Asynchronous Design/Evaluation Methods for Hypertext Technology Development

Gary Perlman

Department of Computer and Information Science
The Ohio State University
2036 Neil Avenue
Columbus, OH 43210-1277
perlman@cis.ohio-state.edu

ABSTRACT

A process model used in the design and evaluation of hypertext systems is discussed. The model includes asynchronous processes of task analysis, document analysis, literature survey and systems evaluation, interpretation of data, designing and building systems, and collecting data. For each process, experiences with NaviText™ SAM, a hypertext interface to a reference source, are discussed. A variety of new methods for evaluation of experimental systems are presented along with several empirical results.

INTRODUCTION

The development of new hypertext systems is a candidate for some of the development techniques that have been successful in other dynamic fields. The iterative design and evaluation prototyping lifecycle used in user interface development and the experimental programming strategy used in artificial intelligence provide us with paradigms for exploring new possibilities for the online delivery of information. Figure 1 shows a data flow diagram of a process model of an asynchronous design and evaluation method I have used for developing systems, most recently hypertext systems. In this paper, I will illustrate this model of system development, discussing the model both in terms of my own work on the NaviText™ family of hypertext browsers ([Perl87], [Perl88], [Perl89b]), and in terms of other research. For each process in the model, there are issues in the methodology of developing a new technology that will also be addressed.

This document is based on a data flow diagram (see the primer below) that has been flattened out for presentation. The original document was built in the "Software Through Pictures" system [IDE86]. The numbered sections in this document are keyed to the activity numbers in the diagram. Being asynchronous, the actual order of activities may not match the order of the activity numbers.

This document will begin with a description of a data flow model of hypertext systems development research, including the processes and data (knowledge) stores. Then, experiences and results with NaviText™ SAM will be discussed in terms of the processes involved in its development. Ideally, the descriptions of the data flow elements would be popup notes, accessible during reading, but instead, some background on the data flow model must be covered. On a first reading, the rest of the introduction can be skimmed.

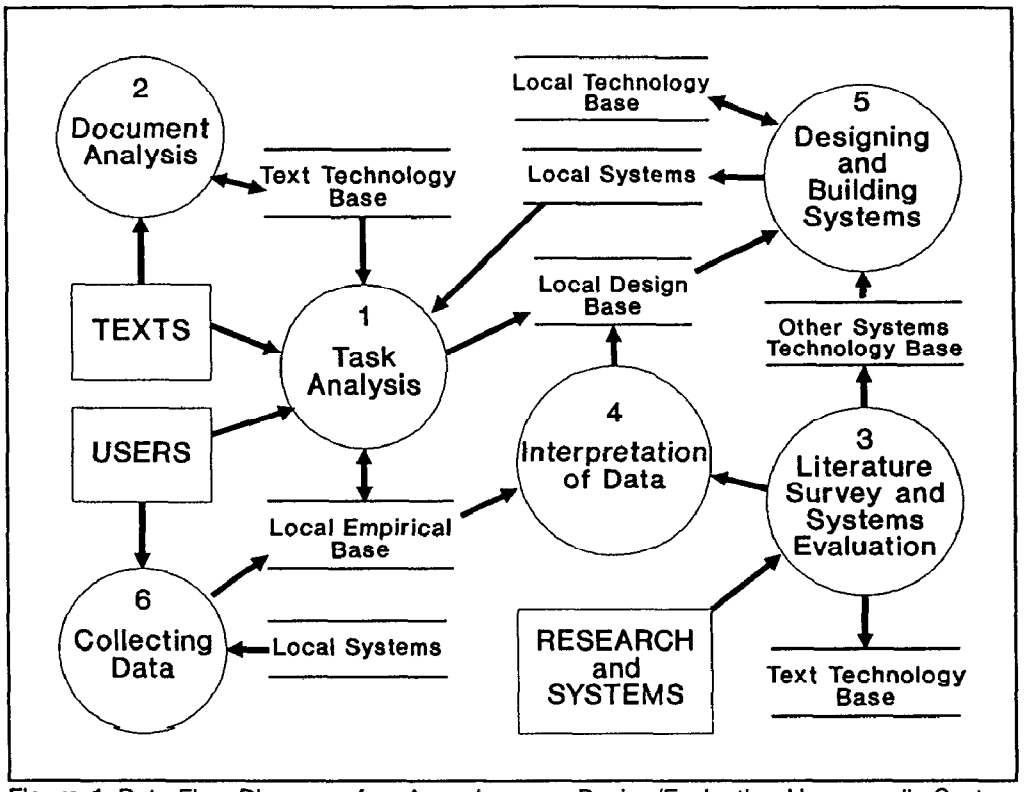


Figure 1. Data Flow Diagram of an Asynchronous Design/Evaluation Hypermedia System Development Process.

A Primer on Data Flow Diagrams

Data flow diagrams are used to model the flow and transformation of data in data processing systems. In data flow diagrams, boxes, bubbles, arcs, and bars are arranged graphically to model how a system works. External entities (shown as boxes) are sources or destinations of information, but are not modeled inside the system. Processes (shown as circles and often called bubbles) operate on data by transforming it or simply by routing it. Data flows (shown as arrows) represent data flowing through the system. Data stores (shown as bars) represent data structures or databases used in and controlled by the system. Data stores may appear more than once in the same diagram to simplify the flows and make the diagram easier to understand. All objects in data flow diagrams can be labeled with a name and sometimes a number. All objects (except externals) in data flow diagrams can be decomposed into sub-parts; processes can be decomposed into sub-diagrams, and stores and flows can be decomposed into component structure definitions. While data stores represent data structures at rest, data flows represent data structures in motion, to or from a process. Data flow diagrams do not represent flow of *control* through a system; any or all processes may be active at the same time, and the order of process activation is sometimes unpredictable. To *read* a data flow diagram, one focuses on the processes, one at a time, to understand their inputs and how they produce their outputs.

A Data Flow Model of Systems Development Research

Each of the processes, flows, and stores in Figure 1 represent an activity or result of hypertext research. The structure of the diagram is a reusable organizer for

experimental hypertext development. Because of the iterative nature of system design, the flow of information in this system is non-linear, so Figure 1 will require a detailed explanation. First, the purpose, inputs, and outputs of processes will be described. Then, all the stores used by these processes will be described (in alphabetical order). After this introduction, a second pass through the processes will show how the model can help organize the asynchronous design evaluation process, as applied to NaviText™ systems. In the discussion, the following special fonts will be used to refer to *processes*, and data stores.

Process 1. *Task Analysis*.

PURPOSE: *Task Analysis* is used to determine the information processing needs of users. These needs depend on (1) the type of user, (2) the tasks being performed by the user, and to some degree, (3) the technology available to the user. Since the user changes over time, by gaining knowledge or becoming fatigued, another important factor is (4) time.

INPUT: The inputs to *task analysis* come from interaction with users and knowledge of their tasks, from the Local Empirical Base (e.g., usage data), from experiences with locally developed systems (Local Systems), and from task-specific analysis of documents (Text Technology Base).

OUTPUT: The outputs from task analysis can go into a Local Empirical Base or directly into a Local Design Base.

Process 2. *Document Analysis*.

PURPOSE: *Document analysis* is used to determine the information content, structure, and format of a (potential) hypertext. The importance of the analysis of logical vs. physical structure, identifying "natural" units of information, and the linking of this information has been discussed in detail in [Glus88] and [Glus89].

INPUT: The inputs to *document analysis* can be an analysis of particular texts, or a summary of understood text technology from a Text Technology Base (e.g., the uses of multiple hierarchical organizations or ways of representing multiple versions).

