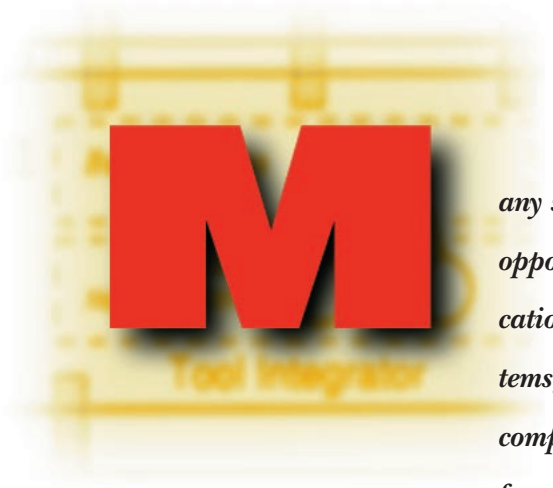
Designing Hypertext Support for **Computational Applications**



Many scientific and business applications are* computational, *as opposed to* display-oriented. *Examples include accounting applications, computer-aided design systems, geographic information systems, expert systems, and statistical analysis packages. People use computational applications primarily for their underlying analytic functionality, not for reading or navigating through large amounts of displayable information.

Michael Bieber and Charles Kacmar

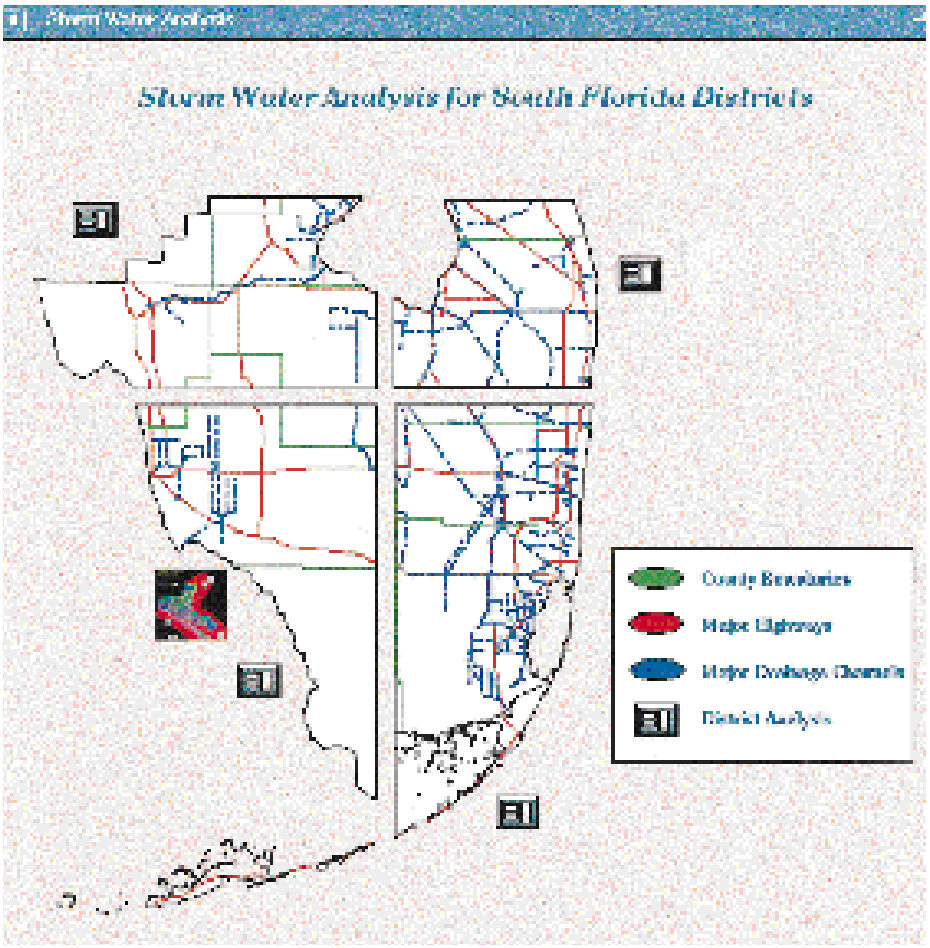
Developers holding a limited view of hypertext may dismiss the possibility of hypertext support for computational applications, when in fact hypertext functionality can supplement an application's computational power. In this article we describe how developers can incorporate hypertext functionality into computational applications.

