EDIT WEAR AND READ WEAR

William C. Hill and James D. Hollan
Computer Graphics and Interactive Media Research Group
Bellcore, 445 South Street, Morristown, NJ 07962-1910

Dave Wroblewski and Tim McCandless
US West Advanced Technologies
4001 Discovery Lane, Boulder, CO 80303

Email: willhill@bellcore.com, hollan@bellcore.com, davew@uswest.com, mccand@uswest.com

ABSTRACT

We describe two applications that illustrate the idea of *computational wear* in the domain of document processing. By graphically depicting the history of author and reader interactions with documents, these applications offer otherwise unavailable information to guide work. We discuss how their design accords with a theory of professional work and an informational physics perspective on interface design.

Keywords: Graphical user interfaces, informational physics, interface mechanisms, professional work, reflective practitioner.

INTRODUCTION

The research described here grew out of a question implicit in Schoen's [11,12] analysis of professional work: how might we employ computation to improve, what Schoen refers to as, the *reflective conversation with work materials*? Previously we have addressed this question in the realm of computer systems administration [14]. That effort focused on the application of an object-oriented system, DETENTE, to embed agendas in complex application interfaces, to track and handle scheduled and unscheduled computer maintenance.

In this paper we address the same underlying question in the more general realm of document creation. The basic idea is to maintain and exploit object-centered interaction histories:

> *Record on computational objects (e.g. documents, menus, spreadsheets, images, email) the events that comprise their use, and then, on future occasions, when the objects are used again, display useful graphical abstractions of the accrued histories as parts of the objects themselves.*

Two document processing applications resulted from considerations of this basic idea: *Edit Wear* and *Read Wear*. The choice of the term *wear* comes from an analogy to physical wear. Specifically, use leaves wear. By modifying an existing editor, these applications arrange for every edit of a document and every episode of reading to leave wear on the document. In the case of *Edit Wear*, this means to graphically portray the document's authorship history by modifying the document's screen representation. In the case of *Read Wear*, it means to graphically portray the document's readership history.

Using a technique called *attribute-mapped scroll bars* [15], wear appears to users as marks mapped onto document scroll bars in positions relative to line positions (see Figure 1). The length of the mark depicts the magnitude of the wear or other wear quantities. Attribute-mapped scroll bars can be used in a variety of ways. For example, we have used them to map word search hits in a document onto their respective scroll bar positions [15]. The placement of the edit wear and read wear information inside the scroll bar serves to frame the wear marks. Since the length of the scroll bar represents the length of the document and relative position in the scroll bar represents relative position in the document, the scroll bar provides a geometry within which to interpret the wear marks in relation to the structure of the document. As a display technique it has the nice property of reusing precious screen space. More importantly it co-locates information display with navigation control points, a topic we analyze later in the theory section.

Figure 1 shows five examples of what users see. Scroll bar (a) is a normal scroll bar unadorned with wear. Bar (b) shows a snapshot of edit wear on a document. The width of individual wear marks is proportional to the the largest magnitude of edits per line. The fact that some sections have been edited more than others is visible and it easy to get to those sections by clicking on them. Bar (c) is the same document at a later stage, with two categories of wear in right and left vertical scroll bar bands. Edit wear dis-

played in (b) has been compressed into the left band. A second category of edit wear is displayed in the right band. Groups of edits in the second category of edit wear are visible along with a smattering of small edits. Bar (d) shows total read wear on a source code file. Bar (e) shows the same read wear as in (d) but now partitioned in three bands according to its three constitutive categories.

