Kim Halskov Madsen and Peter H. Aiken

EXPERIENCES USING COOPERATIVE INTERACTIVE STORYBOARD PROTOTYPING

The interfaces of computer systems embedded in certain types of consumer electronic products are frequently targets of criticism. Perhaps the most frequently cited examples of poor user interfaces (UIs) are those associated with videocassette recorders (VCRs).

To cite a popular reference, the April 29, 1991 issue of *Business Week* reported only 3% of total television viewing time went to shows that were recorded by the users. Further surveys report high percentages of flashing VCR clocks, indicating the clocks are not set properly. Two of the obvious consequences of these poorly developed human-computer interfaces are: 1) a considerable portion of the population is unable to benefit from the primary functionality of these products; and 2) the loss of potential recording tape sales represented by the flashing clocks.

The need for active end-user participation in development activities has been acknowledged and is reflected as increasing interest in professional gatherings such as the Computer Supported Cooperative Work (CSCW) and Participatory Design (PD) conferences. Prototyping has contributed to the success of end-user design activities [6, 7]. Both theoretical arguments and empirical evidence indicate a strong cause-and-effect relationship between development approaches permitting realistic conditions for prototype evaluation and successful interface development efforts. Better solutions are achieved when the user is better prepared to participate in development processes [10, 11].

Storyboard prototyping is a variation of the general 'plan to throw one away in order to get it right' school of software development promoted by Brooks and others [1, 3, 8, 13]. Storyboard prototyping can be defined as ". . . a technique designed to generate consensus and closure via a tangible, interactive systems concept. It permits users to participate in the requirements validation process, and it provides an *audit trail* of the requirements analysis process. . . . The presentation of the interactive storyboard to users and managers is intended to evoke comments and criticisms" [13, p. 39]. As in film production, the use of storyboards in the development of computer systems is a way to 'sketch out' the future system early in the development process. In an effort to verify the requirements, the developer uses nonfunctional mock-ups, a technology dating at least to the 1930s, to illustrate a task-driven view of the proposed system for the user. The concept of *iteration as a discovery process* is the key to prototyping: each successive iteration brings the prototype one step closer to correctly representing the user needs.

Delay associated with production of the next version of the storyboard is a source of frustration for storyboard developers. Too much time between user review sessions leads to loss of cognitive momentum and can introduce errors to and perpetuate omissions in the development process.

Inspired by Scandinavian research into cooperative design [11], the thrust of the Cooperative Interactive Storyboarding Prototyping (CISP) approach is to more actively involve users in the prototype interface development. CISP empowers users with tools and techniques encouraging them to interactively contribute to real-time, storyboard use, evaluation and modification. Crucial here is the concept of the role of the user changing from *reviewer* to *codeveloper*. By drawing the user into the production process itself, CISP seeks to reduce the delays typically associated with the production of the 'next iteration' of the storyboard. In addition to reducing iteration delay, CISP also provides situations in which users can evaluate the storyboard prototypes under realistic circumstances and modify them in real time. Hence, the matter is not merely one of speeding up the design process but of creating a situation in which developers can respond to user requests in real time.

While the effort described in this article is very much 'work-in-progress,' we have some experience using CISP, having developed a VCR interface and evaluated the efforts of a small group of master's-level students, studying Human-Computer Interaction, as they used the interface to complete certain VCR-specific tasks. Preliminary results are promising. When applied to prototyping, the increased, collective, cognitive momentum seems to favorably affect the quality of the resulting development efforts.

THE CISP-TOOL

A large part of the impetus for this investigation came shortly after we received a new piece of multimedia equipment for the Hypermedia Technologies Laboratory. We anticipated the unit, the NEC PC-VCR S-VHS videocassette recorder with an RS-232 interface, would be an important tool—augmenting our efforts in the lab to apply hypermedia technologies to the analysis phase of decision making and problem solving. While the unit was tech-

