Video Prototyping: a technique for developing hypermedia systems

Wendy E. Mackay
Massachusetts Institute of Technology

CHI'88 demonstration
13 April 1988

ABSTRACT

Successful user interfaces create a interactive dialog between the user and the program that develops over time. Specific iterative design techniques, such as rapid prototyping and the *Wizard of Oz*, have proven particularly effective because they enable the program designers to test this interaction before the software is complete. This article describes a new technique called video prototyping which combines and extends these techniques, using real-time digitized video on a networked workstation. Video prototyping is most effective for developing highly interactive, information-intensive software, including hypermedia, educational software, simulations, and multi-media databases. Future uses will support the evolution of user-computer dialogs and user-user dialogs mediated by the computer.

INTRODUCTION

The term "user interface" is used to refer to a variety of issues such as how to lay out a computer screen, the appropriate use of input and output devices, and deciding whether or not to provide menus, commands or icons. Some user interface researchers concentrate on how to increase the perceived speed of a program or reduce the number of keystrokes necessary to accomplish a common task. While these are clearly important, they tend to divert attention from the interaction of the user with the program over time. This issue is particularly important with information- or communication-intensive software. Because different users use each program differently, the most critical user interface problems have less to do with speed and aesthetics and more with comprehensibility and ability to obtain and manipulate the appropriate information.

Individual users develop dialogs with these programs that evolve as the user's needs and level of understanding change. The user interface designer must understand the various interaction patterns possible and decide how best to support a range of user needs. Some dialogs will be fast-paced, as in video games. Others will be slower, as in educational software. The dialog may be initiated by the computer, as in a drill-and-practice teaching program, initiated by the user, as in a spreadsheet, or the control may switch back and forth, as in some hypermedia systems. The computer may be passive, as in a database or intrusive, as with an on-line tutor. Some programs allow users to specify the kinds of possible dialogs, others are limited to one or two. In any case, users develop a model of the system and how to communicate with it that forms the backbone of the user interface.

This article will describe software tools that use iterative design techniques to help create effective dialogs between users and software. These tools have been designed to facilitate development of highly interactive, information-intensive software, including hypermedia systems, databases, on-line documentation, computer-based instruction, simulations and intelligent tutors.

ITERATIVE DESIGN

Side bar: Iterative design refers to the process of creating a version of a program, testing it, and using the information to revise the software. These steps are repeated (iterated) until the authors are satisfied with the results (or run out of time and money). Iterative design is useful in situations where the authors have incomplete information, often the case in developing user interfaces. Rapid prototyping and the Wizard of Oz are both iterative design techniques.

Arguments abound in the software community about how best to design software. Some prefer a structured approach in which all of the components of the final software are identified in the initial design phase and then implemented accordingly. This strategy is suitable when the problem is well-understood, the end goal is well-defined, and the intermediate steps are specifiable in advance. An alternative strategy is to iteratively develop software, by creating prototypes and testing them. This strategy is more appropriate when the problem is poorly understood or evolving, multiple correct solutions are acceptable, and the intermediate steps are not specifiable in advance.

Information-based software, such as on-line documentation, intelligent tutors, and educational programs, tend to fall into the second category. In each case, the problem can be defined as providing the user with the optimal means of obtaining appropriate information. The user-computer dialog is the most critical part of the user interface design here. Designers of these systems must be prepared for the evolution of users' conceptions of the program as needs

---

In *CHI'88 Conference Companion Human Factors in Computing Systems*. Washington, D.C.: ACM/SIGCHI.

change, expertise grows and more possibilities present themselves.

Specifying all this in advance is difficult and sometimes impossible. Although guidelines exist for creating legible screen displays and determining which kinds of input devices are appropriate in which situations, we simply do not know enough to effectively predict the words of dialogs users will have with these systems. Just as it would be difficult to predict the possible avenues of a conversation with a friend, it is hard to predict the possible kinds of questions a user may ask of a system.

Several iterative design strategies have proven effective in addressing these problems. Rapid prototyping involves quickly building a prototype, trying it out on users, and then modifying the prototype as necessary until an acceptable state has been reached. The only problem is that building the prototypes may be as expensive as developing the final software. An alternative is a tool that imitates the software before it is actually built. This strategy has been dubbed the *Wizard of Oz* technique because of the resemblance to a scene in the movie of the same name. When Dorothy and her companions first appear before the Wizard, they see a huge head breathing smoke and speaking with a deep, impressive voice. Only later do they discover that the actual wizard is a frail old man, who creates the illusion of the Wizard by controlling various levers from behind a curtain.

The software version of the Wizard of Oz operates on the same principal. A user sits a terminal and interacts with a program. Hidden elsewhere, the software designer (the wizard) watches what the user does and by responding in different ways, creates the illusion of a working software program. In many cases, the user is unaware that a person, rather than a computer, is operating the system.

What makes this technique so useful? If user interfaces were limited to issues of screen design and input/output devices, the answer would be "not much". But if the scope of the user interface is expanded to include the dialog between the user and the computer, then being able to create, modify and understand interactions over time is critical. The cost of developing software prototypes can be very high, so it makes sense to let people imitate computers and take advantage of their insights to design more effective user-computer dialogs.

GENERALIZED WIZARD OF OZ TECHNIQUE

The most common use of the Wizard of Oz technique has been to develop natural language interfaces. Chapanis (1992) and Kelley (1993) used the term "Wizard of Oz" to describe their iterative development of a natural language interface to a checkbook and calendar management program. Wixon, Whiteside, Good and Jones (1993) used a variation of the technique to develop a natural language interface to an electronic mail system, based entirely on interactions between the user and the designer. Note that these interfaces involve obtaining information, and unlike video games or automatic bank tellers, fast response times are not the most critical issue. Instead, the major problem is to provide a reasonable way to give the user access to appropriate information.

