MenuSelector: Automated Generation of Dynamic Menus with Guidelines Support

Jeremy Spoidenne, Jean Vanderdonckt

Université catholique de Louvain (UCL)
School of Management (IAG) - Unit of Information Systems (ISYS)
Belgian Laboratory of Computer-Human Interaction (BCHI)
Place des Doyens, 1 – B-1348 Louvain-la-Neuve (Belgium)
vanderdonckt@isys.ucl.ac.be

Abstract

This paper presents a software tool that enables developers to quickly specify design options for a menu bar and related menus and sub-menus for any type of application. These design options are then stored in a universal user interface specification language (XIML) for later use. At any time, the current design of the menu can be checked against some basic usability guidelines, depending on currently being specified options. Once a menu is ready for full specification, it is then subject to automated generation of MacroMedia Flash code. Traditional menu abstractions are frequently based on the paradigm of Graphical User Interfaces. Here, we attempted to expand this paradigm to any type of menu interactor, with static or dynamic features.

1 Introduction

Incorporating guidelines in software tool to assist designers and developers has been for a long time a research and development issue that still remains not completely solved. Separating the user interface (UI) design phase from the evaluation phase has been criticized for a long time: not only both phases are decoupled, but the developer may become confused when the evaluation report, along with some feedback, is produced because she is no longer in a design context. This change of focus of attention may disrupt developers. To overcome this problem, we believe that producing evaluation feedback while designing a UI would be better appreciated by developers. While they are designing and developing a UI, they could at any time trigger an evaluation.

For this purpose, we developed a test-bed application, MenuSelector, that allows developers to design a menu bar with menus and sub-menus for any type of interactive application. Then, the developer can perform a guidelines checking by asking the software to directly assess the currently being designed menu. We choose the menu selection interaction style as many usability guidelines exist today to characterize whether a menu is usable or not, for instance (Norman, 1990). In particular, some guidelines (Kiger, 1994) (Norman & Chin, 1988) establish boundaries for keeping a usable trade-off between the breadth and depth. More specifically, even age differences of users can influence this trade-off (Zaphiris, 2001). To reach this goal, a new abstraction of the menu concept was required to specify not only traditional aspects (like menu items, breadth, font, size), but also non-traditional aspects (like presentation, animation effect). When designing a menu, the developer can specify appropriate values for any of these parameters. MenuSelector then stores this specification in XIML (exTensible Interface Mark-up Language) (Puerta & Eisenstein, 2002) for future use. For some combinations of these parameters, MenuSelector automatically generates MacroMedia Flash code in vector presentation style (SWG). Automated generation of menu from a conceptual schema has already been investigated and proved feasible (Vanderdonckt, 1998). However, incorporating support for guidelines evaluation has been underexplored so far (Scapin *et al.*, 2000).

2 The New Menu Abstraction

To support the specification of advanced menus, possibly with dynamic aspects, for traditional Graphical User Interfaces (GUIs) or for non-WIMP menus, possibly in three dimensions, a new abstraction of the menu artefact was needed. This abstraction consists in defining a series of parameters for each menu where appropriate. These parameters are summarized in Table 1.

Menu	General organization	Full screen, localized
	Amount of spatial dimensions	1D, 2D, 2D½, 3D (Vanderdonckt, 2003)
	Shape of main presentation axis	Linear, curve, polygonal
	Menu disposition	Horizontal, vertical, oblique, circular, polygonal, mixed
	Menu orientation	If vertical: left, right If horizontal: top, bottom
	Modality types	Textual, iconic, bimodal
	Status	Compact, full
	Representation	Implicit, explicit
	Duration	Static, dynamic, persistent
	Apparition	Animated, not animated
	Navigation	Continuous, by selection
	Length	Fixed, variable
Sub-menu	Apparition	Expandable: drop-down, cascade, sub-menu, other Not Expandable
	Reference to lower level	Upper level complete, by title, inexistent
Menu item	Type	Command item, dialog item, sub-menu item, toggle item, radio item
	Highlighting method	None, for “on-mouse over” state, for “on-mouse down” state, similar for both, different for both.