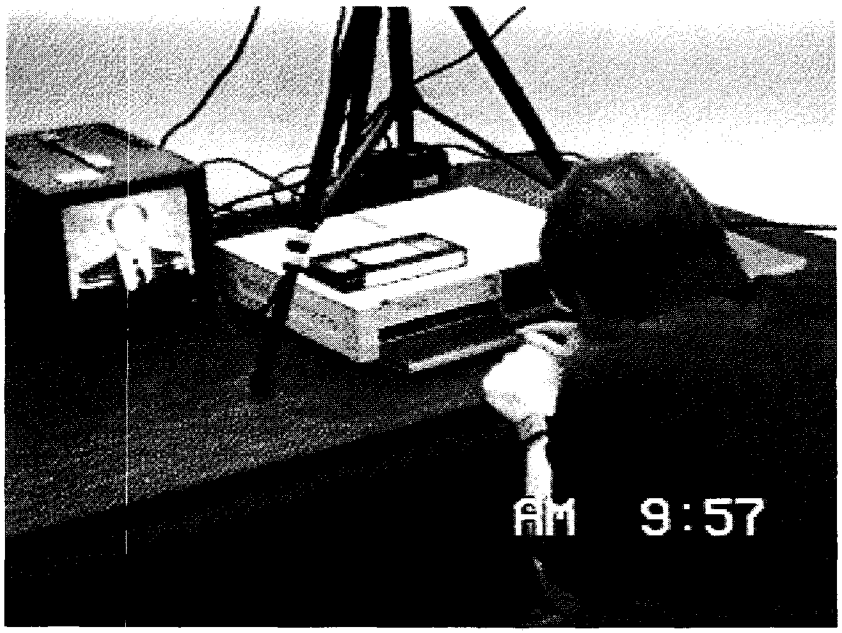


Figure 1. Video recording setup and view from scene camera

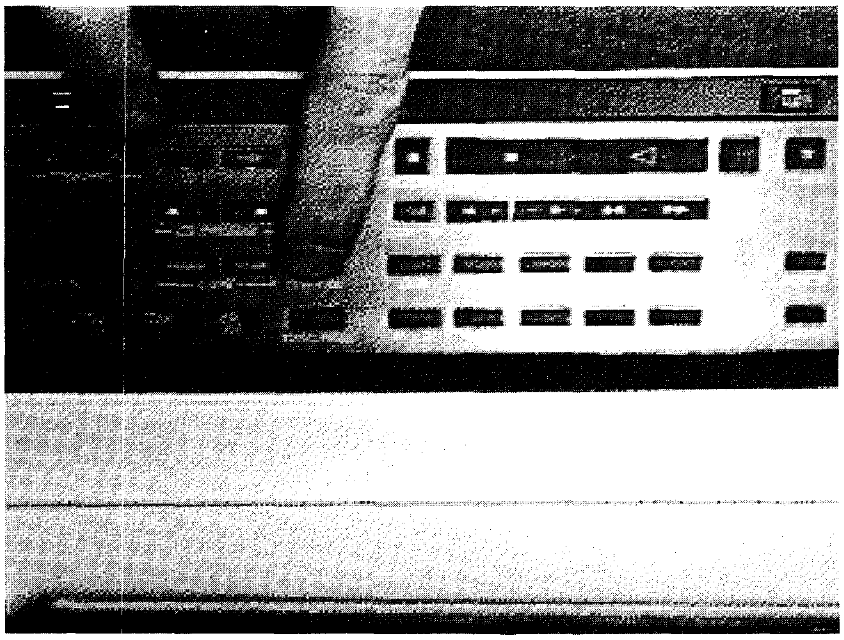


Figure 2. View from overhead camera

nically functional, correctly interfacing with the lab's Macintosh™ computer network, the user-computer interface was no better and in some ways worse than the hundreds of other available VCR interfaces, lacking features such as on-screen programming.

We decided that anything we could do would be an improvement over the current interface and began analyzing user interaction with the existing interface. As part of our analysis, we attempted to use scene and overhead cameras, as shown in Figures 1 and 2. Finding the 'talking-out-loud protocol supplemented with video images' approach unsuitable for eliciting user comments and even determining what buttons were pushed, we began to develop the CISP-Tool with an eye toward creating a solution permitting a general approach to these difficult problems. VCRs have limited interaction modes; most user actions consist of pushing buttons. Limited interaction modes made it easy to develop a storyboard for this prototype.

The CISP-Tool is an extension of Apple's HyperCard™, offering a series of features built on top of the standard HyperCard capabilities. (HyperCard—bundled as an extension of the Macintosh system software—is typically described as a multimedia-based personal tool kit offering a scripting language and a UI metaphor consisting of buttons and text fields, most commonly represented as 3-by-5 note cards.) By building on HyperCard we can use external commands and functions to extend possible user interaction modes beyond button pushing—allowing developers to create fully functional interfaces such as the one we created for the PC-VCR.

Interface Development Using Domain-Specific Building Blocks

The more users can work with familiar objects the better they will be able to relate to the development process. CISP supports collaborative interface development by permitting the user to combine building blocks made of domain-specific objects. Figure 3 shows a sample building block, switches used to move between system states such as "power on/off," "s-vhs on/off," and "record speed fast/slow." The lower part of the illustration represents the VCR control panel and the upper part represents the VCR display panel. As the switch is turned on and off, the display changes accordingly. Typically, this type of building block is created in HyperCard using multiple button and text fields. CISP permits developers to deal with these separate objects as a single composite object. Switch objects (and others) can be duplicated using a single copy/paste operation and, if necessary, modified afterward. It can be moved around the screen as a unit using keyboard adjuncts.

The lower part of Figure 4 represents the VCR control panel, while the upper part represents the VCR display panel. The figure shows two sets of domain-specific building blocks. Users can set the day of the week by clicking the button corresponding to the correct day and the display changes accordingly. The time control can be set by first clicking the 'time' button and subsequently clicking the 'up arrow' or the 'down arrow' buttons.

