38

N91-21951

Automating Hypertext for Decision Support

Michael Bieber
Boston College, Carroll School of Management
Chestnut Hill, MA 02167-3808
(617) 552-3964 Email: bieberm@bcvms.bitnet

Abstract
We are constructing a decision support system (DSS) shell that can support applications in a variety of fields, e.g., engineering, manufacturing, finance. The shell provides a hypertext-style interface for *navigating* among DSS application models, data and reports. To do so we had to enhance the traditional notion of hypertext. Hypertext normally requires manually, pre-defined links. A DSS shell, however, requires that hypertext connections be built "on the fly". In this paper we discuss the role of hypertext in augmenting DSS applications and the decision-making process, and how we automatically generate hypertext nodes, links and link markers tailored to an arbitrary DSS application.

One of our main research goals is to make it easier and cheaper to build and use decision support systems (DSS). We want to enable application builders to construct DSS applications with more functionality than traditional DSSs have, and to do so more quickly. This goal led to the idea of a DSS shell supporting multiple DSS applications[1] (see Figure 1) that produce *interactive documents*, such as those in Figures 2a and 2b. Interactive documents give direct access to reports, operations, and other components of DSS applications. The navigation and clarity of presentation required by interactive documents suggest a hypertext-style interaction, but most current hypertext systems lack the dynamic and general characteristics inherent in a DSS shell. The heart of this research is our model of *generalized hypertext*, which automates and generalizes the standard notion of hypertext for a knowledge-based DSS shell. Generalized hypertext superimposes a hypertext *network* on a DSS application, generating all nodes, links and link markers dynamically from the application's standard, non-hypertext knowledge base. Happily, this occurs unbeknownst to DSS application builders and users; they can take the hypertext-style interface for granted (see Figure 3). We have implemented many of these concepts in a prototype system called Max, which is used by the U. S. Coast Guard [KPBB90].

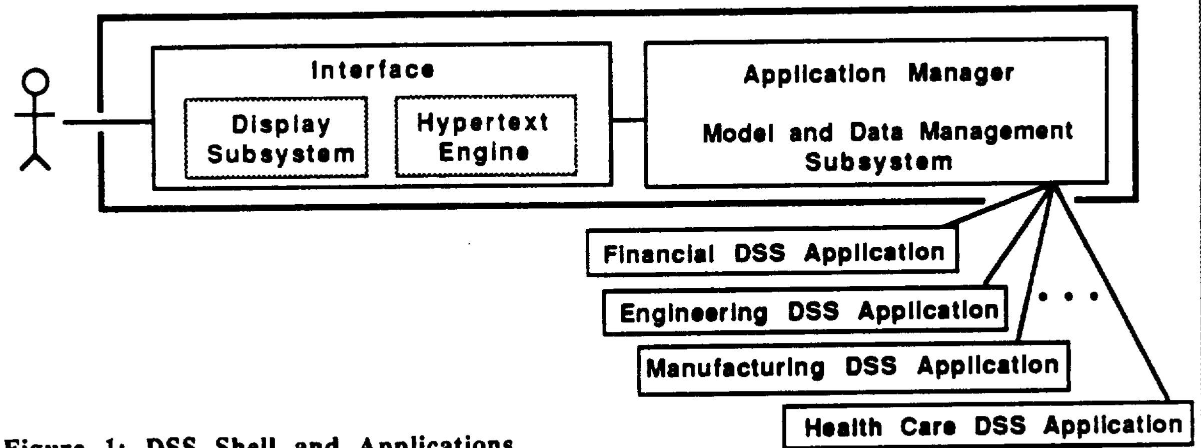
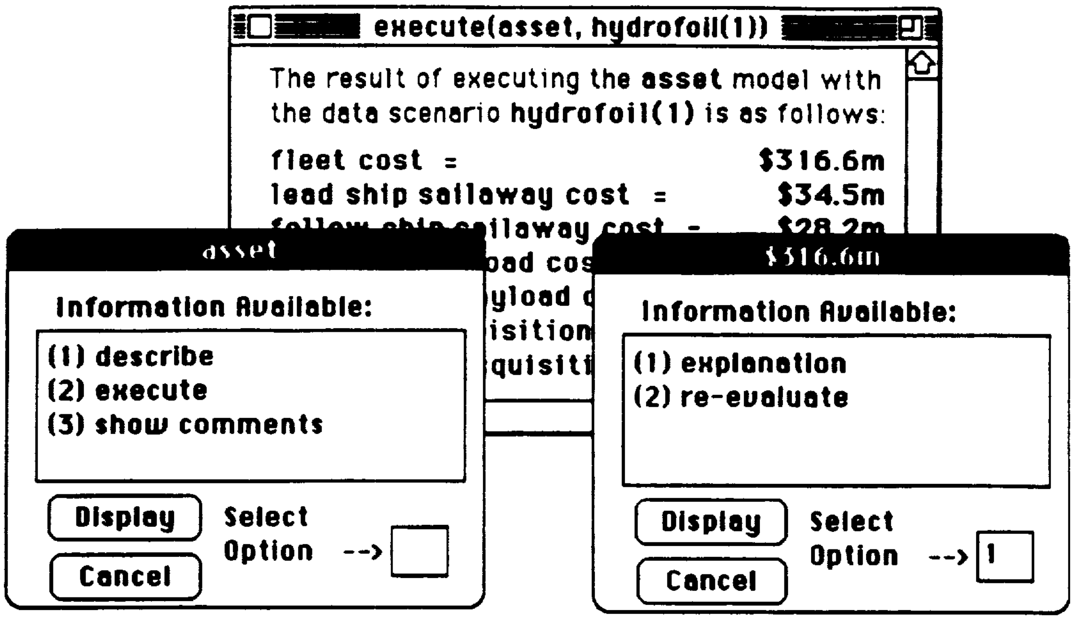

