5 Navigation Consistency, or the Lack Thereof, in Cross-Platform User Interfaces

Mir Farooq Ali, Human Interaction Research,Motorola Labs, 1295 E. Algonquin Rd., Schaumburg, IL 60196, USA farooq.ali@motorola.com

There are many different aspects of consistency with regard to cross-platform user interfaces and their design. One of the important factors in using a cross-platform user interface is its navigation capability across platforms. In many instances, it is impossible to provide navigation consistency due to the inherently different nature of the user interfaces (UIs) on different platforms. In this position paper, we discuss that even while using the same representation of a UI as a starting point, the design of a cross-platform UI necessitates having different navigation operators. We discuss a development process that uses a few navigation operators to provide different navigation capabilities for different platforms.

INTRODUCTION

There are many different aspects of consistency with regard to cross-platform user interfaces and their design. Denis and Karsenty [6] introduce a term “inter-usability” to describe the ease with which users can reuse their knowledge and skills for a given functionality when switching from one device to a different device. A related term that they introduce is called “inter-device consistency”. This is related to the consistency when switching between devices. They describe this consistency at four levels across devices: perceptual, lexical, syntactic and semantic.

One of the other important aspects of this consistency across devices is the navigation capability that is provided to the end user on each of the different devices. Navigation, as performed by the end-user, is inherently a part of accomplishing some task. This navigation capability could differ dramatically from device to device depending on their capabilities and also the way the user interface is designed. It also plays an important role in situations, where the end-user starts a task on one device, then switches to another device to complete or resume the task. Ideally, consistency across devices would imply that the navigation capability is the same across devices. However, this is not necessarily the case and forcibly ensuring consistency, especially in automatically generated UIs, could lead to poor usability.

This paper discusses a design process [2] that allows the UI developer to flexibly create UIs that could have different navigation capabilities on different platforms. The developer has to work with CTT task models [10, 12] and the UIML specification language [1]. This process is semi-automatic and requires developer intervention to help guide the UI design/generation process.

RELATED WORK

Some of the common design mechanisms for cross-platform or multi-platform user interfaces involve model-based design processes and tools that use abstract models as a starting point [2, 4, 5, 8–10, 12] and/or the use of specification languages including UIML [1–3], USiXML [8, 9] and XIML [13]. Some other approaches to ensure consistency of UIs across devices involve UNIFORM [11], which is used to automatically generate consistent remote control UIs, and SUPPLE [7].

PREVIOUS WORK IN THIS AREA

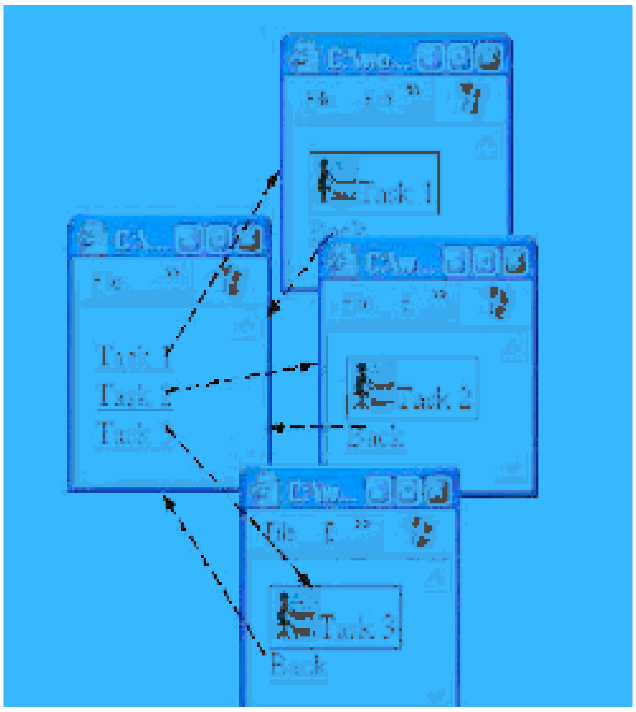
As mentioned earlier, we use the User Interface Markup Language (UIML) as the underlying language for building UIs [1]. Details about the UIML language and its pros and cons with respect to cross-platform UI development can be found elsewhere [1]. The CTT task model notation notation represents a hierarchical tree of tasks and sub-tasks [10]. Each of the tasks in the task tree is categorized as either user, interaction, application or abstraction. In addition to the tasks, there are temporal operators in the CTT notation that specify some relationship between sibling tasks.