Using a single menu selection CISP offers the possibility of altering the display panels and control panel styles. Figure 5 illustrates how the day of the week alternatively can be set by first clicking the 'day' button and subsequently clicking the 'up arrow' or the 'down arrow'. Users and developers can rapidly evaluate the effectiveness of different combinations during design sessions. Again these building blocks were created from collections of buttons and fields—standard HyperCard objects. Additional building blocks of this type can be created with copy/paste and subsequent labeling operations aided by a dialog box. These individual display objects can be manipulated as a single object once an initial configuration has been selected.

An additional concept illustrated in Figures 4 and 5 is use of the gray shade. Gray shades emulate the use of plastic covers on VCR interfaces used to reduce interface complexity by hiding panels controlling more complex features. Users can reduce clutter by determining how much, if any, of the interface features are "hidden" by the gray shade permitting developers to quickly modify the level of interface complexity. In this way, the CISP-Tool encourages rapid generation and efficient evaluation of a large number of completed, or partially completed, potential design alternatives.

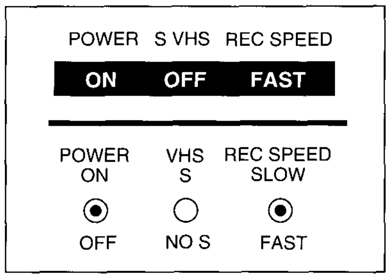


Figure 3. CISP building blocks

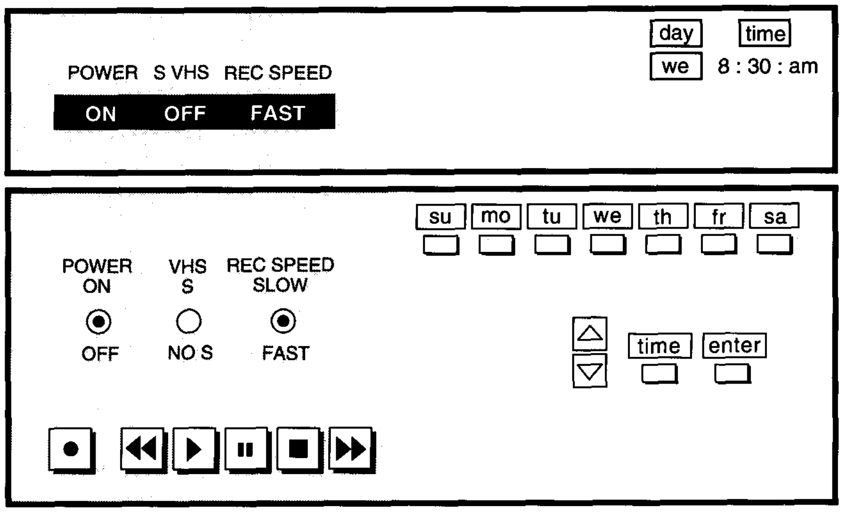


Figure 4. Two kinds of interface building blocks

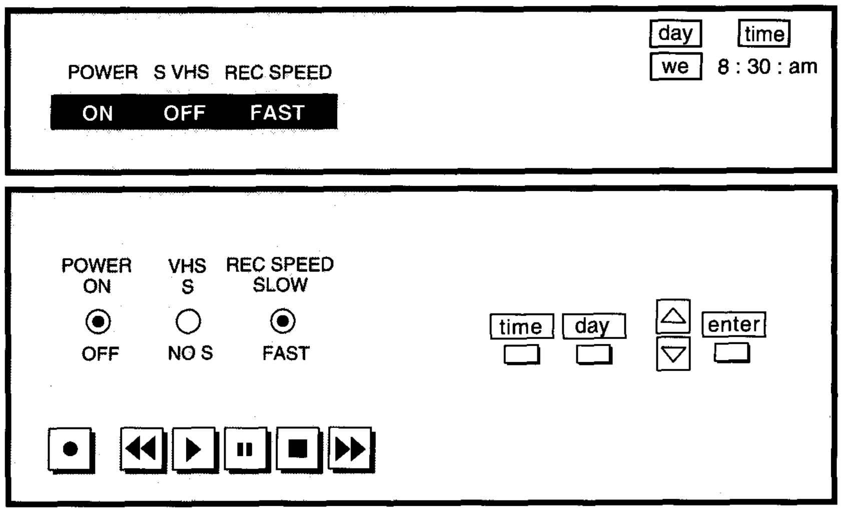


Figure 5. Alternate means of representing system interfaces and states

Support For Interface Use, Evaluation and Modification By Users

Current prototyping tools do not provide much support for system use and evaluation. HyperCard's inherent ability to mimic other interfaces makes it an ideal base for adding application-specific functions via externals enabling the creation of evolutionary prototypes [9]. This allows the user to interact with and evaluate a close surrogate of the system under development in order to quickly 'get a feel' for it. CISP is capable of capturing and reproducing user actions as they interact with the storyboard. At the same time it eliminates one of the most glaring defects in our video capture process—identifying the specific buttons the user actually pushed during evaluation sessions. (Button selections were not clear even when user comments were added to the videotape.) By recording and 'transcribing' actions and permitting 'slow motion' replay of the user interaction, CISP provides users and developers the opportunity to play back and evaluate the *actual* rather than the *hypothesized* user-system interaction.

