On Generalizing the Concept of Hypertext

By: Michael P. Bieber
Computer Science Department
Boston College
Chestnut Hill, Massachusetts
02167-3808 U.S.A.

Steven 0. Kimbrough
Decision Sciences Department
The Wharton School
University of Pennsylvania
Philadelphia, Pennsylvania
19104-6366 U.S.A.

Abstract

Hypertext has quickly become an established paradigm in the design of information systems. The success of products in the software market, evident benefits as reported by users, and the flowering of related research activity all attest to the significance and staying power of hypertext-rich information systems. Although standard hypertext has a number of unquestioned benefits, the concept also has a number of well-known problems and limitations. This article reviews the main problems and limitations of basic (standard) hypertext that constrain the use of hypertext in practical applications. Further, this article presents and discusses our "generalization" of the basic hypertext concept, which we call generalized hypertext. *These generalizations encompass, among other things, automatic creation of hypertext elements. Generalized hypertext promises to be more powerful than standard hypertext as well as less expensive to implement and maintain. To illustrate these concepts, we describe the implementation of a decision support system currently in use by the U.S. Coast Guard.*

Keywords: Hypertext, generalized hypertext, hypertext computation, virtual linking, dynamic linking, decision support systems, information presentation

ACM Categories: H.1.0, H.3.4, H.4.2

Introduction

The core idea of hypertext has been described clearly and accurately:

> The concept of hypertext is quite simple: Windows on the screen are associated with objects in a database, and links are provided between these objects, both graphically (as labelled tokens) and in the database (as pointers) (Conklin, 1987, p. 17)

(See also Nielsen, 1990, and Shneiderman and Kearsley, 1989, for book-length introductions to hypertext.)

Still, as is universally recognized, there is more to the idea of hypertext than linked information items that allow a user to explore ideas and pursue thoughts in a free and "non-linear" fashion. After all, well-designed standard computer application programs, including reporting systems and decision support systems, have long delivered such capability, at least to a respectable degree. What hypertext systems add, with their emphasis on the value of linked information items, is: (1) easier, richer, more highly featured linking of information; and (2) *system-level,* rather than application-level, support for creating, maintaining, exploiting, and managing linked information items (Bieber, 1991). Like database systems, report generators, graphics packages, and user interface management systems, hypertext software can be seen as application-independent, system-level tools for providing useful features for specific applications.

Our aim is to describe and discuss certain extensions *at the system level* to the core ideas of hypertext, which we call *generalized hypertext.* We have been motivated to develop generalized hypertext concepts as part of a larger effort, funded by the U.S. Coast Guard, to develop decision support system shells, i.e., system software for generating particular decision support systems (Bhargava, et al., 1988; Kimbrough, 1986; Kimbrough, et al., 1990a; 1990b; Minch, 1990). Our purpose in this article is mainly to describe these concepts, the reasons for them, and their present implementation.

The paper is organized as follows. In the next section, we present briefly the core concepts and vocabulary for basic hypertext, as well as certain

problems and limitations of this hypertext concept. These problems and limitations are widely recognized. They are the primary motivation behind our concept of generalized hypertext, which is the main focus of this article. We then present the essential ideas of generalized hypertext, followed by a discussion of our implementation of the system.

Basic Hypertext

Our aim in this section is to briefly present and discuss *basic hypertext*. In the following section we shall contrast this with the *generalized hypertext* system that we have conceived, developed, and implemented. Certainly, many existing systems have richer feature sets than we shall describe in connection with basic hypertext, but our focus in this article is on generalizations to the basic hypertext concept. Further, although we shall limit our discussion to hypertext, most of what we say (when not describing our implementation) can be applied as well to hypermedia.[1]

The central concept in hypertext is that of linked collections of information. A hypertext document may be seen as a graph, with *nodes* that are collections of information (called, e.g., windows (Conklin, 1987), documents (Brown, 1987; 1989; Haan, et al., 1992), cards (Apple Computer, 1989; Halasz, 1988), information items (Bhargava, et al., 1988), chunks (or pieces of text) (Koved, 1988; Trigg, 1983), frames (Akscyn, et al., 1988)). *Links* specify relationships between nodes. They may have properties themselves and fall into types. They are maintained by the system, and are named or referred to by *buttons* (also called link icons (Conklin, 1987) and link markers (Halasz and Schwartz 1990)), which normally are found in the nodes. To illustrate, see Figure 1 (based on Conklin, 1987), where nodes *Window A* and *Window B* are presented as windows on the display. Within the nodes are the buttons *x, y, z*. What is displayed represents part of the underlying *hyperdocument* (network of hypertext nodes and links). Window A is a representation of node *AA* in the hyperdocument, and Window B represents *BB*. Similarly, a button, e.g., *x,* represents a particular link in the hyperdocument, e.g., *xx,* which links nodes *AA* and *BB*.

Typically, a user sees a node displayed in a window, its buttons highlighted in some fashion. The user explores the hyperdocument by, e.g., clicking on a particular button, thereby causing the system to find the internal representation of the link named by the button, to then traverse the link, to find the node at the link's endpoint, and to display that node as another text passage. The newly displayed node may have buttons as well, which the user may employ in order to continue exploring the hyperdocument. Alternatively, the user may at any time decide to return to an earlier node and explore from there. Users may continue in this way more or less indefinitely, thereby exploring at will the hypertext network. A particular hyperdocument—a collection of nodes and links—may be thought of as an application written under the hypertext system. It is the system that provides the general means for exploring the particular hyperdocument. Thus, a basic hypertext system may be thought of as operating a *select-traverse-display* loop. The user selects a button, the system traverses the link named by the button, and the system displays the node at the far end of the link, possibly using information picked up in traversing the link.

Typical, basic operations supported by hypertext systems include:

- User-directed navigation (traversal (of links) and display (of nodes)) of the hyperdocument.
- Search and display (for example, the user will provide a search string and the system will search until it finds a node containing that string and then will display the node).
- Map-based navigation (the system displays a graph (called a map or network overview) of the hyperdocument, and the user may direct navigation of the hyperdocument based on the map, whose buttons, when selected, cause the corresponding node in the hyperdocument to be displayed).
- Creation, modification (e.g., editing the contents of a node), and deletion of nodes and links and their attributes.
- Display of link and node attribute information (e.g., the name of the node at a link's endpoint, the type of node or link).

[1] The hypermedia concept extends hypertext to types of information items besides text, such as graphics and sound (Haan, et al., 1991).

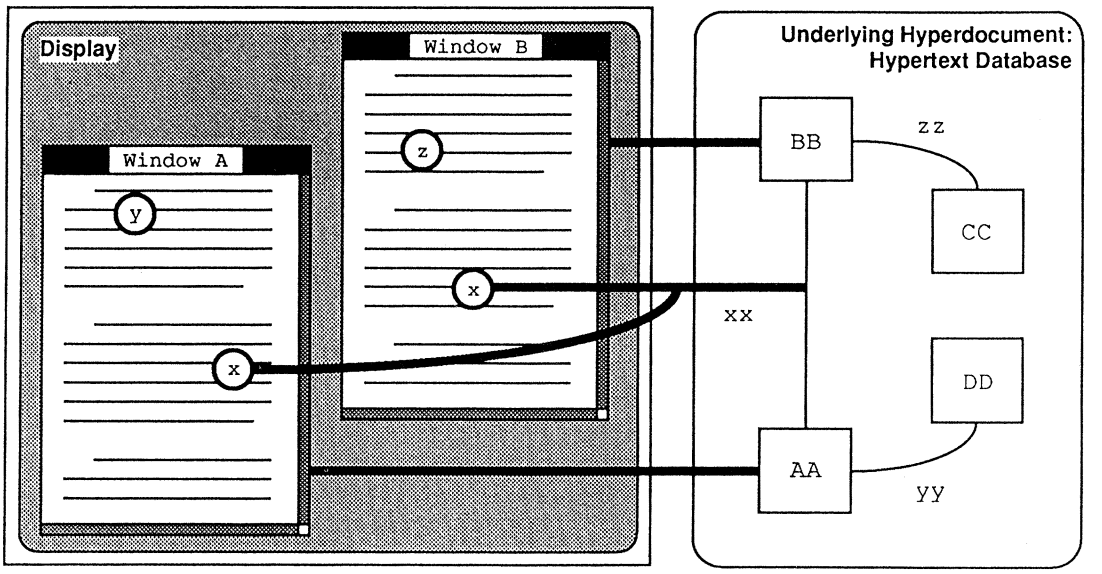

Figure 1. Central Hypertext Concept

- Procedural attachment (link endpoints may be procedures that are activated by traversal of their incoming links. These procedures typically affect how certain nodes are displayed (see Apple Computer, 1989; Halasz, 1988; Koved, 1988, and Thompson, 1990).

