Model-Based Design of Mobile User Interfaces

JEAN VANDERDONCKT, MURIELLE FLORINS, FRÉDÉRIC OGER

Université catholique de Louvain

vanderdonckt@qant.ucl.ac.be
Phone: +32-10/47 85 25
Fax: +32-10/47 83 24

Abstract We propose a set of techniques that will aid UI designers who are working in the domain of mobile computing. These techniques will allow designers to build UIs across several platforms, while respecting the unique constraints posed by each platform. In addition, these techniques will help designers to recognize and accommodate the unique contexts in which mobile computing occurs.

Keywords. Mobile computing, Model-based design, UI models.

Techniques for Supporting Mobile User Interface

We describe a spectrum of model-based techniques that can be used to support mobile computing. Each technique involves creating mappings between the various model components that we have discussed. These mappings are interpreted to produce a UI that is specially customized for the relevant device and context of use.

Handling Platform Constraints

One obvious constraint that is often posed by mobile computing platforms is the display resolution, which can range from a wall-size flat screen to a cellular phone to a head-mounted immersive environment. We will use this screen resolution constraint as an example case, and describe some methods for dealing with it, while showing how these methods might also be applied to other constraints. The screen resolution constraint was chosen because we feel that in many ways it is the most difficult constraint to deal with; whereas other constraints, such as limited interaction capabilities, can be handled through interactor selection, the screen space constraint often requires a more global solution.

The screen resolution is typically expressed in pixels (e.g., 1024x768). It should not be confused with the screen surface area; two displays having different sized surfaces can share the same resolution. Because of screen resolution differences, an optimal layout for one display may be simply impossible to render on another device. This problem is particularly salient for mobile computing, because so many mobile platforms possess small-size, low-resolution displays. There are three parameters that contribute to the amount of display size required by a user-interface design: size of the individual interactors, layout of interactors within a window, and the allocation of interactors among several windows. There are two possible methods for accommodating screen resolution constraints by adjusting the size of the interactors. The first possibility is simply to shrink the interactors, while observing usability constraints related to the AIO type. For example, the length of an edit box can be reduced to a minimum (e.g., 6 characters visible at the same time with horizontal scrolling) while its height cannot be decreased below the limit of the smallest font size legible (e.g., 8 pixels); usability experiments have determined that the minimum size for an icon is roughly 8 by 6 pixels.

Of course, many interactors simply cannot be shrunk to any significant extent. An alternative to reducing the size dimensions of an interactor is to replace that interactor with a smaller alternative. For example, a Boolean checkbox typically requires less screen space than a pair of radio buttons. The technique of automatically selecting an appropriate interactor while considering screen resolution constraints has already been investigated and shown to be feasible [3,4,7].

These techniques can easily be applied to constraints other than screen resolution. Interactors can be parameterized to reduce bandwidth usage or battery consumption–for example, a video player can reduce the frame rate or the number of colors. This parameterization is similar to reducing the display size of interactors. Likewise, interactor selection can be performed to accommodate

reduced interaction capabilities. For example, a handwriting-based interactor from a PDA could be replaced with a voice-based interactor on a cellular phone. When there are several criteria that need to be optimized – e.g., screen size, usability, and power consumption – interactor selection can become a multidimensional optimization problem. An algorithm for solving this problem would be beyond the scope of this paper, but we hope to explore it in the future.

Returning to the matter of screen resolution, we see that both techniques for adjusting the size of the interactors in a UI can help us to find an appropriately-sized presentation. However, we believe that in many cases, a more global solution is necessary. A WAP-enabled cellular phone can display only one or two interactors at a time, no matter how small they are. To achieve the amount of flexibility necessary to support all mobile devices, we will have to examine window layout and the allocation of interactors among windows.