This section discusses some of our past work in creating a design process for multi-platform user interfaces and how some specific operators help aid the developer in customi¬zing the navigation capability to different devices. The de¬veloper initially has to start developing a CTT task model. This task model specifies the different tasks that the end-user performs with the system. The developer then has to add so¬me navigation operators to the task model. This is done at the level of each task in the task model. UIML is generated from the CTT task model after the developer has completed anno¬tating it. The UIML could be further customized for each platform, keeping in mind different aspects of consistency desired.It should be noted again that depending on the particular target device for which the UI is being generated, the navigation operators might vary. As an example, the desired navigation style for a mobile platform family might incorporate a more menu-style navigation style, since the typical UIs for small mobile devices are structurally hierarchical, while there might be more considerably less menu-like navigation used for a desktop family, which might not necessarily be very hierarchical.

Navigation operators: We next describe the navigation operators that could be added to each task by the developer. Each task can have one of three navigation operators. These operators are applicable only for non-leaf level task nodes in the task model tree. The navigation operators are menustyle, contains and independent.

Each operator is explained next.

- Menustyle: This navigation operator for a task indicates that it has to be organized in the form of a menu with selection of each menu-item leading to the particular subtask. An example of this is given below in Figure 1. The figure indicates that a menu is created with three menu-items, each of which leads to a separate container for the sub-tasks. The dashed lines indicate the navigation paths from the menu to the subtask containers and back. There might also be more navigation paths between the subtask containers based on the temporal operators between the subtasks. These are shown in Figure 3 below



.

Figure 1. A menu in HTML with three options, each leading to one of the subtasks

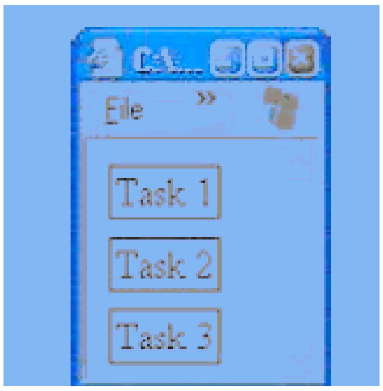


Figure 2. A container with three subtasks

- Contains: This type of navigation operator indicates that the parent task is mapped to some container that contains whatever parts its subtasks are mapped to. And example of this is shown in Figure 2.
- Independent: This navigation strategy indicates that each subtask of a particular task can be in an independent container. Figure 3 shows how this structure might look like. The lines in the figure indicate the navigation paths between the three containers. The name for this particular navigation style indicates independence from the parent container. The navigation paths vary between the various subtasks since the temporal operators between them are different.

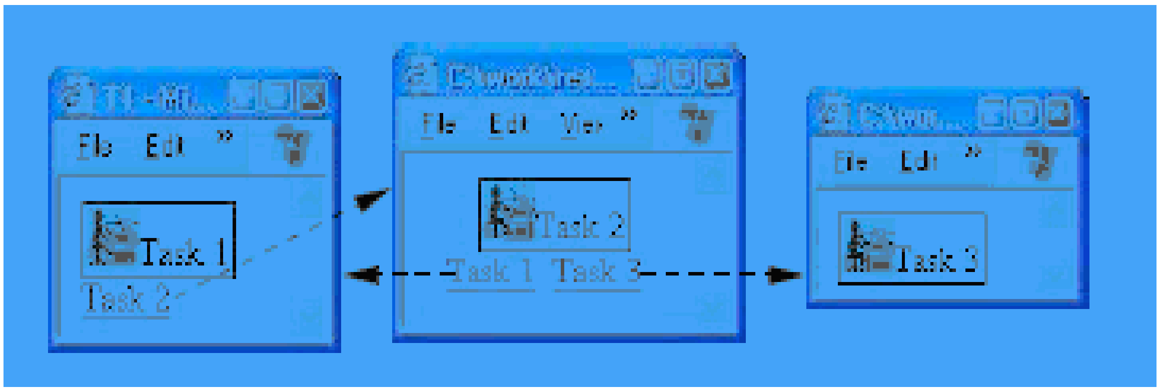


Figure 3. A container with three subtasks.