However interesting this basic hypertext concept is, and however useful various implementations of it have proved to be, a number of problems and limitations have been identified with this basic concept (Bhargava, et al., 1988; Conklin, 1987; Feiner and McKeown, 1991; Halasz, 1988; Van Dam, 1988). For this article we are concerned with the following widely recognized problems and limitations in basic hypertext (and in many current implementations of hypertext).

- **Manual linking** (Bhargava, et al., 1988; DeRose, 1989; Feiner and McKeown, 1991; Halasz, 1988; Jordan, et al., 1989; Van Dam, 1988). Basic hypertext systems provide editing features for linking existing nodes and for creating and manipulating buttons (link icons). These features axe highly useful to the builder—or annotator—of a hyperdocument. The basic hypertext concept, however, does not encompass inferred or *virtual,* linking of nodes by the system at run time. To illustrate the inferred linking concept (called *implicit linking* by DeRose, 1989), consider a system with predefined keyword nodes, whose contents explain and discuss the keyword in question. The hypertext system might infer a link (and thus the existence of an accompanying button) by being able to recognize keywords in arbitrary nodes and dynamically creating buttons out of them that are linked to the appropriate keyword nodes. With such a capability, a builder could simply type text into a node and have the system create many of the needed buttons and links associated with that node.[2] Clearly, there is considerable potential benefit—especially in terms of reducing the cost of building a hyperdocument—to having the hypertext system capable of creating buttons and links automatically.
- **Manual node creation** (Bhargava, et al., 1988; Halasz, 1988; Jordan, et al., 1989; Parunak, 1988). This is the node version of the above link limitation. Under the basic hypertext concept, the hyperdocument builder builds nodes by using an editor to key in or to paste in information. There remains the possibility of the hypertext system generating nodes (along with

[2] Although we will not discuss it further, this feature is supported in the system we illustrate later. Other researchers have been active in exploring this sort of feature, e.g., in the context of extended electronic mail systems (Ackerman and Malone, 1988; Harp, 1988; Lai, et al., 1988; Jackson and Yankelovich, 1991; Schatz, 1988). Other researchers are working on generating links from content analysis on text (Hammwoehner and Thiel, 1987; Parunak, 1990).

embedded buttons) on the basis of user inputs, in conjunction with existing information in its database (Furuta and Stotts, 1990; Schnase and Leggett, 1989). (See Feiner and McKeown, 1991, for mention of work on generation of graphical objects.) Again, it can be hoped that with such a capability the cost of developing a hyperdocument might be reduced. Finally, we note that automatic creation of nodes is quite different from procedural attachment (see above), which has been used to modify—or modify the display of—nodes, rather than to create them.

- **Network disorientation** (Conklin, 1987; Nielsen, 1990b; Parunak, 1989). This is the often-cited "lost in hyperspace" problem. At-will exploration of a rich hyperdocument can easily lead to user bewilderment. Basic hypertext systems use map-based navigation, logging of nodes traversed, and search-and-display commands as tools for ameliorating this problem.
- **Cognitive overhead disorientation: displayed information** (Conklin, 1987; Glushko, 1989). A main virtue of hypertext is that it provides system-level support for building software that both presents cognitively tractable amounts of information on the screen *and* makes easily accessible arbitrarily large amounts of associated information. In the basic hypertext concept, however, the builder must explicitly design the application's displays. System-level support for tailoring the amount of information displayed and its mode of display is functionality beyond that in the basic hypertext concept.[3]
- **Multiple views** (Halasz, 1988; Koved, 1988; Perlman, 1989). Basic hypertext systems typically provide a limited number of ways to view nodes. For example, many systems permit buttons to be displayed with or without highlighting, and some offer both a user's view and a builder's view for nodes. Other views not envisioned in basic hypertext are possible. As a means of reducing cognitive overhead, nodes might be filtered and transformed for pertinent information before display (Beeri and Kornatzky, 1990; Tompa, 1989). For example, displays specialized by type of user (novice, experienced, e.g.) might be implemented in this way.
- **Cost of building hyperdocuments** (Bhargava, et al., 1988; Jordan, et al., 1989; Kimbrough, et al., 1990a; 1990b). Basic hypertext systems provide substantial support for building applications in which the user may interactively explore a large collection of associated information. Nevertheless, much more might be achieved by embedding knowledge into the hypertext system (DeYoung, 1989). For example, contextual information could be used automatically to invoke filtering routines in support of multiple views of nodes. Also, nodes (and embedded buttons) might be generated automatically, at run time, by machine-based inferential processes. There are many other possibilities as well, e.g., automated node creation and linking.

With the basic hypertext concept and a list of some of its pertinent limitations at hand, we shall now discuss our generalization of the concept and how this generalization addresses the list of limitations.

Generalized Hypertext

Our concept of generalized hypertext is basic hypertext plus generalizations with regard to nodes, links, and link traversal. These generalizations are further extended by system-level support for user and domain contextual dependencies. The aim of this section is to articulate our concept of generalized hypertext by presenting and discussing these generalizations. In the following section, we shall illustrate the generalizations with example's from our implementation.

Node generalization

In basic hypertext, nodes are largely document or card nodes: collections of text with embedded buttons and (often) graphics. Further, these nodes are explicitly represented in the system. We generalize nodes in two principal ways. First, while nodes may be collections of text with embedded buttons, under our concept a node

[3] It is not, however, entirely beyond the functionality of some hypertext systems. As noted above, some systems allow procedural attachment for altering node display characteristics. "Card-based" hypertext systems (e.g., Akscyn, et al., 1988; Apple Computer, 1989; Halasz, 1988), restrict the size of nodes in some way in order to limit—in a limited way—the amount of information available on screen at any time.

may be any information item (structured bit stream, e.g., a document, a symbol, a picture, and so forth) about which the system may reason. Just about any sort of entity may be the endpoint of a link (including other links), and all such endpoints are considered to be nodes. Abstractly, nodes are objects that may be named (referred to) in various ways (explicitly or implicitly) and linked to other nodes. Further, information about nodes may be declared in the system, and this information (including contextual information) may be used by the system during its link traversal operations. Our second generalization is that nodes need not be explicitly represented in the system. They may be *virtual,* i.e., inferred (or computed (Halasz, 1988)) at run time from declarations used to build the system, as well as from other information, such as user inputs and attached applications (such as TEFA, see below).

For example, in the decision support system supported by our generalized hypertext implementation, (illustrated later), models are represented in the attached application, TEFA, and every declared model is also a node. Linked virtually to every model are the results of various operations on it, e.g., describing it and evaluating it. These results are themselves nodes, typically document nodes with embedded buttons, and are created in real time during operation of the system.

Link generalization

In basic hypertext, each link establishes a relation between a single source node and a single destination node, called the *link endpoint.* We generalize links in two principal ways. First, links may fork into multiple links. Thus, in selecting a button, which names a link, the user may then be asked to choose among several sub-links. (We call such collections of sub-links *link ensembles.)* For example, later we shall see that the name of a mathematical model may be a button in a document. Upon selecting such a button, the link traversal routine will infer that several analysis options are presently available, e.g., to run the model, to describe the model, and to suggest a scenario (data set) for running the model (see Figure 2). There are also two hypertext documentation options for adding a comment and for initiating a user-declared (i.e., explicit) link.

Each of these sub-links, or link forks, is traversable by the system and may be thought of as a command. In basic hypertext each analysis link may be thought of as a command to display one of the two endpoints of a link. This generalization allows arbitrary commands for operating upon a link endpoint and is a richer concept than that normally encompassed by procedural attachment.

Our second generalization is that links need not be explicitly represented in the system. Like nodes, they may be inferred at run time from declarations used to build the system, as well as from other information, such as user inputs, attached applications, and context. In fact, generalized hypertext buttons will often indicate the presence of such virtual links. These links are not generated until the user actually chooses to traverse them.