Table 1: Parameters of the new Menu abstraction.

Figure 1 shows a menu with the following parameters: localized (the menu only consumes some part of the screen, not the whole screen), 3D (as the presentation of items are in three dimensions, linear (items are aligned on a horizontal axis), horizontal menu disposition, items are oriented from left to right. Each menu item is bimodal as icons are augmented with menu labels. The menu is full (no alternative presentation depending on the user for instance), the representation is explicit, their duration is static (items are presented only during the menu selection), they are animated with a 3D special effect, the navigation is continuous, and the length is fixed.

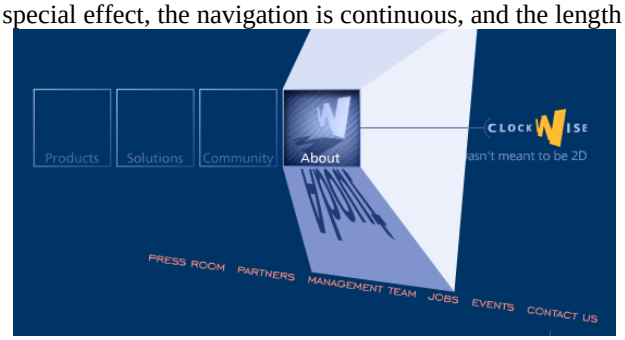

Figure 1: Example of non-traditional menu

3 Tool support

A prototypical tool has been implemented in Microsoft Visual Basic V6.0 that allows developers to specify any menu according to the parameters defined in the menu abstraction. The tool verifies that each menu specification is compliant with some basic guidelines. For instance, it is possible to check that the amount of items at the first level does not exceed the threshold prescribed by usability guidelines. Therefore, if the developer is attempting to introduce to many items at the first level with too few items on the subsequent levels, the tool may suggest re-ordering items.

Depending on previously entered specifications, the tool activates or deactivates further possible specifications (Fig. 2a). For instance, when choosing a design option is restricting the scope of other design decisions, only the possible choices are displayed and activated. For example, when the amount of spatial dimensions is specified as 2 (Fig. 2b), the developer can specify various shapes for the principal axis along which the menu items will be presented, here a curve. Due to this choice, the tool restricts the range of potential values for the other parameters.

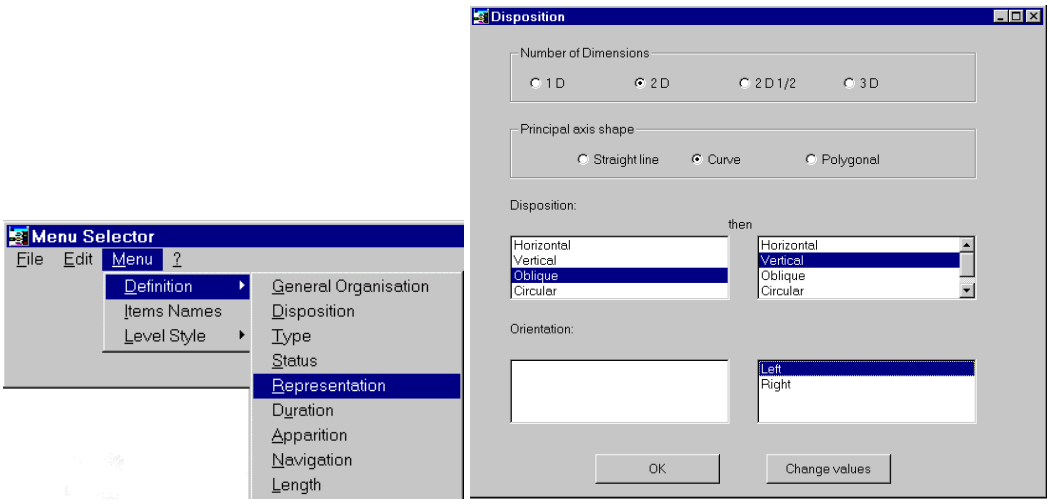

Figure 2: Entering menu specifications with MenuSelector.