Related Approaches

There were three primary inspirations to the development of CISP. Each is discussed in the following paragraphs.

Trillium. Trillium is a computer-based environment for designing interfaces for machines such as copiers and printers [12]. We adopted a key Trillium philosophy: reducing the impact of the use-evaluate-modify cycle by offering developers use of an interactive interface construction kit. Trillium is an industrial design environment, while CISP is in the experimental stage. CISP expands the Trillium concept in two directions: first, it seeks to reduce the amount of programming required by substituting direct manipulation techniques such as clicking, dragging, and copy/paste. Second, Trillium is a tool for software developers. We suggest an expanded role for this kind of tool: to be used by end users working in conjunction with developers as the requirements for the product are being developed.

The Scandinavian Approach. In the Scandinavian countries, a longstanding tradition of focusing on end-user's needs and situation requirements emphasizes ideals such as "quality at work" and "workplace democracy" [2]. Research results of the 1980s included such concepts as "creating a design situation with similarity with the future use situation," "taking practice seriously," and "from human factors to human actors," (see [10, 11]). Inspired by research such as the UTOPIA project [4] and the COOP project, one of the promising paths currently being investigated has end users actively taking part in design by applying a cooperative prototyping approach. According to Bødker and Grønbæk, a major obstacle is the developer's limited ability to respond smoothly to user ideas or requests for prototype changes during design sessions [6]. Joint, cooperative development of the prototype permits users, as domain experts/lay developers, and professional developers to each contribute their knowledge to prototype development tasks. Building on cooperative development aspects of the Scandinavian approach, CISP seeks to further explore the potential resulting from combined user/developer abilities to manipulate computerized domain-specific building blocks during prototyping activities.

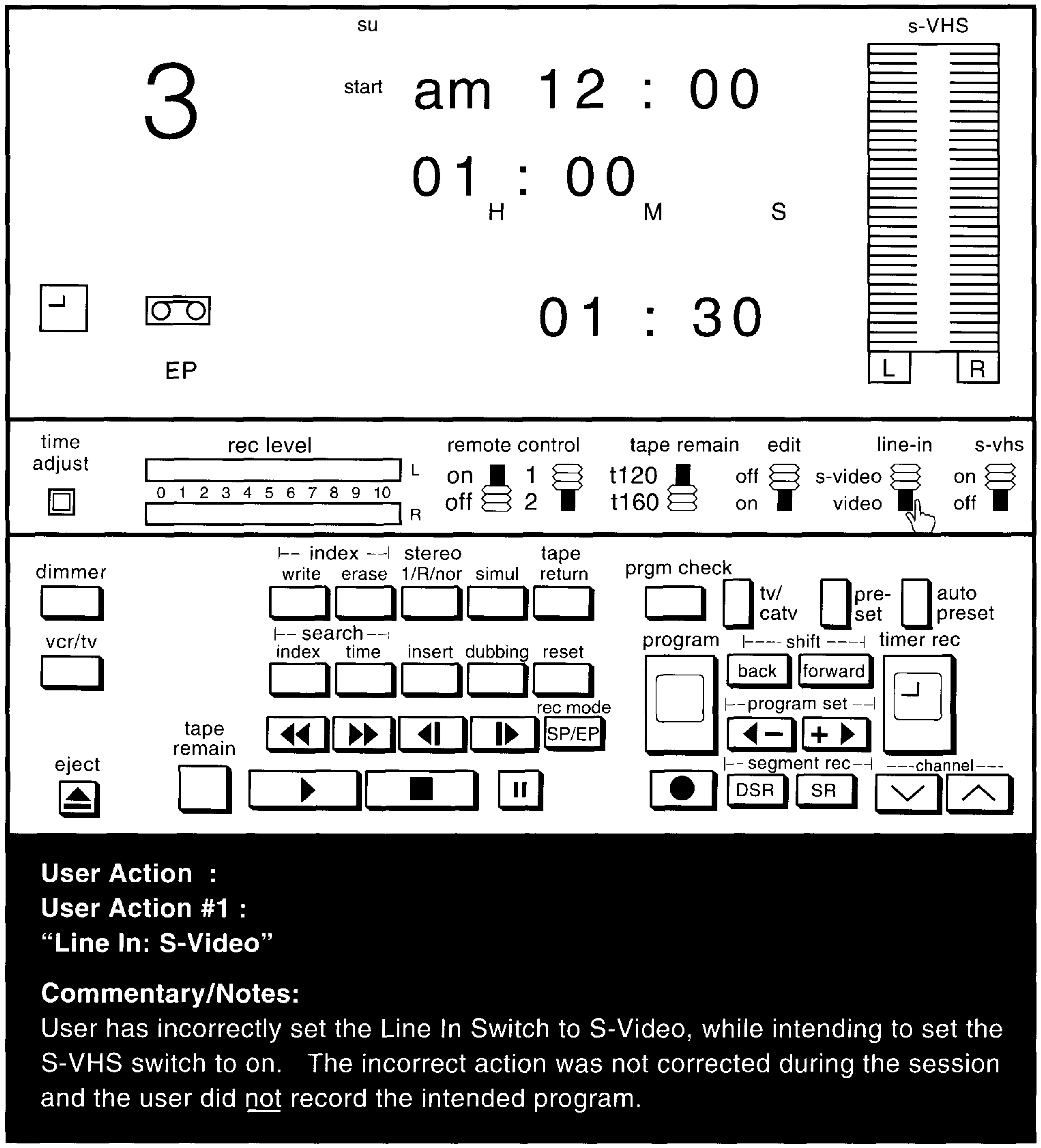
Pictive. Also inspired by the Scandinavian approach, PICTIVE (Plastic Interface for Collaborative Technology Initiatives through Video Exploration) is a participatory design (PD) technique combining the use of 'low tech' objects with video recording technology to increase end-user participation in system design; improve developers' ability to collect information about the use of the proposed system; and improve the sense of "design ownership" of the final product [14]. Major design components are literally made out of plastic and are thus seen by the users as malleable and adaptable. End users prepare and participate in job and task scenarios when they use the plastic components to evaluate aspects of the system development. CISP expands two aspects of PICTIVE: first, offering *computerized building blocks* to increase active user participation, and second, offering the possibility for users to evaluate *use* in context of the system being built.