Generalizing link traversal

In basic hypertext, as noted above, link traversal is normally performed through a select-traverse-display model: the user selects a button (e.g., by pointing to it with a mouse and clicking on the mouse), the system finds the link named by the button, traverses it, and displays the node found at the link's endpoint. (In the case of procedural attachment, the system may find a procedure at a link endpoint. If so, the system calls the procedure, which normally changes the content or display of a node.)

We generalize link traversal as follows. Inference (indeed, arbitrary processing) may occur both before and after traversal of a link. After the user selects a button, the system may perform a series of inferences in order to determine what the available links are (i.e., the system collects the link ensemble), possibly taking context into account. If there are several options available, the user is then prompted to choose a particular sub-link and (when needed) to supply parameters. (Alternatively, the system may invoke a default.) Inferencing is performed again in order to validate the refined request. Upon successful validation, the system determines (finds or generates) the appropriate sub-link and traverses it. Traversal—which may itself be a complex inferencing process and may use application-level procedures—produces a symbol that names a node. Inferencing, or processing, is then performed on that symbol (e.g., for the purpose of formatting and display). Usually, this final inferencing results in display of a new node. Thus, our generalization

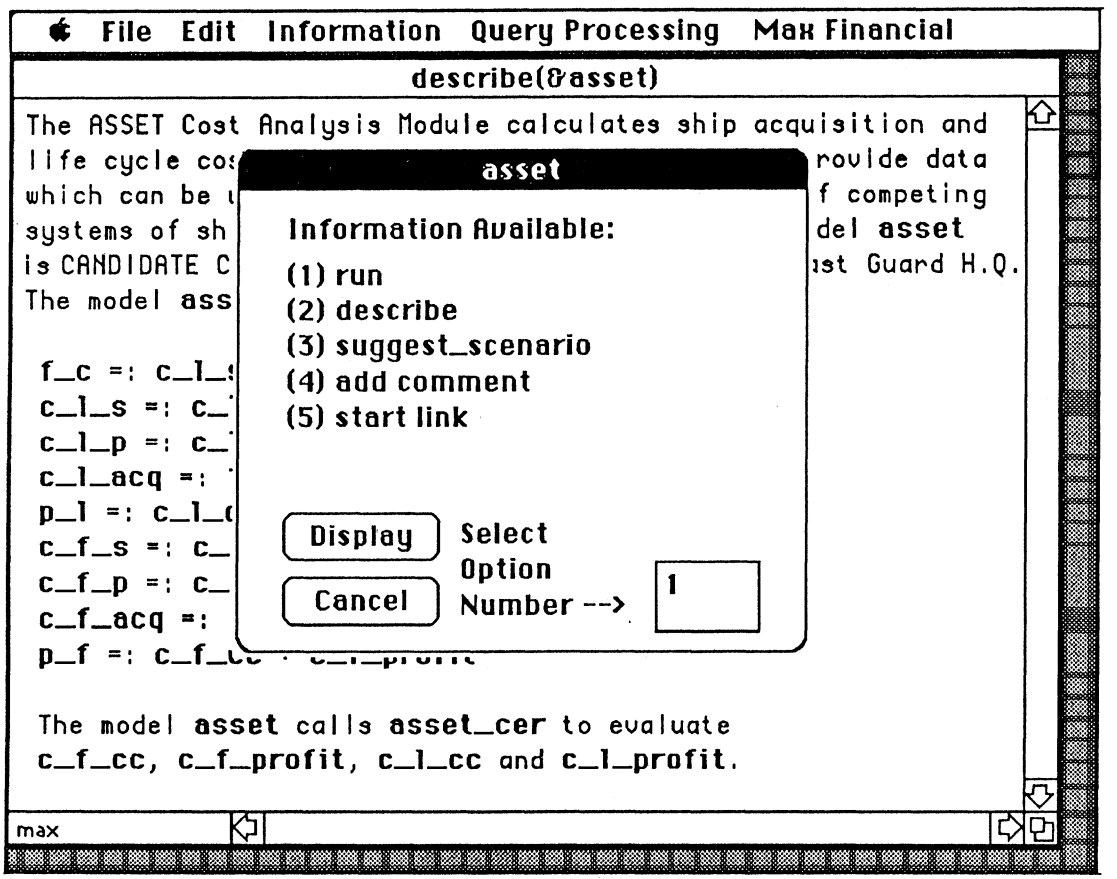

Figure 2. Inferred Link Ensemble Associated With the "Asset" Button

follows a *select-infer-traverse-infer* model. (For more details see Bieber, 1992.)

Use of the generalizations

Under our concept of generalized hypertext, nodes are objects (declared or inferred), and links declare operations that may be applied to objects, usually producing a hypertext document upon completion. These generalizations are the outcome of our intention to construct a hypertext system in which the cost of building hyperdocuments is greatly reduced through automatic creation of nodes and links on the basis of application-dependent declarations. System-level procedures that implement these generalizations work on application-specific declarations in order to make the necessary inferences for automatic linking, automatic node creation, and support for multiple views of nodes, links, and buttons. An important element of our design concept is that the application should declare—explicitly or implicitly—what is important to it, and the generalized hypertext system should exploit these declarations in order to infer links, nodes, and views. (This is done, in our system, through the use of universally quantified generalizations, which we call *bridge laws*. The purpose of bridge laws is to map terms in the attached applications (such as TEFA, see below) to expressions (nodes, links, and so forth) native to the generalized hypertext system. Detailed discussion of this technique is beyond the scope of this article. For further information see Bieber, 1990; 1992; Bieber and Kimbrough, 1990. Further, it is our hope that network and cognitive disorientation are reduced by inferencing procedures that are broadly available for reporting on and explaining various system entities, notably nodes, links, and buttons. The essential idea is to employ a stan-

dard format to declare information about system entities (nodes and links); use generic (application-independent) inferencing procedures to generate nodes, links, and alternate views; and provide system-level explanation features.

We shall now illustrate these ideas with a discussion of our implementation of them.

Illustration: Max, a DSS Shell

Max is a generalized hypertext knowledge-based DSS shell. It is written in Prolog and is currently being used at the Research and Development Center at the Office of Engineering, and elsewhere, in the U.S. Coast Guard. Max has two main modules: a user interface subsystem, called Maxi (Max Interface), and a model and data management subsystem, called TEFA (The Eileen Ford Agency, model management being such a "fashionable" subject). The two subsystems have no code in common and communicate via expressions in a formal communications language, the expressions of which are formatted and interpreted in an elementary message management system (Kimbrough and Moore, 1992). In Figure 3, A is the user, B is the communications path on which messages from the message management system flow, and C is the locus of external procedures (e.g., subroutine libraries and commercial model solvers) and data (e.g., in database management systems).

Maxi is a standalone event-driven generalized hypertext editing and management system. It dynamically creates user interface environments based on requests from TEFA, which are tailored using context information about the task and the user (Halasz, 1988). As seen in Figure 3, Maxi has two main components. The user communicates directly with the dialog subsystem, which handles the physical input and output. The (largely) configuration-independent hypertext subsystem passes information between TEFA and the dialog subsystem, performing hypertext editing and inferencing as necessary. TEFA is a domain-independent model and data management system currently supporting models that