What benefit do users gain from hypertext support in computational applications? Managing the myriad of interrelationships in a computational application's knowledge (data and calculated information) is difficult for a user. It can be improved by streamlining access and increasing user comprehension through interface enhancements (e.g., visualization). Augmenting an application with hypertext support results in new ways to view and manage the application's knowledge, by navigating among items of interest and annotating with comments and relationships (links).

Unfortunately, many computational system developers view hypertext only in terms of accessing and managing documents (or smaller units of static information). Such display-oriented behavior characterizes the majority of hypertext systems. SEPIA (see Thüring et al. and Streitz in this issue), KMS [1], Aquanet and VIKI (see Marshall and Shipman in this issue), NoteCards [10] and Intermedia [25] all are designed to facilitate authoring, creating relationships, displaying information, and navigating through large information spaces. (Hypertext *viewers,* such as Mosaic and Lynx, provide navigation only.) In fact, most hypertext design methodologies and guidelines, such as those in [10], Hypermedia Design Model (HDM) [9], and those presented in this special issue, were developed specifically to optimize these functionalities. In computational applications, on the other hand, document management and object display are second in importance. Here, hypertext must augment both interface and analytical activities. This requires that instead of adding computation to a hypertext application (as in [10,

20]), hypertext must be integrated into the design of the computational application.

A geographic information system (GIS) is an example of a computational application. It computes and displays information about geographic features such as roadways, utility easements, property boundaries, landfills, population densities, and vegetation coverage. In fact, a GIS must support both computational and display facilities, and for this reason serves as an excellent focus for this article.



The primary output from a GIS is usually a map, such as that in Figure 1. Maps serve not only to convey information about geographic features but also as navigational facilities, orienting and providing context during computational activities. Additionally, a GIS may produce textual output, such as a regression or analysis of variance (ANOVA), on geographic data (see Figure 2). Geographic data often resides in relational databases and can contain thousands of variables (attribute/value pairs). Elements such as roads and property boundaries are stored as vectors, rendered upon access to visual specifications (e.g., scale, resolution). Hypertext, when added to a GIS environment, must be integrated seamlessly into the display of these complex data. For example, anchor markers must not obstruct or distort the extremely dense displays that characterize GIS.

Consider a GIS used by a city engineer to analyze storm water flooding problems.[1] Analyzing data that measures storm water depth and flow through a drainage facility produces a graphical display depicting flow in different colors (as in Figure 3). The analysis requires the GIS to extract data from multiple data sets, analyze and synthesize the data within the visual parameters specified by the user, and render the results. Hypertext augments this environment, for example, by providing links that initiate more detailed calculations, direct access to detailed explanations for calculations, and backtracking through analysis steps. Hypertext also enables the engineer to annotate specific components of the analysis with explanations and other remarks.

[1]All GIS examples in this article are hypothetical, as no GIS incorporates the hypertext support we envision (yet). One goal of this article is to show how extensive hypertext functionality could augment a relatively unsupported computational application domain, such as GIS.

Figure 1. Hypertext-augmented map providing access to four hypothetical analyses of storm water capacity for districts in South Florida. County boundaries, major highways, drainage channels (artificial or natural) and coastal boundaries make up its base layers. To further define the boundaries of the water management districts, the map is exploded. The icons adjacent to each district provide access to a detailed analysis of storm water capacity (see Figure 2). The thumbprint graphic provides access to a density analysis of storm water capacity for the district (see Figure 3). In addition, the user may select any object (e.g., road, coastline, county boundary, highway, drainage channel,) to get more information, or select any part of the region to zoom into with more accuracy.

In what follows we identify the role each component of a computational application plays in providing hypertext support. We then describe hypertext features builders of computational applications should consider. We discuss the general benefits and caveats of each, as well as the coordination each requires in computational applications. Next we outline various support tools, so builders do not have to implement hypertext functionality entirely from scratch. We conclude with some observations on the unique nature of hypertext support in computational environments.

A note on our terminology: *Builders* design and implement computational systems or packages and the general hypertext features a system supports. *Developers* design and implement individual applications within a hypertext-enabled computational system. *Users* employ individual applications to accomplish specific *tasks*. In addition, we use the terms *object* and *class* in their broadest sense, which includes their use in object-oriented systems. We make no distinction between "hypertext" and "hypermedia." All concepts we discuss apply equally to both.