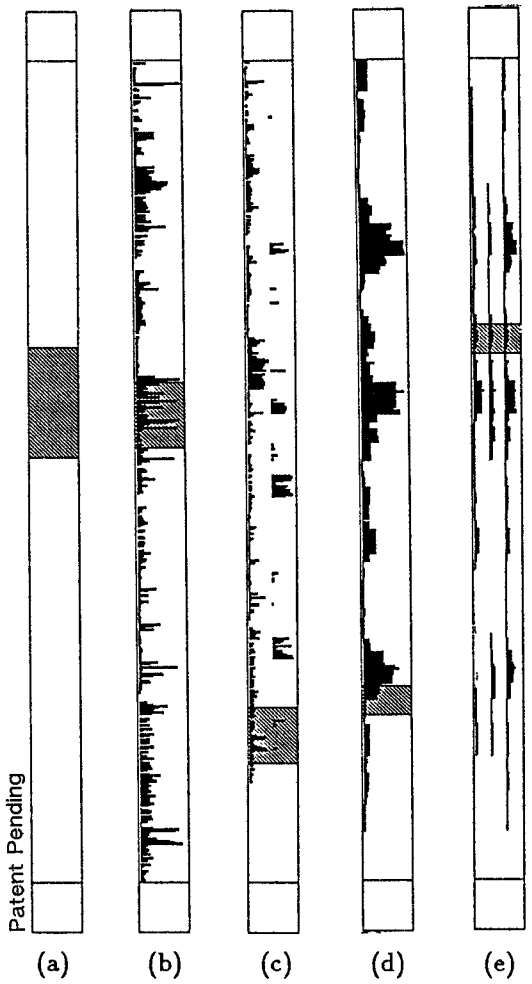

Figure 1. Five Sample Scroll Bars

By displaying an edit-by-edit history of a document in progress, *Edit Wear* graphically answers questions such as: Which sections of the document are most stable (i.e., changing the slowest)? Which sections are most unstable (i.e., current editing hot-spots)? What are the relative ages of document sections? How often have sections of the document been edited? What edits were made during the last editing session? In the case of co-authorship, edit wear distinguishes contributions by author, and answers questions graphically and immediately: Who wrote what? Who edited what and when did they edit it? What have co-authors written and edited since I last saw the document?

Similarly, by use of a line-by-line readership history, *Read Wear* addresses questions concerning how documents have been read: How often and how much have sections of the document been read? Which sections of this document been read by various categories of readers? Who were the last people to read this section, and when?

Answers to these questions should be useful in a number of areas, e.g. co-authored reports, large source-code libraries, and on-line reference document sets. But there is a cost to pay. Without compression techniques, saving all the extra history information on a per line basis results in storage costs being one to two orders of magnitude greater than without *Edit Wear* or *Read Wear*. We have not worked at all on optimizing storage but as storage costs fall, this becomes less of an issue.

From among the myriad ways one might implement the basic idea of *Edit Wear* and *Read Wear*, we chose to base it on a theory of professional work since reading and authoring are ubiquitous professional activities. We follow others [2,3,4,5,9] in applying Schoen's theory of professional work to the design of interfaces. The ways in which *Edit Wear* and *Read Wear* display their information were influenced by Schoen's theory, and much of their utility and usability can be described by Schoen's phrase, *the reflective conversation* that their design engenders.

The rest of this paper divides into two main sections. In the first, we describe the implementation: the role of *categories of wear* as a concept and data structure, selected aspects of how *Edit Wear* and *Read Wear* work internally, the state of our implementation, and undesirable properties of the current implementation. In the second section, we describe the theoretical underpinnings of *Edit Wear* and *Read Wear* and how they illustrate a number of interface design theses. In particular, we examine them from three perspectives: how they embody an interpretation of Schoen's theory of professional work, how they exemplify an informational physics view of interface design and might be generalized, and how they illustrate a computer-supported cooperative work thesis, namely that small group cooperation is better organized by shared artifact than by group process control.

IMPLEMENTATION

Edit Wear and *Read Wear* were implemented by modifying Zmacs, the editor for Symbolics lisp machines. Zmacs is similar to Emacs. It supports both major and minor editing modes on a per buffer basis. Major modes are mutually exclusive specializations for editing specific types of documents. Lisp mode, C mode, and Tex mode are examples. Minor modes are editor specializations that work within all major modes and co-exist with other minor modes. Auto Fill mode, Abbreviation mode, and Electric Font Lock mode are examples. *Edit Wear* and *Read Wear* are implemented as minor modes.