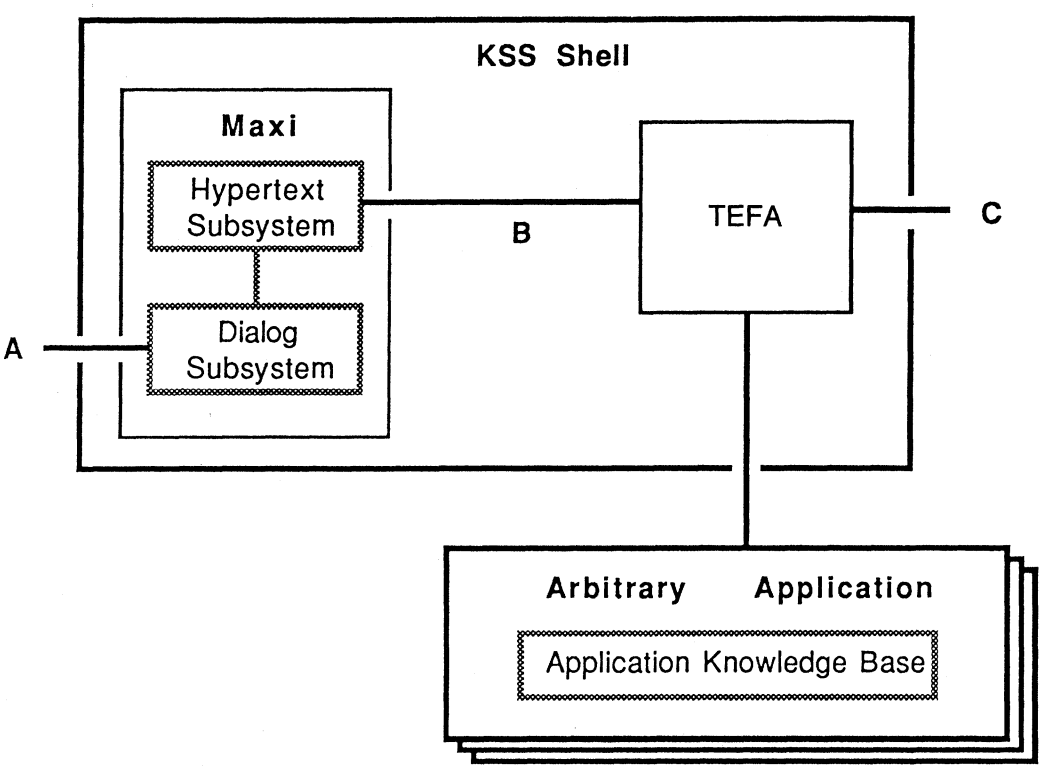

Figure 3. Max KSS Shell High-Level Architecture

can be expressed as mathematical equations. For the application at hand, it provides a modeling language (which DSS builders use to record the domain models and data) and information about the models and data (Bhargava, et al., 1988; Bhargava and Kimbrough, 1990; 1992).

Maxi currently runs only on a Macintosh computer. TEFA is written in generic Prolog and is therefore substantially configuration-independent. When not combined with Maxi it has its own generic Prolog command language component and is currently functioning in this way in the VAX/VMS environment. For TEFA to function in an event-driven windowing environment, these commands are translated by the message management system into the communications language.

Example: Working with a cost model

In order to give a sense of how Max works, we shall discuss some of the features a user would employ in a Max application—called Max Financial—to work with a particular model. We shall use as our example a model called Asset, which is used by the Navy and the Coast Guard to estimate ship acquisition and life cycle costs. Asset was originally implemented in Fortran. We reimplemented it in the model representation language of TEFA. The various reports and features we shall illustrate are produced inferentially at run time by Max, i.e., by our generalized hypertext system. They are automatically available for any model declared in TEFA. (It has been our experience that this strategy significantly hastens the building of a particular DSS (Kimbrough, et al., 1990a; 1990b.)

Max was designed to support two types of users; analysts and executive browsers (or other readers of prepared reports). Each typically approaches the system with a different purpose. Analysts execute models under various data scenarios. Information is returned in standard reports that are dynamically created by the system. The analysts can then copy and paste from these standard reports to create their own ad hoc final reports. Again, it is important to note that Max dynamically generates standard reports and automatically embeds buttons that name generalized hypertext links. Copying and pasting preserves these links in both the original and duplicate copies. (The computational cost of this is not excessive because buttons name—or refer to—links, and these buttons are copied, not the links.) Executives, on the other hand, generally do not build reports. Instead, they typically read reports produced on the system by analysts. Because analysts can easily include generalized hypertext buttons in their reports, executive browsers have access to the same standard reports and other generalized hypertext information as their analysts. This allows executives to explore the information supporting the analyst's recommendations and findings without placing undue burden on the analysts.

Imagine that an executive needs to make a decision based on the costs of two different fleets, one consisting of hydrofoils and the other of SWATH (Small Waterplane Area Twin Hull) vessels. The analyst's task, then, is to perform an analysis exercising the Asset life cycle cost model to create a report. The steps taken are outlined in Figure 4.

Steps 1 and 2: Analyst's Point of View. After starting the Max session, the analyst asks for a full description of the Asset model. To do this he or she selects the *describe* command/query from a menu in Max's menu bar and the Asset model as the subject of the command. The system produces a standard report model description (displayed in an interactive document), shown in Figure 5. Buttons are highlighted in boldface, indicating that further information is available about the objects they represent.

Steps 1 and 2: System's Point of View. During Max's initialization, the Max Financial application (under TEFA) passed a list of command options in the communications language to Maxi, the shell interface subsystem, which installed them in the menu bar under the heading *Max Financial* (see Figure 5). When the analyst chose one of these items, he or she initiated the following dialog with the Max Financial application.

- Dialog Subsystem
  - — Intercepts user input.
  - — Passes the hypertext subsystem a message relaying that the user selected the *describe* menu item and entered the text string "asset."
- Hypertext Subsystem
  - — Receives the *describe* menu item and associated text string.

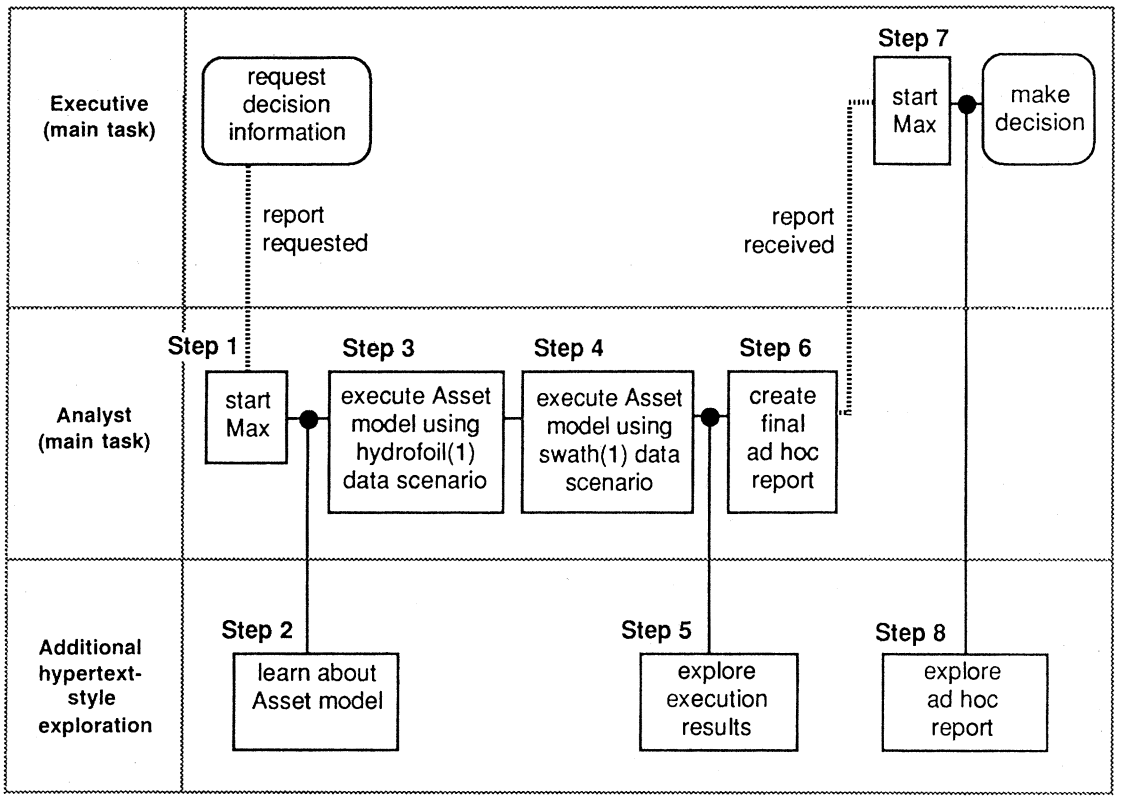

Figure 4. An Analysis Task Supported by Max

— Determines that this is a generalized hypertext link traversal request.
— Infers that processing must be performed by TEFA to determine the destination node.
— Formulates a link traversal request to TEFA (which TEFA will perceive as a command request) in the formal communications language.