Managing hypertext functionality on behalf of an application *requires cooperation from both the computation and interface modules.*

Applications: Interface, Computation, and Hypertext Components

For illustrative purposes, logically we consider computational applications to consist of three components: a computational module, a user interface module, and a hypertext module. Implementation architectures, however, lie beyond the scope of this article. How builders actually implement these logical components can vary. For example, builders could embed hypertext functionality in an application's computation and interface subsystems, develop a hypertext subsystem as part of the application, or deploy a hypertext module as a distinct and separate entity, as in a client-server architecture [12, 13]. Regardless of the physical implementation architecture, we consider the hypertext component logically distinct from computation and interface activities. For simplicity, we assume the computation and hypertext modules access a (separate or shared) data management component to support retrieval, storage, and management of their objects.

The hypertext module manages all hypertext functionality on an application's behalf. Hypertext functionality (navigation, annotation, and view- and structure-oriented features) is constructed using the basic building blocks of nodes, links and anchors. Thus, the hypertext module must represent application objects internally in terms of these constructs. Nodes represent objects of interest to the application: models, variables, data sets, calculation results, computer-aided design (CAD) objects, reports, and so on. Links represent relationships, for example, among related data sets, between a calculation and its explanation, or among the components of a composite GIS map or CAD object. Anchors define the endpoints of links. Depending on a link's focus, its anchors could map to an entire node or to specific objects within the node's content, such as a segment of the coastline on a complex map.

Managing hypertext functionality on behalf of an application requires cooperation from both the computation and interface modules. First, the hypertext module must communicate with the other modules. This implies that a pathway and protocols exist supporting the flow and interpretation of messages [12]. Second, the interface module must display hypertext information, allow users to make hypertext selections, and pass appropriate messages to the hypertext module when hypertext events occur.

The hypertext module intercepts messages from the computational module containing display requests. The hypertext module must know the application's message format to analyze it. Through argument and template positions, declared markup, keyword search, lexical, or content analysis,[2] unique object identification (in object-oriented systems), etc., the hypertext module identifies references to application objects. It maps "standalone" objects to nodes. Nodes generally appear in separate windows or are a major focus of attention in a window. Querying the hypertext database or the application's structural schema (described in the section on automated linking) determines whether an object or a reference to an object within a node's content (such as a keyword or labeled line segment) has a link. If so, the hypertext module maps the object or object reference to an anchor. It then augments the display message (if allowed by the communications protocol) with hypertext identifiers or generates additional messages so the interface knows where to highlight anchors.

When the user clicks on a link anchor, the interface passes its identifier to the hypertext module. The hypertext module determines the anchor's link(s), requesting computational module processing when appropriate. Some systems allow users to preview a

[2] We omit further discussion on automated linking based on lexical or content analysis, which also can be used to determine similarities (relationships) among documents. This falls under the field of information retrieval. See [19] for leads into this literature.

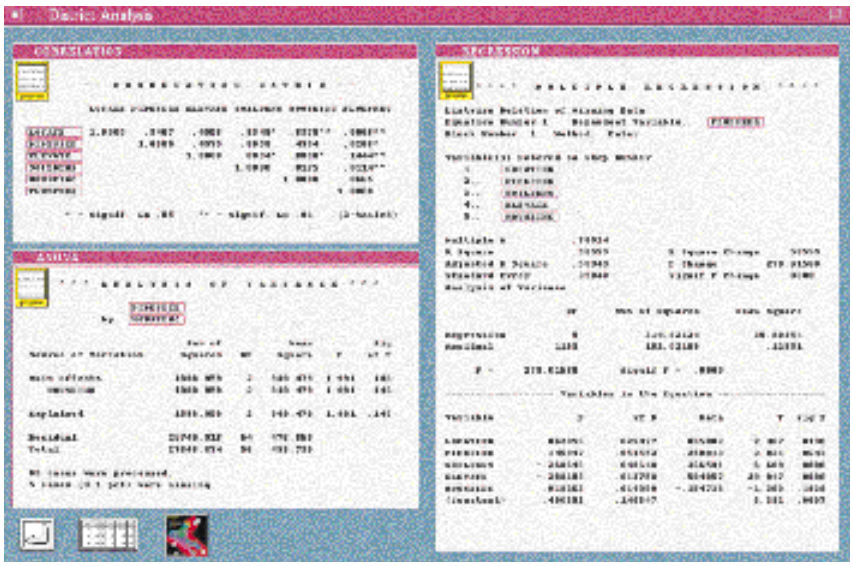


Figure 2. Storm water capacity is analyzed three ways: correlation, analysis of variance (ANOVA) and regression. Users may access the statistical program for each through the small program icon in its upper left corner. A detailed description of any variable may be accessed by clicking on its boxed name. Links at the lower left of the screen provide access to related documents. The back arrow icon returns the user to his or her previous position. The second link accesses the data file used to produce the analyses. The thumbprint graphic provides access to a density analysis of storm water capacity for the district (see Figure 3).