One turns on *Edit Wear* mode and *Read Wear* mode on a per buffer basis. Normally *Edit Wear* is turned on in an editable buffer and *Read Wear* is turned on in a read-only buffer. Usually one doesn't have both modes on in a single buffer,

though nothing prevents this. *Edit Wear* and *Read Wear* history data persist across editor sessions. When document files are saved, a shadow file is automatically generated and saved to permanently record the editing or reading activity. Similarly, Zmacs was modified to restore the existing *Edit Wear* or *Read Wear* history when a file is loaded. From the user's point of view, nothing is different about the way one loads, edits, reads, navigates, or saves document files and no extra work is involved in getting the information that *Edit Wear* and *Read Wear* provide. What changes for the user is how the document looks and the addition of a few additional editor commands.

Categories are unique identifiers that serve to label categories of wear. Their purpose is to serve later as indices into wear history records according to category. A set of such categories is associated with each author and reader. Thus author *A* might have the list: (John Q. Public, project 0891, marketing, manager level A3), while reader *B* might have the list (Mary Doe, research, unix expert). An individual can have both author and reader category lists. When *Edit Wear* mode or *Read Wear* mode is entered for a buffer, the editor begins to record editing or reading events for that buffer for each of the active record categories. Editing or reading activity can then be indexed, filtered, and played according to these categories.

How *Edit Wear* and *Read Wear* Work

The main modification to Zmacs required to implement *Edit Wear* and *Read Wear* was to provide a hook that allowed an arbitrary function to be run whenever a document line was edited. Care had to be taken not to degrade editor performance. We experimented with two versions of edit activity recording. The first and most expensive indexes a timestamp for each edit and line for all active edit record categories. The second keeps a list of edit timestamps per line and increments sums of edits per category.

Read Wear attempts to record for each active category associated with the reader and for each document line how many seconds of reading per category the document line participates in. The resulting measures are approximate but precise enough to be useful. In our implementation, reading a line has three requirements: the line is visible in the editor window, a user is logged in, and no lack-of-interaction timeouts (e.g., screen-dimming) have occurred to indicate that the user is not attending to the screen. *Read Wear* keeps its own lack-of-interaction time-out, set currently to 3 minutes. So for each visible line *Read Wear* figures out how many seconds of viewing time it has received and counts that as read time. *Read Wear* suspends and resumes the count properly as users swap around to different buffers.

A few details are needed to explain how this works. Two buffer variables are kept updated: *last-displayed-lines* and *current-displayed-lines*. When text-movement commands are issued by keystroke or mouse, just before the window is scrolled, *last-displayed-lines* takes the value of *current-displayed-lines*. Just after the text-movement command has taken effect, *current-displayed-lines* is updated to whatever lines are now visible. After these two variable assignments are complete, an equality test is performed. If the two variable values are equal (meaning that the lines that are displayed haven't changed) no action is performed. If on the other hand, new lines are visible and some old lines are not visible anymore, processing proceeds. Newly visible lines are given a timestamp for when they became visible. Lines that went out of the viewer get read seconds computed by subtracting their *became visible* timestamps from the time of the text movement command. Nothing happens to the lines that remain visible across the text movement command. This is enough to produce the desired accumulation of read wear. Initial, save-time, swap-buffer, and end-session special situations proceed according to variants of this method.

Experimentation with the current implementation has pointed out a number of possible improvements. First, wear attaches to lines in the document. We would prefer wear attach to individual characters but the storage requirements jump by another two orders of magnitude. Per-character edit wear offers a precision that would be useful in the case of source code documents where single character differences matter. Per-word or per-token read wear is a compromise. Second, the legends that describe different type of wear show up in the editor mini-buffer window which is a half inch below the bottom of the document scroll bar. It is not as easy to apply the legend to the wear as one would like. Third, we have as yet done nothing about making wear show up on the text itself rather than just in the scroll bar. It would be interesting to explore uses of color and texture mapping for this. Fourth, wear doesn't show up in hardcopy. Since its common to use hardcopy to support co-authoring, a program that generates wear-displays on hardcopy would be useful.