- TEFA
  — Receives the *describe* command request with input text string "asset."
  — Infers that the text string "asset" represents a Max Financial model.
  — Infers the generic *describe* report model for a (financial) model.
  — Executes the generic report model for the *asset* financial model. This involves inferring the financial model's description, source information, equations, and any related submodels. The knowledge base is used to derive these elements, many of which themselves are entities containing subcomponents.
  — TEFA generates a composite report containing this information (which Maxi will perceive as the destination node), formats it in the formal communications language, and passes it to the hypertext subsystem.

- Hypertext Subsystem
  — Receives the link's destination node from TEFA.
  — Processes the node's contents into a format that the dialog subsystem can use. This involves tagging buttons with an internal ID and formulating display information (e.g., the buttons' base display text, and whether each is textual, numeric, monetary).
  — Passes the destination node's contents to the diaglog subsystem.

- Dialog Subsystem
  — Receives an ordered set of text and buttons from the hypertext subsystem.
  — Processes the data to create an interactive document. The button information compiled by the hypertext subsystem is used to

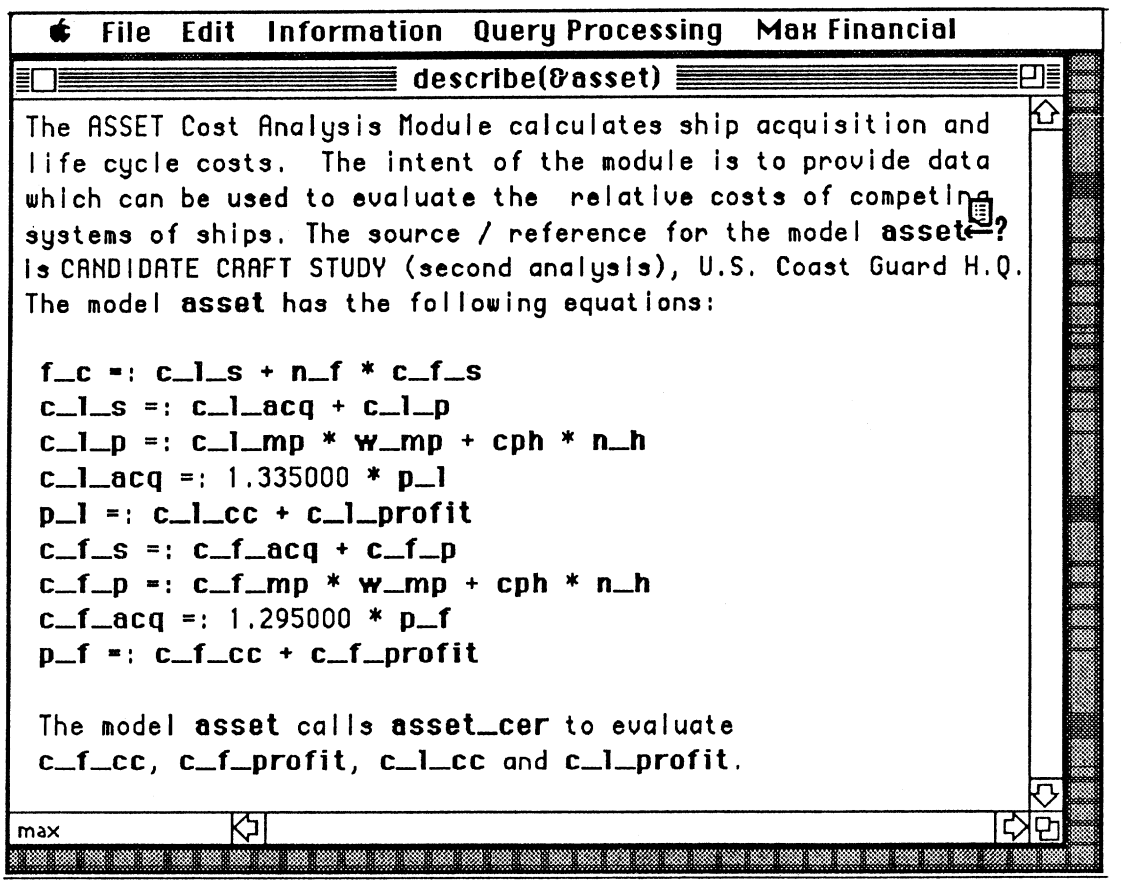

Figure 5. Standard Report Describing the Asset Model

determine the actual text representation of each button on the screen.

— Displays the interactive document shown in Figure 5 on the screen.

The highlighted buttons in an interactive document denote links, real or virtual. Although they indicate a relation to information in the knowledge base, they (usually) are not *explicitly* linked to anything. As we shall see, only when they are queried directly will a link be determined and traversed.

Step 3a: Analyst's Point of View. Next, suppose the analyst wants to execute the Asset model under two scenarios. The analyst presses the (Macintosh) option key, and the "show me all available options" cursor appears as shown in Figure 5 (near the upper right-hand corner). The analyst then clicks the mouse on one of the "asset" buttons. Figure 2 shows the list of available options (i.e., generalized hypertext links), which is generated inferentially. Thus, we say that a button names a *link ensemble,* or bundle of links, rather than a single link as in basic hypertext.

Step 3a: System's Point of View. What happened internally is that the system interface determined that the user clicked on a set of known entities: the *asset* button, the report node, and the interface window itself. By default the system chose to take the most specific entity (i.e., the button) and translated its internal ID to its TEFA identifier in Max's generic "What can I do with this?" query. To the options (sub-links) returned by the application (TEFA), Maxi's hypertext subsystem added the hypertext documentation options for commenting and user-declared linking.

Steps 3b to 5: Analyst's Point of View. From this (filtered) list of options the analyst executes

the Asset model twice, once with the scenario (data set) "hydrofoil(l)" and once with "swath(l)." The analyst believes the variable "f_c" stands for the total fleet costs, but just to be sure, clicks on it to ask for a short description. The generalized hypertext reports generated are shown in Figure 6. These standard reports are quite sparse, but they could be made more explicit for, say, a novice analyst. Alternatively, if only the total fleet cost were needed, a report with only this value could have been returned.

Steps 3b to 5: System's Point of View. In order to execute the Asset model for the user, the system traversed a virtual "execution" link (as opposed to the other options of traversing a "describe" link or a "suggest scenario" link, etc.) from a report button for a model node. As a result of following the execution link, the model was executed, and, as it happens, a standard report node comprising the major resulting values was generated as the link destination (i.e., the node was created) and passed to the interface for display.

Steps 6 to 8. Figure 7 shows the final ad hoc report that the analyst has constructed for the executive browser. Note that the analyst has had to do no explicit linking. Instead, the automatic links were carried over through the buttons (boldface text in the figures) via copy and paste operations and editing of the final report. Although the final report is quite short, an executive or browser can query any button for further detail (Bieber, 1992). In fact, a large amount of information is available in this fashion. For example, in Figure 7 the executive has queried the value representing the SWATH fleet cost. The system recognized that this button represents the result of a model execution and determines that