OUTPUT: The output of *document analysis* is an increased understanding of the structure of documents, stored in the Text Technology Base.

Process 3. *Literature Survey and Systems Evaluation*.

PURPOSE: *Literature survey and systems evaluation* builds on the lessons learned by other researchers. The article by [Conk87] not only surveyed many research and commercial hypertext systems, but also compared these systems along many dimensions. Such dimensions can be used to help organize the structure of design support bases.

INPUT: The inputs to *literature survey and systems evaluation* are (1) published research results and (2) local evaluations of existing systems.

OUTPUT: The outputs of *literature survey and systems evaluation* add to the Text Technology Base and to the knowledge of other systems (Other Systems Technology Base). Empirical results and experiences from external research and systems must also be considered by an evaluative process (*Interpretation of Data*).

Process 4. *Interpretation of Data.*

PURPOSE: *Interpretation of data* is used to filter data and experience from external and internal sources, with the goal of improving the quality of design knowledge. Classical issues of the evaluation of empirical research (measurement, sampling, data analysis and inference) all come into play.

INPUT: The inputs to *interpretation of data* are the results of studies, either from research reports found from a *literature survey and systems evaluation* or from internal studies in the **Local Empirical Base**. These include the results of controlled experiments, studies of usage, and reported phenomena. Particularly compelling for the iterative development of hypertext systems are *critical incidences* of (usually problematic) usage, often gleaned from video protocols.

OUTPUT: The outputs from *interpretation of data* go into a **Local Design Base**, a store of knowledge (design principles, guidelines, and rules) about how systems should be built.

Process 5. *Designing and Building Systems.*

PURPOSE: *Designing and building systems* generates artifacts that can serve a useful purpose and also be used to collect data to improve knowledge of hypertext systems technology. Each such artifact represents a *proof of concept* of the design concepts in the **Local Design Base** and of the implementation tools and techniques in the **Local Technology Base**.

INPUT: The inputs to *designing and building systems* include the **Local Design Base**, the tools and techniques for building systems from the **Local Technology Base**, and information from the **Other Systems Technology Base**.

OUTPUT: The outputs from *designing and building systems* include working **Local Systems**, and new or updated tools and techniques for building systems (i.e., an improved **Local Technology Base**).

Process 6. *Collecting Data.*

PURPOSE: *Collecting data* is used to add to the **Local Empirical Base**. It involves the evaluation of **Local Systems**, either to improve those systems or to ultimately add to the **Local Design Base**.

INPUT: The inputs to *collecting data* include input from users (e.g., via interviews or surveys), or from data collected from use of **Local Systems**.

OUTPUT: The outputs from *collecting data* are stored in a **Local Empirical Base** from which *interpretation of data* is possible.

Local Design Base. The **Local Design Base** contains the accumulated knowledge of how to design new systems by defining what functionality is needed (high-level) and how it is to be presented (lower-level). Typically, the **Local Design Base** is private to an organization, except that reports about design principles are may be published, some good examples of which are [Aksc88] and [Hala88]. [Smit86] provides a format for encapsulating user interface design ideas in a regular format. Such formats promote the effective use of design information. As an example of design information, in many systems, an important high-level design concept is the *bookmark*, which is used to keep track of interesting chunks of information (see

[Walk88]). In specific systems, bookmarks may be implemented in different ways. For example, in NaviText™ SAM, chunks of information can be marked by the user with numerical ratings or automatically marked by the system to indicate if a chunk has been seen before or gathered into a working set. As another example, SuperBook [Egan89] labels super-ordinate chunks with the number of subordinate chunks matching the most recent search key.

Local Empirical Base. The Local Empirical Base contains data collected from users during *task analysis* and about Local Systems while *collecting data*: results of surveys, usage logs, critical incidences of problems, etc. are included.

Local Systems. Local Systems are experimental, demonstration, or production quality systems developed or available locally. Each hypertext research site typically has one or more systems to which they apply ideas from their Local Design Base and their Local Technology Base and from which they can gather data for their Local Empirical Base.

Local Technology Base. The Local Technology Base contains the tools and techniques for the development of Local Systems. The tools typically include proprietary software libraries and the techniques include local software engineering practices. Modifiable source code is needed for a flexible research platform. The Local Technology Base may contain concepts from the Local Design Base that are implemented in software; this can make it difficult to see design decisions and how they were made, but simplifies development.

Other Systems Technology Base. The Other Systems Technology Base contains tools and techniques used in the development of non-Local Systems. Typically, the technology used in other systems must be inferred from published reports describing rationales for design or from careful examination of object code. For example, the NeXT (pre-release version 0.8) indexing software libraries, used in their hypertext help and reference library systems, are documented at the level of function prototypes, but some details (e.g., the list of built-in stopwords) can only be found in the object code of the library.

Text Technology Base. The Text Technology Base, contains knowledge of the structure of documents and how to represent them in an online form.

Application of the Process Model

The process model of asynchronous design and evaluation in Figure 1 has been used in the development of NaviText™ SAM ([Perl87], [Perl88], [Perl89b]) and in the development of the next generation based on experiences with NaviText™ SAM. The process model provides a framework in which the results of experimental systems development can be better understood. To illustrate this, the following sections will describe results (both hard data and informal observations) found with NaviText™ SAM and during the development of a more general NaviText™ system.

1. TASK ANALYSIS

Task demands vary widely, so it should not be surprising that a variety of hypertext models are needed to support a variety of tasks, just as a variety of database

models are used to support various information needs. It probably does not make practical sense to argue about what is and what is not *real* hypertext, but we should be concerned about how system capabilities support (1) what kinds of users doing (2) what kinds of tasks over (3) what periods of use. Results about one type of system that supports a specific user in a task at a specific time should not be used to generalize to the design of systems to support other tasks and users.

One important task is the access of information in large reference documents in which the hypertext system is used as a browser. The most common case is that browsers are used to scan existing documents that have been re-engineered (see [Glus88]) to be in an online form. Examples of document browsers are SuperBook [Remd87], NaviText™ SAM [Perl88], and the Document Examiner [Walk88].

Task analysis was critical in the design of NaviText™ SAM. NaviText™ SAM is a hypertext interface to the [Smit86] collection of 944 guidelines for designing user interface software. The Smith & Mosier report has gone through several revisions during more than five years of development, and in [Mosi86], the results of a user survey provided quantitative information about the ways that the report was used. More so than trying to implement a particular model of hypertext, NaviText™ SAM was designed to support expert users in their information management tasks, the most detailed of which has been called the *checklist method* of using a reference source [Perl89b]. An overview of NaviText™ SAM is given in the section on *designing and building systems*. The checklist method is summarized below.

From the survey by [Mosi86] and from a survey sent to about 500 recipients of the [Smit86] guidelines, we determined that online access should support a variety of methods of *finding relevant user interface design information,* and should do so on a relatively *inexpensive hardware and software platform.* Our survey of potential users of NaviText™ SAM showed that most had access to PC's, and many fewer to Apple Macintosh and workstations. Additionally, the PC's to which potential users had access often did not have a hard disk and almost never used windowing software or a pointing device. Many machines were original PC's or XT's with limited processing power. Such hardware and software limitations placed considerable constraints on the sort of platform from which we could build a hypertext system that would satisfy the needs of many users. Still, a widely accessible platform avoids the irony of the often *cited* but seldom *seen* system.

The Checklist Method

The checklist method supported by NaviText™ SAM is a step-by-step method of applying reference source information to the design and evaluation of systems. In NaviText™ SAM, the design steps, based on [Smit84], include:

- finding relevant information by (1) browsing a dynamically expandable (fisheye view) table of contents, (2) hierarchically inherited keyword search (see *document analysis*), (3) following cross-references, (4) using a citation index, and (5) library-shelf search;
- prioritizing information by attaching ratings (numerical annotations) of relevance to a particular design task;
- defining design rules justified in terms of collections of source information.