link's *metainformation*, such as its type, associated operations, and destination (as mentioned in Kahn's sidebar). This is especially important if link traversal invokes a time-consuming computation or an operation that alters the application contents (e.g., appending to a data set).

Note the responsibilities of the interface module in the preceding paragraphs. This module must display anchors as selectable objects and provide a mechanism for users to select them. Anchors, for example, may overlap (e.g., a data value within an equation where both are anchors, or smaller geographic features embedded within larger ones). Several other compliance issues are discussed in [4] and [7]. Furthermore, the interface module must display new types of information that originate from the hypertext module, including link metainformation, annotations, overviews, and guided tours that restrict user interaction (as well as the hypertext module's own menus, user dialogs, and warning messages).

Hypertext Functionality

What to Link: A Philosophy of Maximum Access

Hypertext functionality includes navigational, annotation, structure-oriented, and view-oriented features. We begin by considering what one should link in a computational application.

What should the hypertext module represent as anchors and links in a computational environment? We view hypertext as a philosophy of *maximum access*—giving the user freedom to access and explore as much information and metainformation as possible, both to better understand an application as a whole and to have more confidence in its results. Under this philosophy, any item of interest to a user is a candidate for an anchor (though developers must take care not to overwhelm novice users). For computational applications this includes:

1.) traditional linkable objects in non-computational applications;
2.) objects for which a definition, attribute value, or other metainformation is available, including executable programs, commands, and parameters; and
3.) the result of any computational operation (e.g., an explanation, calculation, composite of individual components).

Users may need access to many associations or relationships. Possible links include:

1.) the connection between an object and its metainformation;
2.) operations connecting inputs and outputs;
3.) structural and group relationships among objects (e.g., the components of a composite geographic phenomenon or documents within a related cluster);
4.) different views of the same object; and
5.) spatial, temporal or other process relationships (e.g., the next step in a work flow or procedure, subtask in a project management system, or document in a series).

Automated Linking Through Structural Relationships

Manual linking in dynamic computational systems is impractical, in part due to the sheer number of application elements and in part because many computational elements, such as calculation results, do not exist when the developer originally declares the application and would manually link its contents. When the computational module passes a message to the interface module, the hypertext module automatically analyzes the message (see earlier discussion) and dynamically maps instances of *structural anchors* and *structural links* to application objects [3]. This eliminates almost all manual linking except for objects of interest and relationships that cannot be generalized (e.g., the nonstructural links we discuss under annotation functionality). For example, whenever a geographic region is displayed, users could have access to some or all of the structural links shown in Figure 1, such as all geographic features (e.g., roads, rivers), its water management spatial data set, or storm water analyses with any neighboring region. HDM refers to these structural relationships as design "in the large" [9], and it is these relationships that Isakowitz et al. discusses in this issue. Structural relationships equate to an application's schema. Builders and developers can declare them, for example, as expert system rules or database tuples, or as with one author's current prototype under development [4], as bridge laws [3, 24] in Prolog predicates.

Following the hypertext philosophy of maximum access prompts developers to scrutinize an application in a new light. In our own TEXPROS document management system, for example, this helped us identify indirect structural relationships between individual documents and related document folders, which the system could infer automatically and users thus could access directly [23]. In another project involving a detailed analysis of NASA's Knowledge-based Software Engineering Environment [15], we found no way to access (cross-reference) an object across the system's four design environments, although structurally, the system could have provided such links automatically. Developers need to consider carefully the types of possible relationships and granularities that a user can access and ensure that the application structure is flexible enough to represent these. Can a GIS user, for example, trace a data value's use across different analyses (from the data set through resulting computations in both textual reports such as Figure 2 and graphics such as maps)?