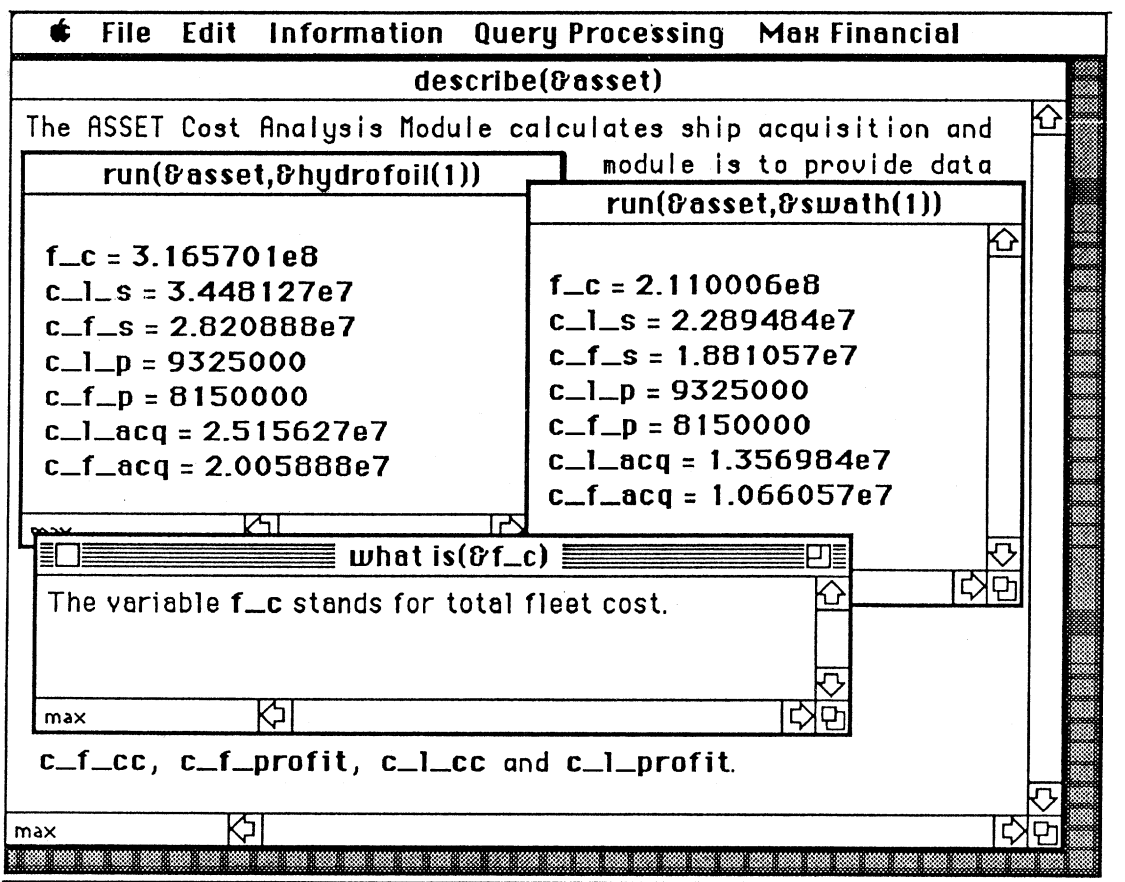

Figure 6. System Generated Reports From Executing the Asset Model and Describing the f_c Variable

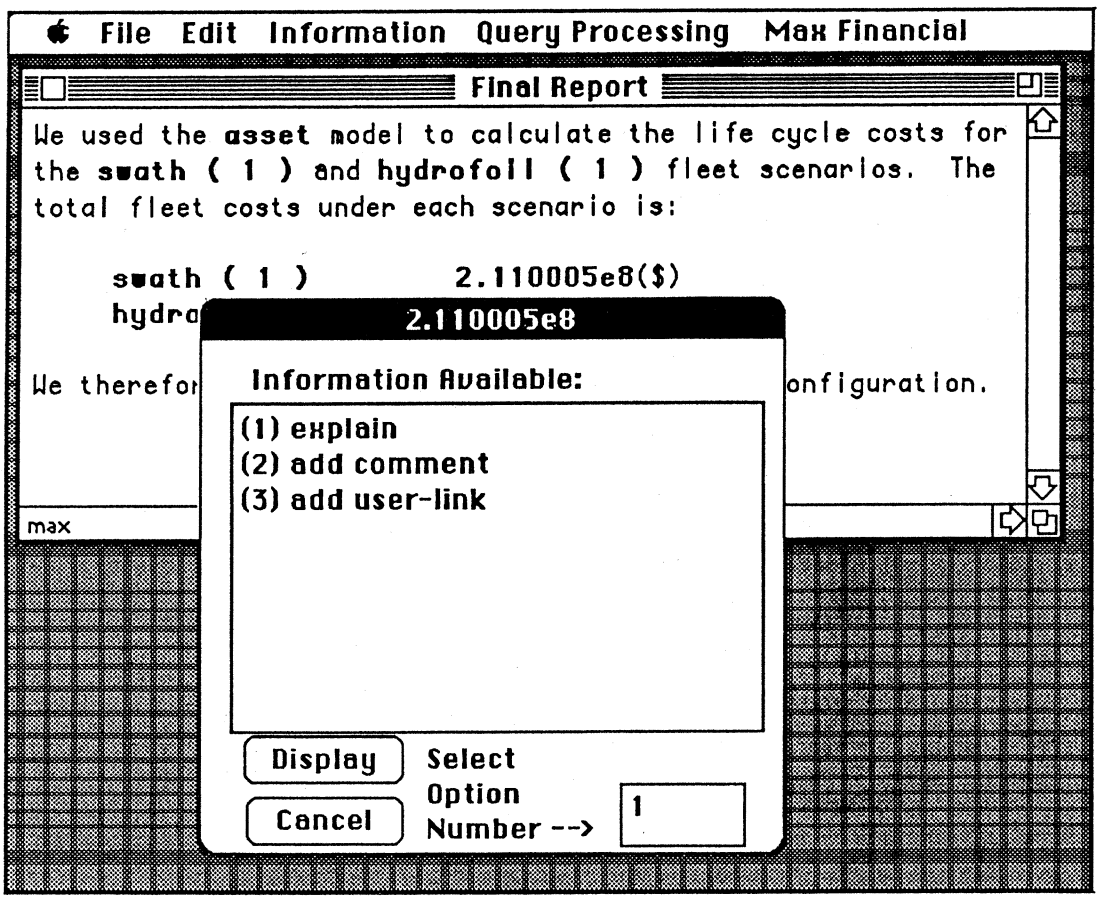

Figure 7. Analyst's Final Report and Link Ensemble for SWATH Fleet Cost

an explanation can be generated dynamically. This explanation appears in Figure 8, complete with automatic links from the embedded buttons to provide further information for the executive.

Max supports many other features (e.g., explicit creation of nodes, links, and buttons; user-induced, machine-interpretable comments on models and data) and contains several substantial mathematical models, but discussion of these lies beyond our current purpose, which is to present and discuss generalized hypertext and the essentials of our implementation of it.

Discussion and Conclusion

Hypertext is recognized as a method for reducing the human operating cost and cognitive overhead of using information systems. Generalized hypertext is a method for reducing the cost and effort of creating and using hypertext systems. By providing efficient and standard techniques for incorporating necessary and recognized features, our generalization of the concept of hypertext provides a robust basis for knowledge-based hypermedia systems.

Our experience with the Max system has demonstrated that generalized hypertext concepts are both computationally feasible and useful for builders, as well as users, of hypertext systems. By way of summary, we shall comment briefly on the relation of generalized hypertext to the problems and limitations of basic hypertext, presented earlier.

- **Manual link creation, manual node creation.** Under our concept of generalized hypertext, both links and nodes may be either explicit (as in basic hypertext) or virtual (created during run time by the system, based on inference