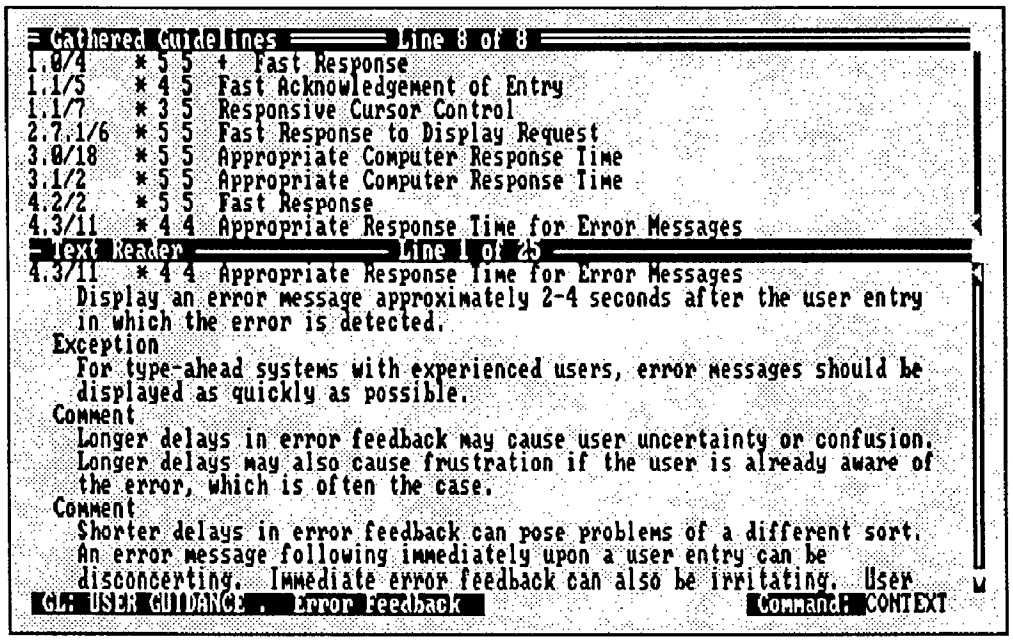
The evaluation steps include:

- using relevant information as evaluation criteria for areas to which it was applied in design;
- rating conformance to the source design information by attaching ratings (numerical annotations of a new type) to the source design information;

- referring back to the full source information for important areas of design for which evaluated conformance was low.

Display 1 (top pane) shows a design/evaluation checklist used in decisions about NaviText™ SAM's response speed. The first column contains the guideline identifier. The next column (*) indicates that the guidelines have been read. The third column is the rated importance (of response time in NaviText™ SAM) of each guideline. The fourth column is the rated conformance to the guideline. These guidelines are sorted by identifier number, but they could be sorted by conformance and importance to highlight serious violations of critical guidelines. With long lists of guidelines, such sorting is essential. In NaviText™ SAM, the text of the guidelines can be accessed directly from the checklist, as shown in the bottom window of Display 1.

A checklist is not a great intellectual achievement, unless perhaps it is complete, but a generalizable method for managing checklists is an important concept for a design and evaluation tool that acknowledges human memory limitations.



Display 1. A NaviText™ SAM design/evaluation checklist on response time.

Task Analysis with Friendly Users

In the continuing development of NaviText™ systems, we are working with standards documents like MIL-STD-1472 [DoD89]. With surprisingly little effort, we enlisted the help of what we call our *friendly users*, potential users of systems who are willing to put up with preliminary versions of systems because they are interested in the potential to help mold a system design. Via surveys and detailed personal interviews, we analyzed the tasks that system designers and evaluators have done with preceding versions of *1472*. While we had anticipated that such standards, being contractual obligations for multi-million dollar contracts, would be used extensively, we were surprised by the extent of the information demands made by such documents. During design and evaluation periods, our friendly users of 1472 spend anywhere from 10 to 30 hours per week with the document, using close variations of the checklist method, primarily with paper. In particular, we were impressed that checklists mapping system design to specific statements in 1472 were

used to manage and later demonstrate conformance to the standard. Such non-linear associations between large documents resulted in the development of several essential new features for any hypertext system supporting standards use. There is a need for extremely flexible annotations, not only that can be attached to chunks of information, but that can be attached to paraphrases of parts of chunks of information. The rational chunking of information for one person (e.g., a document author) may not map well onto natural units for another person (see [Glus88]). There is a need for flexible viewing of documents, both in the amount and type of detail that is visible, and also to allow users to view documents in their original (paper, paged) form, so that they can communicate with colleagues who do not have access to an online version. One striking problem that we encountered was that our friendly users, although technically sophisticated, were poor at being able to appreciate the concepts of hypertext capabilities, as evidenced by their suggestions for the most obvious capabilities like following cross-references, long after such capabilities were explained to them. Evidently, the promise of hypertext is something that takes considerable explanation.

Cognitive Analysis

Underlying the utility of hypermedia systems is their ability to adapt to human cognitive limitations and capabilities. The transition from *browsing* to *keyword search* modes discussed in *collecting data* is analogous to the transition of novice to expert use of interactive systems in which menus and forms are supplanted by command languages or in which graphical interaction is replaced by function *quick* keys. Classic results on human memory limitations [Mill56] can be applied to the design of hierarchical and network browsers, and more recent work on spatial ability in hypermedia systems [Camp89] should underlie the design of systems.

2. DOCUMENT ANALYSIS

Although NaviText™ SAM has some annotation capabilities, those capabilities are mainly for annotations of existing *static* structures, not the definition of new *dynamic* structures. As such, the emphasis with NaviText™ systems has been on analyzing document structure and getting them online. Some of the lessons learned while representing the structure of documents may have applications to the design of authoring systems (see [Scha89]), but the main application we have found is in suggesting how to design documents to make them easier to get online and use effectively. In any case, reading is a more common activity than writing.

Getting the Information Online. The original document ([Smit86]) used with a NaviText™ system was obtained in online form from the authors. The base form was used to generate a well-human-engineered printed document, but it was not in a reasonable form for parsing the structure of the document because direct typesetting codes (e.g., point size 8, bold, etc.) were used instead of a structured markup language. Hand-editing and structural markup of the megabyte of text took about 30 hours. Although this was a lot of tedious work, it was minimal compared to the years of effort that went into the *content*. Other documents that we work with now require OCR (optical character recognition) and considerable cleanup, data conversions from one word processing format to another, parsing common formats like SGML or *troff*, and hand-entry.

Once a document is online, many formatting decisions must be made about which text can be filled and which must be displayed verbatim and about how to display figures that do not conveniently fit on the screen. The need to reformat and

sometimes clip information is much like the adaptation of wide-screen movies to videotape; the *"window-boxing"* format, preferred by Siskel & Ebert on *At the Movies*, shows all the information but does not use the full screen. The more common method of scanning part of the image and panning can show more detail, and use the full screen, but sometimes with a loss of content. In any sort of media translation, a detailed knowledge of the content is critical, and decisions about difficult tradeoffs must be made.