Interface
Display Subsystem
Hypertext Engine
Application Manager
Model and Data Management Subsystem
Financial DSS Application
Engineering DSS Application
Manufacturing DSS Application
Health Care DSS Application
. . .

Figure 1: DSS Shell and Applications

The purpose of this paper is to discuss generalized hypertext and show how automating hypertext can enhance decision making with a DSS. We shall introduce generalized hypertext in §3 and show how it overcomes

[1] Shells provide standard functionality and a common "look" to a range of applications, with the goal of decreasing the effort involved in building and using them. For example, spreadsheet packages such as Lotus 1-2-3 can be considered shells. They provide a standard interface (the tabular worksheet), a set of functions (e.g., statistical, financial, as well as plain arithmetic) and set of menu commands. The applications are the individual spreadsheets built by entering formulas and data values in the interface provided. For a further discussion of shells see [Kim86, KPBB90].

Figure 2a: A hypothetical *interactive document* showing the results of executing the "asset" model from a DSS application. Interactive documents provide the following functionality. All text highlighted in boldface are hypertext *link markers* indicating the presence of hypertext *links* to documents and other related information. In this figure the user has selected two markers, the "asset" model and the execution result "$316.6m." The system then displays all appropriate commands and connections to information related to these markers, e.g., describe the selected model, execute the selected model, explain the selected model execution result. Each of these options represents a hypertext link. The user then may choose one of these links to *traverse*. In this illustration the user chooses an "explanation" link for the marker "$316.6m", i.e., option (1). We continue this scenario in Figure 2b.



execute(asset, hydrofoil(1))

The result of executing the **asset** model with the data scenario **hydrofoil(1)** is as follows:

```
fleet cost =                       $316.6m
lead ship sailaway cost =           $34.5m
follow ship sailaway cost =         $28.2m
```

asset

Information Available:

(1) describe
(2) execute
(3) show comments

Select Option -->

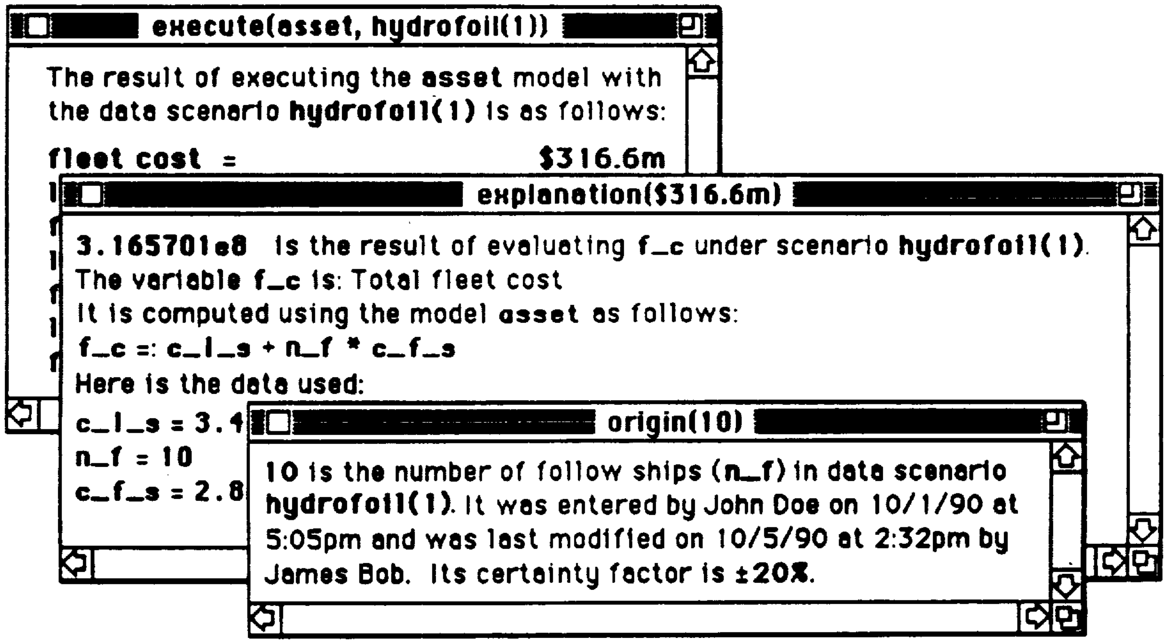
$316.6m

Information Available:

