaches. The first is to decide a resizing rule for each window and apply that as tailored code for each window. The second is to have a general layout adjustment algorithm doing it for all windows.

Validation

The patterns collection was presented at a half day tutorial at the HCI International conference in 2007 [6]. At the tutorial, the structure of the patterns collection was presented, and all patterns were presented at a very brief level. Then, 12 of the 26 patterns were presented in more detail. During the presentation, the participants filled in a questionnaire. They scored the relevance of the main problem areas, the relevance and usefulness of each of the presented patterns, and finally the relevance and usefulness of the patterns collection and such as well as their expectations for future use of the patterns collection. Relevance, usefulness and future use were scored on a scale from 1 (lowest) to 6 (highest).

29 of the participants at the tutorial handed in the questionnaire. There was a small majority of male, age varied from 25 to 50, most being around 30 years old. A majority had their origin in Asia, the rest coming from Europe and America. The majority has an education on master level, the rest split among undergraduates and PhD holders. The educational background was equally split between technical and non-technical. UI development experience varied from 0 to 12 years,

the majority having 5 years or less experience. Experience in developing mobile solutions varied from 0 to 6 years, the majority having 2 years or less experience.

Results

Scores on the main problem areas show an average score on relevance of 5.3 for Utilizing screen space, on 5.4 for Interaction mechanisms, and 4.9 for Design at large. Scores on the patterns collection as such show an average score on relevance of 5.0, and on both usefulness and future use of the patterns collection of 4.5. All these scores verify that the patterns collection both addresses relevant problems and gives useful and practical advices on how to solve these problems.

Looking at the scores for the individual patterns that were presented in more detail, the scores vary a bit, but are still fairly high. Figure #-1 below shows the average scores for relevance and usefulness for the 12 patterns, sorted descending on scores for relevance.

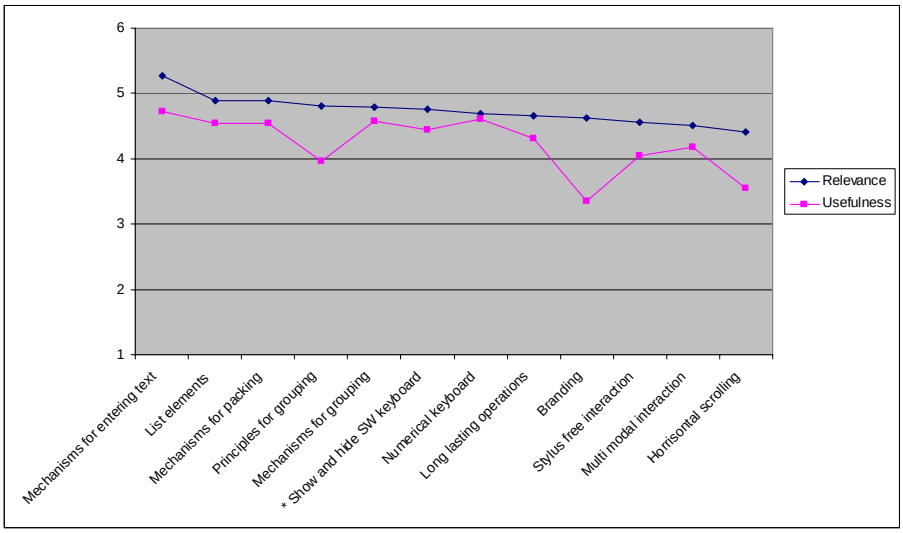

Fig. *#-1.* Average scores for relevance and usefulness

As for the patterns collection as such, the average scores for relevance are higher than the corresponding scores for usefulness. This is not surprising, as it is usually easier to agree with a problem description than a proposed solution. Although all but one of the average usefulness scores are on the top half of the scale, the patterns where the usefulness scores are lowest, and the patterns where the difference between relevance score and usefulness scores are highest, are candidates for further work. It may also be noted that correlation analyses show that the scores for relevance and usefulness correlates on the 0.01 level for 7 of the 12

problems, and on the 0.05 level for 2 of the other 5. Also, the scores for relevance and usefulness for the patterns collection as such correlates on the 0.01 level.

