HYPER MODEL MANAGEMENT SYSTEMS

by

P. Balasubramanian
Doctoral Program in Information Systems
Leonard N. Stern School of Business
New York University

Tomás Isakowitz
Assistant Professor of Information Systems
Leonard N. Stern School of Business
New York University

Hardeep Johar
Doctoral Program in Information Systems
Leonard N. Stern School of Business
New York University

Edward A. Stohr
Professor of Information Systems
Leonard N. Stern School of Business
New York University

July 1991

Center for Research on Information Systems
Information Systems Department
Leonard N. Stern School of Business
New York University

Working Paper Series

STERN IS-91-16

Abstract

In this paper we study the integration of Model Management and Hypertext systems to produce a Hyper Model Management System (HMMS). Model Management Systems constitute a class of software that is designed to support the construction, storage, retrieval, and use of models in the context of decision support systems (DSS). Hypertext systems allow users to split information into data fragments which the user can browse to find information by taking non-linear paths in computer based texts. It has been suggested that DSSs should be conceived as environments which support decision making. We support the view that such environments can be readily provided for the subtask of model management by hypertext systems. The different kinds of model knowledge can be captured within different types of nodes and the relationships among these can be maintained by hypertext links. In this paper we describe some aspects of model management where hypertext will have a significant impact. However, plain hypertext is ineffective in dealing with the dynamic nature of information in model management tasks where data is revised, models executed, and reports are created on the fly. Dynamic domains require *dynamic hypertexts.* In this paper we also study the requirements for *dynamic hypertexts.* These can be satisfied within the class of *generalized hypertext systems* by using special hypertext nodes and links which we describe. We explore different architectures to integrate MMS and Hypertext systems to obtain HMMSs. This paper emphasizes the need for a shift to integrated Model Management environments and proposes hypertext as an integrating technology.

Contents

List of Figures

1 INTRODUCTION

Hypertext systems [Nel80] provide a new form of computer- based support for reading documents. Rather than being constrained to the linear order of conventional documents, users are able to move through a hypertext document in a non-linear fashion, traversing hypertext "links" in order to explore concepts in more depth or access chains of related concepts. Model Management Systems (MMS) [Bla89] constitute a class of software that is designed to support the construction, storage, retrieval, and use of models in the context of decision support systems (DSS) [KM78].

In [BWD+89, BHW80] the idea of a special type of environment for DSS, *the hyperknowledge environment.* is proposed. This calls for a union of the ideas underlying hypertext and MMS. It is argued that a DSS can be conceptualized as an environment within which various kinds of knowledge are managed. Among these are "descriptive knowledge (e.g. data, information), procedural knowledge (i.e. algorithms), reasoning knowledge (e.g. rules) , linguistic knowledge (e.g. problem statement grammars) and presentation knowledge (e.g. forms, templates)." It is suggested in [BWD+89] that hypertext would provide an appropriate user-interface for such a DSS. Indeed, the different kinds of knowledge can be captured within different types of nodes and the relationships among these can be maintained by hypertext links. The need for "controlled focusing of attention by cognitively navigating among the universe of available concepts" is recognized as fundamental for effective utilization of DSS in general. Other recent work on hypertext and MMS is contained in Minch [Min90] who explores a number of potential applications and [BBK88] who focus on the need to provide links and nodes to support virtual structures such as explanations, model execution and report creation.

In this paper, we describe how hypertext technology matches the requirements of MMS in two senses: (1) as a useful style of DSS interface as outlined above, and (2) as a technology that provides mechanisms on which to build both an MMS and specific models. In sections 2 and 3 we briefly overview the features of MMS and hypertext, respectively. In section 4, we provide several brief examples showing how hypertext can support and enhance the func-

tionality of an MMS. In section 5, we argue that an enhanced kind of hypertext, a *dynamic hypertext* is required to support the desired features of an MMS. In section 6, we describe alternative architectures for combining hypertext and MMS technologies to form a "Hyper Model Management System" (HMMS). The paper concludes with some research ideas for developing the exciting possibilities that arise from the union of these two technologies.

2 Hypertext

Hypertext is an emerging technology in the field of information storage and retrieval. The concepts of hypertext have been around for some time [Bus45], but the recent widespread spurt in interest in this area is due to the availability of commercial versions [GUI87, Shn87b, App87]. The term hypertext was coined by Ted Nelson [Nel80] and it is being increasingly used as a model for information presentation and user navigation in information systems. It gets its name from the fact that users can browse the data fragments and find information by taking non-linear paths in computer based texts [AM84]. We refer the reader to [Con87, Nie90] for introductions to hypertext.

