The Coast Guard's KSS Project

STEVEN O. KIMBROUGH — *Department of Decision Sciences/6366, University of Pennsylvania, Philadelphia, Pennsylvania 19104-6366*

CLARK W. PRITCHETT — *United States Coast Guard Research and Development Center, Avery Point, Groton, Connecticut 06340*

MICHAEL P. BIEBER — *Computer Science Department, Boston College, Chestnut Hill, Massachusetts 02167-3808*

HEMANT K. BHARGAVA — *Naval Postgraduate School, Code 54BH, Monterey, California 93943-5000*

Since June 1986, the University of Pennsylvania has been under contract with the Coast Guard to provide support for the Coast Guard's KSS (knowledge-based decision support systems) project. A principal output of the KSS project has been Max, a DSS shell or environment, originally designed to support modeling tasks during decision making for ship acquisition. Max is innovative in a number of ways: (1) it is a document-oriented DSS, (2) Max documents are generalized hypertext documents for which the buttons and linkages to associated information are set up dynamically at run time by the system, and (3) Max contains a model management system, called TEFA, which can represent and evaluate a broad range of models. In addition, TEFA is able to represent a rich body of information about its models and data, and this meta-information is available through the generalized hypertext facilities of system. Max has been delivered to the Coast Guard and is in use. Several important Coast Guard and Navy models have been implemented in Max at very low cost.

Since June 1986, the University of Pennsylvania has been under contract with the Coast Guard's R and D Center at Groton, Connecticut to provide support for the Coast Guard's KSS (knowledge-based decision support systems) project. This

0091-2102/90/2006/0005$01.25

DECISION ANALYSIS—SYSTEMS
GOVERNMENT SERVICES—COAST GUARD

INTERFACES **20:** 6 November–December 1990 (pp. 5-16)

support has included systems analysis, management science modeling, concept development, and software design and implementation. Work on the KSS project has proceeded through the close cooperation of the University of Pennsylvania, the R and D Center, and the Coast Guard's Office of Acquisition in Washington, DC. The University of Pennsylvania and the R and D Center have done the actual work on the project. The Office of Acquisition, which is charged with acquiring major systems in the Coast Guard, has been a willing customer and provided information and feedback on various aspects of the project. A principal output of the KSS project has been Max, a DSS shell or environment [Kimbrough 1986], originally designed to support modeling tasks during decision making for ship acquisition [Bhargava, Bieber, and Kimbrough 1988; Bhargava and Kimbrough 1990; Bieber and Kimbrough 1990; Kimbrough et al. 1986].

A DSS is an interactive software tool for working with models and data. A KSS (the Coast Guard's term) is a DSS plus intelligence and a richer concept. In developing new software, our practice has been to begin by developing an application theory, or hypothesis, regarding why the software is needed and what it should be used for. Following this, we develop a concept for what the software should be about, how in the abstract it should work. We then design representation schemes for the system's objects (the various entities, for example, models and data, that the system must reason about and operate upon), the operations supported in the system, and the control mechanisms for the system. Our debt to the Sprague and Carlson ROMC (representations, operations, memory aids, and control) framework should be clear [Sprague and Carlson 1982]. To our way of thinking, memory aids, the M of the Sprague and Carlson ROMC framework, are elements of the system concept whose specific implementation aspects are covered under representations, operations, and control.

A KSS is a DSS plus intelligence and a richer concept.

We subscribe to the argumentation theory of DSS, according to which a main purpose of a decision support system is to support the construction, evaluation, and comparison of arguments for courses of action [Kimbrough 1987]. (DSSs are also useful for reasons of convenience—a DSS can make working with data and models much easier—and for reasons of insight—a DSS can deliver information that is critical for gaining insight into the decision problem at hand. Reasons of argumentation, however, were a main motivating impetus for our work.) We use the term argument in its logical sense, to indicate a collection of statements, including a conclusion and zero or more premises. Put differently, DSSs are for constructing and examining reasons for doing things. The Coast Guard's Office of Acquisition is interested in acquiring ships and aircraft. This costs money and must be justified with reasons [Kimbrough 1982]. Their underlying purpose for a DSS is to support the development of the best possible reasons for the best possible course of action in acquiring ships and aircraft.