Dynamic Regeneration

In computational environments, several of the hypertext features we will discuss next encounter the challenge of dynamic regeneration. Suppose a GIS user wants to determine whether a correlation exists between household income and the amount of money a city spends on storm water drainage in various neighborhoods. The user selects a region of interest on the map displayed and invokes the correlation computation by choosing items and variables of interest from menus or directly from displays. The user then creates an annotation to record his or her motivation for this calculation. A few days later, while preparing a report, the user needs

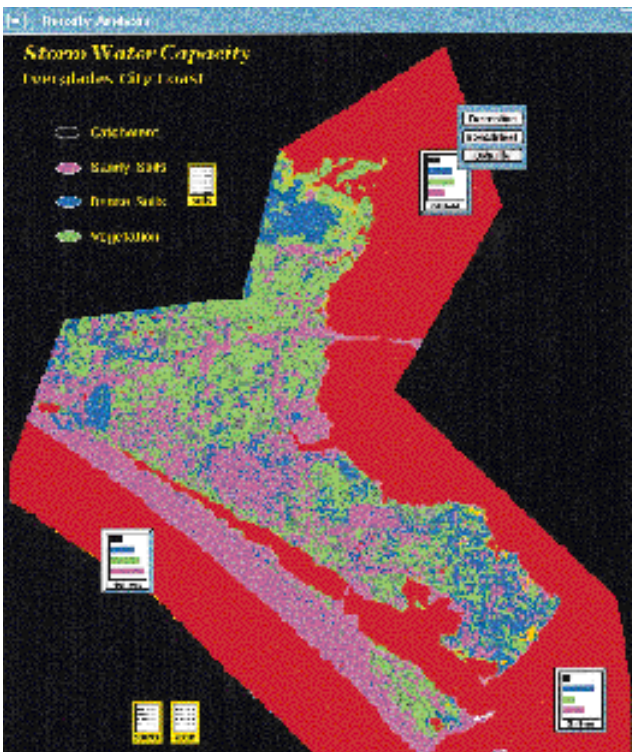


Figure 3. This density graphic represents the Everglades City coastal area. The three bar graphs serve two purposes: first, to provide graphical information about storm water capacity (in thousands of gallons), and second, to act as link markers. A comment (annotation) link marker in the legend provides an additional explanation of the region's soils. The comment icons at the bottom document source and accuracy information for the entire graph's data. The user has activated the link marker associated with the upper right bar graph, causing a menu to appear. The menu offers the user a choice of links to (1) a written description of the bar graph data, (2) a spreadsheet containing the bar graph data, or (3) the ASCII file containing the bar graph data.

the calculation result and annotation. This time the user wishes to return to them without first finding the original map, displaying it, and respecifying the same exact calculation parameters. Thus, the hypertext module should have a way to call for the calculation result to be regenerated automatically.[3] As noted later, the hypertext module has the further challenge of relocating previously declared annotation endpoints in regenerated documents, especially when the contents change between generations.

Navigation Functionality

Navigation transports the user among locations (e.g., analyses, maps, data sets) in a hypertext network. Navigation features include browsing, backtracking, component-based query and structural query.

Browsing allows users to select and traverse a link. In computational applications, traversing an operational link such as Figure 1's district analysis causes the computational module to generate (or refresh) the endpoint. Browsing updates the system's session log, which the hypertext engine uses for backtracking. Session log entries can include the link, the departure and destination nodes, anchors, and contextual settings such as active filters. Entry values may consist of hypertext and computational module identifiers, parameter values to instantiate procedure calls, and any other information necessary to regenerate the departure endpoint when backtracking (see footnote 3).

Backtracking enables users to retrace their steps—for example, to perform a subtask or follow an interesting tangent or detour and easily return to its starting point within the main analysis. Backtracking employs the session log to return the user to a previously visited node, but in its current state. Suppose the storm water analysis discussed under dynamic regeneration is a subtask within a much larger project requiring separate calculations for each area in the geographic region. Calculations involve different levels of accuracy or granularity. The user begins with a high-level view from a regional water management perspective, analyzing spatial data having an accuracy of 1:24000 (e.g., from the U. S. Census Bureau's TIGER file[4] data, which provides roadway, property, census, and other information). The user drills down to increasingly finer levels of accuracy to reach the city storm water drainage map with accuracy 1:15. Each progression is accompanied by calculation, traversal to results and, where appropriate, annotation. Although this scenario sounds similar to navigating a document-based hypertext, it differs in that navigation results from computation. Nodes are computed dynamically with content being a factor of context, display and the link selected. Backtracking now allows the user to review the calculations—all now complete—at each level of accuracy, from the finest to coarsest, before continuing to the next subtask.