The basic building blocks in hypertext are nodes, links and buttons. Each node is associated with a chunk of information related to a document and nodes could be of different types [Hal88]. The node type depends on the class of data stored (e.g. plain text, graphics, audio or an executable program.) Another node type, action nodes, can cause the execution of procedural code [HMT87]. In most systems, nodes can be contained in different files allowing inter document traversal. A link defines a relationship between a source node and a destination node and can be traversed to access the destination node. This provides the users with the ability to navigate through the hypertext document, and discover new relationships. Buttons are areas within nodes on which links are defined [GUI87, BBK88]. It is now common to find hypertext systems with a one-to-one correspondence between nodes and windows; each window is associated with an object in the database and links are provided between these objects [Con87].

There can be many links associated with each node - e.g. a piece of text may be linked

to a more detailed document, to bibliographic references, to other associated concepts, and so on. The type of information associated with a link is usually indicated by the icon used as its button. As discussed in more detail below, there can be a number of link types - e.g. links that simply support traversal and others that also support the transfer of data between nodes. We also distinguish between hypertext systems that support only *explicit* links that are placed in the document explicitly by its author ahead of actual use, and hypertext systems that support *implicit* links that can be inferred by the system and dynamically created as required.

Current hypertext systems provide users with sophisticated user interface tools that enable them to inspect node contents and navigate through the network by selecting a path to follow [Luc90]. In addition to allowing users to traverse links at their own discretion, systems may provide users with predefined paths through the network, or the ability to specify search conditions for the selection of nodes. These queries may be content-based (searching the content of nodes) or structural (depending on the topography of the hypertext network). Because a major problem with hypertext is the potential for users to get lost in the detail of the information that can be accessed, hypertext systems usually provide path tracing and other navigation aids such as a view, in either text or graphical form, of the table of contents (or outline) of the document.

Hypertext, has been successfully used as an interface in several domains including portfolio management [BIKM91], software engineering systems [Del86, GS87, CB88], auditing [You89], idea organization [CB88, HMT87, AMY88]. Users have been shown to prefer hypertext "embedded menus" to conventional menu systems [Shn87a]. While a number of studies have shown a high level of satisfaction by hypertext users [DJT+89, MS88], more study is needed to confirm the almost universally held expectation that hypertext improves the efficiency of search and understanding of users.

Minch [Min90] provides an excellent overview of the potential applications of hypertext in decision support systems. Of interest here is his observation that many decision making paradigms have graph-based interpretations that can be represented naturally by hypertext systems. Hypertext systems are appropriate vehicles for DSS because of the diverse informa-

tion types that are permitted in the nodes of a hypertext [Con87], and the ability to provide a "context-based" access to the knowledge base [BBK88]. Also relevant to DSS is the ability of hypertext to improve the process of finding relevant information from voluminous written materials [Luc90] systems and to aid in the organization and manipulation of irregularly structured information.

3 Model Management Systems

Sprague and Watson [SW75] proposed that decision models should be managed by means of an MMS just as a DBMS manages data. This lead to the development of software that insulated users from the physical aspects of the organization and processing of decision models, just as a DBMS insulates users from the physical and organizational aspects of stored data. It has been realized that models are an important resource that need to be managed and modeling environments are being developed to help in this task [MMS89]. For example, Bhargava et al. [BBK88] envision an MMS as an operating system that, in addition to handling models like an operating system handles files, also fills the role of a human expert in modeling. In this section we examine two approaches to model management and attempt to integrate these approaches to identify general requirements for an MMS.

The decision-oriented approach to MMS emphasizes the use of models to support decision making. The role of an MMS is to provide an efficient means for the representation, storage, retrieval, and use (to support decisions) of models [DB84, BWD+89, BBK88, BHW80], and to handle the communication between the users and the model base [SC80, BHW80]. In [BHW80], an MMS consists of three components, a knowledge system, a language system, and a problem processor system. The knowledge system stores models, data, and other types of knowledge; the language system serves as the interface between the users and the MMS; and the problem processor system acts as the interface between the language system and the knowledge system by interpreting user requests, activating models, collecting data required by the models, and presenting results to the users in suitable presentation styles. According to the decision-oriented approach, the MMS should support each stage of decision

making identified in the DSS literature (e.g. [Ger71]): (1) intelligence, (2) diagnosis, (3) formulation, (4) model selection, (5) input of data, (6) model generation, (7) validation, (8) model execution, (9) sensitivity analysis, and (10) interpretation of results.

The second approach, the process approach, with roots in artificial intelligence, emphasizes the need for purposefully capturing knowledge to be used in modifying models to match the changing needs and perceptions of the user. Dhar and Jarke [BWD+89] for example, argue that the knowledge that goes into model development is as important as the model, and that therefore an MMS should also capture knowledge used in the building of models. Lerch and Prietula [BWD+89] suggest that an MMS should support the decision process and that the user interface should be designed to help the user's model construction process. The objectives of the process approach are therefore twofold, first the MMS should capture knowledge used to formulate models, and second, the MMS should assist the users in the process of refining and changing the model. To support the process approach, an MMS should provide: (1) automated model documentation, (2) the storage of meta-knowledge about models, (3) the composition of models from component sub-models, and (4) the revision of models as dictated by changed circumstances.