(1) explanation
(2) re-evaluate

Select Option --> 1

Figure 2b: A hypothetical example of *drilling down* for further detail in a DSS. In Figure 2a the user selected the link marker "316.6m" and chose to *traverse* an "explanation" link. The system displayed the resulting explanation shown here in the middle interactive document report. Then (s)he selected the link marker "10" in the second to last line. One of the available options was to show the origin of this value. The user chose this and the system *traversed* its corresponding link. This creates the interactive document report entitled "origin(10)".



execute(asset, hydrofoil(1))

The result of executing the **asset** model with the data scenario **hydrofoil(1)** is as follows:

fleet cost = $316.6m

explanation($316.6m)

3.165701e8 is the result of evaluating **f_c** under scenario **hydrofoil(1)**.
The **variable f_c** is: Total fleet cost
It is computed using the model **asset** as follows:
f_c =: c_l_s + n_f * c_f_s
Here is the data used:
c_l_s = 3.4
n_f = 10
c_f_s = 2.8

origin(10)

10 is the number of follow ships (**n_f**) in data scenario **hydrofoil(1)**. It was entered by John Doe on 10/1/90 at 5:05pm and was last modified on 10/5/90 at 2:32pm by James Bob. Its certainty factor is ±20%.

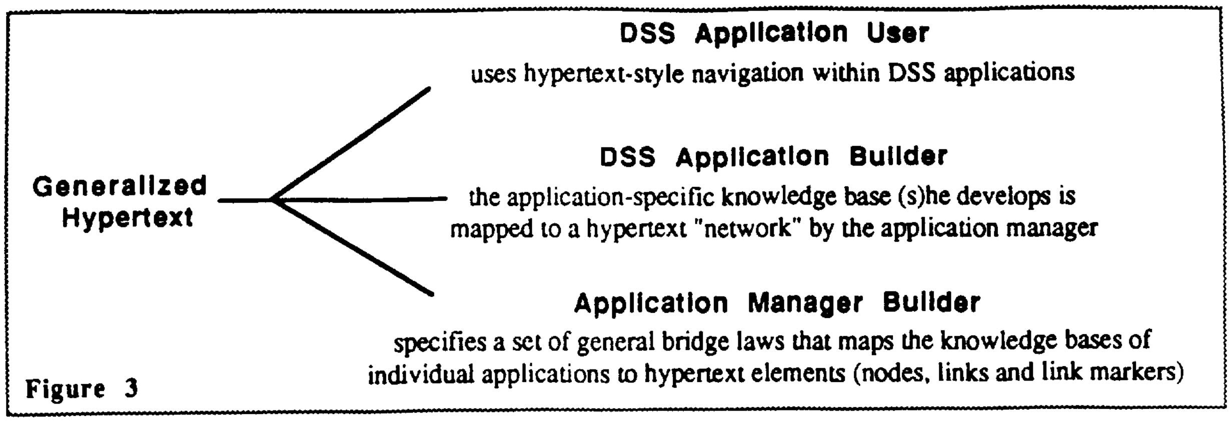

DSS Application User
uses hypertext-style navigation within DSS applications
DSS Application Builder
the application-specific knowledge base (s)he develops is mapped to a hypertext "network" by the application manager
Application Manager Builder
specifies a set of general bridge laws that maps the knowledge bases of individual applications to hypertext elements (nodes, links and link markers)
Generalized Hypertext

Figure 3

limitations inherent in what we call *standard hypertext*, the way hypertext traditionally has been implemented. First, in §2 we set the stage by explaining how a hypertext-oriented DSS can support decision makers.

§2 Hypertext and Decision Support Systems: Supporting the Four Stages of Decision Making

In this section we describe how DSS applications can assist decision makers and how hypertext can augment DSS functionality. To do so we shall use a framework comprising Herb Simon's *intelligence-design-choice* model of decision making [Sim77[2]] and Henry Mintzberg et al.'s additional step, *authorization* [MRT76 p257]. *Intelligence* is the act of gathering information. *Design* involves developing alternative scenarios or problem solutions. *Choice* is selecting the "best" option. *Authorization* is the process of justifying the recommendation or decision made to those affected by it.[3] In the paragraphs that follow we discuss the role for DSS and hypertext in each stage of decision making and justification.

Intelligence

Intelligence is the act of exploring a decision domain, either looking for or responding to a problem or opportunity. A DSS can help by maintaining information—the models[4], data and reports pertinent to understanding the general decision domain and, in particular, the issue-at-hand. For example, for a mortgage domain the DSS would contain information about the economy, interest rates and the housing market. Its models would help a loan officer evaluate a client's risk factor and determine what rate to charge. In an acquisition domain the DSS would contain life cycle cost models (e.g., construction, operation and maintenance equations) as well as exact or statistical data on the items being purchased or constructed. Similarly, scientific and engineering domains would have their own sets of models, data and standard reports.

Not only should the DSS maintain these models and data, but it should help the decision maker (or the analyst working for him) access and understand these. Now consider hypertext, which many believe reduces the

[2] For an interesting analysis see [Sil86 p24].

[3] Of course, instead of being performed sequentially, these stages typically are intertwined during a decision analysis.