Structuring Information. Even with a document online, it is still necessary to determine the structure of the document and how that structure will be represented in an online form. Like [Glus88], we have found it useful to be able to discuss document structuring with authors, and in the case of [Smit86], the logical structure of the document was discussed in detail in an introduction. The highly structured format of the document, both in the regular hierarchical structure of sections and subsections leading to guidelines, and in the structure of the guidelines themselves, was a major contributing factor to the selection of the report as a good candidate for a hypertext interface. Each of the six sections has a number, a title, an introduction, and subsections. Each subsection has a number, a title, an introduction, an optional example display, and guidelines. Each of the 944 guidelines has its own structure that made it attractive to offer dynamic view capabilities, like *"now you see examples, now you don't."* Each guideline has a number, a title, a statement of the guideline, and optional paragraphs containing comments, exceptions, examples, references to outside sources, and cross-references to other parts of the report. An example guideline is shown in Table 1. Work with [Smit86] exemplified that specific texts will suggest new capabilities that generalize to other systems.

In (re)structuring information, it is useful to consider the goals of users. Different tasks may be better supported by different chunking and structuring. A long-term benefit of hypertext might be to allow multiple structurings of the same content, adapting them to user goals. NaviText™ SAM allows users to create custom subsets of reusable parts of [Smit86].

Removing Redundant Information. A good reference document contains many aids to using the document effectively. Tables of contents (of different levels of detail), subject and author indexes, and structurally-based formatting all aid to the ease with which the information can be accessed. In the analysis of [Smit86], most such aids were determined to be redundant with information that could be derived dynamically. For example, [Smit86] contains tables of contents of three levels of detail: (1) one at the beginning of the report with the first two of three levels, (2) another with each section with the middle level, and (3) a 17-page reference table with 1021 entries at the back of the report. These tables are all redundant with the titles of the section, area, and guideline entries; any such table could be dynamically generated from an outliner and browsed with fisheye views [Furn86], so fixed tables of contents were avoided in NaviText™ SAM. There was considerable redundancy among guidelines (see the duplication of guideline titles in Display 1), because similar guidelines in different areas of the report were partially adapted to their context. They were left unchanged in NaviText™ SAM, but their presence suggests that authors of modern texts may want to adapt their writing to take advantage of hypertext capabilities.

A corollary of the ability of hypertext systems to remove redundant information is their ability to dynamically produce (or compute) additional information. In the guideline in Table 1, there is no context for the title, although this is provided as a running heading in the [Smit86] printed report. In NaviText™ SAM, context is available by a simple traversal up the spine of the hierarchy, and is bound to the

3.1.3/13 Letter Codes for Menu Selection

If menu selections are made by keyed codes, design each code to be the initial letter or letters of the displayed option label, rather than assigning arbitrary letter or number codes.

EXAMPLE

```
( Good)   m  =  Male
          f  =  Female

( Bad)    1  =  Male
          2  =  Female
```

EXCEPTION

Options might be numbered when a logical order or sequence is implied.

EXCEPTION

When menu selection is from a long list, the line numbers in the list might be an acceptable alternative to letter codes.

COMMENT

Several significant advantages can be cited for mnemonic letter codes. Letters are easier than numbers for touch-typists to key. It is easier to memorize meaningful names than numbers, and thus letter codes can facilitate a potential transition from menu selection to command language when those two dialogue types are used together. When menus have to be redesigned, which sometimes happens, lettered options can be reordered without changing codes, whereas numbered options might have to be changed and so confuse users who have already learned the previous numbering.

COMMENT

Interface designers should not create unnatural option labels just to ensure that the initial letter of each will be different. There must be some natural differences among option names, and special two- or three-letter codes can probably be devised as needed to emphasize those differences. In this regard, there is probably no harm in mixing single-letter codes with special multiletter codes in one menu.

REFERENCE

BB 1.3.6; MS 5.15.4.2.11; Palme, 1979; Shinar, Stern, Bubis, & Ingram, 1985.

SEE ALSO

4.0/13

Table 1. A sample guideline from [Smit86] with a variety of paragraph types.

context command. (The context is: Sequence Control, Dialogue Type - Menu Selection.) More compelling is the ability to let readers *look before they leap* and following a cross-reference. In Table 1, the SEE ALSO pointer to 4.0/13 in the printed form is optionally partially expanded to show the title of the text to which it points: *"Consistent Coding Conventions."* Similarly, references to outside sources can be partially expanded, automatically or interactively, as shown in NaviText™ SAM Display 2.

Reverse-Engineering Indexes. The [Smit86] report contains a hand-made subject index (but no author index). Rather than provide a special-purpose interface to the subject index, a decision was made early on to develop a keyword searching scheme that would replace the functions of an index. The logic is as follows: Instead of searching the index for terms that, once found, lead you to a location in the text (i.e., a page or paragraph number), keywords would be attached to the chunks of information. That way, a keyword search would lead to the same chunks of information as would an index search. The index entries (with **primary**, and secondary terms) in [Smit86] leading to the guidelines in Table 1 are shown below. The terms that are not redundant with those in the title are underlined. They are *index keywords* to be added to the title keywords.

Coded menu options, code design
Keyed data entry, menu selection
Menu option codes, code design
Menu selection, keyed entry

An index like [Smit86] is reverse-engineered, or *un-indexed*, as follows. All primary

and secondary index terms in the index that lead to a chunk (page or chunk identifier) are added to the list of index keywords for that chunk, if they are not in the title of the chunk. Non-content words (i.e., traditional stopwords like: and, of, the) are filtered. Index keywords add richness to search without some of the false-alarm problems of full-text retrieval.

In a hierarchically structured system, in which matching a high-level chunk will also match all subordinates (using an inheritance mechanism), index keywords that appear in the index keywords or titles of superordinates can also be removed to conserve space. Such a scheme is used in NaviText™ SAM. An added benefit of the un-indexing is that it is not necessary to have multiple entries that ensure that searches for *"response time"* (indexed under R) and *"time to respond"* (indexed under T) will have the same result. Un-indexing is an effective process for aiding keyword search via the multiple access methods (e.g., synonymy) built into good indexes. When used as a replacement (as opposed to a complement) to indexes, it requires that users predict the terms in the index more so than when the index is present. Such term guessing can be aided by browsing. An implication of this approach is that keywords should be attached to chunks in the first place, and that search can be based on fields like *title*, *keywords*, and *text body*.

Although, there was no author citation index for the [Smit86] report, it was easy to generate one from the reference sections in the guidelines, automatically linking author names to guidelines citing them.

The Book Metaphor. Several systems have advocated a book metaphor so that the online version of information closely matches printed formats, even if there is no printed version. The rationale is that systems presenting online documents will be easier to learn and use if they closely match the way books look and work. For example, the Sun Help Viewer presents a paginated display that looks remarkably like a book page, and research has shown the system to be easy to learn [Camp89]. In other systems, like NaviText™ SAM, there are parallels between the original text and the hypertext presentation, but the mapping eluded some users if the mapping is not explained. NaviText™ SAM is like a book, but it attempts to make better use of information structure needed by *"power"* users:

- The tables of contents with varying amounts of detail are replaced by a dynamic outliner.
- The index is replaced by keywords attached to structures.
- Cross-references are made *dynamic* to allow *jumps* to related information.
- References to outside sources can be partially expanded or used as a citation index.
- Running headings on printed pages are replaced by context information.
- Format to reflect information type is made dynamic, e.g., information of generally low interest can be made invisible while other types of information can be highlighted.

With NaviText™ SAM we have found that an appropriate mental model must be provided to help users transfer their knowledge of how they use printed media.