These two approaches take complementary views of the purpose of an MMS. The decision-oriented approach emphasizes the efficient storage and retrieval of models and their effectiveness in supporting specific decision making activity. The process approach emphasizes the maintenance of meta-knowledge about model construction and that models exist independently of particular decisions and serve classes of decisions. The combination of these two approaches leads to a more eclectic vision for model management.

Regardless of the approach, models are complex and their formulation, manipulation, and interpretation taxes the limits of our cognitive abilities. Model builders and users need considerable support to help them understand both the MMS system itself and the specific models that are developed and executed within the framework of the MMS. The interactive, non-linear quality of hypertext interfaces, should help reduce the complexity of the modeling process. In the next section, we examine the natural match between hypertext systems and model management systems in supporting several of the above requirements.

	HMMS	Model specific
Explicit Links	System documentation Context sensitive help	Model documentation Model storage and retrieval
Implicit links	Model selection Model construction	Context sensitive help for model user Interpretation of Results

Figure 1: Categories of Selected Modeling Facilities

4 Hypertext capabilities in support of MMS requirements

In this section we illustrate how several of the requirements for an MMS that were discussed in the previous section can be supported through the use of hypertext. The selected examples are assigned to different quadrants in the 2 by 2 matrix of Figure 1.

The horizontal axis emphasizes the distinction between modeling facilities that describe the HMMS itself or are built into the HMMS as permanent capabilities, and facilities that describe classes of models or model instances within those classes. Features in the former class are designed and developed by the original HMMS developers, while the builder of each different class of model must specify the model specific features to be provided by the HMMS. The vertical axis distinguishes between modeling facilities that can be provided by explicit (predefined) hypertext links and those that require implicit (dynamic) links for their implementation. In general, it is easiest to provide the capabilities in the upper left quadrant and hardest to provide those in the lower right quadrant. The following illustrations are designed to reveal the kinds of hypertext technology required as described more fully in the next section of the paper.

4.1 HMMS System Documentation

Here, the HMMS provides online information about itself - how to use the functions it provides to build and execute models, etc. Three features of hypertext systems are especially useful for documentation support: *Navigation, Browser Maps*, and *granularity adjustment.* Navigation helps the documentation browser move through the documentation in a non-linear fashion. Adjusting the granularity of information permits individual users to explore documentation at a level that matches their expertise. Browser Maps can be used to orient the users while navigating through the maze of help information available. Existing hypertext authoring systems can provide the features necessary to implement stand-alone HMMS system documentation.

4.2 Context Sensitive System Help

Pressing the help button while using any specific HMMS feature causes a window to be opened into the hypertext documentation that explains that feature. This is similar to the previous system documentation facility, except that the MMS and Hypertext subsystems must be linked tightly.

4.3 Model Documentation Support

To support model documentation, the HMMS must provide convenient hypertext authoring support for the model builder. Features of models such as variables, data coefficients, algebraic equations, and so on can be represented as hypertext nodes. Links can be constructed between these features and their model specific definitions. Other links would represent relationships between the different objects in a model. In the simplest implementation, the model builder would have to create these links, while in a more sophisticated implementation of model documentation support (one that would be classified in the lower right quadrant), the authoring system would understand model features and relationships and be capable of generating links between appropriate classes of objects automatically. In the second case,

the HMMS would be helpful in model construction as described below.

4.4 Model Storage and Retrieval

Storage and retrieval of models has been addressed extensively in the DSS literature and a number of model representation and retrieval languages have been suggested [DK84, BWD+89, DB84, Geo87]. Hypertext provides a useful and natural interface for these tasks [BBK88, Geo89]. Storage, retrieval and reuse can be facilitated by the use of a model dictionary that is analogous to a data dictionary. Each entry in the model dictionary could have several buttons for links accessing a description of the associated model, the source code for the model, the results from the last time it was executed, etc. All versions of a model could be accessed by following a chain from its dictionary entry. A hypertext network could be used to classify all models in the model base by type and function to aid access and understanding and to simplify reuse. A simple way to implement this feature is to require each model builder to use a hypertext authoring system to establish the node for his/her model in the model dictionary and the links to all information sources and to the classification scheme.

4.5 Model Selection

The HMMS can play an active role in helping users find the correct model for their task. Rather than having an ad hoc model classification scheme built by the model builders as in the last example, the system can have a comprehensive and authoritative model classification scheme (e.g. for statistical models) as a built-in feature. If the models are classified by the task they support, users could search through the hypertext network till they find the node that corresponds to the task they need performed and then retrieve the associated model and run it. This can be done using explicit links. A higher level of support for model selection can be obtained by utilizing intelligent agents to create implicit links to candidate models. For example, an expert system is proposed in [BBG89] to identify appropriate data analysis techniques for statistical analysis. The proposed techniques depend on the data.

Given a scenario with actual data, the expert system will interact with the users and with the hypertext to provide a list of candidate analysis techniques. These candidates will be connected via implicit links to the scenario.