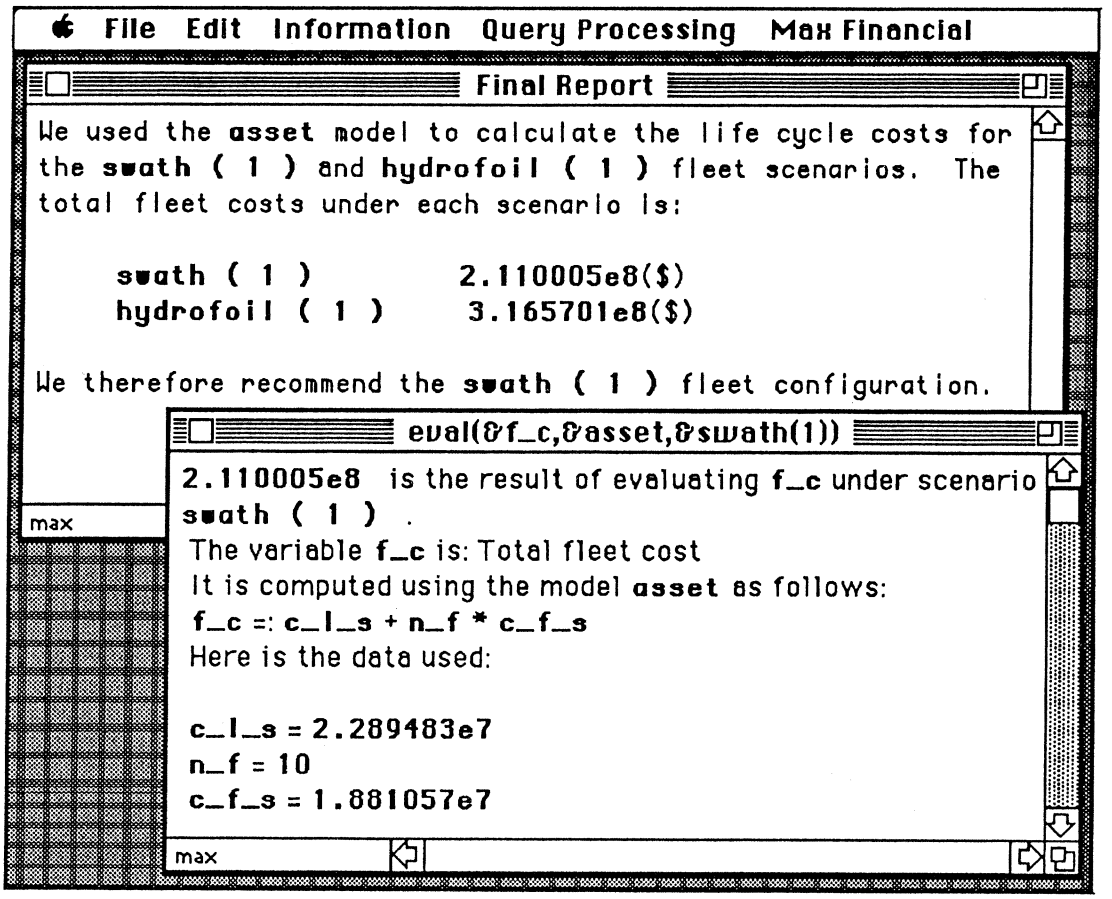

Figure 8. System Generated Report Explaining the SWATH Fleet Cost Number in the Analyst's Final Report

from general declarations). We have demonstrated this concept with an implementation of a DSS shell and model management system, called Max, which is currently in use by the U.S. Coast Guard. Applications of Max beyond DSS have been developed to the prototype stage (e.g., for project management and for executive information systems) but are not yet deployed. The generalized hypertext system is indeed quite general and application-independent. Automated link and node creation can be thought of as being based on a declared theory of what is important to the application. In the Max Financial system (the DSS application described above), models are important, data are important, and things that can be done with them (e.g., describe, explain, run, suggest a scenario) are important, and the system contains internal declarations that say so. In the case of DSS and model management, it is fairly clear what sorts of things are important and hence should be used for automatic link and node creation. The extensibility of the generalized hypertext idea to other application domains depends upon whether it is possible to state a general, broad theory (set of declarations) regarding what is important in that domain. Without such declarations (bridge laws or something like them), it would of course be impossible for automatic node and link creation to be practicable.

- **Network disorientation.** We do not claim to have a major advance on this problem. We have, however, learned something that suggests some progress in this area. Our system is, as we have seen, oriented toward helping

an analyst construct an interactive document with generalized hypertext functionality. The analyst or document builder can create one or more documents that are organized as the analyst sees fit. Text and buttons can be copied from other documents, normally generated by the system, into a document of the analyst's choice. (Again, this was illustrated above.) Working with the system in this way, the hypertext network tends to take on a nice structure. Instead of being an essentially arbitrary network of nodes (documents) and links, the hyperdocument begins to have the structure of a collection of trees with a common root, where that root is the document the analyst is using the system to create. One begins at the document, forays out for information, copies the information (including the buttons), returns to the document (with a click on its window), pastes in the retrieved information, makes any needed editing changes, and repeats the basic cycle until done. Similarly, browsers may use a single document or set of documents as a home base for exploration. We believe this structure for hypertext sessions reduces the problem of network disorientation, but proving it is another matter and will be the subject of future research efforts.

- **Cognitive overhead disorientation, multiple views.** Because links and nodes may be created automatically based on declarations from the application, it is possible to set up the declarations in a way that results in salient chunking of the information. What counts as salient chunking, of course, is difficult to determine and will likely be application-dependent. We have illustrated above the outcome of our choices in the context of DSS and model management. How, in general, information should be organized for presentation under hypertext is a proper subject for an extended research program. Generalized hypertext, as we have presented it, does not solve this problem or complete the research. Instead, it facilitates implementation once a solution—a salient chunking—is chosen. By making the application-specific declarations (including contextual information) in the right way, the generalized hypertext system can be made to present information in cognitively useful chunks, and this can be done in a context-sensitive fashion (Bieber, 1991b). Much remains to be learned on this subject.

- **Cost of building hyperdocuments.** It is evident that by relying on automatic generation of nodes and links through runtime inferencing on general declarations, generalized hypertext can greatly reduce the cost of creating specific hyperdocuments. This is particularly clear for the application we have chosen to illustrate here. In Max Financial the number of virtual nodes is essentially infinite. There is a generalized hypertext node corresponding to every possible parameter setting for every model in the system, but only a few nodes are ever actualized in any given session. Also, it should be emphasized that from the DSS builder's point of view the generalized hypertext features "come for free." Specifically, models, data, and information about them are declared in a simple, straightforward language. (The same technique is used in other application areas as well, e.g., project management.) Once declared, this information is fully available to the generalized hypertext system. The builder inputs (explicit) information about models and data, not about windows, buttons, and links. It is this latter information that the system provides and manages.

Finally, it might be asked, "What does generalized hypertext do for the user?" In one sense it does nothing, nor should it. Debuggers and compilers do not deliver new functionality for end users. Instead, they make it easier and cheaper to produce application software. Similarly, generalized hypertext—based on our experience—makes it easier and cheaper to produce hyperdocuments for those who can use them in their jobs. In Max, for example, a few pages of declarations describing a mathematical model (such as the Asset model) can result in thousands of (virtual) hypertext links that can be generated dynamically by the system. Most of these links get used. Without automatic generation, however, it simply would not be cost effective to set up the links manually, i.e., with a standard hypertext editor.

In sum, while we have installed a functioning generalized hypertext DSS, there is still very much room for further work. We view our current system as a platform for investigating orientation, scaling, networking, and knowledge-base maintenance issues, and particularly for further exploiting contextual clues. We are improving our specifications of contexts and task environments

and are working on incorporating a hypermedia model input and modification subsystem, as well as a hypermedia project management system, as standard features of the DSS. We are also modeling a "hypertext engine" to make generalized hypertext available to information systems other than DDS (Bieber, 1991). These are topics for forthcoming systems and future papers, both of our own and, we hope, of others.

Acknowledgements

Special thanks to Hemant K. Bhargava and Christopher V. Jones for stimulating discussions, ideas, and comments on an earlier draft. Dr. Bhargava is largely responsible for the TEFA subsystem mentioned in the body of the article. This work was funded in part by the U.S. Coast Guard under contract DTCG39-86-C-E92204 (formerly DTCG39-86-C-80348), Steven O. Kimbrough principal investigator.

References

Ackerman, M.S. and Malone, T.W. "Intelligent Agents, Object-Oriented Databases, and Hypertext," in *AAAI-88 Workshop, AI and Hypertext: Issues and Directions,* St. Paul, MN, August 23, 1988, pp. 1-3.

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

Apple Computer, Inc. *HyperCard User's Guide,* Cupertino, CA, 1989.

Beeri, C. and Kornatzky, Y. "A Logical Query Language for Hypertext Systems," *Hypertext: Concepts, Systems, and Applications. Proceedings of the European Conference on Hypertext '90,* Cambridge University Press, Cambridge, England, 1990, pp. 67-80.

Bhargava, H.K. and Kimbrough, S.O. "On Embedded Languages for Model Management," *Proceedings of the Twenty-Third Hawaii International Conference on System Sciences,* 1990, pp. 443-452.

Bhargava, H.K. and Kimbrough, S.O. "Model Management: An Embedded Languages Approach," *Decision Support Systems,* forthcoming, 1992.

Bhargava, H., Bieber, M., and Kimbrough, S.O. "Oona, Max, and the WYWWYWI Principle: Generalized Hypertext and Model Management in a Symbolic Programming Environment," *Proceedings of the Ninth International Conference on Information Systems,* Minneapolis, MN, November 30-December 3, 1988, pp. 179-191.

Bieber, M.P. *Generalized Hypertext in a Knowledge-Based DSS Shell Environment,* doctoral dissertation, University of Pennsylvania, Philadelphia, PA, 1990.