Multiple Versions. In work with MIL-STD-1472 [DoD89], we have found it necessary to be able to represent multiple versions of documents. Military standards follow a predictable hierarchical format (there is a military standard for their format). When a standard is released, it may be updated by *notices*. For example, there were three notices to the predecessor of 1472D. Each notice contains new pages to be inserted, and pen and ink changes to be made by hand. There are no references to structures (i.e., paragraph numbers like 5.15.3.3.2) so it is a difficult task to determine where real changes occur. Additionally, there are

unpredictable changes in the layout of text, making line-by-line comparison impractical. We have found it useful to develop a revision control tool to represent changes at three parallel levels: (1) the printed page level, (2) the meaningful structure level, and (3) the word level. The last level is needed to highlight changes between successive versions.

Multiple Documents. Part of the vision of hypertext is that we will be able to move with ease from document to referenced document. In the [Smit86] report, there are 944 references from (coincidentally 944) guidelines to 173 outside sources. It would indeed be a tremendous achievement to get all these sources in an online form, but we have done some work to see what it would be like to get some. There are over 200 citations of an early version of [DoD89] from [Smit86], and these have been linked in a system. The two documents have had an incestuous relationship for years, and much of [DoD89] is copied *verbatim* from [Smit86]. Tracing for circular justification loops is not possible because in DoD standards, providing a reference source for a rule is not required. Based on a combination of *task analysis* and *document analysis*, we feel there is a need for specific functional support for each specific document type like [Smit86] and [DoD89], although there is considerable overlap of support needs among these and other reference documents. This suggests a need for customizability of the functionality of hypertext browsers, if they will be more than generally mediocre.

3. LITERATURE SURVEY AND SYSTEMS EVALUATION

A Hypertext Technology Assessment Project

[Conk87] compared many hypertext systems along 12 dimensions (i.e., features, capabilities of systems) and offered plausible classifications for different types of systems. [Hala88] discussed seven issues for the next generation of hypermedia systems. An evaluative survey of hypermedia functionality and its delivery form is one of the most common sources of design information, despite possible copyright and patent problems [Samu89]. One goal of our hypermedia technology assessment project is to find *all* dimensions on which systems can differ, and assess the practical utility of these differences for different types of users working on different types of tasks. For example, many systems have a capability for keyword searching. How important is it for there to be a full Boolean combination search capability? How does this depend on the type of users and the tasks they are working on? The particular implementations (including the user interface) and the methods of evaluation of such capabilities are critical; it is easy to set up a weak strawman using an *ad hoc* user interface to a limited, inefficient Boolean scheme, compare it to an interactive browsing system, and invalidly conclude that differences observed are meaningful. The field of software systems development is full of such comparisons of rotten apples to pale yellow oranges. Issues of comparisons among systems are discussed in more detail in the section on *interpretation of data*.

Another goal of the hypermedia assessment project is to develop a taxonomy of hypertext capabilities alongside those of more traditional information management technologies like information retrieval and databases. It should not matter if a system or a capability is *really* hypertext, only that it supports users in their tasks. For example, the GUIDE system added string search to present users with a marketable system even though it was outside their hypertext model [Brow87]. As another example, some hypertext researchers require that links (such as those to connect a software engineering design document with correlated code) must be

independent of the source and destination to be considered *real* hypertext. Some links, like those to manage software project documents, may require *rich* links that include such essential information as the application to launch when traversing the link, but other applications may require little information in a link. With that in mind, our evaluations of a wide variety of systems focus on the effectiveness of functionality, to determine what tasks are possible with what capabilities. For example, there are many advantages to having information online, without using any hypertext concepts. An advantage of online access is full text search, but structured text allows people to use structure in search, in online display, and in preparing special purpose subdocuments. The following progression of capabilities shows that many benefits are obtained before we reach any functionality that any researcher would consider hypertext.

online	search, cut, and paste (by lines or proximity)
typed chunks	elision by type, simple formatting, level of detail, annotation
structures	views (e.g., hierarchical), formatting based on structure
hypertext	cut/paste/view by reference

As an example of an analysis of the wide possible range of functionality, consider the NaviText™ SAM *expand* function. The semantics of this function depend on the type of object being expanded and the workspace from where it being expanded. Different objects behave differently in different contexts. There are also options that control to where an object will be expanded: using stretch text, in the same window, or in another window. Such richness of functionality is not uncommon in object-oriented systems where each class of object has its own functionality. It is a challenge to understand such diversity of capability well enough to predict when it will be useful for a particular task.

4. INTERPRETATION OF DATA

The Method of Specific Advantages

Hypertext systems can be complex, with many functions and new user interfaces. Data collected on such systems must be analyzed with the assumption that it is difficult to control all conditions relevant to data collection. The effectiveness of hypertext capabilities should be demonstrated, but it can be difficult to separate the functionality from the systems that implement them. A capability may *outperform* another only because of the choice of tasks, users, or because of artifacts of the implementation of the system in which the capability is implemented. Rather than criticize specific systems or researchers' evaluations of capabilities, I will present a method that can better quantify the performance advantages of capabilities within systems. The method also applies to the comparative evaluation of user interfaces and to systems in general. I will begin with an anecdote.

A student of mine wanted to show that graphical displays of networks were superior to tabular displays. He could have made some really bad tables and shown a gigantic superiority of graphical displays over tabular displays. Such bad tables would have been a *weak strawman*. For a *plausible strawman*, he needed to show that his tables had some intrinsic merit. To do this, he was careful to devise some tasks that would show *specific advantages* of tables over graphs, if the tables were well designed. His results showed task-specific advantages (faster responses to questions) of both graphics and tables; on some tasks tables were better than graphs, and in others, graphs were better than tables. This *cross-over interaction* is a critical aspect of argumentation in many empirical fields.

A *plausible strawman* for a condition provides data that shows a specific-advantage of the strawman over the condition. Mutually plausible strawmen provide specific advantages over each other. This allows for the possibility that both are mediocre, but it adds to the credibility that they are reasonable foils for each other. At the system level, some common plausible strawmen are existing (commercial) systems, their specific-advantage at least being notoriety. Another group includes print media, their specific-advantage being that they have been in use for a long, long time. At the feature level, there may be more than one way in a hypertext system to accomplish the same task. If a feature is never used, then perhaps it is useless or hard to use, but if feature A shows a specific-advantage (is used more, or is used more effectively, or is better liked) in one context, and B shows it in another, then more sound conclusions can be drawn about the merits of both features.

5. DESIGNING AND BUILDING SYSTEMS

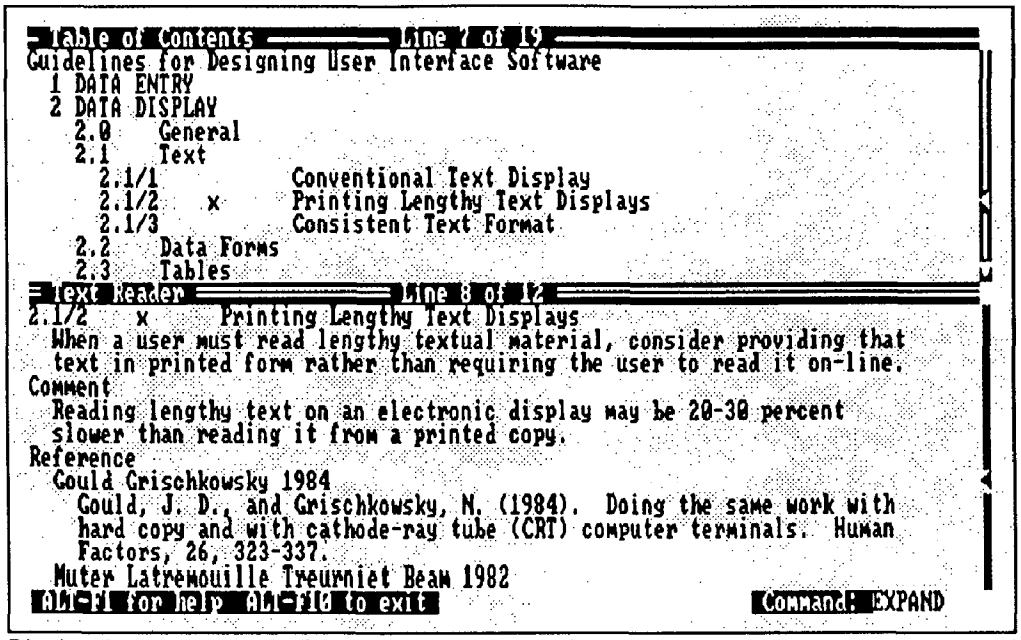
An Overview of NaviText™ SAM

NaviText™ SAM provides PC-based support of the checklist method of design and evaluation with Smith and Mosier's [Smit86] *"Guidelines for Designing User Interface Software."* NaviText™ SAM was developed to create and explore a variety of hypermedia technologies in the context of solving specific information management problems. To evaluate new ideas in hypermedia, a research platform with control over source code is a *sine qua non.*