4.6 Model Construction

An important capability for a DSS is to let the users construct and run decision models on the fly. Hypertext with its flexible structure and variable link types provides a natural interface to deliver such functionality. An intelligent agent could interpret the user's actions an assist him in building a model. For example, in linking two nodes, one with definitions about variables and the other with equations, the intelligent hypertext system could automatically establish explanatory links to the variables occurring in the equations. Furthermore, it could check whether the usage of the variables is consistent with their type.

4.7 Context Sensitive Help for Model Users

Context-sensitive help is the ability of the system to tailor its response for guidance to the user's context. The hypertext system can determine the context based on the user's location in a network of nodes. Thus, the response for a help about a variable might inform the user about its definition or provide a history of values depending on the node from which the request is issued. On a more sophisticated level, by keeping track of the nodes visited, a response tailored to what the user's knowledge can be constructed.

4.8 Interpretation of Results

For effective decision support it is necessary to help the users of a model interpret the results from the model. The users should be able to understand how the results relate to the model (e.g. a number is obtained as the solution of a quadratic equation), how the results relate to each other (e.g. cities and suppliers in a transportation model) and he should also be able to perform sensitivity analyses by changing some of the data. Since the reports are created by

the HMMS, it can utilize implicit links to establish the necessary connections. For example, values in the report will link to the variables in the model. The use of hypertext to help in sesitivity analysis is explained in the next section.

5 Dynamic Hypertext

The Hypertext models described in the literature so far are mostly static. By this we mean that the user has to physically create the links. *Dynamic* domains such as decision support systems require the capability to generate nodes and links in real time. This requires an implementation of hypertext that supports *virtual structures* and *computational mechanisms* which are used to generate a hypertext in real time.

The Max system prototype [KPBB90, BBK88] combines a dynamic hypertext front-end module with a model management back-end, providing a "proof of concept" that virtual structures and computation can supply the needed liaison to integrate the model management and hypertext systems. The Max system is based upon a *Generalized Hypertext* [Bie90], a term coined in [BBK88]. Link traversal in a plain hypertext can be described as a *select-traverse-display* operation: select the link (e.g. click on a button), traverse the link, display the node at the link's destination. In a generalized hypertext however, link traversal is extended so that the system deduces the link to be traversed and infers its next action based on this traversal. The node at the destination of the link might not exist, in which case it should be created as a result of this traversal. This corresponds to an *infer-traverse-infer* operation.

In order to support MMS activities, hypertext should support interaction between its components. This requires that information be automatically transmitted between nodes, and that nodes have the ability to engage in communication. This fits within the generalized hypertext framework, with a specialized type of traversal: *infer-transmit-infer*. *Valuation links* and *valuation nodes* are introduced in [BI91b] to this effect. A *valuation link* performs a *data push* operation by transmitting a value from origin to destination and *pasting* it in the latter, thereby changing its contents. A *valuation node* is a node that represents a

computation that can be triggered from within the hypertext. The result of *evaluating* a valuation node is a *value* which can be transmitted by valuation links originating in this node. If the link's destination is again a valuation node, the value transmitted by the valuation link activates its *evaluation*. In such a way computations can take place on a hypertext network. We will clarify these ideas with two examples.

Execution: the user has provided a scenario which contains model data, a solver has been selected. To solve the problem, the scenario and model statement could pass data as arguments via valuation links to the solver which could then execute and produce a solution.

Sensitivity Analysis: As the solver produces a report, it embeds valuation links that relate specific values to the portion of the solver used to compute them. In order to perform sensitivity analysis, the user might want to change some of the data in the scenario. Since this data is linked via valuation links to the solver, a *data push* operation will take place which can trigger re-execution of the solver with the new data. The result of this execution updates the report via the embedded valuation links. Since dependencies between values are explicitly maintained, the hypertext might reduce the amount of computation needed by re-computing just the affected portions of the network.

We now describe some of the features required from a dynamic hypertext for model management.

1. **Open Code:** the links and buttons should be kept separate from node contents so that these contents can be utilized for other purposes. For example, the code for a solver might have embedded links to specific models and scenarios. However, these links should not interfere with the compilation and execution of the solver.

2. **Implicit or automatic links:** these are links that are created by the system as the user interacts with it. They are called *implicit* as opposed to *explicit links*, which are created by the user who fully determines their origin and destination. For example, context sensitive on-line help requires that the destination of some links be changed depending on the context from which the call for help is executed.

3. **Valuation nodes and valuation links:** as described above, these provide for interaction between model components to support a variety of tasks, sensitivity analysis among them. Both explicit and implicit valuation links are required.

4. **Intelligent agents for link creation:** some reasoning is required for more sophisticated operations such model selection (via an expert system) or on-the-fly link creation based on a conceptual representation of the domain (e.g. as in a solution to a transportation problem). In the next section we will describe *Bridge Laws* that could incorporate some of this intelligence.