The remaining two parameters—layout of interactors within a window, and allocation of interactors between windows—can be grouped together under the mantle of *presentation structure*. We want to select the appropriate presentation structure, given the constraint of the amount of screen-resolution afforded by a platform. To solve this problem, we need a set of alternative presentation structures, which can be generated by the designer or with the help of the system. The simplest solution would then involve creating mappings between each platform and an appropriate presentation structure. However, in this case a more dynamic solution is possible. Recall that the screen resolution of each device is represented declaratively in the platform model. Similarly, it is possible to represent the amount of screen space required by each presentation structure in the presentation model. We can exploit this knowledge by constructing an intelligent *mediator* agent [1,2]. This mediator should determine the maximum usable screen resolution for the relevant device, and evaluate the amount of screen resolution required by each presentation structure alternative. It can then select the presentation structure that consumes an amount of screen resolution that falls just under the maximum (fig. 1,2).

This more dynamic solution is preferable because it accounts for the fact that the screen resolution of a device may change while it is in use. Moreover, it eases the integration of new devices into the platform model. Rather than forcing the user to explicitly specify the appropriate presentation structure for a new device, the user needs only to specify the amount of available screen space, and the mediator will find the correct mapping.

Under our proposed architecture, it is still left to the interface designer to specify a set of alternative presentation structures. However, it would be better if the correct presentation structure could be generated by the system, given a set of user-defined constraints [1,4,5,7]. For this purpose, we need to consider additional elements in our presentation model besides AIOs and CIOs. Two new abstractions need to be defined:

1. *Logical Window* (LW): this can be any grouping of AIOs—a physical window, a subwindow area, a dialog box or a panel. Every LW is itself a composite AIO as it is composed of other simple or composite AIOs. All LWs should be physically constrained by the user's screen.
2. *Presentation Unit* (PU): a PU is defined as a complete presentation environment required for carrying out a particular interactive task. Each PU can be decomposed into one or many LWs, which may be displayed on the screen simultaneously, alternatively, or in some combination thereof. Each PU is composed of at least one window called the *main window*, from which navigation to other windows is allowed. For example, a tabbed dialog box can be described as a PU, and it should be decomposed into LWs corresponding to each tab page.

The abstractions can be structured into a hierarchy (fig. 3) that serves to expand modeling capabilities of a presentation model. We can use this hierarchy to construct an automated design tool that generates several platform-optimized presentation models from a starting presentation model that is platform independent. This tool could function by employing one of the redesign strategies described below.

Re-modeling LWs within a PU: the contents of initial LWs are redistributed into new LWs within a single PU. Basic operations involve: ungrouping an LW into several smaller LWs; grouping the contents of several LWs into a single, comprehensive LW; and moving AIOs from one LW to another. For example, if ample space is available, then the LWs representing several tab sheets can be grouped into a single LW. Alternatively, if space is at a premium, the contents of a single window can be broken into pages of a tab sheet. Some existing tools apply some of these operations. SEGUIA suggests configurations based on the semantics [7]: minimal (as many logical windows as possible), maximal (one logical window for a PU), input/output (one logical window gathering input while another gathers output for a same function), functional (LWs depending on the functional analysis), and free (as many LWs as the designer wants to).

Re-modeling AIOs within a LW: according to existing constraints, the contents of an initial LW are redistributed into new AIOs, composite or simple. For example, AIOs contained in a group box are split into smaller group boxes or individual AIOs. A group box can also be replaced by a push button that displays the AIO contained in this box on demand. This technique remains unexplored today. Moreover, we are aware of no unambiguously successful algorithm for the automatic generation of a presentation structure based on these abstractions that has yet been documented. Obviously, both of these techniques leave out the difficult question of dialog layout.

As we have said, screen resolution is not the only constraint that must be negotiated. Other constraints include bandwidth usage, battery power consumption, number of display colors, and interaction capabilities. However, unlike the screen-space constraint, these problems do not appear to require a global strategy; they can usually be accommodated through AIO selection. Previous research has addressed the problem of dynamic interactor selection at length, and the issue of platform-specific input constraints has been considered [3,4,6]. We realize that additional constraints may present themselves in the future. While the techniques described in this paper cannot be expected to handle any and all platform-based constraints arising from mobile computing, we feel that this work can serve as a starting point for the development of further model-based solutions.