It is important for developers not to feel forced to enter all parameters at the same time. Indeed, UI design process remains very flexible, largely incomplete or underspecified at the beginning. The early requirements may evolve with the rest of the interactive application during the traditional development life cycle. As this cycle is progressing, new requirements can be elicited that can be transformed into new or updated specifications for the menu. At any time, the developer can specify the hierarchy of menu, how many levels, what are the items (the text and the icon where appropriate), what kind of transitions should be supported in the menu selection. This can be updated at any time. Although the abstraction represented in Table 1 goes further than the usual menu definition, developers should be enabled to add any extra, possibly unsupported, definition of parameters.

Each menu specification is internally stored in XIML (exTensible Interface Markup Language) (Puerta & Eisenstein, 2002), which is a UI specification language supporting various models (i.e., task, domain, user, presentation, dialog, platform, and general purpose). Each model may contain attributes defined locally or globally. Between models or within models, relationships can be defined. For the basic parameters defined in table 1, a canonical representation has been defined in

XIML so as to express any combination of these parameters. For custom, possibly unsupported, parameters, the general purpose attribute definition is used. Therefore, each menu is modelled in a presentation model, which is recursively decomposed into presentation elements. At the first level (e.g., a menu bar), a first presentation element of the class "Menu Bar" is instantiated. To add pull-down menus, instances of the class "Pull-down menus" are created. Similarly, for any sub-menu or final menu item, each parameter can be specified where appropriate. This XIML specification can then be integrated with a larger UI specification, for instance the other models. Or the menu definition can itself become a particular presentation element of the global presentation elements hierarchy. At any time, the current specifications can be exported in a XIML-compliant file to be integrated in a larger UI design repository or for another tool to generate final code. A summary of the specifications can be displayed in XIML format on demand. As this XIML file is itself XML-compliant, it can be the source of any XSLT transformation so as to create other definitions, or XML compatible UI definitions.

MenuSelector is intended to automatically generate code corresponding to the menu in MacroMedia Flash format. This language allows much more flexibility in the presentation styles and dynamics than traditional mark-up languages (e.g., HTML) or programming languages (e.g., Visual Basic), unless special libraries are used (e.g. AfterEffect). Fig. 3 represents a 1D vertical menu where the first-level items can be deployed at any time and separately. Clicking on any coloured tab open or closes the corresponding sub-menu. Contrarily, fig. 4 presents the same menu items, but in a purely hierarchical menu, where only one level of menu can be open at a time. Clicking on any other tab automatically closes the previously opened menu. Although this variation may seem very limited, the tool allows the developer to change parameters at a higher level of abstraction (the one proposed in Table 1) than at the code level. MenuSelector does not automatically generate code for all parameters combinations that can be inferred from Table 1. Rather, some parameters, that have been identified as fundamental, are currently being supported. The exported XIML file can be edited manually or may initiate another UI generation or interpretation thanks to an external tool. If this tool supports the abstraction introduced in Table 1, the specification task is considered as a good starting point. Not all abstractions should be supported by all external tools. Some parameters can be left unsupported, such as the general purpose attributes that can be kept only for the sake of specifications.