[4] There are many types of decision models: algebraic (e.g., sets of formulas found on a spreadsheet), optimization (e.g., sets of equations used in linear and integer programming), simulation, etc. Models are *executed* by applying a *data scenario*—a set of data values—to the model's variables.

cognitive overhead in viewing a complex domain [Con87 p40]5. We can think of hypertext as a method for presenting short information displays embedded with links to further details or other related information. Hypertext-style *navigation* or *browsing* allows the user to explore this "network" selectively, choosing to see what he wants to and by-pass what he already understands or feels is less important at the moment. What would we want to link in a DSS? Related models, data and reports should be connected. Models should be linked to their submodels and variables. These variables should be linked to all possible data values registered in an application data base. All of the above should be linked to any execution results derived from them, especially when these results appear in reports. The user should be able to retrieve definitions, explanations and any background information (including any comments or annotations) inferable about items of interest. We illustrate access to such information in Figures 2a and 2b. The DSS interface can use hypertext concepts when structuring and presenting information, so that relevant data (e.g., based on these types of links) is readily accessible to the analyst or decision maker in a comprehensible and useful format [HM89 p47]. We have dubbed this hypertext-style accessibility the *WYWWYWI ("what you want, when you want it") principle* [BBK88].

Design

The analyst will design potential scenarios to address the issue-at-hand. This may involve creating additional models or developing alternate data scenarios (sets of data values for the model equation variables), perhaps through spreadsheet-style "what-if" analysis. For example, an analyst in the acquisition domain may explore the effects of possible changes in petroleum prices or the inflation rate on operation and maintenance costs over a thirty-year period.

To support the hypertext-style navigation described earlier, the DSS must both register the new models and data scenarios, and create the hypertext network—the nodes, links and link markers accessed during exploration and execution.

Choice

According to the *principle of decision support* [Kim88], the analyst will use the DSS to investigate different alternatives until he comes up with an optimum choice or quits (e.g., for timing reasons or perhaps because there is no optimum), in which case he will choose the best alternative he has found (assuming it is "satisfactory"). Normally the DSS's operations include executing models and performing *what if* analysis.

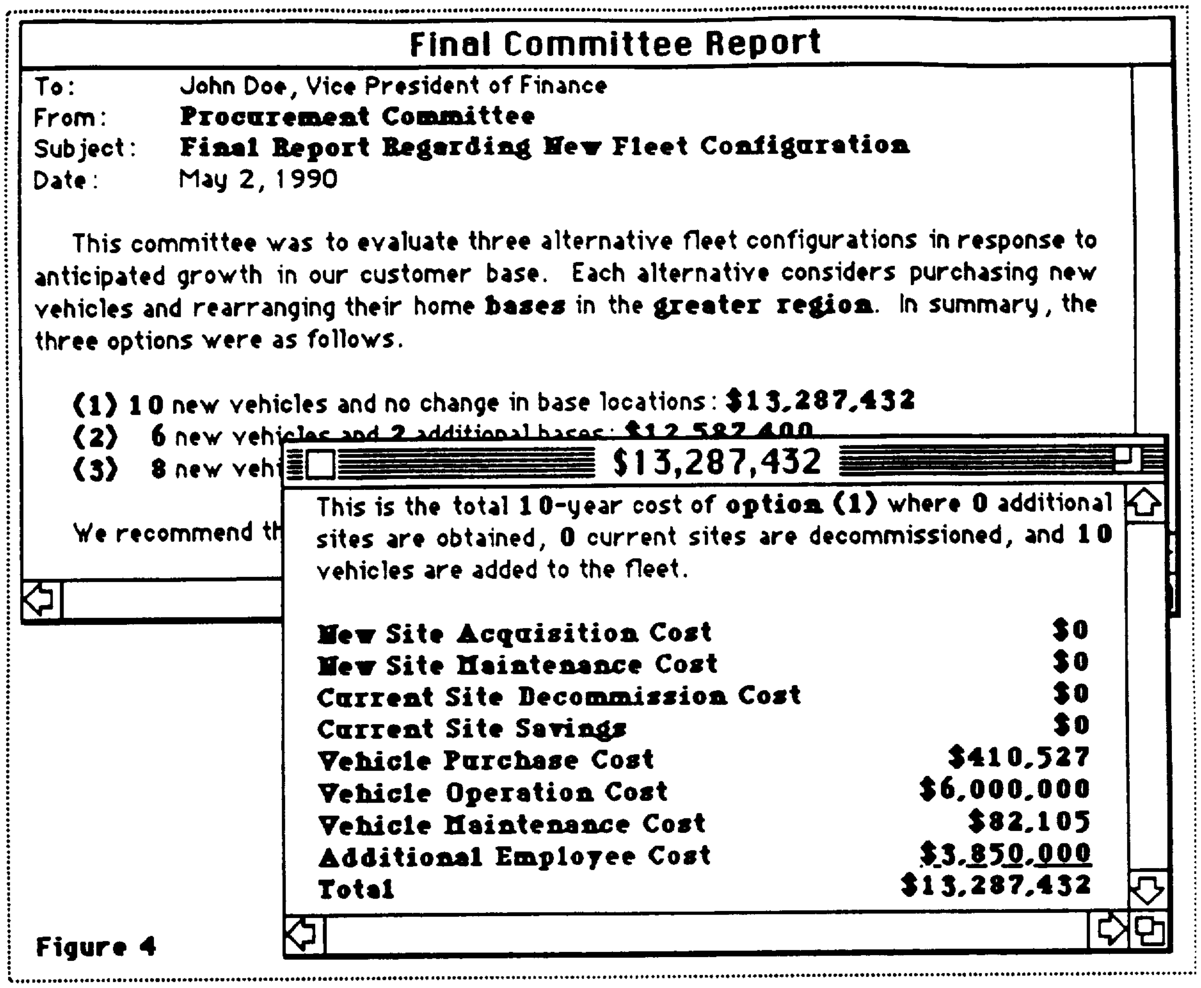
How does hypertext help? Again, hypertext provides easy access to explanations and other information. Furthermore, hypertext-style markers can act as *embedded menus* [KS86], giving the analyst "context-sensitive" access to normal DSS operations, such as model execution. We see this in Figure 2a.