Focusing on Contexts of Use

So far we have assumed that on each device, the user will want to accomplish the same set of tasks. Often this is not the case. A device may be especially suited for a specific subset of the overall task model. For example, in the scenario in Section 2, we know that the cellular phone is especially suited for finding driving directions, because we can assume that if the user were not driving, she would use the PDA. The desktop workstation cannot be brought out into the field, so it is unlikely that it will be used to enter new annotations about a geographical area; rather, it will be used for viewing annotations. Conversely, the highly mobile PDA is the ideal device for entering new annotations.

Through the application of a task model, we can take advantage of this knowledge and optimize the UI for each device. The designer should create mappings between platforms (or classes of platforms) and tasks (or sets of tasks). Additional mappings are then created between task elements and presentation structures that are optimized for a given set of tasks. We can assume these mappings are transitive; as a result, the appropriate presentation model is associated with each platform, based on mappings through the task model. The procedure is depicted in fig. 4. In this figure, the task model is shown to be a mere collection of tasks. This is for simplicity's sake; in reality, the task model is likely to be a highly structured graph where tasks are decomposed into subtasks at a much finer level than shown here.

There are several ways in which a presentation model can be optimized for the performance of a specific subset of tasks. Tasks that are thought to be particularly important should be represented by AIOs that are easily accessible. For example, on a PDA, clicking on a spot on the map of our MANNA application should allow the user to enter a new note immediately (fig. 5). However, on the desktop workstation, clicking on a spot on the map brings up a rich set of geographical and meteorological information describing the selected region, while *showing* previously entered notes. On the cellular phone, driving directions are immediately presented when any location is selected (fig. 6). On the other devices, an additional click is required to get the driving directions. The "bottom arrow" button of the cellular phone enables the user to select other options by scrolling between them.

In this way, our treatment of optimizations for the task structure of each device is similar to our treatment of the screen-space constraint: global, structural modifications to the presentation model are often necessary, and adaptive interactor selection alone will not suffice. Here, we propose to solve this problem through the use of several alternative user-generated presentation structures. In many cases, automated generation of these alternative presentation structures would be preferable. We hope to explore the automated generation of task-optimized presentation structures in the future.

References

1. Y. Arens and E.H. Hovy, "The Design of a Model-Based Multimedia Interaction Manager", *Artificial Intelligence Review*, Vol. 9, Nos. 2-3 (1995), pp. 167-188.

2. Y. Arens, L. Miller, S.C. Shapiro, N. Sondheimer, "Automatic Construction of User-Interface Displays", *Proc. of AAAI'98* (St. Paul, 21-26 August 1988), AAAI Press / The MIT Press, 1988 pp. 808-813.
3. J. Eisenstein and A. Puerta, "Adaptation in Automated User-Interface Design", *Proc. of IUI'2000* (New Orleans, 9-12 January 2000), ACM Press, New York, 2000, pp. 74-81.
4. S. Prasad, "Models for Mobile Computing Agents", ACM Comput. Surv. 28 (4), Dec. 1996, Article 53.
5. M. Satyanarayanan, "Fundamental Challenges in Mobile Computing", *Proc of the fifteenth annual ACM symposium on Principles of distributed computing*, ACM Press, New York, 1996, pp. 1-7.
6. J. Vanderdonckt, F. Bodart, "Encapsulating Knowledge for Intelligent Interaction Objects Selection", Proc. of InterCHI'93, ACM Press, New York, 1993, pp. 424-429.
7. J. Vanderdonckt, P. Berquin, "Towards a Very Large Model-based Approach for User Interface Development", Proc. of 1st Int. Workshop on User Interfaces to Data Intensive Systems UIDIS'99, IEEE Computer Society Press, Los Alamitos, 1999, pp. 76-85.