NaviText™ SAM Workspace Windows. There are eight workspace windows used by NaviText™ SAM. At any one time, at most three can tile the screen, the combinations of which were based on task analysis and feedback from users. The **Table of Contents** window is a dynamic outliner that shows all possible views of the main hierarchical structure of its document. The **References** window displays a list of references, with the facility to show detail about references and to gather (bookmark) chunks that cite a reference. The **Text Reader** is used for displaying larger blocks of text, such as section introductions and individual guidelines. The *view* of a guideline is determined by settings in the **Options** window, a data-entry form. The **Copy** window allows the comparison of any window contents with any other, and helps compensate for the size of the PC screen. Online expandable help is available in the **Help** window.

As potentially useful pieces of information are seen, they can be gathered into the **Gathered** set. This set can be scanned or searched, and details of the set can be expanded into the **Text Reader**. As texts are expanded, their identifiers (and linked titles) are placed in the **Expanded Text** window to allow backtracking and review. All NaviText™ SAM windows support a wide variety of operations: file interface, sorting, deleting and inserting text, and navigation using the standard arrow and paging keys. Examples are in Displays 1 and 2.

NaviText™ SAM Functions. The main functions in NaviText™ SAM are the dual *expand* and *conceal* operations that can be applied to sections, functional areas, guidelines, references, and other objects. The same objects can be *gathered* into the working set, on which several specialized functions are possible, most notably ratings by annotation, sorting on multiple keys, and report generation controlled by a hierarchy of display and format options. A hierarchical index keyword search with inheritance can also augment the gathered working set.



Display 2. A NaviText™ SAM fisheye view (top) with a guideline with an expanded reference (bottom pane).

A Hypertext Technology Base

NaviText™ SAM was developed as a special-purpose system, and as such, its applicability to other reference texts is limited (It has been applied to the UNIX® manual pages). The experiences with NaviText™ SAM have shown that there are generalizable capabilities in the system, and many of these have been extracted for use in the development of more general NaviText™ systems. Our current strategy is to use two complementary technologies in the development of hypertext browsers for large reference documents: (1) general information retrieval indexing and search engines, and (2) hierarchical outliners. The storage technology used in NaviText™ SAM includes: (1) dynamic memory allocation with caching of text for speed and garbage collection to free memory on small machines, (2) text compression to save disk space, and (3) version control. This technology has shown itself to be easily generalized. The NaviText™ SAM windowing software is also reusable. Some key developments for new documents have been necessary to support programmable operations and search of general annotations, particularly for group-shared annotations on documents. We plan to increase our use of established information management technologies to achieve efficient access to more document types.

6. COLLECTING DATA

Tasks and Performance Measures

In [Perl89a], I discuss methods for gathering data on user interfaces, many of which can be adapted to the empirical investigation of the utility of hypertext capabilities. With NaviText™ SAM, most data collection has been observational -- monitoring of usage patterns -- although there have been experiments in which users were placed through a series of tasks and observed over time. For gathering longitudinal data from most of the users of NaviText™ systems, we have methodological problems of controlling conditions and security problems,

particularly for work using military standards. Like other empirical researchers ([Egan89], [Marc88], [Furn86], and [Camp89]), we have used a variety of tasks, collected a variety of measures, and compared them to performances of a variety of strawmen (see the section on *interpreting data*). On the task dimension, we have had users search for material relevant to specific topics, controlling the familiarity of the topics and the texts in use. Our measures have been frequency of use, task completion time, and discriminability (see below). We have used printed versions of texts and competing methods within systems for comparisons.

A Measure of Discriminability. In comparing access of information with NaviText™ SAM to the printed form, we were faced with a problem of retrieval evaluation (see [Salt83]). Research uses of large reference sources, such as one might go through in designing a user interface, are open-ended. For a particular design area, some information is clearly relevant while some is clearly irrelevant. In a resource like [Smit86], there are 944 guidelines, so it is no easy task to determine if all and only the relevant information has been found. In Display 1 there are eight guidelines on system response time, spanning four sections of the report. Because each guideline has been read and rated for relevance, we can be reasonably sure that they are relevant, but are these *all* the relevant ones? Any search-success measure must compensate for both retrieval errors: false alarms and misses. Signal detection theory [Coom70] may provide us with a fair method of evaluating the effectiveness of retrievals. There is a problem of deciding the truth of whether a piece of information is truly relevant, and this can only be overcome with expert ratings checked for inter-rater-reliability. Once a search task is set up, with ratings of relevance in hand, different access methods can be compared by their ability to discriminate among alternatives in the search space. This measure was developed and exercised to compare student prototype hypertext interfaces to [Smit86], and there is ongoing work on its refinement and validation.

Empirical Results from NaviText™ SAM

We have found that inexperienced users learning NaviText™ SAM needed help with the (lack of a) book metaphor. Without hard data on which to base our conclusion, we have concluded that it was extremely useful to tell users how to map their paper-based information finding skills to the NaviText™ SAM implementation of hypertext. A common new user comment was "I am not sure where to start" which was avoided with training on online reference documents.

Two phenomena are apparent in experienced users of NaviText™ SAM: (1) To become oriented with a new information space (particularly with its terminology), users explore the space with a fisheye view [Furn86] outliner and gradually migrate to a title and index keyword search strategy. This is related to a transition from recognition of menu options to recall of exact terms in a command language. (2) To avoid disorientation (see [Mant82]), users adopt a breadth-first-search (BFS) strategy of putting possibly interesting cross-references at the *end* of their browsing agenda, instead of following them as they are encountered (i.e., they avoid a depth-first-search (DFS) strategy).

The following time-line graphs show the sequence of actions taken by an experienced user of NaviText™ SAM. In the first task (Figure 2), the user is looking for user interface design guidelines for designing a window title bar. In the second task (Figure 3), the same user is looking for guidelines on the use of color in displays. The topics were chosen so that a topic familiar to the user would precede a less familiar topic to see how search strategy might be affected.

NaviText™ SAM has a built-in monitoring capability used for several purposes: creating macros, running demonstrations, and collecting usage data. Usage data can be analyzed to see what actions are being taken in which windows, and displayed in time-line graphs. The time-line graphs can be interpreted as follows. Across the horizontal dimension are actions taken by the user (not including help or navigation within and between windows). Along the vertical dimension are the windows in which the actions take place. The meaning of a command, like *expand*, depends on the window in which it takes place. For example, in an outliner, expansion means *"show detail"*, while in a text reader, it means *"follow a cross-reference."* Similarly, *"search"* in a window means to look for a combination of terms inside the window, while *"global search"* searches through the full information space. The type and position of the commands used in a search give good insights into the strategies being used. The action types, and their window-specific interpretations, are tabulated in Table 2.