Authorization

Analysts often use a DSS to develop a recommendation. The tangible result of a completed DSS analysis is a report documenting it. Decision makers can use this report to reach a final decision and then convince others of its logic.

For example, Figure 4 shows the final report of a procurement committee. Hypertext's role would be to facilitate the connection between this report and all supporting information linked to the highlighted markers. This should help the decision maker in several ways. The DSS (ideally) would contain the answer to any questions he would have, especially if the analyst has added comments. The decision maker can investigate any piece of information, exploring all the way "down" to its origin (recall Figure 2b), thereby increasing his overall understanding of, and confidence in, his decision. The most important point is that the decision maker is now self-sufficient. He can do all this by browsing on the computer without active assistance from the report's author.

5 At its most basic level, *hypertext* is simply the concept of *linking* any two pieces of information, for example, a number with an explanation of how it was generated. For surveys of hypertext history and systems see [Con87, SK89 and Nie90]. [IRIS89 and Nie89] provide further bibliographic references. See [Min89] for a general discussion of hypertext and decision support systems.



Final Committee Report

To:	John Doe, Vice President of Finance
From:	**Procurement Committee**
Subject:	**Final Report Regarding New Fleet Configuration**
Date:	May 2, 1990

This committee was to evaluate three alternative fleet configurations in response to anticipated growth in our customer base. Each alternative considers purchasing new vehicles and rearranging their home **bases** in the **greater region**. In summary, the three options were as follows.

(1) 10 new vehicles and no change in base locations: **$13,287,432**
(2) 6 new veh
(3) 8 new veh

We recommend th

$13,287,432

This is the total **10**-year cost of **option (1)** where **0** additional sites are obtained, **0** current sites are decommissioned, and **10** vehicles are added to the fleet.

New Site Acquisition Cost	**$0**
New Site Maintenance Cost	**$0**
Current Site Decommission Cost	**$0**
Current Site Savings	**$0**
Vehicle Purchase Cost	**$410,527**
Vehicle Operation Cost	**$6,000,000**
Vehicle Maintenance Cost	**$82,105**
Additional Employee Cost	**$3,850,000**
Total	**$13,287,432**

Figure 4

In summary, these stages implement the *argumentation theory of DSS*, according to which "the main purpose of a DSS is to support the construction, evaluation and comparison of arguments for courses of action" [Kim88, KPBB90]. These "arguments" then are used to justify the course of action taken. The hypertext concepts employed by the interface facilitate such support, giving the user easy access to information and operations within the DSS application without overwhelming him with details.

Can this process be made more efficient? Consider the second interactive document in Figure 4, which presumably had to be constructed manually. Suppose the system could generate this report, along with all other lower-level definitions and explanations. This would expedite the analyst's task dramatically! It is this DSS functionality that we shall describe in the next section.

§3 Generalized Hypertext: Overcoming the Limitations of Standard Hypertext

The DSS environment described in §2 could be supported by most hypertext systems today, but with a great restriction in functionality. To illustrate this, in Technical Appendix 1 we give the internal code for part of the knowledge base defining the interactive documents shown in Figure 2b. (A systems programmer would arrange such code based on the nodes, links and link markers created by the application builder, the application builder and user never see this view of hypertext[6].) What is wrong with this internal representation? The problem is that the knowledge base consists of static, explicit, pre-defined entries. The builder must anticipate the user and provide all the nodes, links and markers he believes the user will want to access in the future. He therefore must do all DSS analyses and prepare their explanations in advance so he can specify the hypertext link markers embedded in the appropriate reports. Each application builder must establish the hypertext nodes, links and markers related to his application.

Considering all the possible executions, execution results and explanations hinted at in Figures 2 and 4, we see that in a typical application, an immense amount of analysis is possible, too much to do in advance. Furthermore, it would be unreasonable to ask the application builder to declare all the hypertext elements in his application. The effort in doing a thorough job would frighten away the most willing application builder. Besides, a DSS generally is a dynamic beast. Its knowledge base contains definitions, models, data and documents, but not execution results or their explanations. The latter are produced dynamically upon user request, and so must any hypertext links mapped to them. Our only feasible option is to automate the generation of the hypertext network.

How do we automate hypertext in a knowledge-based DSS? First, it is important to note that hypertext functionality (e.g., creating, exploiting and managing links and linked information items) must be implemented in a way that is accessible to all current and future shell applications. Therefore we treat hypertext functionality as *system-level* (as opposed to application-level) support. Recall Figure 1, where the hypertext engine was embedded in the shell's interface component. Like database systems and user interface management systems, the hypertext engine is an application-independent, system-level tool for providing hypertext functionality for specific applications.

