An interface design tool based on explicit task models

Tom Hinrichs, Ray Bareiss, Lawrence Birnbaum, and Gregg Collins
The Institute for the Learning Sciences
Northwestern University
1890 Maple Ave., Evanston IL 60201
(847) 467 -1870
E-mail: {hinrichs, bareiss, birnbaum, collins}@ils.nwu.edu

ABSTRACT

Producing high-quality, comprehensible human interfaces is a difficult, labor-intensive process that requires experience and judgment. In this paper, we describe an approach to assisting this process by using explicit models of the user's task to drive the interface design and to serve as a functional component of the interface itself. The task model helps to ensure that the resulting interface directly and transparently supports the user in performing his task, and serves as a scaffolding for providing in-context help and advice. By crafting a library of standardized, reusable tasks and interface constructs, we believe it is possible to capture some of the design expertise and to amortize much of the labor required for building effective user interfaces.

Keywords

Model-based Interface Design Tools, Task Analysis

INTRODUCTION

Because the success of an interface turns on how well it supports the user in what he is trying to do, task analysis and modeling have assumed a central role in the process of interface design. The current standard practice in this area is to model tasks empirically, on a case-by-case basis. We believe there are a number of advantages to taking a more design-oriented approach to the problem, in which models of particular user tasks are developed by combining and parameterizing entries selected from a library of standardized, abstract, and explicitly represented task models. In particular, by tying model components to appropriate interface objects, and combining and parameterizing those objects as the designer combines and parameterizes the corresponding models, it is possible to automatically compile a significant portion of the user interface [3, 5, 6]. We have developed a prototype tool, MODEST (MOdel-based Design Employing Standardized Tasks), based on this approach.

CHI '96 Companion, Vancouver, BC Canada
© 1996 ACM 0-89791-832-0/96/04..$3.50

THE DESIGN TOOL

MODEST consists of three components: A model editor, a dialog manager, and an interface previewer/graphical editor.

The Model Editor is a window that displays a graphical representation of a portion of the current model of the user's task. The designer chooses the next sub-task to refine by selecting a portion of the model.

The Dialog Manager is an interaction window in which the designer answers questions in order to set parameters of the task model, by either selecting from a menu of preexisting objects, or creating a new kind of object. This dialog proceeds in two phases. In the first phase, the designer specializes and instantiates the actions and entities of the task. In the second, the designer specifies the graphics, sounds, and movies that display the entities to the user.

The Interface Previewer/Editor permits the designer to arrange and scale the graphical objects on the screen and to run the interface to verify its behavior. Rather than trying to automate graphical layout, MODEST focuses on generating the appropriate interface behavior for graphical objects from the task model.

THE DESIGN PROCESS

As an initial design experiment, we have used MODEST to rationally reconstruct the interface for Sickle Cell Counselor, an educational program we previously developed [1]. That program involves a simulation in which the student takes a blood sample from a patient, runs it through a gel-electrophoresis machine, and interprets the results to determine the patient's blood hemoglobin type.

We developed an abstraction of this task called **Sample-Test-Interpret**, which comprises taking a sample of something, testing it for some property, and interpreting the results. Driving down a level, the **Sample** sub-task involves extracting samples (e.g., blood) from one or more sample sites (e.g., a person), perhaps by using some sample extraction device (e.g., a syringe), and placing the samples in a storage container (e.g., a test tube).

In the first phase of the dialog, the designer associates the entities to be manipulated in a particular application (e.g., a

syringe) with their corresponding roles in the abstract task model (e.g., sample extraction device). The key point is that the behavior of these entities, in functional terms, is largely determined by the nature of the task at this rather abstract level: A sample extraction device can either be empty or full. It can be filled, if empty, only when positioned over an appropriate sample site. It can be emptied, if full, only over an empty container.

In the second phase of the dialog, the designer must associate the entities with media elements that represent them in their different states. For example, what does a syringe look like when it is empty? What does it look like when it is full? Finally, he must choose interface idioms that correspond to actions in the model, e.g., dragging and dropping to represent movement of an object, or clicking a button to represent starting a process. This process continues until all the necessary media resources and interface idioms have been identified, at which point the designer previews, edits, and runs the graphical interface.

ISSUES

The viability of our approach depends on addressing two key challenges: Generativity and appropriate abstraction.

Generativity

Flexibility in supporting the design of interfaces for a wide range of tasks entails the ability to compose more complex, aggregate tasks from simpler ones. Achieving this sort of generativity is one of the chief design goals for our task modeling language. The temptation here is to try to provide a fine-grained, fully compositional programming language for task modeling. Even if achievable, however, this would not necessarily give designers the leverage we seek; in the worst case, task model-based design would simply reduce to programming once again.

The alternative we are pursuing is to develop hierarchical libraries of reusable, standardized sub-tasks that can be connected together like circuit boards on the back-plane of a computer, using a variety of combiners. This simplifies the job of the application designer, while maintaining a high degree of flexibility. The design of the sub-task models themselves, like the design of circuit boards, would be a more specialized job in our approach.

Appropriate Abstraction

The appropriate level of abstraction for task models is determined by a tradeoff between the desire for broad coverage and the need to provide as much specific information as possible about the functional behavior of the corresponding interface entities. The long-term feasibility of our approach depends upon being able to find a rich set of models that strike the appropriate balance. In other words, we are banking on the existence of a basic level for tasks, similar to Rosch's basic level for object categories [4]. Our investigations into such abstract tasks as Sample-Test-Interpret, Threat Detection, and Strategic Planning provide us some measure of optimism in this regard [2].

CONCLUSIONS AND CURRENT STATUS

We have described an approach to interface design based on combining and parameterizing explicit, standardized task models selected from a library. Our approach is embodied in MODEST, a prototype tool implemented in ScriptX, that automatically compiles large portions of the user interface for Sample-Test-Interpret tasks as the designer carries out this process. MODEST currently assumes that tasks are discrete and serializable, and so does not yet deal with real-time, concurrent, or continuous tasks. We are in the process of developing a task model for Threat Detection, which is leading us to address these issues.

ACKNOWLEDGMENTS

We would like to thank Nathalie Grué, Bob Hooker, and Chris Johnson for programming and content analysis. This work was supported in part by the Advanced Research Projects Agency, monitored by Rome Laboratory under contract F30602-94-C-0219. Chris Johnson is supported by a National Science Foundation Graduate Research Fellowship. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, and receives additional support from Ameritech.

REFERENCES

1. Bell, B., Bareiss, R., and Beckwith, R. Sickle Cell Counselor: A prototype Goal-Based Scenario for instruction in a museum environment. *Journal of the Learning Sciences 3*, 4 (1993/1994), 347-386.

2. Birnbaum, L., and Collins, G. Towards a general theory of planning and design. Technical report no. 44, Northwestern University, The Institute for the Learning Sciences, Evanston, IL, 1993.

3. Puerta, A., Eriksson, H., Gennari, J., and Musen, M. Model-based automated generation of user interfaces. In *Proc. AAAI '94*, pp. 471-477.

4. Rosch, E., Mervis, C., Gray, W., Johnson, D. & Boyes-Braem, P. Basic objects in natural categories. *Cognitive Psychology 18*, (1976), 382-439.

5. Szekely, P., Luo, P., and Neches, R. Beyond Interface Builders: Model-Based Interface Tools. In *Interchi '93*, ACM, (1993), 383-390.

6. Wilson, S., Johnson, P., Kelly, C., Cunningham, J., and Markopoulos, P. Beyond hacking—A model based approach to user interface design. In *People and Computers VIII, Proceedings of the HCI'93 Conference*, Cambridge University Press, 1993.