Code	Meaning
+	Gather (Set Bookmark) (in CONTENTS, it means to save guideline for review) (in READER, it means to put guideline on BFS agenda)
D	Delete (Unmark) * implies many
N	Next (Library Shelf Browsing Expansion)
P	Previous (Library Shelf Browsing Expansion)
R	Reorder (Using Multiple Sorting Keys)
S	Search (GLOBALly, it means hierarchical search with gathering)
U	Make Unique List (Remove Duplicates)
•	Expand (in CONTENTS, it means to show local detail) (in READER, it means to follow guideline cross-ref using DFS) (in GATHERED, it means to review guideline detail)

Table 2. Key for Interpreting Time-Line Graphs.

In the search for window-title guidelines (Figure 2), the user immediately begins with a keyword search (for window title) and after a few guidelines are gathered for later review, another keyword search (for display label) follows. This results in over 50 guidelines being gathered, and the user deletes many guidelines from the gathered set based on the titles, without expanding the detail of the guidelines. When pockets of interesting guidelines are found, a library-shelf search of adjacent guidelines is used. With little more expansion to full text, the list pruning is completed. Note that there was no use of the table of contents, designed to provide context to the user, nor the expanded set of guidelines, designed to provide a backtrackable trail (made unnecessary here because of the BFS strategy).

The second time-line graph (Figure 3) shows the sequence of actions taken by the same user in a subsequent search for information about the use of color in displays. This session is typical of one by an experienced NaviText™ SAM user who is marginally familiar with the coverage in the text of a topic; the table of contents is used for orientation in a new topic area, and a BFS strategy is used to avoid getting lost. Early actions involve the table of contents, which is being used as a fisheye view of the information space. As candidate information is encountered, it is gathered for later evaluation. After initial browsing of the table of contents, the gathered information is reviewed in more detail, and cross-references from useful chunks are gathered for later analysis to avoid getting lost

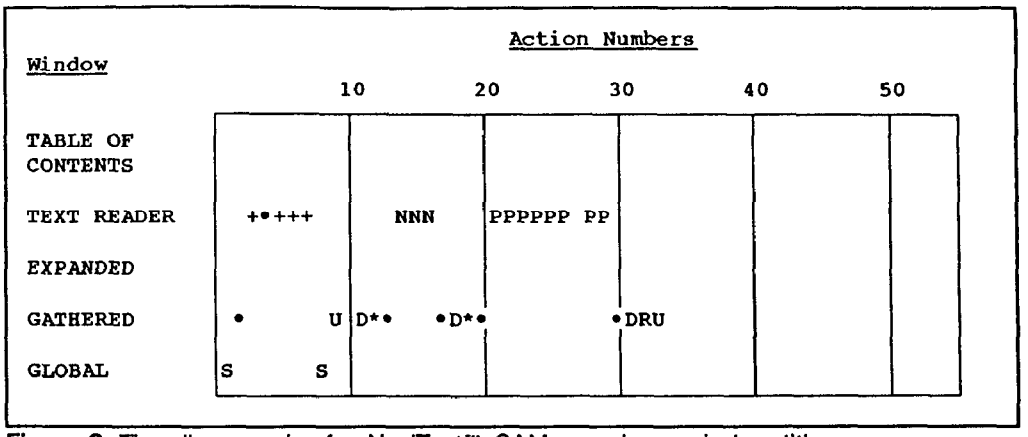


Figure 2. Time-line graph of a NaviText™ SAM search on window titles.

in hyperspace. The unusual expansion (following) of a cross-reference at action 17 is followed by gathering of the information and the expanded text (trail) is used to backtrack. A series of expansions from the gathered set are occasionally supplemented by further gathering of cross-references, which are added to the end of the gathered set for later consideration. After a series of decisions about the relevance of information in the gathered set, two global searches are attempted using terms seen in the body of the text (e.g., spectr, which abbreviates both spectrum and spectral).

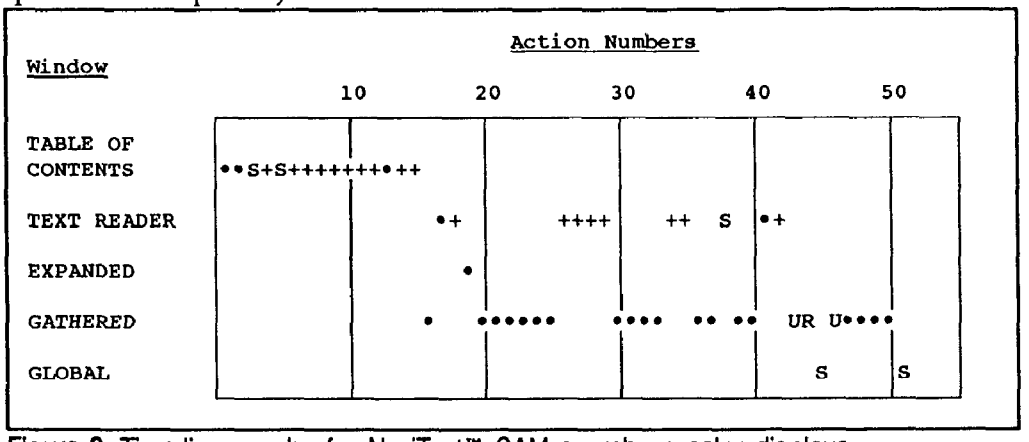


Figure 3. Time-line graph of a NaviText™ SAM search on color displays.

The NaviText™ SAM table of contents outliner and keyword index search are plausible strawmen for each other because the choice of which is used depends on the context. If one was always used first, then we would not know if the other was less useful, poorly implemented, was hard to use, or had some other deficiency.

SUMMARY

We need a framework to capture and organize information for the development of new hypermedia technology. The data flow diagram in Figure 1 is an attempt to reflect the asynchronous process of design and evaluation of hypermedia systems, a process more complex than an iterative design, implementation, and evaluation lifecycle. Many aspects of the diagram apply to systems more general than

hypermedia systems; some apply to human-computer systems, some to any system.

Task analysis with friendly users and document analysis with specialized texts can produce ideas for new functionality in hypertext systems (e.g., dynamic views, checklist support). New techniques and demanding decisions are required to bring information online, and an appreciation of these would be useful to designers of documents targeted for print and online access. Hypertexts include structures of typed information chunks, but many benefits of accessibility come from being online, or having some structure. Users of full-featured hypertext systems need special training to take advantage all their capabilities. This training should help map mental models of how to use printed reference texts onto hypertext system functionality. A technology base of information management tools is a prerequisite for experimental development and evaluation of new hypertext technology.

A variety of usage data have been collected on NaviText™ SAM, a hypertext interface to a reference document ([Smit86]). Experienced users of NaviText™ SAM orient themselves in a novel information space using browsing strategies with a hierarchical outliner and by following cross-references. The same users choose the more direct keyword search to access information in familiar information spaces. It is necessary to have a technology base capable of supporting competing access methods (e.g., outliners and full-text search) to allow a fair comparison using the method of specific advantages. Experienced NaviText™ SAM avoid getting lost in hyperspace by using bookmarks to formulate a breadth-first-search rather than follow cross-references and effectively use a depth-first-search.

Current research is aimed at providing new methods to formulate queries and specify views (both online and for generated reports), and to evaluate their ability to help users find relevant information. A measure of discriminability, based on signal detection theory, is being validated as part of this effort.