When automating hypertext, the hypertext engine cannot take an arbitrary application's knowledge base and magically infer what elements correspond to hypertext nodes and links. Instead, the shell's *application manager* subsystem must provide some translation routines that the engine can use to make its inferences[7]. We call these translation routines *bridge laws* [Nag61, Hau78, Kim79] because they serve as a "bridge" or connection between elements defined in the language of the application's knowledge base and those in the shell's hypertext engine. As we shall see, bridge laws exploit *logical quantification*, enabling individual laws to map entire classes of application objects (e.g., models or variables) to hypertext entities; the same bridge law will map any number of instances in the application knowledge base. For example, recall the three possible links emanating from the "asset" model in Figure 2a—"describe", "execute" and "show comments." How could this set be generated in a general manner? Given any link marker that represents the name of a model, in Technical Appendix 2 we describe a set of eight *predicates* that generates this trio of links. These predicates will work for every model ever to be declared by an application builder, now and in the future. In [Bic90] we describe similarly compact sets of bridge laws for variables, data scenarios, etc. Additional bridge laws may be developed by the systems programmer responsible for the shell's application manager subsystem. Application builders and users need have no knowledge of bridge laws. To them, hypertext functionality occurs automatically!

In developing generalized hypertext we had to solve three of the outstanding problems in hypertext research as identified by Halasz in [Hal88]. These are the creation and manipulation of virtual hypertext entities, computation over the knowledge base during link traversal, and the tailorability of the hypertext network. We describe these in

[6] The shell's interface is much "friendlier" than this code. For example, in a typical hypertext system, an application builder would construct a link by selecting a portion of text with his mouse, choosing the "create link" menu, and selecting a second portion of text in a different window. The system would then generate the internal code for the link, similar to that shown in Technical Appendix 1.

[7] Analogous to a spreadsheet package that provides all necessary functions and menu commands to an individual spreadsheet, a DSS shell *application manager* subsystem would provide both the standard report templates and the tools for manipulating models and data (e.g., execution and explanation), as long as application builders declare their models and data in a standard format, much as one is forced to in a spreadsheet. Based on this standard format, the application manager in a hypertext-oriented DSS shell provides general bridge laws for *mapping* application models, data, template reports, functions and commands to hypertext nodes, links and link markers.

the following paragraphs.[8]

Bridge laws are examples of *virtual entities*[9]. By *virtual*, we mean not fully identified or *resolved*. The parameters of a virtual entity are not known or *instantiated* in their entirety and therefore can assume any compatible value. For example, the "316.6m" link marker in Figure 2a is virtual when the user selects it because two possible links can be associated with it, i.e., its "link" parameter can be filled by one of the two compatible links. One can think of virtual entities being represented by templates prescribing both the entity's internal format and how the hypertext engine should *infer* compatible values to fill its parameter *slots* during link traversal. (We describe the actual technique used in Technical Appendix 2.) When the hypertext engine has filled all the slots then we say that the virtual entity is *resolved* and that we have either found or created (generated) a specific *instance* of that entity. When the user selects the "316.6m" link marker, the hypertext engine uses the bridge laws for link markers and links to *infer* which links could exist for the markers selected and tentatively fills in the templates with values drawn from models and data in the application knowledge base. The user then chooses one of these tentative links to traverse, as shown in Figure 2a. As Appendix 2 indicates, several pieces of information now must be inferred to resolve the link chosen. One is the type of link (e.g., "explanation" or "re-evaluation"). Another is the link's destination. The hypertext engine uses additional bridge laws to infer these from the contents of the application knowledge base. Once these are known, we consider the link fully resolved because the template is complete.

The next step is *computation*, i.e., actually generating the contents of the destination node specified by the link. A DSS operation (e.g., a model execution) normally must be performed to create the destination node (e.g., a decision report describing the results of the operation). Once this is done, the shell prepares the outcome for display in an interactive document. Part of this process involves the hypertext engine, which uses bridge laws to determine which portions of the interactive document should be highlighted as link markers. These markers will be virtual, for as we saw, they will not be associated yet with actual links. When the user selects one of these markers, the resolution cycle repeats. In summary, we say that link traversal in generalized hypertext follows the *select-infer-traverse-infer* pattern described here.

We want an application to be able to *tailor* its processing and resultant reports to a given user's skill level and the specific task he is using the DSS application to accomplish. The shell can do this by maintaining multiple *contexts* or modes, each associated with a different *view* of application knowledge base components and with a different set of report formats. The hypertext engine checks contexts as part of every operation it performs. For specific details we refer the reader to [Bie90]. Contexts are also instrumental in providing the task environments discussed in the next section.

§4 Discussion and Future Developments

Generalized hypertext is a logic-based technique for automating hypertext within a knowledge-based decision support environment. Generalized hypertext adds value by providing a hypertext-style interface to a decision support system (DSS) application without an *author* having to create any nodes or links. (Of course, users can add their own comments and other annotations, just as in regular hypertext systems.) Generalized hypertext can be applied to any domain that is well enough understood and expressible for the systems programmer building an application manager subsystem to map the components of the application knowledge base to generalized hypertext entities and attributes.[10]

The direct access to information provided by a generalized hypertext DSS makes its use easy for someone who is new to a decision domain. He can explore the domain at his own pace, read the comments and definitions, and

[8] We discuss this entire process more fully in [BK90] and [Bie90].