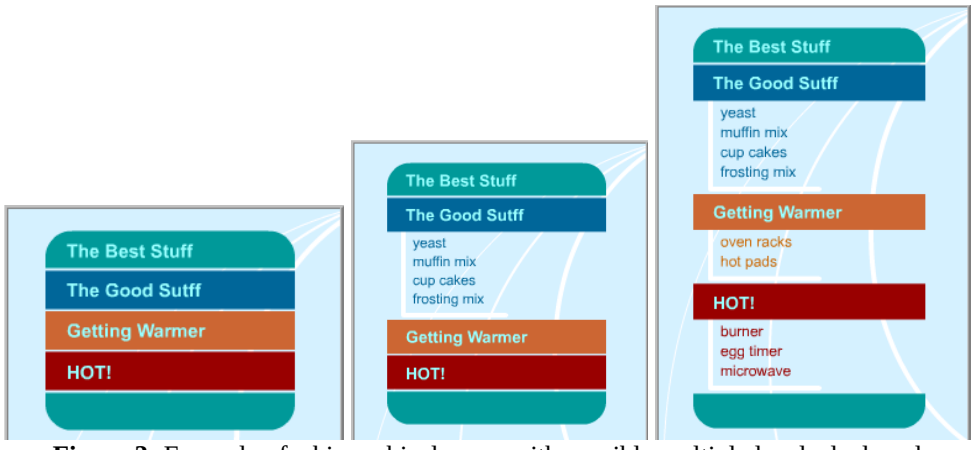

Figure 3: Example of a hierarchical menu with possible multiple levels deployed (Source: http://www.freehandsource.com/_test/hierarchical_menus.html)

Figure 4: Example of a purely hierarchical menu

Appendix

The web site of MenuSelector (Computer-Aided Design of Animated Web Menus in Flash) is accessible at http://www.isys.ucl.ac.be/bchi/research/selector.htm.

References

Jacko, J., & Salvendy, G. (1996). Hierarchical Menu Design: Breadth, Depth, and Task. *Complexity. Perceptual and Motor Skills*, 82, 1187-1201.

Kiger, J.I. (1984). The Depth/Breadth Tradeoff in the Design of Menu-Driven Interfaces. *International Journal of Man-Machine Studies*, 20, 201-213.

Norman, K.L., & Chin, J. (1988). The effect of tree structures on search in a hierarchical menu selection system. *Behaviour and Information Technology*, 7, 51-65.

Norman, K.L. (1990). The Psychology of Menu Selection: Designing Cognitive Control of the Human/Computer Interface. Norwood: Ablex Publishing Corporation.

Puerta, A., & Eisenstein, J. (2002). XIML: A Common Representation for Interaction Data. In Proceedings of 7th ACM International Conference on Intelligent User Interfaces IUI'2002 (San Francisco, January 13-16, 2002). New York: ACM Press. Accessible at http://www.iuiconf.org/02pdf/2002-002-0043.pdf

Scapin, D., Leulier, C., Vanderdonckt, J., Mariage, C., Bastien, Ch., Farenc, Ch., Palanque, Ph., & Bastide, R. (2000). A Framework for Organizing Web Usability Guidelines. In Kortum, Ph. & Kudzinger, E. (Eds.), Proceedings of 6th Conference on Human Factors and the Web HFWeb'2000 (Austin, 19 June 2000). Austin: University of Texas. Accessible at http://www.tri.sbc.com/hfweb/scapin/Scapin.html and http://www.isys.ucl.ac.be/bchi/publications/2000/Scapin-HFWeb2000.htm.

Vanderdonckt, J. (1998). Computer-Aided Design of Menu Bar and Pull-Down Menus for Business Oriented Applications. In Martins, F.M. & Duke, D.J. (Eds.), *Proceedings of 6th International Eurographics Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'99* (Braga, 2-4 June 1999). Vol. 2. Vienna: Springer-Verlag, pp. 73-88.

Vanderdonckt, J. (2003). Visual Design Methods in Interactive Applications. Chapter 7. In Albers, M. & Mazur, B. (Eds.), Content and Complexity: Information Design in Technical Communication. Mahwah: Lawrence Erlbaum Associates, pp. 187-203. Additional web page accessible at http://www.isys.ucl.ac.be/bchi/members/jva/pub/visual.htm.

Zaphiris, P. (2001). Age Differences and the Depth-Breadth Tradeoff in Hierarchical Online Information Systems. In Stephanidis, C. (Ed.). Proceedings of 1st International Conference on Universal Access in HCI UAHCI'2001 (New Orleans, August 5-10, 2001). Mahwah: Lawrence Erlbaum Associates, pp. 540-544.