THEORETICAL PERSPECTIVES

We now turn to examining *Edit Wear* and *Read Wear* from three perspectives: how they embody an interpretation of Schoen's theory of professional work, how they can be generalized as examples of an informational physics view of interface design, and how they illustrate a CSCW thesis, namely that small group cooperation is better organized by shared artifact rather than by group process control.

Schoen's Theory of Professional Activity

Schoen's theory implies a scheme of interface evaluation in terms of *reflective conversation.* We describe two theoretical constructs concerning reflective conversations and their application to interface design. We then note how physical wear in the world considered as an *interface* often succeeds in the implied interface evaluation scheme. From this we derive the notion of computational wear and show how *Edit Wear* and *Read Wear* fair in the same evaluation scheme.

Opposing the analytical view that "professional activity consists in instrumental problem-solving made rigorous by the application of scientific theory and technique" [11, p.

21], Schoen proposes a reflection-in-action analysis of professional work:

> *When the practitioner tries to solve the problem he has set, he seeks both to understand the situation and to change it. ... Through the unintended effects of action, the situation talks back. The practitioner, reflecting on this back-talk, may find new meaning in the situation which leads him to a new reframing.* ***Thus he judges a problem-setting by the quality and direction of the reflective conversation to which it leads.*** [11, pp.134-135, emphasis ours]

Here, Schoen emphasizes problem-setting over problem-solving. What does Schoen mean by problem-setting? For Schoen, *problem setting is a process in which, interactively, we name the things to which we will attend and frame the context in which we will attend to them.* [11, p. 40] Problem-setting precedes problem-solving. Considering interfaces for professional work in the light of Schoen's analytical point of view, we might add and paraphrase:

> *Interfaces permit and encourage certain problem-settings and should be judged by the quality and direction of the reflective conversations that result from the problem-settings they engender.*

Often interfaces presume an implicit immutable problem-setting and concentrate on supporting problem-solving within the resulting constrained framework. Interfaces don't often support the fuzzy work of problem-setting which according to Schoen is the hallmark of professional work. What Schoen terms the *perceptual emergence of the unnamed and unframed* is a critical aspect of supporting professional problem-setting in computation. For, unless an interface displays perceptual groupings that have yet to be named and framed in Schoen's sense, all phenomena are already labeled and classified and creative problem-setting is constrained.

Edit Wear and *Read Wear* were designed with this in mind. We wanted patterns of editing activity and reading activity to emerge on the documents. We meant the displays of wear to foster sense-making out of otherwise unavailable data. As documents are edited or read, wear builds up in various categories on a per line basis. Eventually, these new wear spots attract attention relative to other wear spots, occasioning the opportunity for authors or readers to name the pattern of wear (actually patterns of author or reader activity) with such phrases as "The July work", "Dave's section", "The network guys' release 2.0". Authors and readers can then use these now named patterns of wear to adjust what they are doing. *

Another notion from Schoen of importance to the design of *Edit Wear* and *Read Wear* is his concept of the *action present*:

> *A practitioner's reflection-in-action may not be very rapid. It is bounded by the* ***action present****, the zone of time in which action can still make a difference to the situation. The action-present may stretch over minutes, hours, days or even weeks or months depending on the pace of activity and the situational boundaries that are characteristic of the practice.* [11, p. 62)]

As a result of the desire to embed prior activity information in the action present, *Edit Wear* and *Read Wear* were designed to make patterns of past use apparent during all phases of editing and reading rather than as an after-the-fact summary to be consulted in some other context.

Edit Wear and *Read Wear* were designed to display themselves in the action present of document navigation, a document's scroll bar. One of the most beautiful aspects of this design is that document wear appears in the exact screen position on which a user clicks to scroll to the document section that has that self-same wear. SuperBook [1], a document-oriented information retrieval system, provides another example of this by graphically displays retrieval hits in the context of a document's table of contents. This embedded display method renders the distribution of retrieval hits interpretable and useful to readers.

Physical and Computational Wear

Another way of thinking about this is to employ the notion of physical wear as an organizing metaphor for what we wanted computationally. Physical wear is emergent and generally remains unnamed and unframed until it causes a problem. It is also embedded, unavoidably tattooed directly on the worn objects. It appears exactly where it can make an informative difference.