A problem with this technique is that designers have had to build a special-purpose Oz program for each project. The Educational Services R&D group at Digital was interested using the technique for a variety of user interface design questions, so we developed a generalized version that allowed the designer to intercept text and graphics from almost any program running under the VAX/VMS operating system (Mackay, 1984, Mackay, 1986b). The program was used to help develop an automatic natural language generator (Van Praag, 1985), an intelligent tutor for a text editor (Mackay, 1985, 1988) and as a tool for exploring different ways of accessing a hypermedia database of educational material (Mackay, 1986a).

The generalized Wizard of Oz system runs in a window environment on a workstation. The designer can see different versions of the user's responses, including a picture of the user's screen and a log of the user's keystrokes. The latter is sometimes useful in interpreting just what the user did. When the user types something or makes a choice, the designer can allow the input to go directly to the program or intercept it and modify it. Similarly, the designer can intercept the response from the computer and modify it before sending it on to the user. In this way, the designer can control the interaction between the user and the program and handle problems from either the user's or the program's side.

Once this system was developed, it became clear that the Oz system could be extended beyond the simple one user-one researcher model. For example:

1. Controlled Experiments: A single researcher can provide exactly the same input to users in different conditions in an experiment.

2. On-Line Consulting: A single teacher can monitor several users at once and provide real-time advice.

3. Parallel Processing: Several researchers can observe the same user, but react to different kinds of behavior, simulating a real-time parallel processing system.

4. Expert Tutor Development: A chain of a student, an expert who watches the student and a knowledge engineer who watches the expert can help determine the rules the expert uses to tutor the student.

5. Teleconferencing: Several users may share information from any software package (spreadsheets, databases, editors, etc.) with each other, real-time.

6. Video prototyping: By adding video, users can do all of the above while watching each other (as in teleconferenced meetings). Video allows users to share anything that appears in the real world, not just each

other's faces, but software on incompatible machines, objects in the room, or scribbles on a piece of paper.

VIDEO PROTOTYPING

Video prototyping combines the generalized Wizard of Oz approach with video and sound to create a high-bandwidth system for developing and supporting user-computer dialogs. One can think of it as a method of developing functional specifications for interactive, multi-media software. Customers, marketing groups and upper level management can explore different possibilities and develop a deeper understanding of the eventual user-computer interaction. A version of a video prototyping system is being developed at MIT's Project Athena, using the video windows on the X Window system (Hodges, 1986)

Video is more than just another type of data, presented at 30 frames per second. A video camera can capture the sights and sounds of the real world. People can talk to each other, react to changes in body language and facial expressions and listen to changes in tone of voice. A user can send scribbled messages on a desk top to a colleague without translating them into computer-readable formats. A marketing manager can evaluate a competitor's software by watching a demonstration. He or she can even get a sense of what it's like to interact with it, if another person will act as the wizard. Video prototyping also allows designers to create both interactive and non-interactive demonstrations of software that has not yet been designed. Video is not a replacement for other kinds of information, but it certainly enhances the range of communication possible.

Multi-window workstations that support video are decreasing in price and increasing in popularity. Networks of these workstations will change the way we view both the development and the delivery of information within organizations. The concepts developed for video prototyping are the basis for a multimedia communication system, that blurs the distinction between prototype and finished software. People will be able to use the video prototyping environment to share information and interact with each other in a variety of ways. Individuals will change over time, organizations will evolve, and approaches such as video prototyping will be required to support both user - computer interaction and user - user interactions mediated by the computer.

REFERENCES

Chapanis, A. (1982) Man/Computer Research at Johns Hopkins, *Information Technology and Psychology: Prospects for the Future*. Kasschau, Lachman & Laughery (Eds.) Praeger Publishers, Third Houston Symposium, NY, NY.

Hodges, M. (1986) The Visual Database Project: Navigation. *Educational Services R&D Technical Report Series*, Digital Equipment Corporation, Bedford, Massachusetts.

Kelley, J.F. (1983) An empirical methodology for writing user-friendly natural language computer applications. In *Proceedings of CHI '83 Conference on Human Factors in Computing Systems*. Boston, Massachusetts.

Mackay, W.E. (1984) Wizard of Oz. *Educational Services R&D Newsletter*. Burlington, MA.

Mackay, W.E. (1985) Does Tutoring Really Have to Be Intelligent? (*Technical Report)* Digital Equipment Corporation, Educational Services. Bedford, Massachusetts.

Mackay, W.E. (1986a) Interactive Videodiscs: Database Driven Courseware. In M. M. Geerling (Ed.), *Computer Based Education in Banking and Finance*. Amsterdam: Elsevier Science Publishers B.V., North Holland, pp.169-178.

Mackay, W.E. (1986b) Integrated Learning Environments. In *EURIT 86: Developments in Educational Software and Courseware. Proceedings of the first international conference on education and information technology*. Oxford: Pergamon Press, pp. 29-34.

Mackay, W.E. (1988) Tutoring, Information Databases and Iterative Design. In D. Jonassen (Ed.), *Instructional Designs for Microcomputer Courseware*. Hillsdale, New Jersey: Lawrence Ehrlbaum Associates.

VanPraag, J. (1985) A New Approach to English Language Learning by Computer by Means of Template Extraction from Wizard of Oz Transactions. *Software Human Engineering Technical Report*, No. 413: Digital Equipment Corporation.

Wixon, Whiteside, Good and Jones (1993) Building a User-Derived Interface. In *Proceedings of CHI'83 conference on Human Factors in Computing*. pp. 24-27.