To date with Max we have aimed at a modest approximation of this goal largely because DSSs in general and DSS features in particular have traditionally been categorized as either data-oriented or model-oriented [Alter 1977]. This may be correct as a description of actual practice, but we believe that a third orientation is needed in DSS. Assuming the argumentation theory for decision support, our application theory (for DSS in the Coast Guard's Office of Acquisition) has been that the KSS software should be a tool that is useful in constructing decision reports, reports that recommend courses of action and document the reasons for them. While they incorporate data and the results of model runs, these reports essentially make recommendations and provide support or reasons for them. Our fundamental hypothesis in designing Max has been that what is needed is a document-oriented DSS that could also be an effective tool for working with models and data. We think that a document-oriented DSS is a reasonable step towards a more thoroughgoing argumentation DSS. (We credit David Ness with persuading one of us [Kimbrough] to appreciate the importance of document handling in DSS.)

Major Systems Acquisitions at the Coast Guard

The Coast Guard uses ships (cutters) and aircraft (airplanes, helicopters, and lighter-than-air vehicles) in conducting its missions. Eventually, a particular class of asset (cutter or aircraft) wears out through use or becomes technologically obsolete and economically unsupportable or is no longer matched to the missions it is called upon to perform. Some of these problems can be delayed or overcome by upgrading systems or overhauling or modernizing, but eventually an entire class of resources must be replaced. The Coast Guard, for example, is currently examining fleets of high-endurance cutters and medium-range helicopters.

The federal government has established and defined a process that must be followed in acquiring new systems, including classes of cutters and aircraft. The process was delineated by the Office of Management and Budget in its now famous A-109 circular (1977). The acquisition process for major systems, such as a fleet of ships, can take 10 or more years from inception to the point where a contract is actually let to build the first vessel. The responsible office must establish mission needs and operational requirements for the new system. All reasonable alternatives must be considered. Each alternative might be played through a scenario or series of models to forecast its performance on items of interest, such as mission performance or logistics. Analyses are periodically required for various aspects of the problem throughout the life of a major system acquisition project. For large contracts, prototypes may be built and tested before the final contract is let. Ultimately, a trade-off between cost and performance must be made and the alternatives must be ranked. At various stages in the process, these evaluations and analyses must be documented to satisfy the executive department that wants the resources (in the case of the Coast Guard, the Department of Transportation) and the Congress, which is being asked to appropriate the money.

During the next 10 years, the US Coast Guard will invest more than a billion dol-

lars in acquiring replacements for its present, aging fleet. The cost of staffing, operating, and maintaining these assets will be several times the acquisition cost. Because of the importance and complexity of the problem of acquiring such major systems, the Coast Guard initiated the KSS project. It was aimed, in part, at systematically developing software to support decisions on acquiring major systems. The goal is to produce a series of specific KSSs to support particular acquisition projects that would be based on generic, reusable software.

During the next 10 years, the US Coast Guard will invest more than a billion dollars in acquiring replacements for its present, aging fleet.

Important elements of the context for the KSS project follow:

(1) The Coast Guard needs to replace its aging fleet of ships and aircraft.

(2) Replacement of the Coast Guard fleet will occur in a policy and political environment in which funds are limited.

(3) Vessel acquisitions are complex and difficult and require years of effort and planning.

(4) The complexity and difficulty are increased by new technology, changing mission requirements, and the fact that finding the best possible mix of ships and aircraft becomes more critical with limited resources [Bhargava, Kimbrough, and Pritchett 1990].

(5) The Coast Guard must conform to the A-109 process.

(6) Coast Guard analysis personnel and decision makers are often not trained in management science and information systems and often serve as acquisition analysts for only one or two years.

(7) Historically, the analysis of alternatives has been weak and has relied very little on management science techniques and decision support systems.

(8) Although the existing Coast Guard personnel are overextended, they must respond to difficult questions from the Department of Transportation and the Congress quickly and accurately.

(9) The costs of mistakes are high: a loss of credibility with the Department of Transportation and Congress and a loss of dollars paid to contractors for Coast Guard mistakes and to attorneys for litigation support.