Bieber, M.P. "Issues in Modeling a 'Dynamic' Hypertext Interface," *Proceedings of Hypertext '91,* San Antonio, TX, December 1991a, pp. 203-218.

Bieber, M.P. "Template-Drive Hypertext: A Methdology for Integrating a Hypertext Interface into Information Systems," working paper, BCCS-91-4 Computer Science Department, Boston College, Boston, MA, 1991b.

Bieber, M.P. "Automating Hypermedia for Decision Support," *Hypermedia,* forthcoming, 1992.

Bieber, M.P. and Kimbrough, S.O., Towards a Logic Model of Generalized Hypertext," *Proceedings of the Twenty-Third Hawaii International Conference on System Sciences,* 1990, pp. 506-515.

Brown, P.J. "Turning Ideas into Projects: The Guide System," *Hypertext '87 Proceedings,* Chapel Hill, NC, November 1987, pp. 33-39.

Brown, P.J. "A Hypertext System for UNIX," *Computing Systems* (2:1), 1989, pp. 37.53.

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

DeRose, J. "Expanding the Notion of Links," *Hypertext '89 Proceddings,* Pittsburgh, PA, 1989, pp. 249-258.

DeYoung, L. "Hypertext Challenges in the Auditing Domain," *Hypertext '89 Proceedings,* Pittsburgh, PA, 1989, pp. 169-180.

Feiner, S.K. and McKeown, K.R. "Automating the Generation of Coordinated Multimedia Explanation," *IEEE Computer* (24:10), Ocober 1991, pp. 33-41.

Furuta, R. and Stotts, P.D. "The Trellis Hypertext Reference Model," *Proceedings of the Hypertext Standardization Workshop,* NIST Special Publication SP500-178, Gaithersburg, MD, January 1990, pp. 83-94.

Gloor, P.A. "CYBERMAP: Yet Another Way of Navigating in Hyperspace," *Hypertext '91 Proceedings,* San Antonio, TX, December

1991, pp. 107-121.

Glushko, R. "Design Issues for Multi-Document Hypertexts," *Hypertext '89 Proceedings,* Pittsburgh, PA, November 1989, pp. 57-60.

Haan, B.J., Kahn, P., Riley, V., Coombs, J.H., and Meyrowitz, N.K. "IRIS Hypermedia Services," *Communications of the ACM* (35:1), January 1992, pp. 36-51.

Halasz, F.G. "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems," *Communications of the ACM* (31:7), July 1988, pp. 836-855.

Halasz, F.G. and Schwartz, M. "The Dexter Hypertext Reference Model," *Proceedings of the Hypertext Standardization Workshop,* NIST Special Publication SP500-178, Gaithersburg, MD, January 1990, pp. 95-134.

Hammwoehner, R. and Thiel, U. "Content Oriented Relations between Text Units—A Structural Model for Hypertexts," *Hypertext '87 Proceedings,* Chapel Hill, NC, November 1987, pp. 155-174.

Harp, B. "Facilitating Intelligent Handling by Imposing Some Structure on Notes," *AAAI-88 Workshop, AI and Hypterext: Issues and Directions,* St. Paul, MN, August 23, 1988, pp. 79-83.

Jackson, S. and Yankelovich, N. "Intermail: A Prototype Hypermedia Mail System," *Hypertext '91 Proceedings,* San Antonio, TX, December 1991, pp. 405-409.

Jordan, D.S., Russell, D.M., Jensen, A.S., and Rogers, R.A. "Facilitating the Development of Representations in Hypertext with IDE," *Hypertext '89 Proceedings,* Pittsburgh, PA, November 1989, pp. 93-104.

Kimbrough, S.O. "On Shells for Decision Support Systems," working paper, Decision Sciences Department, University of Pennsylvania, Philadelphia, PA, 1986.

Kimbrough, S.O. and Moore, S.A. "Message Management Systems: Concepts and Prospects," *Proceedings of the Twenty-Fifth Annual Hawaii International Conference on System Sciences,* January 1992.

Kimbrough, S.O., Pritchett, C.W., Bieber, M.P., and Bhargava, H.K. "An Overview of the Coast Guard's KSS Project: DSS Concepts and Technology," *Transactions of DDS-90, Tenth International Conference on Decision Support Systems,* Boston, MA, May 21-23, 1990a, pp. 63-77.

Kimbrough, S.O., Pritchett, C.W., Bieber, M.P., and Bhargava, H.K. "The Coast Guard's KSS Project," *Interfaces* (20:6), November-December 1990b, pp. 5-16.

Koved, L. "Imposing Usability and User Performance with Hypertext Documents," *AAAI-88 Workshop, AI and Hypertext: Issues and Directions,* St. Paul, MN, August 23, 1988, pp. 121-122.

Lai, K., Malone, T.W., and Yu, K. "Object Lens: A 'Spreadsheet' for Cooperative Work," *ACM Transactions on Office Information Systems* (6), 1988, pp. 332-353.

Minch, R.P. "Application and Research Areas for Hypertext in Decision Support Systems," *Journal of Management Information Systems* (6:3), Winter 1989/90, pp. 119-138.

Nielsen, J. *Hypertext and Hypermedia,* Academic Press, New York, NY, 1990a.

Nielsen, J. "The Art of Navigating Through Hypertext," *Communications of the ACM* (33:3), March 1990b, pp. 296-310.

Parunak, H.V. "AAAI Hypertext Position Paper," *AAAI-88 Workshop, AI and Hypertext: Issues and Directions,* St. Paul, MN, August 23, 1988, pp. 140-142.

Parunak, H.V. "Hypermedia Topologies and User Navigation," *Hypertext '89 Proceedings,* Pittsburgh, PA, November 1989, pp. 43-50.

Parunak, H.V. "Toward a Reference Model for Hypermedia," *Proceedings of the Hypertext Standardization Workshop,* NIST Special Publication SP500-178, Gaithersburg, MD, January 1990, pp. 197-209.

Perlman, G. "Asyncronous Design/Evaluation Methods for Hypertext Technology Development," in *Hypertext '89 Proceedings,* Pittsburgh, PA, 1989, pp. 61-81.

Schatz, B.R. "Proposal to Attend Workshop on 'AI and Hypertext'," *AAAI-88 Workshop, AI and Hypertext: Issues and Directions,* St. Paul, MN, August 23, 1988, pp. 147-149.

Schnase, J.L. and Leggett, J.J. "Computational Hypertext in Biological Modelling," *Hypertext '89 Proceedings,* Pittsburgh, PA, November 1989, pp. 181-198.

Schneiderman, B. and Kearsley, G. *Hypertext Hands-On! An Introduction to a New Way of Organizing and Accessing Information,* Addison-Wesley Publishing Company, Reading, MA, 1989.

Thompson, C. "Strawman Reference Model for Hypermedia Systems," *Proceedings of the Hypertext Standardization Workshop,* NIST Special Publication SP500-178, Gaithersburg, MD, January 1990, pp. 197-209.

Tompa, F. "A Data Model for Flexible Hypertext Database Systems," *ACM Transactions on Information Systems* (7:1), January 1989, pp. 85-100.

Trigg, R.H. *A Network-Based Approach to Text Handling for the Online Scientific Community,* doctoral dissertation, University of Maryland, College Park, MD, 1983.

Van Dam, A. "Hypertext '87: Keynote Address," *Communications of the ACM* (31:7), July 1988, pp. 887-95.

About the Authors

Michael P. Bieber is assistant professor of information systems at Boston College, Carroll School of Management. He received his B.S.E. in computer science and Ph.D. in decision sciences at the Wharton School, University of Pennsylvania. His research interests include hypertext, information presentation, and knowledge-based decision support.

Steven O. Kimbrough is associate professor in the Decision Sciences Department, the Wharton School, University of Pennsylvania, and is currently the William Davidson Visiting Professor of Computer and Information Systems at the School of Business Administration, University of Michigan. He received his M.S. (industrial engineering) and Ph.D. (philosophy) degrees from the University of Wisconsin. His research interests include hypertext, decision support systems, electronic commerce, and logic modeling. Since 1986, he has been principal investigator for the U.S. Coast Guard's KSS (knowledge-based decision support systems) project.