[9] By hypertext entities we mean hypertext nodes, links and link markers. Further hypertext entities are presented in [Bie90].

[10] For an interesting discussion of domains where this is not possible see [RT88].

experiment with application models and data. Another benefit of hypertext is the degree of user control. More advanced users can by-pass information easily with which they are familiar. To assist the user further, one direction we shall explore in our research is *task environments*, local environments organized around the procedure the analyst should follow when he is performing an individual task [Bie90]. They are tailored specifically to the task (and perhaps to the user's skills or preferences). Only relevant information (data, documents, reports, models) and commands will be available. Also, task environments can serve as an enhanced form of documentation, building upon the notion of, e.g., *guided tours* in NoteCards [MI89] or Zellweger's *scripted documents* [Zel89]. An advanced implementation of task environments could incorporate "active" agents that guide the user through the process, reminding him of steps left out and making suggestions.

Generalized hypertext enhances the functionality and ease of use of DSS applications within a DSS shell. This is evident from our prototype system, Max, described in [KPBB90]. It is our belief that generalized hypertext will advance both the state of hypertext research in knowledge-based environments and the art of information presentation in decision support systems.

Acknowledgement

I want to thank Steven O. Kimbrough of the Wharton School, University of Pennsylvania for his help in developing generalized hypertext, which was invaluable.

This work was motivated by and funded in part under contract DTCG39-86-C-80348, between the U. S. Coast Guard and the University of Pennsylvania, with Steven O. Kimbrough as principal investigator.

Technical Appendix 1: Inside a Standard Hypertext Knowledge Base

This appendix presents a systems programmer's view of the nodes, link markers and links in the hypertext knowledge base representing Figure 2b's reports. Each report, link marker and connection is represented by a "node", "marker" or "link" *predicate*. We precede each set of predicates by a legend explaining its three arguments.

Legend: node(unique report id, report title, list of the text and link markers comprising its contents)

```
node(1, title("execute(asset, hydrofoil(1))"),
    content(["The result of executing the", marker(1), "model with the data scenario", marker(2), "is as
        follows:<cr>", marker(3), "=", marker(4), "<cr>", marker(5), "=", marker(6), ...]))
node(2, title("explanation($316.6m)"),
    content([marker(17), "is the result of evaluating", ...]))
node(3, title("origin(10)"), content(["This value for the number of helicopters..."]))      ... etc.
```

Legend: marker(unique marker id, the link associated with the marker, the text representing the marker)

```
    marker(1, link(1), content("asset"))
    marker(2, link(2), content("hydrofoil(1)"))
    marker(3, link(3), content("fleet cost"))
    marker(4, link(4), content("$316.6m"))                                        ... etc.
```

Legend: link(unique link id, the report where the link originates, the link's destination report)

```
    link(1, node(1), node(4))
    link(2, node(1), node(5))
    link(3, node(1), node(2))
    link(4, node(1), node(3))                                                     ... etc.
```

Technical Appendix 2: Generalized Hypertext

This appendix presents simplified examples of bridge laws, which are specified using quantified predicates in first order logic. The set of predicates here, which includes both bridge laws and some of the application knowledge base declarations they find, generates the hypertext links emanating from all keywords in interactive documents that are names of application models. For clarity, we precede bridge law names with the code "ght" and application knowledge base declarations with "appl".

Legend: The format of the generalized hypertext link predicate

We declare generalized hypertext links with the following predicate. As we shall see next, bridge laws for links have this same format.

```
ght_link(id, originating node, originating marker, destination node, link operation type)
```

Bridge Law Declaring Links from Keyword Markers to Destination Nodes

This bridge law establishes links to all keywords *i* that are names of elements in the DSS application. It calls the application predicate *appl_type/2* to identify the elements, bridge law *ght_operation/2* to determine the link operation type *o*, and bridge law *ght_identifier/3* to generate the name of the destination node *x* given the link operation identified. (Arguments with underscores accept any values passed.)

```
(∀i, o, t, x) (ght_link(i, _, keyword(i), x, o) <--
    (appl_type(i, type(t)) &
     ght_operation(type(t), i, o) &
     ght_identifier(i, o, x)))
```

Identifying the Type of Application Element Selected

It is reasonable to assume that applications can identify each of their elements. This predicate determines that *i* is an application model if the application has an *appl_model/3* predicate declared for it.

```
(∀i) (appl_type(i, type(model)) <--
    (appl_model(i, _, _)))
```

Bridge Laws Inferring Link Operation Types for Models and Elements with Comments

The first two bridge laws find "describe" and "execute" links for application models.

```
ght_operation(type(model), _, operation_type(describe))

ght_operation(type(model), _, operation_type(execute))
```

The next determines whether any comments are registered for the element passed.

```
(∀i) (ght_operation(_, i, operation_type("show comment")) <--
    (comment(i, _)))
```