The CISP Technique

Besides facilitated iteration, the CISP approach to interface development involves five specific techniques that function to complement existing prototyping strategies. Each is examined in the following paragraphs.

Realistic Prototype 'Look and Feel'. As stated previously, all of the user actions can be recorded, stored, and played back for later discussion between users and developers. This created a more realistic evaluation situation without the need for "think-out-loud" techniques. Instead, more realistic evaluation sessions permit more accurate assessments to be made of user-prototype interaction. Figure 6 shows the storyboard prototype for the NEC PC-VCR created using CISP. The upper half of the storyboard represents the display panel and the lower half represents the two panels with the various buttons and switches. The third section of the storyboard records the target of each button selection and provides additional space for user/evaluator comments as described in the previous section. (For more detail see our chapter in [13].) Since we chose not to simulate the television monitor, we asked users to perform tasks in which it was not essential. User interaction with the storyboard was evaluated as users attempted specific tasks. Replaying the actions of each user during subsequent analysis permitted the developer to ask questions such as: "I notice that when you were trying to set the clock you clicked twice on the timer button—was there some confusion there?" (An enhancement to CISP would involve user ability to add voice annotations to the storyboard.) The results of this analysis helped us to better understand the actual effectiveness of the proposed interface design solutions.

Interactive Modification. Because the composite domain-specific build-



User Action :
User Action #1 :
"Line In: S-Video"

Commentary/Notes:
User has incorrectly set the Line In Switch to S-Video, while intending to set the S-VHS switch to on. The incorrect action was not corrected during the session and the user did not record the intended program.

Figure 6. NEC PC-VCR interface layout created using the CISP tool

ing block can easily be copied and moved around as a single object, CISP enables users and developers to interactively and collaboratively modify the storyboard in real time. This is accomplished by dragging controls and displays to different locations (responding to comments such as "How about if we move that switch over there?"), duplicating existing screen objects ("Can you make this function react like that one?"), and creating new objects ("This function really shouldn't be lumped together with those controls!").

Generating and Comparing Design Alternatives. An essential part of CISP is to generate and compare several completed, or partially completed, potential development solutions, analyzing the trade-offs made in each potential solution. Consider an information display-oriented interface such as a monitoring/reporting system for a vehicle. Various types of user evaluation can be conducted to determine the effectiveness of alternative display modes during user evaluation sessions. A