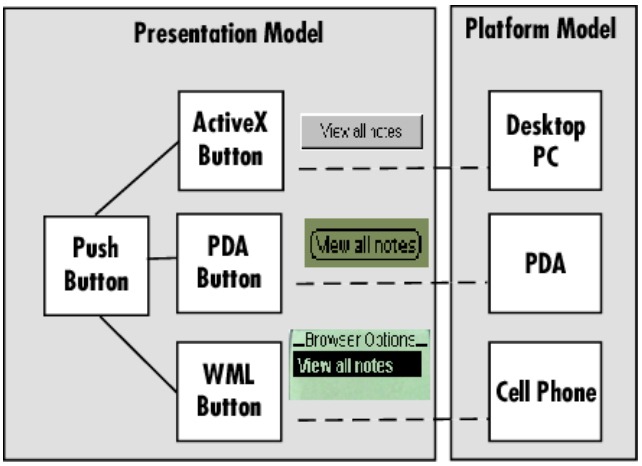

Figure 1. Concrete Interaction Objects are subclassed from an Abstract Interaction Object, and are then mapped on to specific platforms.

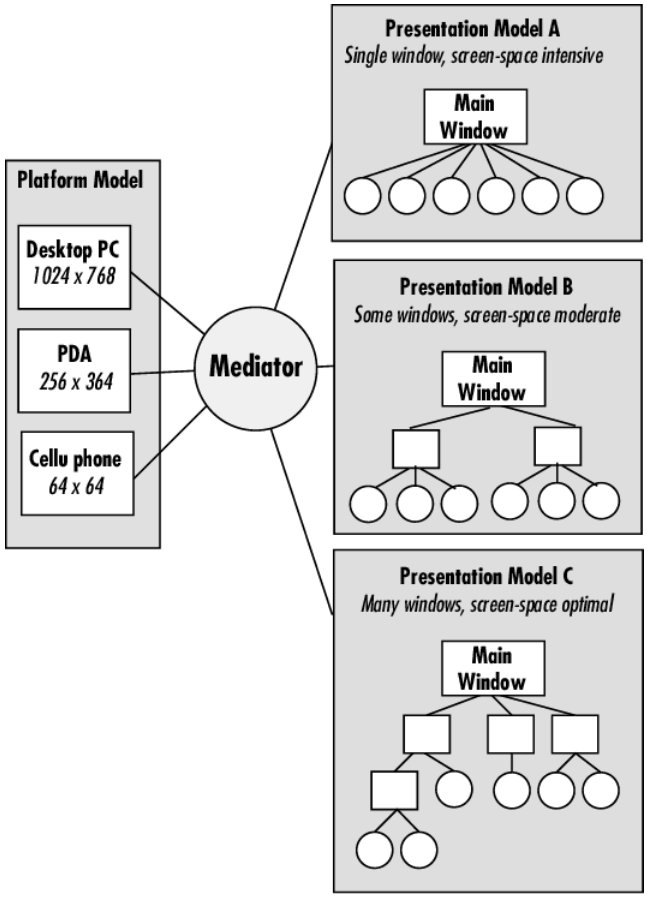

Figure 2: A mediating agent dynamically selects the appropriate presentation model for each device.

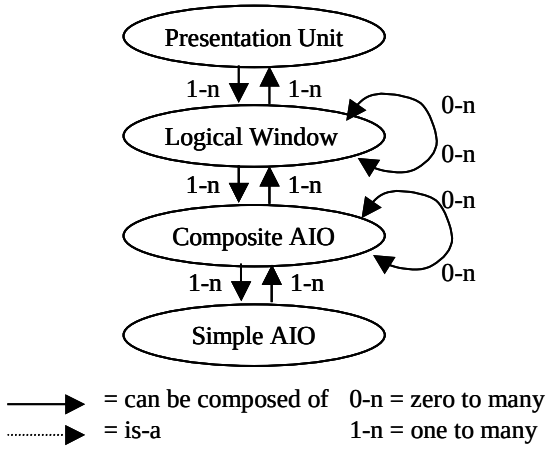

Figure 3. Hierarchy of presentation concepts.