Design Goals

The R and D Center (through a group led by Pritchett) and the project team at the University of Pennsylvania (Bhargava, Bieber, and Kimbrough) identified the following design goals for the KSS software early in the project:

(1) To design and build reusable KSS shells [Kimbrough 1986];

(2) To integrate models and data from a variety of sources;

(3) To produce KSS software that is useful and easy for novices to work with;

(4) To employ commercially available software tools whenever possible;

(5) To use highly modular, flexible, and maintainable code that can be located on a variety of machines;

(6) To represent large amounts of information about models and data, and to provide easy access to this meta-information [Bhargava and Kimbrough 1990]; and

(7) To identify the main classes of users and design the system for their needs.

Why have these high-level design goals? The answer is pretty much self-evident, but we paid special attention to several points (5, 6, and 7). The key idea behind how we built intelligence into the KSS, that is, into Max was to declare information about models, data, and other system entities (6) as well as to represent these entities directly. Also we identified three major classes of users (7): analysts, browsers, and builders. Analysts are in the business of developing (major systems acquisition) options and comparing them. Analysts make recommendations to decision makers, and in doing so they create reports after exercising models and examining data. Browsers are more casual users of the KSS. They need an easy way to explore documents and information, including data and models. Builders need to put together specific models, data, and other information directly and rapidly. The builders (whom we envisioned to be R and D center management scientists) configure the KSS shell environment to create specific DSSs for use by analysts and browsers.

We will discuss building highly modular, flexible code (5) later. We assumed a build-and-revise (iterative) strategy—rather than a specify-and-implement strategy—for the project. (The present version of Max is a much-revised version of the second rewrite.) This universally recognized strategy produces workable, usable early releases of the software, which can then be modified in response to informed customers' directions. Happily, we are in that position. We delivered an early, workable version of the system which is in use and is undergoing frequent revisions.

To justify the need for a DSS to Coast Guard management, and specifically for the DSSs to be produced under the KSS project, we made the following points:

(1) A DSS delivers detailed information to decision makers effectively.

(2) A DSS produces reliable answers to queries quickly. (This is particularly important for maintaining credibility with executive agencies and with Congress.)

(3) A DSS can reduce the costs of producing information, analyses, and reports for decision makers.

(4) A DSS facilitates more effective use of existing models and data. (This was a particularly telling point with the Coast Guard managers, because they had invested heavily in modeling.)

(5) A DSS can lead to better decisions and recommendations (valuable for their own sake and for maintaining credibility with executive agencies and Congress).

(6) A DSS can facilitate more effective presentation and justification of decisions and recommendations.

(7) A DSS can, by collecting models and data in a single system, help with the institutional memory problem.

Results

We developed two KSS concepts: level 1 and level 2. Level 2 builds upon level 1. We developed both concepts in response to a Coast Guard captain's request for a system that could "tell me where the numbers come from" (B. C. Miller, then of the Office of Acquisition). The captain understood that acquisition recommendations are based on assumptions, some of which are solid, some of which are not. He was asking for a system that could tell him eas-

ily and quickly the source and quality of a given assumption, as well as the assumptions underlying a given figure. His request is certainly consistent with the argumentation theory of DSS: he wanted DSS support to help him examine the premises of arguments or the assumptions behind the reasons and reasoning presented in DSS reports.

We set out to provide such a system and to have the KSS shell software—rather than the system builder—determine and manage the links between the numbers and the information about them. Moreover, we sought to do so in a very generalized fashion, applicable very broadly.

Level 1 KSS Concept: Document-Oriented DSS

The delivered and working version of Max (which is still undergoing modifications) is an instance of the level 1 KSS concept; it is a document-oriented DSS. The KSS shell software provides two main classes of features: interactive (hypertext-style) documents and model management. (Its data management capabilities are currently quite basic, but we are enhancing them.) Our DSS argumentation theory, as applied to Max, has it that a DSS should support an analyst in developing a report recommending a course of action and giving the reasons for it. The KSS shell is oriented towards constructing such reports. This is reflected in the high-level design of the shell, which contains two main modules, Maxi and TEFA.