Having shown that desirable features for HMMSs are met by the dynamic hypertext described here, we now turn to an analysis of the different architectures that can be used to construct an HMMS.

6 Architecture

We envision the HMMS as an integration of an MMS and a hypertext. There are a number of different approaches for such an integration. Before engaging in a detailed discussion on the architecture of an HMMS, we would like to highlight the important features to be supported by such a system:

1. *modularity:* it should be relatively easy to independently change the MMS and hypertext components thereby offering the ability to incorporate updates and experiment with new environments;

2. *seamless integration:* the connection between the model management and the hypertext should be transparent to the user, i.e. the fact that there are two systems should be undetectable by the user;

3. *minimal user overhead:* there should be no penalty for using the HMMS as opposed to the MMS, in fact the HMMS should be easier to use than the MMS;

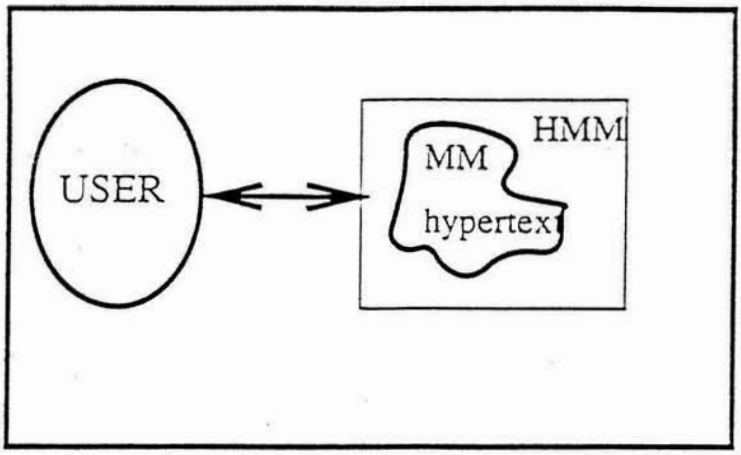

Figure 2: The *ad-hoc* architecture

4. *reliability:* the integration of the two systems should be as reliable as its components; this calls for a conceptually simple and easy to maintain integration mechanism;

5. *efficiency:* the integration mechanism should not slow down system performance; and

6. *enhanced functionality:* the user should be able to utilize with the same ease MMS and hypertext features, furthermore the hypertext features should enhance model management operations by providing intelligent search and browsing features.

We will consider four different architectures to support the desired integration.

6.1 Ad-hoc

A straightforward approach shown in figure 2 consists in developing a totally new system with a hypertext component that has hard-coded MMS features.

This system will probably achieve seamless integration, low user overhead (none at all), efficiency and enhanced functionality; it will not be modular since the systems and their integration are fixed. Nothing can be said about its reliability since this is a brand new

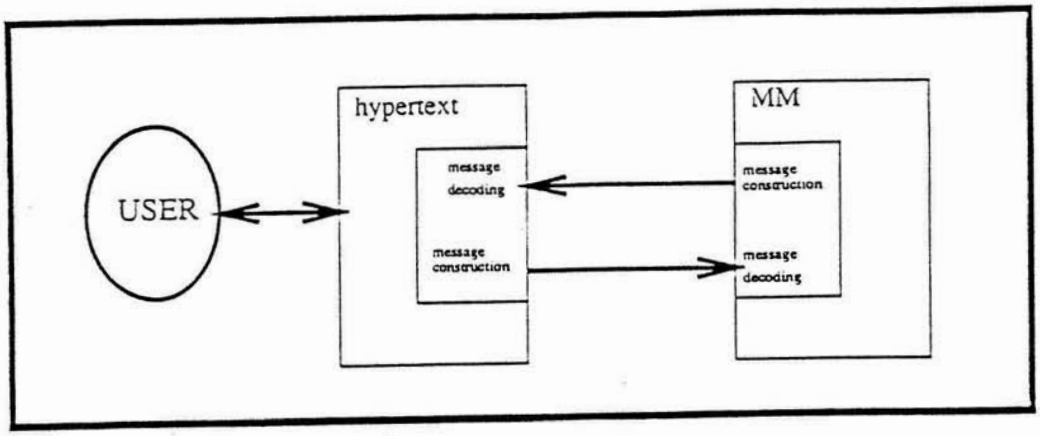

Figure 3: The *message passing* architecture

system. Furthermore, in this case we would not be considering an integration of MMS and hypertext, but a brand new product which would suffer from isolation since it will be the lone system using a specific user-interface. This goes against the fundamental concept of a DSS environment.

6.2 Message Passing

This type of architecture has received particular attention in the literature. It is proposed in [BWD+89] and used in Max [BBK88]. As shown in figure 3, both systems, the MMS and the hypertext, run concurrently in compatible environments and the two systems intercommunicate via messages. Whenever the user wants to engage in a model management activity, he/she interacts with the hypertext via *surface messages* which elucidate the MMS operation required to satisfy the user's request; the hypertext system then constructs a *deep message* which is passed to the MMS. The MMS processes the message and if information needs to be presented to the user, sends a *presentation message* to the hypertext which appropriately presents the information to the user. The cycle repeats itself forming the basis of the interaction.