It should be noted that these navigation operators have been applied to the same initial task model yielding different navigation styles in the final generated UI.

SUMMARY

We have presented in this paper our position that, depending on the desired target platforms, different navigation styles might not just be desired – they might be a necessity. We presented some ways of specifying these in conjunction with the CTT task model and UIML specification language as an aid to the developer. The entire developement process is transformation based and involves many phases [2]. We have just touched on one particular aspect of this process involving specification of navigation operators that is relevant to this workshop.

Note: This work was done as part of the author’s Ph.D. dissertation research at Virginia Polytechnic Institute and State University

REFERENCES

[1] Abrams, M., Helms, J., 2002. User Interface Markup Language (UIML) Specification: Language Version 3.0 (Draft).http://www.uiml.org/specs/docs/ uiml30-revised-02-12-02.pdf

[2] Ali, M. F., A Transformation-based Approach to Building Multi-Platform User Interfaces Using a Task Model and the User Interface Markup Language, Ph.D. Dissertation, Virginia Tech, 2005, http://scholar.lib.vt.edu/theses/ available/etd-05172005-041721/

[3] Ali, M. F., Pe´rez-Quin˜ones, M., Abrams, M., 2004. Building Multi-Platform User Interfaces with UIML. In: Seffah, A., Javahery, H. (Eds.), Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces. John Wiley & Sons, Ltd, West Sussex, England, Ch. 6, pp. 95–118.

[4] Bandelloni, R., Paterno`, F., January 2004. Flexible Interface Migration. In: Ninth International Conference on Intelligent User Interfaces: IUI’2004. Madeira, Funchal, Portugal, pp. 148–155.

[5] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J., June 2003. A Unifying Reference Framework for Multi-target User Interfaces. Interacting with Computers 15 (3), 289–308.

[6] Denis, C., and Karsenty, L. Inter-Usability of Multi-Device Systems – A Conceptual Framework, In: Seffah, A., Javahery, H. (Eds.), Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces. John Wiley & Sons, Ltd, West Sussex, England, Ch. 17, pp. 381–383.

[7] Gajos, K., and Wu, A and Weld, D. CrossDevice Consistency in Automatically Generated User Interfaces, in Proceedings of the 2nd Workshop on Multi-User and Ubiquitous User Interfaces. 2005. San Diego: pp. 7-8.

[8] Limbourg, Q., 2004. Multi-Path Development of User Interfaces. Ph.D. thesis, Universite´ catholique de Louvain, Louvain-la-Neuve, Belgium.

[9] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Florins, M., Trevisan, D., May 2004. USIXML: A User Interface Description Language for Context-Sensitive User Interfaces. In: Luyten, K., Abrams, M., Vanderdonckt, J., Limbourg, Q. (Eds.), Developing User Interfaces with XML: Advances in User Interface Description Languages. Gallipoli (Lecce), Italy, pp. 55–62.

[10] Mori, G., Paterno`, F., Santoro, C., August 2004. Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions. IEEE Transactions on Software Engineering 30 (8), 507–520.

[11] Nichols, J., and Myers, B., and Rothrock, B., UNIFORM: Automatically Generating Consistent Remote Control User Interfaces, To Appear in Proceedings of CHI’2006. April 22-26. Montreal, Canada.

[12] Paterno`, F., 2004. ConcurTaskTrees: An Engineered Notation for Task Models. In: Diaper, D., Stanton, N. (Eds.), The Handbook of Task Analysis for Human-Computer Interaction. Lawrence Erlbaum Associates, publishers, Mahwah, New Jersey, Ch. 24, pp. 483–502.

[13] Puerta, A., Eisenstein, J., 2004. XIML: A Multiple User Interface Representation Framework for Industry. In: Seffah, A., Javahery, H. (Eds.), Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces. John Wiley & Sons, Ltd, West Sussex, England, Ch. 7, pp. 119–148.