The TEFA module manages a broad class of mathematical models. (In the delivered version of Max, TEFA handles hierarchical, conditional equational models. It supports models that consist of many equations, but not systems of simultaneous equations. Basically, TEFA's present model representation and execution capabilities substantially exceed those of spreadsheet programs. We have developed a successful prototype for representing mathematical programming models. TEFA can translate such models from its internal format to that required by a commercial solver.) Further, TEFA can express essentially any information about a model or a variable for a model—for example, its source, its reliability, and its dimensional characteristics. Moreover, TEFA can automatically use such information to provide such features for the KSS as validity checking and reporting on models. The documentation for models in TEFA can be made remarkably clear, as is the actual declaration of the models in the system. Model documentation includes a typeset mathematical formulation of the model as well as the actual declarations for the model in Max.

We designed TEFA with the entire modeling life cycle in mind and continue to improve it. The modeling life-cycle framework that we worked with includes the following elements:

(1) Identification of the sort of model likely to suit the problem at hand,

(2) Formulation and specification of a particular model,

(3) Implementation and validation of the model from step 2,

(4) Determination of parameter values (this typically requires data collection and analysis, and can benefit from data management tools),

(5) Solution of the model,

(6) Fielding and use of the model, and

(7) Modification of the model in re-

sponse to field-based experience.

The Maxi (Max interface) module provides the interactive documents, or hypertext features. (See Bhargava, Bieber, and Kimbrough [1988]; Conklin [1987]; Halasz [1988]; Marchionini and Shneiderman [1988]; and Minch [1990] for discussion of hypertext and hypermedia.) The idea is to build reports in which items—including numbers—are linked automatically to pertinent information about them. What we have is a broader, deeper, more general form of hypertext, which we call generalized hypertext [Bieber and Kimbrough 1989; 1990]. Standard hypertext systems provide system-level support for the user to navigate among the nodes in a hyperdocument, using links that have been set up by a builder using the editing facilities of the system. In generalized hypertext, as implemented in Max, the system can create links and nodes (reports) dynamically at run time, thereby eliminating a great deal of work by the document builder. (See Bieber and Kimbrough [1989, 1990] for a detailed discussion of these points.)

The architecture for the level 1 implementation is highly modular and robust. There is an internal communications path between the Maxi and TEFA modules, which communicate by sending messages in a recursively defined formal language. This structure allowed Maxi and TEFA to be developed quite independently. Also, the architecture for TEFA is itself highly modular and robust, and has proven successful for iterative development. The module consists of a number of knowledge bases, inference engines, and utility processes, all potentially in a many-to-many relationship, with everything ultimately controlled by a meta-level inference engine that tasks out work to the appropriate inference engines and utility processes. Using this architecture (TEFA has more than a score of inference engines), we have been able to add features to the system mainly by adding new inference engines and (occasionally) new knowledge bases and utilities, and by making very minor changes to the meta-level controlling inference engine. We call this architectural technique generalized meta-level inference.

In sum, the level 1 concept may be summarized as consisting of the following ideas:

(1) Model management (which includes declarative representation of models, declaration and utilization of information about models and data, model evaluation, and features to support activities throughout the modeling life cycle);

(2) Interactive documents (which include generalized hypertext, an interface to the model management system, and automated linking of information items with one another);

(3) Generalized meta-level inference; and

(4) Internal communication via a formal language.

Max, level 1, has been delivered and is being used. Improvements in Max are continuing to be made. A number of Coast Guard and Navy models have been incorporated into the system (including ASSET, CAPS, Maintenance, and PATROL) and others are being developed for implementation in Max. With the KSS software, we were able to re-implement ASSET and PATROL, two large models, in less than four person-weeks. ASSET was originally de-

veloped by the Navy, and PATROL was developed by Pritchett [1986] of the Coast Guard's R and D Center. Both of the original implementations cost more than $100,000. The Max KSS greatly reduces the costs of implementing and maintaining models. Further, these models are much more valuable in the KSS on a desk-top Macintosh, than they were implemented in FORTRAN on a distant minicomputer, using different user interface conventions. The models are more accessible in the KSS, and the user has much more information about them and can create reports directly from the outputs of the model runs.