Consider some serendipitous uses of wear that everyday life presents. The bindings of cheap paperbacks bend and crack in a manner that allows one to find the last page read. In an auto parts store, the most often consulted pages among many linear feet of catalog are identifiable by smudges, familiar tears, and loose pages. The smudges, tears, and loose pages index to information users are likely to consult. Switching from auto parts catalogs to door handles, the polished part of an otherwise patinaed brass door handle shows where others succeeded in grasping it. The best recipes cards in a stack are often dogged-eared and stained. Weaver [12, pp.270-271] describes a rediscovery of the law of first significant digit distribution due to odd smudge patterns on logarithm tables. These examples remind us that wear sometimes encodes useful information.

Wear is gradual and unavoidable change due to use. As a source of useful information, wear is particular appealing

* We [6,10] previously demonstrated advantages of displaying unnamed/unframed perceptual configurations. It was observed that unplanned configurations of process control displays, computed from deviations from states of normal operation, have potential for assisting users in coming to consider alternative hypotheses and helping with what Norman [8] has termed *cognitive hysteresis.*

since it is a by-product of normal activity and thus essentially free. No extra effort, nor scheduling of additional tasks are required to get its effects. In the realm of computation we have rudimentary analogs of wear. For example, command histories and system activity logs accrete automatically. File descriptors usually identify the date and time the file was last touched. These examples bear a kind of superficial resemblance to physical wear. We are interested in extending the notion of wear to *computational wear* that might engender reflective conversations with useful task-specific properties. The notion of computational wear exemplifies a new conceptual framework for theorizing about and implementing interfaces. It is related to our work on informational physics [7]. We think of *Edit Wear* and *Read Wear* as examples of an informational physics for documents.

Taking a physics perspective provides another way of looking at the design of interfaces. Computation enables the creation of virtual worlds that resemble the real world and allow us to exploit our extensive knowledge of the world in interacting with them. This is certainly the primary benefit of taking a metaphor-based view of interface design. Of greater importance from an informational physics perspective is the fact that these same techniques also allow us to create virtual worlds that give concrete existence to abstract entities operating according to a physics of our choice. The entities and their physics can be designed to highlight aspects of phenomena not normally available to us but that are important for supporting understanding and task performance. For our point of view it is crucial to emphasize that the physics can be motivated by understandings of the characteristics of cognition and tasks. The lawful relations that make up the informational physics should be such that the interface operates in ways that are specifically designed to facilitate our apprehension of important relationships.

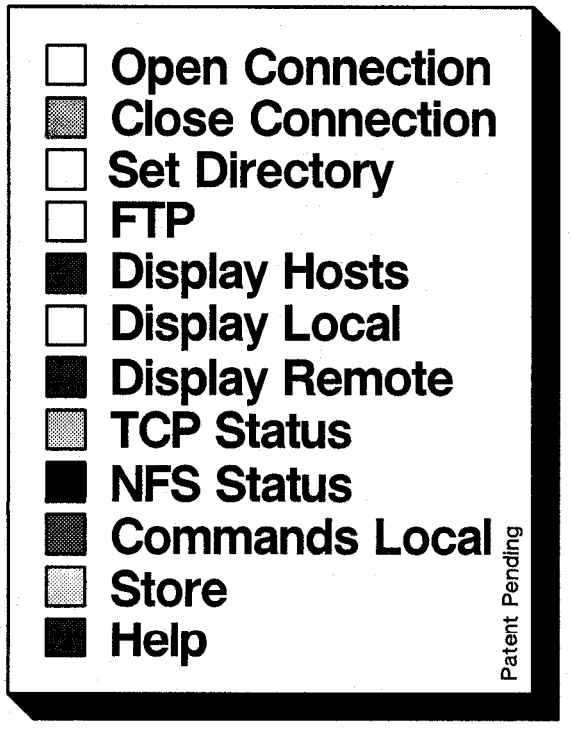

Figure 2. Mockup of Menu Wear exhibits embeddedness and emergence of the unnamed/unframed

Generalization of Edit and Read Wear

Edit and *Read Wear* are useful document processing facilities. But suppose we generalize their basic idea of recording user activity and displaying it later in useful ways to other areas of the interface. For example, consider menus. The idea of *Menu Wear* is that statistics of previous menu-selections by category of user and by category of context get painted onto the menu items themselves (see Figure 2)

Edit wear also makes sense for use with spreadsheets employed in *what-if* scenarios. Cells are colored according to the number of times they have been edited. Horizontal and vertical scrollbars are colored to show total edits by row and column. This kind of edit wear makes it apparent where users have been reworking budget lines the most.