ACKNOWLEDGEMENTS

Northern Lights Software's friendly users provided task analysis, document analysis, and usage data. Bob Glushko provided key conceptual advice. Sid Smith and Jane Mosier provided the first text for a NaviText™ system, and Tony Moorhead provided programming support. The following companies have donated software for the PC-based Hypertext Technology Assessment Project: askSam Systems, Brightbill-Roberts, Cognetics, Group L, Lotus Development, Northern Lights Software, Persoft, and over ten other packages are also being evaluated on loan. I would like to thank Stuart Bertsch and Phil Smith for their comments.

REFERENCES

[Aksc88] Akscyn, R. M., McCracken, D. L., & Yoder, E. A. (1988) "KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations." **Communications of the ACM**, 31:7, 820-835.

[Brow87] Brown, P. J. (1987) "Turning Ideas Into Products: The GUIDE System." **Proceedings of Hypertext'87**. New York: ACM.

[Camp89] Campagnoni, F. R. & Ehrlich, K. (in press) "Information Retrieval Using a Hypertext-Based Help System." **ACM Transactions on Office Information Systems.**

[Conk87] Conklin, J. (1987) "Hypertext: An Introduction and Survey." **Computer, 20:9,** 17-41.

[Coom70] Coombs, C., Dawes, R., & Tversky, A. (1970) **Mathematical Psychology.** New York: Academic Press.

[DoD89] DoD (1989) MIL-STD-1472D: Human Engineering Design Criteria for Military Systems, Equipment and Facilities. Washington, DC: Department of Defense.

[Egan89] Egan, D. E., Remde, J. R., Landauer, T. K., Lochbaum, C. C. & Gomez, L. M. (1989) "Behavioral Evaluation and Analysis of a Hypertext Browser." **Proceedings of the CHI'89 ACM Conference on Human Factors in Computer Systems.** New York: ACM. 205-210.

[Furn86] Furnas, G. W. (1986) "Generalized Fisheye Views." **Proceedings of the CHI'86 ACM Conference on Human Factors in Computer Systems.** New York: ACM. 16-23.

[Glus88] Glushko, R. J. Weaver, M. D., Coonan, T. A. & Lincoln, J. E. (1988) "Hypertext Engineering: Practical Methods for creating a compact disc encyclopedia." **Proceedings of the ACM Conference on Document Processing Systems.** New York: ACM. 11-20.

[Glus89] Glushko, R. J. (1989) "Transforming Text into Hypertext for a Compact Disc Encyclopedia." **Proceedings of the CHI'89 ACM Conference on Human Factors in Computer Systems.** New York: ACM. 293-298.

[Hala88] Halasz, F. G. (1988) "Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems." **Communications of the ACM, 31:7,** 836-852.

[IDE86] IDE (1986) Software Through Pictures™ Data Flow Editor (DFE) Manual. San Francisco: Interactive Development Environments.

[Marc88] Marchionini, G. & Shneiderman, B. (1988) "Finding Facts vs. Browsing Knowledge in Hypertext Systems." **Computer, 21:1,** 70-80.

[Mant82] Mantei, M. M. (1982) **A Study of Disorientation Behavior in ZOG.** PhD dissertation, University of Southern California.

[Mill56] Miller, G. (1956) "The Magical Number Sever Plus or Minus Two: Some Limits on Our Capacity for Processing Information." **Psychological Review,** 63, 81-97.

[Mosi86] Mosier, J. N. & Smith, S. L. (1986) "Application of Guidelines for Designing User Interface Software." **Behaviour and Information Technology,** 5, 39-46.

[Perl87] Perlman, G. (1987) An Overview of SAM: A Hypertext Interface to Smith & Mosier's *'Guidelines for Designing User Interface Software.'* Wang Institute Tech. Report WI-TR-87-09.

[Perl88] Perlman, G. & Moorhead, A. J. (1988) "Applying Hypertext Methods for the Effective Utilization of Standards." **Proceedings of the IEEE COMPSTAN'88 Conference on Computer Standards.**

[Perl89a] Perlman, G. (1989a) "Evaluating How Your User Interfaces Are Used." **IEEE Software, 6:1,** January. 112-113.

[Perl89b] Perlman, G. (1989b) "The Checklist Method for Applying Guidelines to Design and Evaluation." **Proceedings of INTERFACE 89.** Santa Monica, CA: Human Factors Society.

[Remd87] Remde, J. R., Gomez, L. M., & Landauer, T. K. (1987) "SuperBook: An Automatic Tool for Information Exploration -- Hypertext?" **Proceedings of Hypertext'87.** New York: ACM. 175-188.

[Salt83] Salton, G. & McGill, M. J. (1983) **Introduction to Modern Information Retrieval.** New York: McGraw-Hill.

[Samu89] Samuelson, P. (1989) "Why the Look and Feel of Software User Interfaces Should Not be Protected by Copyright Law." **Communications of the ACM. 32:5,** 563-572.

[Scha89] Schank, C. & Mamrak, S. A. (1989) "A Composition Environment to Support Scholarly Writing." Columbus, OH: Ohio State University, Department of Computer and Information Science. Technical report OSU-CISRC-6/89-TR27.

[Smit84] Smith, S. L. & Mosier, J. N. (1984) "A Design Evaluation Checklist for User-System Interface Software." Report MTR-9480. Bedford, MA: MITRE Corporation.

[Smit86] Smith, S. L. & Mosier, J. N. (1986) "Guidelines for Designing User Interface Software." Bedford, MA: MITRE Corporation.

[Walk88] Walker, J. H. (1988) "Supporting Document Development with Concordia." **Computer, 21:1,** 48-59.

Towards a Design Language for Representing Hypermedia Cues

Shelley Evenson and John Rheinfrank

Exploratory Design Lab
Fitch RichardsonSmith

Wendie Wulff

Exploratory Design Lab
Fitch RichardsonSmith
Department of English
Carnegie Mellon University

Background

Hypermedia systems are no longer just interesting experimental software environments. They are common tools in the world of everyday work. People who do not program, but who are computer literate and who want to go beyond the capabilities of word processing, spreadsheet and presentation software packages now use systems like Apple's Hypercard, Owl's Guide, Silicon Beach's Supercard and Xerox's Notecards not only to communicate, but to perform tasks that involve creating and integrating knowledge. This raises some important issues for designers of hypermedia systems. One of the largest is how to represent which pieces of information are linked (or hyper) and which pieces aren't, within a given system or task domain. This, in turn, raises the issue of standards. Should representations of hyperness be consistent across systems and work domains, or should there be individual standards for representing hyperness within systems and work domains? The advantage of a standard is that it may assist users in discovering or labeling what is or isn't hyper across a wide variety of systems. The disadvantage is that a standard severely limits the opportunities for creating systems that are closely connected to the content of specific areas of work, work environments and work tools. Thus, the apparent choice is between adopting a rigid hypermedia cuing standard, or redesigning hypermedia cues for each application.

What is a design language?

One promising alternative is to create a set of tools that will encourage people (both interface designers and users) to directly shape (and reshape) hypermedia cues according to an evolving variety of needs, circumstances and subject matters. By creating such tools we would also be creating a shift in the conditions surrounding the use of the tools--we would, in effect, be creating a language that would enable people to design their own solutions. A design language, then, is a flexible collection of tools (elements), plus indications of the conditions determining their use (guidelines), that together allow people to use (and generate) expressions in response to situations.*

*Our work with design languages in graphic, product and user interface design has prompted us to apply design language theory to the representation of hyperness in multi-level documents. Our current work has focused on design languages for hypertext rather than hypermedia. In this paper we concentrate on the hyperness of words and related units of text. We expect that design language theory will be extensible to other forms of hypermedia and other aspects of hyper system design.