Level 2 KSS Concept: DSS Executive

Our level 2 concept for the KSS originated with a simple thought: it is illuminating to distinguish between the subject matter of a decision problem and the process or procedure undertaken to make the decision. For example, the Office of Acquisition may be deciding what to do about replacing an aging fleet of buoy tenders. So, the subject of the decision is replacement of buoy tenders. In making the decision, however, the Office of Acquisition must perform certain tasks: it must assess mission needs, consider alternative means of keeping buoys in working condition, approve interim recommendations, and so on. These tasks make up the process of coming to the decision.

The level 2 KSS concept started with the idea that it would be possible to distinguish in the KSS software between process-level features and subject-level features. At the subject level, we would have a set of features very much like that of the level 1 KSS concept. Models could be run, data examined, reports issued, and documents produced. At the process level, we would have a model of the procedure by which the decision is to be made. This model, and information declared about it, would be used to provide process-level features.

Specifically, we proposed to model a decision process by representing it in an extended work breakdown structure. This is in accord with the Office of Acquisition's management of acquisition projects and ship building [Levine 1986; Meredith and Mantel 1985; Morris and Hough 1987]. The basic KSS level 2 concept is this. A user of the system could "go" to a particular project represented in the KSS and find certain data, documents, and reports about the project and its status. Further, the user could go to a particular node in the project's work breakdown structure. That would put the user in a local environment, in which he or she could work on the project by collecting information, creating or modifying documents, or by causing information to be added to that node of the project. At each node, certain data, documents, reports, and models are available. Since each node constitutes a local environment, what is available varies from node to node.

Information at a node may be keyed in by the user or may come—as MIS reports or as data—from another computer. In either case, the KSS may have information about the added information. For example, the KSS may have knowledge about the meaning of the row and column headings in an MIS report from an external system. Similarly, the KSS may record and use information about the entries in an external report and may automatically incorporate

data from the external report into other reports internal to the KSS. Operationally, the user can call up an MIS report (associated with a particular node) in which every item in the report is also a generalized hypertext button (or link icon). Using a mouse, the user can direct the KSS to produce a report about any item. For example, a report might supply the source, date, and original location of the data item. Thus, in some sense the KSS may know more about a report imported from an external system than the system that originally produced it, since the new report is linked to information in the KSS.

In sum, the level 2 concept (DSS executive) may be described as the level 1 concept (document-oriented DSS) plus

(1) Intelligence-based integration of distributed MISs,

(2) Interactive MIS reports, and

(3) Knowledge-based project management, organized around a work breakdown structure.

Our concept of project management is very different from that of existing commercial project management systems, which deal only with certain information—primarily timing information—pertaining to the lowest level nodes in the work structure, which are called work packets. Given this data, existing systems are designed to compute schedule-related information and to help the project manager to manage the schedule. We wanted to make the system the central repository for all information relevant to management of the project. Specifically, for the Office of Acquisition we want the KSS to become a computerized and interactive implementation of the Office's *Project Manager's Handbook*, the policy statement by which acquisition projects are managed.

Currently, we have developed the level 2 KSS to the point where it demonstrates our concept. All the essential elements and features we have described have been demonstrated with an implementation. We are developing the needed infrastructure in the KSS software, refining concepts, and initiating a complete redesign. In addition, we have working prototypes in HyperCard and in HyperLisp (HyperCard front-ending a Lisp machine), which we are using to display concepts to users and to elicit feedback from them.

Working with Max

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 ASSET, a model used by the Navy and the Coast Guard to estimate ship acquisition and life-cycle costs. Originally implemented in FORTRAN, we reimplemented it in the model representation language of TEFA by making a series of declarations. The various reports and features we will describe are produced inferentially at run time by Max, that is, by our generalized hypertext system. They are automatically available for any model declared in TEFA. This strategy significantly hastens the building of particular DSSs.

We designed Max to support both analysts and executive browsers. Analysts execute models under various data scenarios. Information is returned in standard TEFA reports. The analysts can then "copy and paste" from these standard reports to create their own ad hoc final reports. The standard reports are themselves interactive

documents, dynamically generated with buttons (naming generalized hypertext links) automatically embedded in them. Copying and pasting preserves these links in both the original and duplicate copies. (The computational cost of this is not excessive, since buttons refer to links, and the buttons are copied, not the links.) The executive browsers have access to these final reports, as well as automatic access to the standard reports via generalized hypertext linking.

