

MODEL-BASED USER INTERFACE DESIGN BY EXAMPLE AND BY ANSWERING QUESTIONS

Martin R. Frank, James D. Foley
Georgia Institute of Technology
{martin,foley}@cc.gatech.edu

ABSTRACT

Model-based user interface design is based on a description of application objects and operations at a level of abstraction higher than that of code. A good model can be used to assist in designing the user interface, support multiple interfaces, help separate interface and application, describe input sequencing in a simple way, check consistency and completeness of the interface, evaluate its speed-of-use and generate context-specific textual and animated help. However, designers rarely use computer-supported application modeling today and prefer less formal approaches such as using a story board of interface prototypes. One reason is that available tools use special-purpose languages for the model specification. Another reason is that these tools force the designers to specify the application model before they can start working on the visual interface, which is their main area of expertise. We present a novel methodology for concurrent development of the user interface and the application model which overcomes both problems by combining story-boarding and model-based interface design.

KEYWORDS: Story-boarding. User Interface Management Systems. Model-based User Interface Design.

METHODOLOGY

Model-based user interface design uses an application model as an executable specification of the interface to the application rather than a paper specification. The key idea is to explicitly represent knowledge that had been buried in code before. For example, the objects and operations of an application are represented in code but they are not normally accessible from outside the application code. An application model is accessible by the interface, the application and external tools at design and at run time. Examples of model-based user interface management systems (UIMS) are UIDE [1,4] and HUMANOID [3,5] which both use a high-level object-oriented application model. ITS [6] is an example of a UIMS which encodes detailed interface style rules in order to generate very high quality user interfaces from the model.

Model-based UIMSs impose a design methodology on their users which requires them to textually specify a model of the application behind the interface before they specify the interface itself. However, the model can be complex, and

many user interface designers prefer starting the design with a story board. In our new approach, we combine the power of model-based design with the intuitive appeal of story board based design.

We use an existing user interface design tool for the story-boarding which is a combination of an interface builder and an object-oriented drawing program [2]. Sequencing of the interface is achieved through an interpreted script language or through communication with an application program. This tool can be used for interface prototypes as well as for the actual application interface.

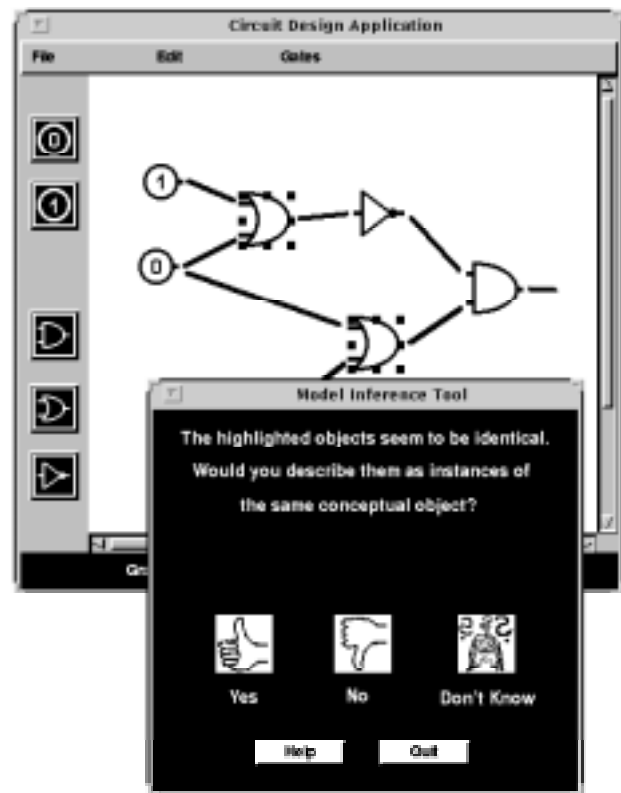


Figure 1: Inferring An Application Model By Asking The Designer Questions

In our methodology, the designer has the choice of starting a new design with the application model or with the user interface. If the application model is specified first our tool can generate a default interface from it, similar to other systems with an interface generation component. If the designer chooses to build the interface or story boards first, our tool

Cite as: M. Frank and J. Foley. Model-based user interface design by example and by answering questions. In *Adjunct Proceedings of INTERCHI, ACM Conference on Human Factors in Computing Systems*, pages 161-162, (Amsterdam, The Netherlands, April 24-29) 1993.

can infer an application model from the interface and from a dialog with the designer. Figure 1 shows a story-boarded prototype interface for a circuit design application in the background. Our tool infers an application model from one or more story-boarded interfaces. It is shown in the foreground asking the designer a question about the nature of the prototyped user interface elements. Not shown in this figure is the application model editor. As the designer answers a question the corresponding change of the model is highlighted in the model editor. In this way, the user learns about the effect of the dialogue and will soon be able to directly edit the model.