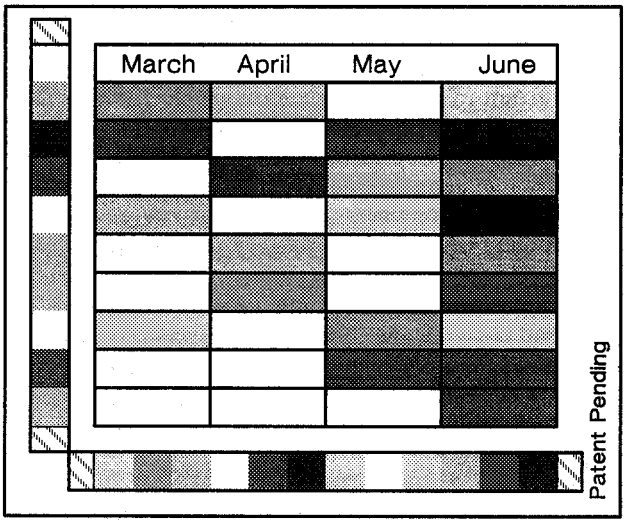

Figure 3. Mockup of Spread Sheet Wear

It is in keeping with the generalization of *Edit Wear* and *Read Wear* that *all* interaction histories should be recorded permanently, event by event, and be made accessible for later redisplay by interface objects such as menu-items, individual text characters in editor buffers, color swatches in paint programs, individual cells in spreadsheet programs, etc. Taking the generalization of *Menu Wear* and *Spreadsheet Wear* to the limit, we arrive at an intriguing and problematic interface design issue associated with permanently registering the interaction events of interface objects. We mention it here primarily as a topic worth further discussion.

Suppose *all* interaction history is recorded, structured and indexed for later use. In such a world, *every* menu choice, *every* document edit would be available. But to whom and for what purposes? To make the discussion more concrete, consider the records of editing and reading activity that

we've already collected with *Edit Wear* and *Read Wear*. To whom do they belong? Who should use them for what purposes?

We believe this raises significant ethical issues. One position is that the use of interaction event records should be untransferrably subject to the will of the participating users. *Edit Wear* would belong to the editor and *Read Wear* belongs to the reader. This topic is complex and we expect there to be vigorous discussion and debate about it.

Edit Wear and *Read Wear*: A CSCW Thesis

Edit Wear and *Read Wear* serve as kinds of computer-supported cooperative work applications in the sense that they mediate coordination and cooperation. Co-authors get more precise information about what each other is doing. Co-readers find out who else is reading a particular topic or which sections are read by specific individuals or groups. The CSCW thesis that *Edit Wear* and *Read Wear* illustrate is that there may be advantages to enhancing already-existing interaction-organizing artifacts, such as reports, diagrams, and code rather than posing new separate social process control artifacts.

Process control models of CSCW tend to be too restrictive for small groups. By virtue of their explicitness, process control models curtail improvisation which is frequently a resource for small groups. In contrast, *Edit Wear* and *Read Wear* do not attempt to control social processes at all. They present information that encourages efficient exchanges. A second difficulty with separate process control artifacts such as meeting schedules, work flow charts, and process control software, is that they must be continually updated to reflect changes that occur despite them. By using artifacts such as *Edit Wear* and *Read Wear* documents to organize interaction, there is no additional updating work required to maintain the process model. The documents and their wear are self-updating.