We want the KSS to become a computerized and interactive implementation of the *Project Manager's Handbook.*

Imagine that an analyst wants to create a report comparing the total life-cycle costs for two different ships, a hydrofoil and a SWATH (a small waterplane area twin-hull) vessel. After starting the Max session, the analyst asks for a description of the ASSET model. To do this the analyst selects the describe command from a menu in Max's menu bar and then chooses the ASSET model as the subject of the command. The system produces a report that describes the ASSET model. The report resembles a word processing document. It contains text and mathematical formulae. Buttons, however, are highlighted in boldface, indicating that further information is available about the objects they represent.

During Max's initialization, the Max application (under TEFA) passes a list of command options in the communications protocol language to Maxi, the user interface subsystem, which then makes them available from the menu bar, under Max Financial. When the analyst chooses one of Max's menu items, this initiates a dialog with the Max Financial application. TEFA, the model management subsystem, determines (infers) that the text string "ASSET" is the name of a model node and executes the describe query as a Max command to generate (an instance of) a generic report model containing a description, an equational listing, and related top-level models.

The knowledge base is searched for these components, many of which themselves are nodes containing subcomponents. The text string "ASSET" is treated as referring to a bundle of virtual (determined inferentially, at run time) links, namely the run, describe, and suggest-scenario links. When the user chooses describe, a virtual report node is generated after traversal of a link of type describe, producing the report. Traversing the describe link does not cause an automatic display of canned information. Instead, it causes a Max procedure (in TEFA) to execute and process data in the knowledge base to create a composite report node which, as it turns out, is to be displayed. The model management subsystem formats this report in the communications protocol language and passes it to the interface, which in turn processes it and passes it on to the screen management subsystem. Here it is mapped to the computer screen, and its buttons—which have been marked by the model management subsystem—are highlighted. Buttons are tagged with an internal ID and display information (for example, their basic display text and whether it is textual, numeric, monetary, and so forth). The interface uses this information to determine the actual

text representation of the button on the screen. Buttons denote links, actual or virtual. Although they indicate a relation to information in the knowledge base, they are usually not explicitly linked to anything. Only when they are queried directly will a link be determined and traversed. When the user clicks on an "ASSET" button, Maxi sends a message to TEFA, which then determines the three things (run, describe, suggest a scenario) it can do with the ASSET model. Nearly every element presented to the user is created dynamically at run time by the KSS.

Next, suppose the analyst wants to execute the ASSET model. In order to execute the ASSET model for the user, the system traverses a virtual "execution" link (as opposed to traversing a "describe" link, a "suggest scenario" link, and so forth). As a result, the system executes the model and generates a standard report node comprising the major resulting values. Each of these values is represented by a hypertext button and each button is generated and maintained by the KSS software.

The final report constructed for the executive browser is typically short. The analyst need do no explicit linking but may copy portions of system-produced reports into the final report, which the analyst may also edit. The KSS software automatically preserves the buttons and linked information for text copied from one document to another. Max Financial supports many other features and contains several substantial mathematical models.

Conclusion

While we have accomplished a great deal, much remains to be done on the KSS project. We continually identify level 1 features that appear useful. In fact, the world of generalized hypertext and model management in DSS has only begun to be explored. In addition, nearly all of the level 2 world remains to be explored and investigated, as well as levels not yet envisioned. In addition, we must determine how best to use and deliver these technologies.

Far from being daunted by these prospects, we are excited and deeply engaged. We hope others will work on similar problems and ideas and that technical dialog in the DSS community will expand and become yet more lively.

Acknowledgments

This work was funded in part by the US Coast Guard under contract DTCG39-86-C-80348 with the University of Pennsylvania, Steven O. Kimbrough principal investigator. Special thanks to Rosie Milán for patience in implementing required editorial changes.

References

Alter, S. 1977, "A taxonomy of decision support systems," *Sloan Management Review*, Vol. 19, No. 1 (Fall), pp. 39–56.