Bridge Laws Inferring Destination Node Identifiers

Given the link operation type, these laws infer the identifier of the destination node that the link operation will generate. The destination identifier is passed in the third argument.

```
(∀i) (ght_identifier(i, operation_type(describe), describe(i)) <--
    (appl_type(i, type(model))))

(∀i, d) (ght_identifier(i, operation_type(execute), execute(i, d)) <--
    (appl_type(i, type(model)) &
     appl_data_scenario(i, d)))

(∀i) (ght_identifier(i, operation_type("show comment"), comment(i)))
```

References

[BBK88] Bhargava, Hemant K., Michael P. Bieber and Steven O. Kimbrough, "Oona, Max, and the WYWWYWI Principle: Generalized Hypertext and Model Management in a Symbolic Programming Environment," in *Proceedings of the Ninth International Conference on Information Systems*, Minneapolis, 1988, pages 179-192.

[Bie90] Bieber, Michael, Generalized Hypertext in a Knowledge-based DSS Shell Environment, Ph. D. Dissertation, Decision Sciences Department, University of Pennsylvania, 1990.

[BK90] Bieber, Michael and Steven O. Kimbrough, "Towards a Logic Model for Generalized Hypertext," *Proceedings of the 23nd Hawaii International Conference on System Sciences*, Hawaii, 1990.

[Con87] Conklin, Jeff, "Hypertext: a Survey and Introduction," *IEEE Computer*, volume 20 number 9, 1987, pages 17-41.

[Hal88] Halasz, Frank G., "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems", *Communications of the ACM*, volume 31 number 7, July 1988, pages 836-855.

[HM89] Herrstrom, David S. and David G. Massey, "Hypertext in Context," in The Society of Text: Hypertext, Hypermedia, and the Social Construction of Information, Edward Barrett (ed), MIT Press, Cambridge, MA, 1989, pages 45-58.

[IRIS89] "Hypermedia Bibliography," Institute for Research in Information and Scholarship (IRIS), Box 1946, Brown University, Providence, RI 1989.

[Kim79] Kimbrough, Steven O., "On the Reduction of Genetics to Molecular Biology," *Philosophy of Science*, volume 46 number 3, September 1979, pages 389-406

[Kim86] Kimbrough, Steven O., "On Shells for Decision Support Systems," Wharton School, Department of Decision Sciences, working paper #86-07-04, July 1986.

[Kim88] Kimbrough, Steven O., "The Argumentation Theory for Decision Support Systems," draft of Chapter 2 of Decision Support Systems: Technologies for Reasoning and Argumentation, Wharton School, Department of Decision Sciences, 1988.

[KPBB90] Kimbrough, Steven O., Clark W. Pritchett, Michael P. Bieber and Hemant K. Bhargava, "An Overview of the Coast Guard's KSS Project: DSS Concepts and Technology," *Transactions of DSS-90*, TIMS, Boston, MA, May 21-23, 1990, pages 63-77; accepted for publication in *Interfaces*.

[MI89] Marshall, Catherine C. and Peggy M. Irish, "Guided Tours and On-Line Presentations: How Authors Make Existing Hypertext Intelligible for Readers," in *Proceedings of the 1989 Hypertext Conference*, Pittsburgh, PA, November 1989, pages 15-42.

[Min89] Minch, Robert, "Application and Research Areas for Hypertext in Decision Support Systems," *Journal of Management Information Systems*, volume 6 number 3, Winter 1989-90, pages 119-138.

[MRT76] Mintzberg, Henry, Duru Raisinghani and André Theoret, "The Structure of Unstructured Decision Processes," *Administrative Science Quarterly*, volume 21, June 1976, pages 246-275.

[Nag61] Nagel, Ernest, The Structure of Science: Problems in the Logic of Scientific Explanation, Harcourt, Brace &World, Inc., New York, NY, 1961

[Nie89] Nielsen, Jakob, "Hypertext Bibliography," *Hypermedia*, volume 1 number 1, Spring 1989, pages 74-91.

[Nie90] Nielsen, Jakob, Hypertext and Hypermedia, Academic Press, 1990.

[RT88] Raymond, Darrell R. and Frank W. Tompa, "Hypertext and the Oxford English Dictionary", *Communications of the ACM*, volume 31 number 7, July 1988, pages 871-879.

[Sim77] Simon, Herbert, The New Science of Management Decision, Harper and Row, NY, NY, 1960, 1977 (revised edition).

[Sil86] Silver, Mark S., Differential Analysis for Computer-Based Decision Support, Ph. D. Thesis, Wharton School, 1986.

[SK89] Shneiderman, Ben, and Greg Kearsley, Hypertext Hands-On! An Introduction to a New Way of Organizing and Accessing Information, Addison-Wesley, Reading, MA, 1989.

[Zel89] Zellweger, Polle, "Scripted Documents: A Hypermedia Path Mechanism," in *Proceedings of the 1989 Hypertext Conference*, Pittsburgh, PA, November 1989, pages 1-14.

Session 7

Future of Hypermedia

Chair: David Palumbo

Hypermedia as Medium

Chris Dede

Hypermedia = Hypercommunication

Mark R. Laff

Moving from Knowledge Presentation to Knowledge Representation

David Palumbo