SUMMARY

In summary, *Read Wear* and *Edit Wear* modify document processing as we know it in three significant ways. They move some reading and editing from the realm of private to semi-public activity. In specific settings, the cost of this subtle cultural upheaval may be offset by advantages in coordination that the techniques offer. Second, they exemplify the trend of intentionally designing the forensic qualities of new computation-based media, a trend we may expect to see continue. Third, the wear patterns they display occasion useful conversations, increasing the probability of efficient and effective exchanges among collaborating professionals. The categorical indices into *Read Wear*, for instance, allows readers to find other readers with similar interests.

The entwined concepts of authorship and readership are changing in these ways to accommodate their emerging computational forms. *Read Wear* and *Edit Wear* are examples that provide novel utility in the realm of document processing, without imposing new demands on authors and readers. We have viewed these applications from the perspective of Schoen's theory of professional work, showed how they are examples of a more general informational physics perspective on interface design, and argued that their generalizations have wide applicability and raise important issues for interface design.

ACKNOWLEDGEMENTS

We thank Gerhard Fisher, Ray McCall, Andreas Lemke, Anders Morch, Mark Rosenstein, Larry Stead, and Loren Terveen for discussions about this paper and about the applicability of Schoen's theory to interface design.

REFERENCES

1. Egan, D.E., Remde, J.R., Gomez, L.M., Landauer, T.K., Eberhardt, J. and Lochbaum, C.C. Formative Design-Evaluation of SuperBook., ACM Transactions on Information Systems, 1989, 7:1, pp.30-57.

2. Fischer, G. and Lemke, A.C. Construction Kits and Design Environments: Steps Toward Human Problem-Domain Communication, Human-Computer Interaction, 1988, 3:3, pp. 179-222.

3. Fischer, G., McCall, R., and Morch, A. JANUS: Integrating Hypertext with a Knowledge-based Design Environment, Proceedings of Hypertext'89 (Pittsburgh, PA), ACM, New York, November, 1989, pp. 105-117.

4. Fischer, G., McCall, R., and Morch, A. Making Argumentation Serve Design, Tech. report, Department of Computer Science, University of Colorado, Boulder, CO, 1991.

5. Fischer, G., Lemke, A. C., Mastaglio, T., and Morch, A., The Role of Critiquing in Cooperative Problem Solving, ACM Transactions on Information Systems, 1991, (in press).

6. Hollan, J. D., Hutchins, E. L., McCandless, T. P., Rosenstein, M., Weitzman, L. Graphical Interfaces for Simulation. In B. Rouse (Ed.) *Advances in Man-Machine Systems Research*, Greenwich, CT: JAI Press, Inc, 1987, 129-163.

7. Hollan, J.D. and Hill, W. C. Towards an Informational Physics Perspective for Interface Design, In preparation.

8. Lewis, C. and Norman, D. Designing for Error. In D. Norman & S. Draper (Eds.) *User Centered System Design: New Perspectives on Human-Computer Interaction,* Hillsdale, NJ: Lawrence Erlbaum Associates, 1986, 411-432.

9. McCall, R., Fischer, G. and Morch, A. Supporting Reflection-in-Action in the Janus Design Environment in M. McCullough et al., (Eds.), The MIT Press, Cambridge, MA, 1990, pp. 247-259.

10. McCandless, T. P., PDP Mechanisms of Intelligent Display Control, Society for Computer Simulation Conference

on Intelligent Simulation Environments, San Diego, pp. 87-91, 1986.

11. Schoen, D. *The Reflective Practitioner: How Professionals Think in Action* Basic Books, New York, 1982.

12. Schoen, D., *Educating The Reflective Practitioner.* Jossey-Bass Publishers, San Francisco, 1987.

13. Weaver, W. *Lady Luck.* Double Day, Garden City, NY, 1963.

14. Wroblewski, D., McCandless, T., and Hill, W.C., DETENTE: Practical Support for Practical Action, Human Factors in Computing Systems, CHI'91 Conference Proceedings (New Orleans, LA), ACM, New York, 1991, pp.195-202.

15. Wroblewski, D., Hill, W.C., Mccandless, T. Attribute-mapped Scroll Bars, U.S. Patent Applications: Serial No. 07/523,117 filed May 14, 1990, Serial No. 07/626,130 filed Dec 11, 1990.