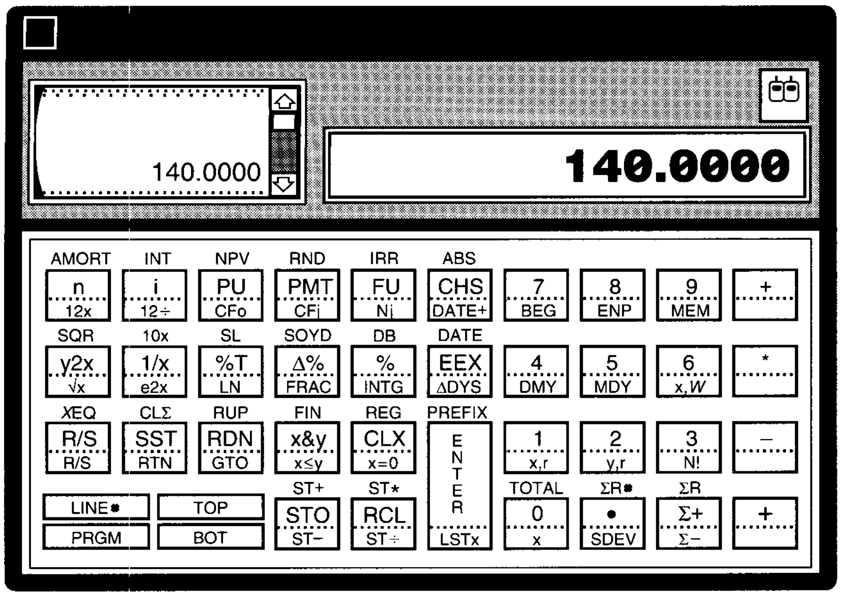

140.0000
140.0000
AMORT INT NPV RND IRR ABS
n 12x | i 12÷ | PU CFo | PMT CFj | FU Nj | CHS DATE+ | 7 BEG | 8 ENP | 9 MEM | +
SQR 10x SL SOYD DB DATE
y2x √x | 1/x e2x | %T LN | Δ% FRAC | % INTG | EEX ΔDYS | 4 DMY | 5 MDY | 6 x,W | *
XEQ CLΣ RUP FIN REG PREFIX
R/S R/S | SST RTN | RDN GTO | x&y x≤y | CLX x=0 | ENTER LSTx | 1 x,r | 2 y,r | 3 N! | –
ST+ ST* TOTAL ΣR• ΣR
LINE• | TOP | STO ST– | RCL ST÷ | 0 x | • SDEV | Σ+ Σ– | +
PRGM | BOT

Figure 7. Another realistic interface prototype

single user can interact with a whole series of development alternatives and then choose the screen display elements producing the "best" information display. Another use of this facility could be the development of a "creativity audit trail"—an automated means of tracking the way in which new solutions are created without having to interrupt the creative process.

Creation of Families of Systems. CISP facilitates the creation of families of systems in which all systems share several core components but differ slightly from one another in other respects (consider real-time command and control systems such as those being created for the NASA Space Station Freedom Project). This is accomplished by constructing the storyboard of common composite objects and introducing slight variations. Another example could be a real estate multiple-listing system that needs to be customized for each individual county because of particular local reporting requirements. Systems for counties needing specific groups of information could also be prototyped using this theme and variations approach, having each prototype start with the core screen elements and then adding group 'X' objects in response to type X requirements, for example.

Realistic Interface/Prototypes. By building on the existing HyperCard/SuperCard strengths, CISP users are able to interact with realistic prototypes and get a feel for the system being developed. This includes the ability to use interface components such as sliders, switches, and other types of specialized controls. CISP-Tool can be used to create and evaluate specific UIs, responding to conditions such as low light or presentation of highly complex information. This approach to prototyping is not unique—Figure 7 illustrates a realistic prototype interface for a Hewlett-Packard 12-C calculator created using a 'calculator construction set' and placed on a freeware disk.

Initial Results/Experiences

To evaluate the effectiveness of the CISP approach, we repeated the series of exercises conducted using the 'talking-out-loud protocol, supplemented with video images' approach previously mentioned. We evaluated tasks, including several variations of taping programs and controlling the playback of taped programs. After this evaluation we decided to ask volunteers to: 1) set the VCR clock to the current time and 2) program the VCR to record a segment from a particular channel. We explained the two tasks to each volunteer, provided a copy of the operating manual and left them alone as they attempted to complete the tasks.

We evaluated the attempts of 10 student volunteers from the Masters in Software Systems Engineering program at George Mason University, who joined us in the Hypermedia Technologies Laboratory for the evaluations. All but one of these students completed the two tasks, the average taking 15 minutes and longest taking 45 minutes. Half of the student volunteers used the documentation supplied with the PC-VCR to aid in task completion. As each student volunteer clicked on buttons, attempting to get the VCR to perform desired operations, the CISP-Tool recorded the name of each button selected by the user. After completion of each attempt we used the CISP-Tool to produce a HyperCard stack capable of physically reproducing each mouse click. That is, the tool created a HyperCard stack with a card corresponding to the location of each mouse click. As we "page" through the stack from card to card, the tool indicates the name and location of each mouse click. By selecting an additional option, we can have the results of each individual click performed by the tool. For example, moving from one card to the next would show the student volunteer clicked on the "Time" button and the tool would send the VCR the instructions to react as if someone had pushed the "Time" button on the VCR. This enabled us to recreate and play back the student volunteer's interaction with the VCR interface with the user present, so we could more carefully study the interaction.

To analyze user interaction, we used CISP capabilities to play back the solution executed by the user. We asked the users to 'walk us through' the various keystrokes representing their path through the interface. The combination of the button names presented in context with the appropriate interface response seemed to prompt the memory of many users, who were able to describe what they were considering as they tried to complete the tasks. Their comments were recorded on a comment field provided below the interface shown in Figure 6.