Using patterns format to document design knowledge

The chosen patterns format is in many ways well suited to document user interface design knowledge, as it captures the essential aspects of a problem. Also, as design patterns may be on different abstraction levels they can be used to describe problems of different "sizes". Furthermore, dividing a problem field into a limited number of well defined problems makes it possible to handle a set of manageable problems separately. Finally, having a patterns collection makes it possible to combine the just mentioned "divide and rule" principle with having an overall structure.

The biggest challenge we had using the patterns format is the connection between problems and solutions. Very often this is a many to many connection. Presenting the same or very similar solutions to a number of problems, either causes a lot of cross-references or large amounts of repetition. We chose to use cross-references, as is being done in other patterns collections [2]. Currently we are considering restructuring the patterns collection so that each pattern represents either one solution or a unique combination of one problem and one solution. Both these approaches will reduce the need for cross-referencing, but both will increase the number of patterns, thus making it more difficult to get an overview of the patterns collection.

Related work

There are a number of patterns collections and even collections of patterns collections on the web, see also [1] for an assessment of such collections. There are also a few collections of patterns for mobile user interfaces, like The Design Pattern Wiki[1] and Little Springs mobile UI design patterns[2]. The latter overlaps with two of our main problem areas, but while our collection is organized by problems, this collection is organized by solutions. The patterns presented in [8], although focusing on mobile interaction, are much wider in its scope than our collection, with only two user interface patterns. [2 & 3] present a design patterns collection for ubiquitous computing that is fairly large, but has a broader scope with patterns that are on a higher abstraction level and/or are less comprehensive in the suggested set of solutions than our patterns.

[1] http://www.gibbert.net/DPWiki (in German)

[2] http://patterns.littlespringsdesign.com/~newlsdpatterns/index.php/Main_Page

Conclusions and future work

In this paper we have presented a structured collection of user interface design patterns for mobile applications. The structure is valuable as an index and it gives a comprehensive overview of design problems for mobile UIs.

The patterns collection has been validated using a questionnaire at a tutorial. This validation shows that both the individual patterns assessed and the whole collection were perceived as relevant and useful by the participants, and that it is likely that they will use the collection in future work. It also identifies patterns that need more work.

Finally, we have discussed pros and cons of using a patterns collection for documenting design knowledge. The main pro being the ability to divide a large problem area into a structured set of manageable problems, the main con being that the same solution may apply to a number of problems, causing a lot of cross-references between the patterns. The collection is continuously being improved and enhanced, e.g. in the areas multi-modal interaction and exploiting context [7].

Acknowledgments The work on which this paper is based is supported by the UMBRA and FLAMINCO projects funded by the Norwegian Research Council and the industry partners in these projects. I would also like to thank my colleagues Asbjørn Følstad and Jan Heim for contributions to designing the questionnaire used for the validation and in analyzing the results.

References

[1] Deng J et al (2005) Managing UI pattern collections. In Proceedings of the 6th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction
[2] Duyne D K & Landay J A (2004) Design Patterns, Course documentation
[3] Landay J A & Borriello G (2003) Design Patterns for Ubiquitous Computing. In IEEE Computer August 2003
[4] Gamma E, Helm R, Johnson R, and lissides J (1995) Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley
[5] Nilsson E G (2005) Design guidelines for mobile applications. SINTEF Report STF90 A06003, ISBN 82-14-03820-0
[6] Nilsson E G (2007) Design patterns for user interfaces on mobile equipment. Tutorial documentation, HCI International
[7] Nilsson E G, Floch J, et al. (2006) Model-based user interface adaptation. Computers & Graphics 30(5): 692–701
[8] Roth J (2002) Patterns of Mobile Interaction. Personal and Ubiquitous Computing, Volume 6, Issue 4 (September 2002)