This type of architecture has to provide an efficient messaging mechanism to support the integration. The MMS has to understand the language of the hypertext in order to provide useful *presentation messages*; and vice versa, the hypertext has to understand the MMS language in order to issue *deep messages*. As a consequence, the integration is not completely modular, since a change in either of the two systems will require changes in the other one. For example, if a new hypertext interface is installed, all presentation message building activities that take place within the MMS have to be adapted to the new hypertext's language. The level of seamless integration, the reliability and the efficiency of the HMMS will depend upon the quality of the message passing mechanism - for example, errors due to incomprehensible messages by either of the systems might surface to the user. There will be minimal user overhead since the hypertext will be intuitive. The level of enhanced functionality will depend upon the message constructing capabilities embedded within both systems and might thus be compromised.

The main difficulty with this approach is the compromise in modularity. Each system has to issue messages using the other system's language and this is an obstacle for modularity. The next approach provides a partial solution to this problem.

6.3 Bridge Laws

Instead of burdening each of the systems with knowledge about the other one, it is proposed in [BK90, BI91a] to use a separate *translation schema* external to the systems involved. An additional system: the *translator* is in charge of establishing proper communication between the systems. In figure 4 we see the interaction of both systems through the translator. Another difference between this approach and the message-passing one is the use of general *bridge laws* to achieve translations of whole classes of concepts from one system to the other one. This acts as a *concept translator* relating concepts in one environment to concepts in the other one. The message passing approach required one message and one translation per activity, here we use schemas to translate whole classes of messages. As an example, consider a specific link that is to achieve *model execution*. The relevant bridge law might specify that

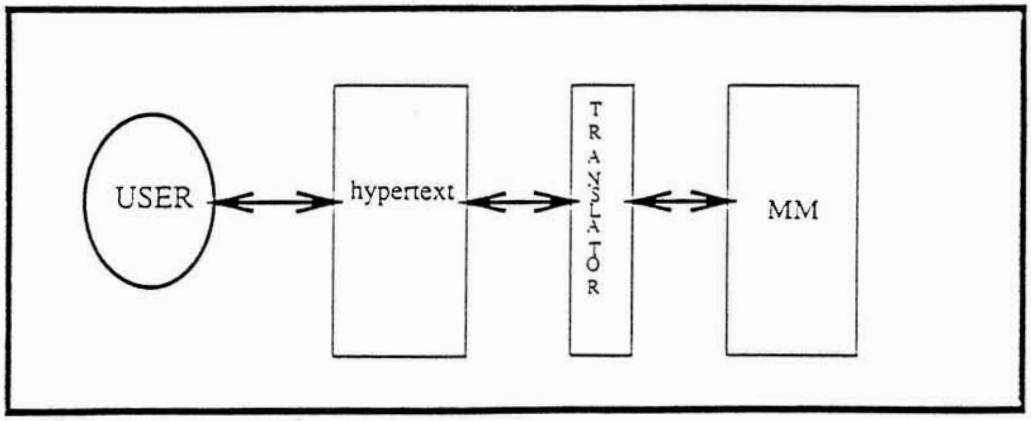

Figure 4: The *Bridge Laws* architecture

> "*all* execution links *will invoke a specific* MMS:model-execution *function upon traversal*"

Thus there is no need to construct a message each time.

What is achieved with this architecture is a separation of the translation mechanism from the application. The hypertext is no longer required to build messages in the MMS language, nor is the MMS required to know about nodes, links and buttons; the translator will take care of this. However, this might result in a performance slowdown since the translator will be an intermediary that will take time to perform its translation task.

In terms of the criteria set forth above, we can see that all criteria except for efficiency and reliability are met. We explained above the issues related to efficiency. Reliability of the integration will depend upon the reliability of the translation mechanism. The power of this approach lies in it generality. The heaviest burden of the integration falls within the translation schema. In [BI91a] some technical details about this mechanism are provided.