For the final third of each protocol, the users were shown how to use the CISP-Tool features, permitting them to modify the interface. Then they were invited to make any changes they desired to the interface. With assistance from the authors, users rearranged button clusters, added graphics, eliminated comment fields, and made a number of other modifications to the original NEC interface to the PC-VCR. Analysis of these experiences has permitted us to identify some promise and problems associated with CISP.

Promise

Though more effort is needed to make the domain-specific building blocks, nonetheless a significant work-load savings was achieved by using CISP to create prototype interfaces of how an improved system might look. The effort reduction seems to be particularly applicable to the development of families of systems. For instance, building the switch shown in Figure 3 from scratch requires numerous switches between the HyperCard tools and a number of other actions to make each element separately. Our environment has reduced this task to a single copy/paste operation. Similarly, the ability to move composite objects without the HyperCard grouping tool is a noticeable improvement. Features such as these make it significantly easier to build and modify multiple development alternatives. Users and developers particularly liked the ability to replay the session for subsequent evaluation. A typical reaction from a user was "(without the replay facility) I wouldn't have been able to give as thorough comments."

Problems

Building our tool in HyperCard has provided us with a useful collection of interface components. But it was difficult to implement some data structures, and missing object-oriented features made it harder to create and modify domain-specific building blocks. Aggregation of primitive objects such as buttons and field into composite objects, like switches, has been handled either by naming conventions or use of the ID numbers automatically assigned to objects by the HyperCard environment. Though easy to handle in small scale, the approach could become unwieldy if care is not taken during the scaling-up process. Another problem we have encountered is HyperCard performance. In a complex prototype such as Figure 6, performance was less than desirable on the top of the line Motorola '030-based CPUs. We believe that while some recoding could speed things up, there is no substitute for faster hardware.

Conclusions

We have two primary motivations for increasing the effectiveness of prototyping efforts: 1) to obtain more active/interactive user participation in the use, evaluation and redevelopment of storyboards, and 2) technological tool development to help shorten the use-redevelop loop used in many storyboarding efforts. In the ideal scenario, users and developers will be able to use the types of storyboarding tools we have presented in this article to interactively change storyboard elements and immediately try out their changes. We are continuing our efforts to enable users and developers to modify the storyboard while minimizing the amount of programming required.

Postscript: Recently the makers of the PC-VCR have ceased production of the unit for unknown reasons.

Acknowledgments

We would like to thank Jonathan Grudin, Kaj Grønbæk, Jennifer Papp, Philip Sage, Kristine Thomsen, and Randy Trigg for many useful comments on earlier drafts. We would like to thank the participants from the SWSE 632 (UI Development) class who helped us to debug, participated and gave us feedback on this project. The article was written during Kim Halskov Madsen's stay as a visiting professor at the department of Information Systems and Software Engineering, George Mason University, during the academic year 1990–1991. The stay was financially supported by Knud Højgaards Fond, Christian og Ottilia Brorsons Rejselegat, Aarhus Universitets Forskningsfond, The Danish Research Council (grant number 5.26.18.29-1) all Denmark and by George Mason University. Peter Aiken's research was supported in part by the Virginia Center for Innovative Technology. ◘

References

1. Andriole, S.J. *Storyboard Prototyping: A New Approach to User Requirements* Analysis, First ed., QED Information Sciences, Wellsley, Mass., 1989.
2. Bjerknes, G., Ehn, P. and Kyng, M. *Computers and Democracy—A Scandinavian Challenge.* Aldershot, Avebury, Great Britain, 1987.
3. Brooks, F. Grasping reality through illusion—Interactive graphics serving science. In *Proceedings: ACM/SIGCHI Conference on Human Factors in Computing Systems* (May 1988), pp. 1–10.
4. Bødker, S., Ehn, P., Kammersgaard, J., Kyng, M. and Sundblad, Y. *A Utopian Experience: On Design of Powerful Computer-Based Tolls for graphical workers.* In [2].
5. Bødker, S., and Grønbæk, K. Cooperative prototyping: Users and designers in mutual activity. *Int. J. Man-Machine Studies 34* (1991).
6. Bødker, S., Knudsen, J., Kyng, M. and Madsen, K. Computer support for cooperative design. In *Proceedings for the Conference on Computer Supported Cooperative Work* (Portland Ore., Sept. 26–29, 1988). ACM, New York, 1988, 1981, pp. 377–394.
7. Connor J., and Shafer, L. *Structured Rapid Prototyping.* Prentice Hall, 1989.
8. Curtis, G., and Vertelney, L. Storyboards and sketch prototypes for rapid interface visualization. CHI-90 Tutorial, 1990.
9. Davis, A. *Software Requirements: Analysis and Specification.* Prentice-Hall, Englewood Cliffs, N.J., 1990.
10. Ehn, P. *Work-Oriented Design of Computer Artifacts.* Lawrence Earlbaum, Hillsdale, N.J., 1989.
11. Greenbaum, J. and Kyng, M., Eds. *Design at Work: Cooperative Design of Computer Systems.* Lawrence Earlbaum, Hillsdale, N.J., 1991.
12. Henderson, A. *The Trillium User Interface Design Environment.* In *Human Factors in Computing Systems,* M. Mantei and P. Orbeton, Eds. *SIGCHI '86 Proceedings,* (Apr. 13–17), pp. 221–227.
13. Madsen, K. and Aiken, P. *Cooperative Interactive Storyboard Prototyping.* In *Storyboard Prototyping: A New Approach*

to User Requirements Analysis, S.J. Andriole, Ed. Second ed., 1991.