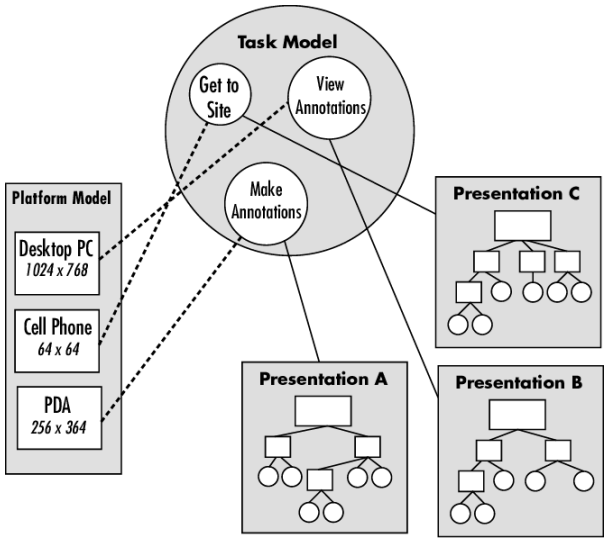

Figure 4. Platform elements are mapped onto high-level task elements that are especially likely to be performed. In turn, the task elements are mapped onto presentation models that are optimized for the performance of a specific task.

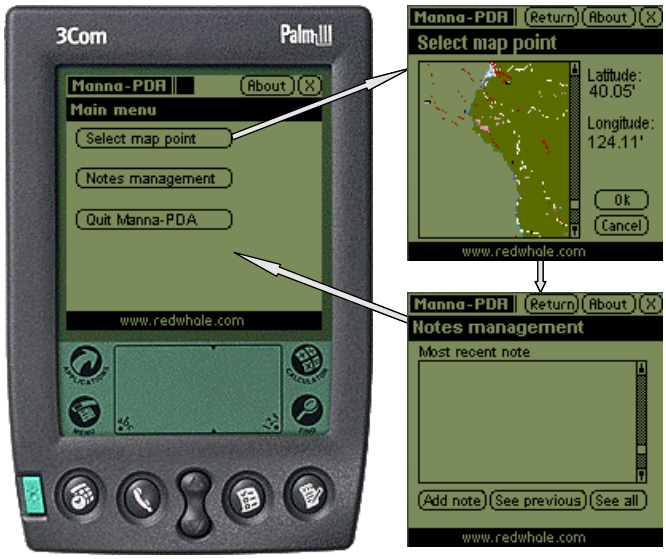

Figure 5. Final presentation for the PDA.

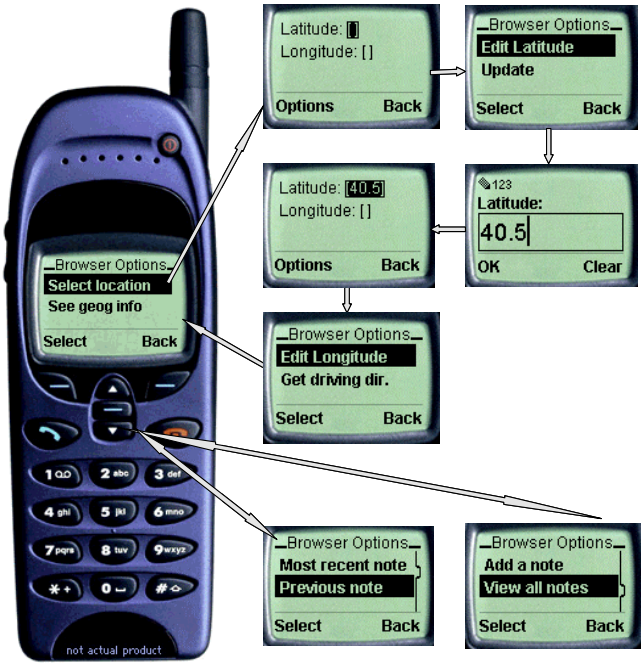

Figure 6. Final presentation for the cellular phone.