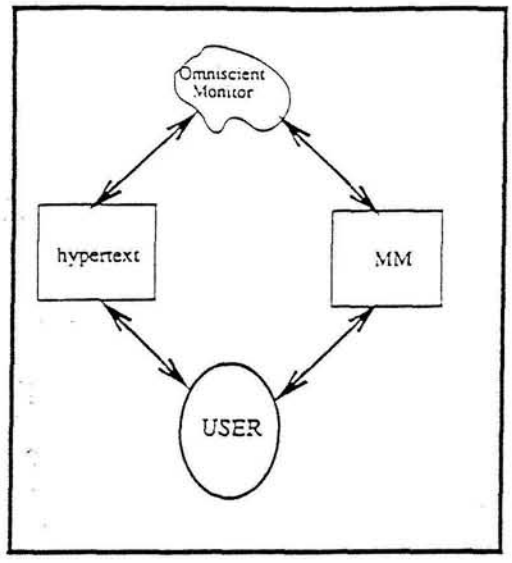

Figure 5: The *Omniscient* architecture

6.4 Omniscience

At the core of this unexplored approach lies an omniscient *monitor* that observes the activities of the systems and relates one to the other without their participation. This monitor is an *all powerful* entity that is able to inspect the inner states of each system and has access to all of its public functions. For example, when the hypertext system traverses a link labeled *execution*, the monitor will understand this as a model-execution operation and will instruct the MMS to execute the corresponding model. When the MMS completes execution it generates a report. As the monitor watches the MMS report creation it instructs the hypertext to import the report into a window. Both systems will interact with the user and with the monitor and neither will know of the other's existence. Moreover, the MMS could be interfacing with a different user through some other mechanisms without compromising the HMMSs performance. Figure 5 gives a high level view of this approach. The details of this mechanism and its feasibility need to be determined. The basic idea is that each system will be analyzed in terms of its *internal state* and its transitions between states. The monitor will map internal states from one system to the other, and artificially cause state transitions in one system to reflect changes in the other. This architecture is at the opposite

end of the spectrum from the *ad-hoc* approach. Its basic philosophy is one of *no intervention.* It will not require any special mechanism from the hypertext and MMS systems involved. This architecture will score high in modularity, seamless integration and reliability. However its efficiency will be compromised by the monitor's actions. It is questionable whether the integration of the systems will provide more functionality than its components because the user interacts with each system independently.

7 Conclusion

Hypertext provides a natural and flexible interface to a model management system and enables the MMS to become an active participant in the model management process rather than being a passive storage facility. Here we described how different features of hypertext can be used to meet the dynamic requirements of an MMS. The integration of Model Management and Hypertext systems naturally leads us to propose Hyper Model Management systems for DSS. We have explored some of the salient features of these systems and we have pointed out avenues for further research in the realm of dynamic hypertext and in the domain of intelligent assistants to model management. We are working on a prototype implementation of an HMMS on the Maluar [Isa91] system on which we will report separately.

References

[AMS4] R. Acksyn and D. McCracken. ZOG and the USS Carl Vinson: Lessons in Systems Development. In *Proceedings of the First IFIP Conference on Human Computer Interaction.*, Amsterdam, Netherlands, 1984.

[AMY88] Robert .M. Akscyn, D. L. McCracken, and E.A. Yoder. KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations. *Communications of the ACM*, 31(7):820–835, 1988.

[App87] Apple Computer, Inc., Cupertino, CA. *Macintosh HyperCard User's Guide*, 1987.

[BBG89] I. Bockenholt, M. Both, and W. Gaul. A Knowldeg Based System for Supporting Data Analysis Problems. *DSS*, 5:345–354, 1989.

[BBK88] Hemant Bhargava, Michael Bieber, and Steven O. Kimbrough. Oona, Max and the WYWWYWI Principle: Generalyzed Hypertext and Model Management in

a Symbolic Programming Environment. In Janice I. DeGross and Margarethe H. Olson, editors, *Proceedings of the Ninth ICIS*, pages 179–192, 1988.

[BHW80] R. H. Bonczek, C. H. Holsapple, and A. B. Whinston. Future Direction for Developing Decision Support Systems. *Decision Sciences*, 11(4):616–631, October 1980.

[BI91a] Michael P. Bieber and Tomás Isakowitz. General Hypertext Interfaces via Bridge Laws, a Logic Modeling Approach. Technical report, NYU, 1991.

[BI91b] Michael P. Bieber and Tomás Isakowitz. Valuation Links: Formally Extending the Computational Power of Hypertext. Technical report, CRIS, New York University, 1991.

[Bie90] Michael P. Bieber. *Generalized Hypertext in a Knowledge-based DSS Shell Environment.* PhD thesis, Decision Sciences Department, University of Pennsylvania, Philadelphia, PA 19104, December 1990.

[BIKM91] P. R. Balasubramanian, Tomás Isakowitz, Rob Kauffman, and Raghav K. Madhavan. Exploiting Hypertext Valuation Links for Business Decision Making: A Portfolio Management Illustration. Technical report, NYU, 1991.

[BK90] Michael P. Bieber and Steven O. Kimbrough. On Generalizing the Logic of Hypertext. In *Proceedings of the 23^{th} Hawaii International Conference on System Sciences*, January 1990.

[Bla89] R Blanning. Model Management an Overview. In *Proceedings ISDP-89*, 1989.

[Bus45] Vannevar Bush. As We May Think. *Atlantic Monthly*, (176):101–108, July 1945.