14. Muller, M. PICTIVE—Exploration in participatory design. In [15] pp. 225–231, 1991.

15. Robertson, S., Olson, S., and Olson, J., Eds. Reaching through technology. In *Proceedings of CHI '91* (New Orleans), Addison-Wesley, Reading, Mass., 1991.

CR Categories and Subject Descriptors: D.2.2 [**Software**]: Software Engineering—*Tools and Techniques*; K.6.1 [**Management of Computing and Information Systems**]: Project and People Management

General Terms: Design, Human Factors

Additional Key Words and Phrases: CSCW

About the Authors:

KIM HALSKOV MADSEN is an associate professor in the department of Information and Media Science at Aarhus University, Denmark. Current research interests include PD, object-oriented design, tailorability, and metaphorical design. **Author's Present Address:** Aarhus University, Information and Media Science, DK 8230 Aarhus N, Denmark; email: halskov@daimi.aau.dk

PETER H. AIKEN is the director of the Hypermedia Technologies Laboratory at George Mason University. Current research includes application of hypermedia-based tools and techniques to the process of software requirements engineering and the applications involving multimedia-based support for group decision making. **Author's Present Address:** ST2-430 George Mason University, Fairfax, VA 22043-4444; email: paiken@gmuvax2.gmu.edu

'Equal Opportunity' PD Using PICTIVE

Michael J. Muller
USER INTERFACE PRODUCT RESEARCH
US WEST ADVANCED TECHNOLOGIES
BOULDER, COLO.

Daniel M. Wildman
USABILITY ANALYSIS
BELLCORE,
PISCATAWAY, N.J.

Ellen A. White
USABILITY ENGINEERING AND TECHNOLOGY
BELLCORE
PISCATAWAY, N.J.

PICTIVE (Plastic Interface for Collaborative Technology Initiatives through Video Exploration) is a low-tech PD technique that has been used on products and research projects [4, 5, 7].

PICTIVE was a response to two PD trends: rapid prototyping and the Scandinavian mock-up approaches. Unlike rapid prototyping, PICTIVE does not involve a *technology* environment for design activity (e.g., Halskov, Madsen and Aiken, this issue). In rapid prototyping, the users often must express their ideas through an intermediary—the developer

Graphic Facilitation

Darlene Crane
CRANE CONSULTING
BERKELEY, CALIF.

Real work is not done in procedures manuals or workflow charts. People do work with movement, sound, thought, words, and sometimes passion in living color. Graphic facilitation, a conceptual analysis methodology used for 10 years, captures important and meaningful elements of work from the worker viewpoint. This technique is meant to be used with other more structured techniques. These visual working sessions give designers a record of work in a few days. The raw data from these sessions have enabled product designers to produce system designs in a few weeks.

Graphic facilitation sessions work this way: The facilitator plans the working sessions with key people in the design team and work group. They develop a working session plan, outlining group process and graphic formats. The physical site for the working sessions is set up in advance with butcher paper panels about 8- to 10-feet long and 4-feet high taped to the walls. Lots of colored pens are placed around the room. Technology can be used for recording, but tends to distract from group interaction.

During the session, the facilitator's role is to guide the workers in exploring their project. Freeform storyboards are usually the first step. The workers are encouraged to tell the "story" of their work in their own words. The key people, procedural steps, important systems support functions, feelings and stresses are recorded. The graphic facilitator asks questions and encourages the workers to talk. Simultaneously, the facilitator or a separate recorder captures the story on the walls covered with paper in simple images, shapes, key words and phrases. The facilitator and recorder DO NOT change the words of the workers or editorialize. Later steps in the session plan will clarify and analyze information necessary to design new technology systems and the corresponding organizational improvements.

What do designers get from having work illustrated graphically?

• *A selective record about work.* Video tapes and lengthy interviews sometimes provide so much data, key concerns of workers can be missed. Storyboards, wall paintings and other graphics help workers focus on the essential elements of the work process.

For example, a team was lost in the redesign of a complex front-end transaction-processing system. The central design theme of the system was identified when a worker finally said in a session, "All I want to know is the source of any errors!" The system was redesigned to focus on preventing and isolating errors rather than cranking out reports.

• *Encourages long-term memory of important elements of work without cumbersome written documents.* Workshop participants remember even detailed content of the sessions because the graphic records are in color and large (usually 4- by 8 feet.) Participants in sessions can readily recall what happened