Bhargava, Hemant; Bieber, Michael; and Kimbrough, Steven O. 1988, "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*, eds., Janice I. DeGross and Margrethe H. Olson (November 30-December 3), pp. 179–191.

Bhargava, Hemant, K. and Kimbrough, Steven O. 1990, "On embedded languages for model management," *Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences*, ed. Jay F. Nunamaker, IEEE Computer Society Press, Los Alamitos, California, pp. 443–452.

Bhargava, Hemant, K.; Kimbrough, Steven O.; and Pritchett, Clark W. 1990, "A balance

sheet approach to fleet mix planning," working paper, University of Pennsylvania, Department of Decision Sciences.
Bieber, Michael P. and Kimbrough, Steven O. 1989, "On generalizing the concept of hypertext," working paper, University of Pennsylvania, Department of Decision Sciences.
Bieber, Michael P. and Kimbrough, Steven O. 1990, "Towards a logic model for generalized hypertext," *Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences*, ed. Jay F. Nunamaker, IEEE Computer Society Press, Los Alamitos, California, pp. 506–515.
Conklin, J. 1987, "Hypertext: An introduction and survey," *IEEE Computer*, Vol. 20, No. 9 (September), pp. 17–41.
Halasz, F. 1988, "Reflections on NoteCards: Seven issues for the next generation of hypermedia systems," *Communications of the ACM*, Vol. 31, No. 7 (July), pp. 836–852.
Kimbrough, Steven O. 1982, "Pragmatic aspects of justifying decision support systems," *Transactions of DSS-82, Second International Conference in Decision Support Systems*, ed. Gary W. Dickson, San Francisco, California (June 14–16), pp. 138–145.
Kimbrough, Steven O. 1986, "On shells for decision support systems," working paper, University of Pennsylvania, Department of Decision Sciences.
Kimbrough, Steven O., Coe, Thomas; Pritchett, Clark; Roehrig, Stephen; Smith, Joseph A; and Sprague, Michael 1986, "A decision support system for evaluation of advanced marine vehicles," *Transactions of DSS-86, Sixth International Conference on Decision Support Systems*, ed. Jane Fedorowicz, Washington, DC (April 21–24), pp. 218–227.
Kimbrough, Steven O. 1987, "The argumentation theory for decision support systems," working paper, University of Pennsylvania, Department of Decision Sciences.
Levine, Harvey A. 1986, *Project Management Using Microcomputers*, Osborne McGraw-Hill, Berkeley, California.
Marchionini, G. and Schneiderman, B. 1988, "Finding facts vs. browsing knowledge in hypertext systems," *IEEE Computer* (January), pp. 70–80.
Meredith, Jack R. and Mantel, Samuel J., Jr. 1985, *Project Management: A Managerial Approach*, John Wiley and Sons, New York.
Minch, Robert P. 1989/90, "Hypertext in DSS," *Journal of Management Information Systems*, Vol. 6, No. 3 (Winter), pp. 119–138.
Morris, Peter W. G. and Hough, George H. 1987, *The Anatomy of Major Projects: A Study of the Reality of Project Management*, John Wiley and Sons, New York.
Pritchett, Clark W. 1986, "PATROL: Volume I, Model description and analyst's guide," USCG Report Number CG-D-05-87, Government Accession Number ADA-178 168.
Sprague, Ralph H., Jr. and Carlson, Eric D. 1982, *Building Effective Decision Support Systems*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

W. R. Johanek, Captain, US Coast Guard, Chief, Project Support Division by direction of the commandant, Washington, DC 20593-0001 writes "Although the final version of the knowledge-based decision support system (KSS) has not been completed, the prototype shows potential for the Coast Guard. Plans are underway to expand this effort to support large segments of information systems and office management. The advanced features such as automatic linking of data offer great potential for improving management effectiveness and efficiency. The model building capability of the KSS has already proved useful. In one working day, a proficient user of KSS took six pages of equations and turned them into a working model. The benefits to the Coast Guard of this software lie in its ability to help us do our job better. As we continue to use it and exploit its capabilities, we anticipate significant improvements in the areas of analysis, information systems, and project management."