[BWD+89] Robert Blanning, Andrew Whinston, Vasant Dhar, Clyde Holsapple, Mathias Jarke, Steven Kimbrough, Javier Lerch, and Michael Prietula. Precis of Model Management and the Language of Thought Hypothesis. In *Proceedings ISDP-89 (forthcoming)*, 1989.

[CB88] J. Conklin and M.L. Begeman. gIBIS: A hypertext tool for exploratory policy discussion. In *Computer Supported Cooperative Work*, pages 140–152. ACM Transaction on Office Information Systems, September 26 - 28 1988.

[Con87] Jeff Conklin. Hypertext: An Introduction and Survey. *IEEE Computer*, 20(9):17–41, September 1987.

[DB84] Amitave Dutta and Amit Bose. An Aritificial Intelligence approach to Model Management in Decision Support Systems. *Computer*, pages 89–97, 1984.

[Del86] N. Delisle. Neptune: A Hypertext System for CAD Applications. In *Proceedings of ACM SIGMOD International Conference on Management of Data, Washington, D.C.*, pages 132–143 (Also available as SIGMOD Record Vol 15, No. 2. June 1986), 1986.

[DJT+89] Eagan D.E., Remae J.R., Landauer T.K., Lochbaum C.C, and Bellcore L.M. Behavioral evaluation and analysis of a hypertext browser. *Working paper, Bell Communications Research*, 1989.

[DK84] D. Dolk and B. Konsynski. Knowledge Representation for Model Management Systems. *IEEE Transactions on Software Engineering*, SE-10(6), November 1984.

[Geo87] A. M. Geoffrion. An Introduction to Structured Modeling. *Management Science*, 33:547–588, April 1987.

[Geo89] A. M. Geoffrion. Hypertext and Structured Modeling. *Informal Note*, January 1989.

[Ger71] T. P. Gerrity. Design of Man-Machine Decision Systems: An Application to Portfolio Management. *Sloan Management Review*, 12(2):59–75, 1971.

[GS87] Pankaj K. Garg and Walt Scacchi. On Designing Intelligent Hypertext Systems for Information Management in Software Engineering. In *HyperText-87 Proceedings*, pages 409–432, 1987.

[GUI87] GUIDE. *Guide User's Manual.* Owl International Inc., 1428 NE 21 St., Bellevue, WA 98007, (206) 747-3203, 1987.

[Hal88] Frank G. Halasz. Reflections on Notecards: Seven Issues for the next generation of Hypermedia Systems. *Communications of the ACM*, 31(7):836–852, 1988.

[HMT87] Frank G. Halasz, T.P. Moran, and Randy H. Trigg. Notecards in a Nutshell. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 1987.

[Isa91] Tomás Isakowitz. MALUAR - A Computational Hypertext Environment. Technical report, NYU, 1991.

[KM78] Peter G.W. Keen and Michael S. Scott Morton. *DECISION SUPPORT SYSTEMS: AN ORGANIZATIONAL PERSTPECTIVE.* Addison-Wesley, Reading, Massachusetts, 1978.

[KPBB90] Steven Kimbrough, Clark Prichett, Micahel Bieber, and Hemant Bhargava. The Coast Guard's KSS Project. *Interfaces*, 20(6):5–16, November/December 1990.

[Luc90] Dario Lucarella. A Model for Hypertext-based Information Retrieval. In A. Rizk, N. Streitz, and J. André, editors, *Proceedings of the European Conference on Hypertext*, pages 81–94, France, November 1990. INRIA, Cambridge University Press.

[Min90] R.P. Minch. Applications and Research Areas for Hypertext in Decision Support Systems. *Journal of Management Information Systems*, 6(3):119–138, Winter 1989-90.

[MMS89] P. Ma, F.H. Murphy, and E.A. Stohr. A Graphics Interface for Linear Programming. *Communications of the ACM*, 32(8):996–1012, May 1989.

[MS88] G. Marchionini and Ben Shneiderman. Finding Facts Vs. browsing knowledge in hypertext systems. *IEEE Computer*, pages 70–80, January 1988.

[Nel80] T. H. Nelson. Replacing the Printed Word: A Complete Literary System. In *IFIP Proceedings*, pages 1013–1023, 1980.

[Nie90] Jakob Nielsen. *HyperText & HyperMedia.* Academic Press, 1990. A very readable introduction to the field.

[SCS0] Ralph Sprague and Eric Carlson. A Framework For Decision Support Systems. *Database*, 4:1–13, 1980.

[Shn87a] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley, Reading, Massachusetts, 1987.

[Shn87b] Ben Shneiderman. User Interface design for the Hyperties electronic encyclopedia. In *Proceedings ACM Hypertext'87 Conference*, pages 189–194, 1987.

[SW75] Ralph. H. Sprague and Hugh J. Watson. Model Management in MIS. In *Proceedings of the* 7^{th} *National AIDS Meeting*, pages 213–215, 1975.

[You89] L. De Young. Hypertext Challenges in the Auditing Domain. In *HyperText-89 Proceedings*, pages 169–180, November 1989.