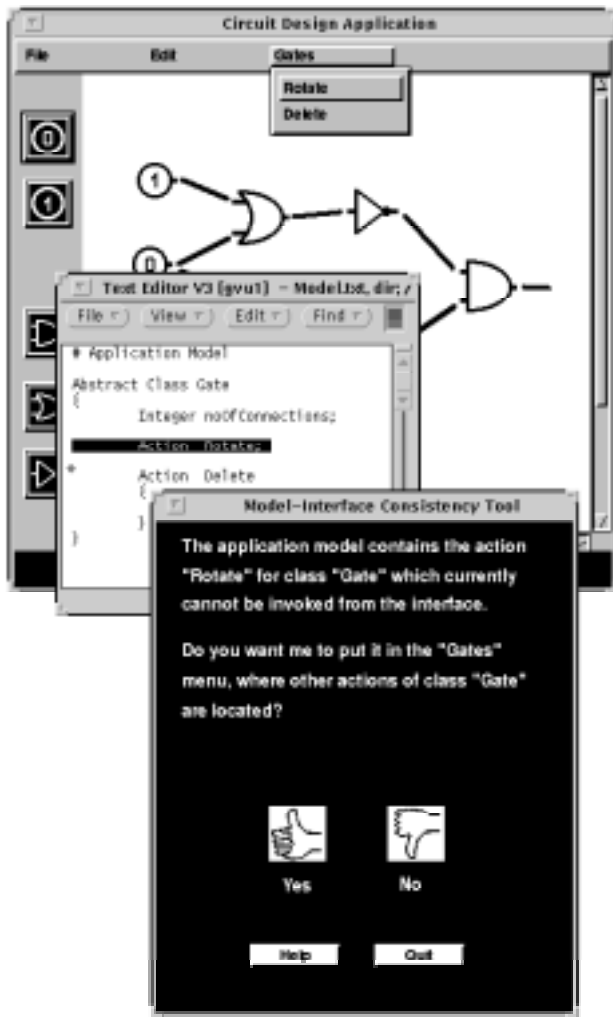


Figure 2: Keeping Model and Interface Consistent

In the later stages of application development, both the interface and the application evolve. In this stage, our tool keeps application model and interface consistent by asking questions. Figure 2 shows the circuit design interface in the background. The textual model of the circuit design application is located in front of it. Our tool in the foreground has detected that there is a model action which cannot be accessed through the interface and makes a suggestion for incorporating it into the interface.

The question generator is a crucial component of our system. It decides what questions are applicable in a certain situation and which one should be asked first. It is crucial because irrelevant or repetitive questions can easily become annoying to the user. One provision to avoid this is a two-level reasoning engine. For example, if nothing is known about a menu entry certain standard questions at the basic level are asked. If the user gives similar answers for consecutive menu entries the system will ask a question of the kind "Are the remaining menu entries of the same nature?", which is at a meta level. The graphical presentation of the questions and their context in the model and the interface also helps in making the question answering process less dry.

We chose the somewhat unconventional approach of asking natural language questions for knowledge acquisition because inferring an application model from a user interface is much harder than the reverse process. For interface generation, the model alone is sufficient. For model inference, the interface alone is not sufficient because the interface design decisions are not contained there. Thus, we must incorporate knowledge from the designer. We believe that asking natural language questions is the best approach for gathering this knowledge from a non-programmer.

CONCLUSIONS

The tool is in an early stage of implementation. Our belief is that this approach has the potential to bring application modelling to a wider audience because it does not require the interface designer to learn a formal model specification language but to just answer questions about why an interface was designed in a certain way.

REFERENCES

- [1] Foley, J., W. Kim, S. Kovacevic, and K. Murray, Defining Interfaces at a High Level of Abstraction, *IEEE Software*, 6(1), Jan. 1989, pp. 25-32.
- [2] Kuehme, T. and M. Schneider-Hufschmidt, SX/Tools - An Open Design Environment for Adaptable Multimedia User Interfaces, *Computer Graphics Forum*, 11(3), Sept. 1992, pp. 93-105.
- [3] Luo, P., P. Szekely and R. Neches, Management Of Interface Design in HUMANOID, *Proceedings of INTERCHI'93*, Amsterdam, Netherlands.
- [4] Sukaviriya, P., J. Foley and T. Griffith, A Second Generation User Interface Design Environment: The Model And The Runtime Architecture, *Proceedings of INTERCHI'93*, Amsterdam, Netherlands.
- [5] Szekely, P., P. Luo and R. Neches, Facilitating the Exploration of Interface Design Alternatives: The HUMANOID Model of Interface Design, *Proceedings of CHI'92*, ACM Conference on Human Factors in Computing Systems, Monterey, CA, pp. 507-515.
- [6] Wiecha, C. and S. Boies, Generating User Interfaces: Principles and Use of ITS Style Rules, *Proceedings of UIST'90*, ACM Symposium on User Interface Software and Technology, Snowbird, UT, pp. 21-30.