It is important to differentiate between *backtracking* and *undoing*. As illustrated, backtracking returns the user to a previous location, restoring the context and preserving the state of any objects created during the session. For example, SPI's Navigator preserves and restores context for single-step backtracking as well as multi-step *backjumping* (see Thüring et al. in this issue). In contrast, an *undo* operation preserves context but does not preserve objects created, such as annotations. In the previous example, backtracking enabled the analyst to view the completed calculations in the context of previous analysis stages. However, backtracking over operations that modify application objects can present challenges. In TEXPROS, backtracking over a "merge folders" command should continue to show the folders in their current merged state. Backtracking merely transfers the user to a prior context or viewpoint. Similarly, within a CAD system, backtracking to a node concerned with a prior state of the analysis over an operation that groups all parts of an aircraft would leave the aircraft composite grouped. Builders should consult Nielsen [16] for an insightful discussion of static backtracking, Garzotto et al. [8] regarding parametric and conditional backtracking, and Bieber and Wan [6] for backtracking within multi-window applications, in which users work on different tasks and subtasks simultaneously.

Component-based query and *structure-based query* can lead to a starting point for browsing a large information space or can produce a starting point for a pruned subset of application data. The queries the computational module normally provides could be augmented with hypertext functionality (e.g., allowing users to select elements of a query specification and request metainformation or even a guided tour before determining how to complete it). In addition, users could query the hypertext metainformation base (e.g., display all comments containing the text "drainage"). Hypertext also can augment a computational application with structure-based query [10], combining application components with hypertext structure. For example, a new analyst may wish to regenerate all storm water analyses with comments by a predecessor or all analyses with at least two nonstructural links labeled with semantic link type "exception."

In a computational environment, both component-based and structure-based query may cause the system to generate and test uninstantiated applica-

[3] We provide for this in one of our current non-GIS prototypes [4]. The computational module's builder declares a procedure call for regenerating each structural node type, which the hypertext module can pass to the computational module for recomputation after instantiating it with an object identifier and parameters, for example, from the session log.

[4] Topologically Integrated Geographic Encoding and Referencing files

tion components to see if they meet the search criteria, which can be time-consuming.

Annotation Functionality

Annotation enables a user to add to the hypertext structure of a computational application beyond the links that the hypertext module infers from the application structure. Users and developers can declare nonstructural links to represent uninferable or *ad hoc* links. *Comments* can record personal notes and analyses, which provide metainformation to the author or other users. *Bookmarks* are "one-way" links similar to Mosaic's "hot list." They identify components that users wish to access at any time, from any location in the application. Users select a bookmark to traverse to its associated endpoint. Users could make computational annotations, associating a computational analysis with a nonstructured link's, bookmark's, or comment's operation type. Alternatively, noncomputational annotations could lead to computational endpoints which are invoked upon arrival. An analyst, for example, may associate an additional graph to answer a particular manager's question. Traversal could require users to input parameters or could use parameter values stored while creating the annotation.

Users could link multiple analyses using the same parameter values for each, such as a storm water analysis and a beach erosion analysis for the same stretch of coastline and season. A bookmark holding these parameter values could point to the paired analyses, or a nonstructural link could connect them if they do not share a predefined structural relation. Ambitious builders could allow users to annotate metainformation such as object classes (a comment on regression analyses in general or on any data values used by a particular analysis) or hypertext constructs (e.g., a comment about a particular comment or computational link).

Builders may decide to differentiate between annotating a specific instance of an object (e.g., commenting on a data value within a particular calculation or on a particular district's, storm water analysis) and annotating that object in general (e.g., when the comment should be associated with any appearance of that data value in any document, map, data set—existing or generated—or with all district storm water analyses).

Annotations raise several other issues for computational environments. Following a nonstructural link or bookmark can require the computational module to regenerate its endpoint or to switch contexts. When converting a message from the computational module to the interface module, the hypertext module must find all possible annotations in addition to the structural nodes and anchors. Finding annotated components that are not otherwise identifiable as anchors (e.g., a plain text string selected by the user and commented on) can be challenging within regenerated documents. If the document does not change substantially between generations, then the anchor will at least be in the same position or have the same display value. (Davis' sidebar suggests several approaches to this problem.) In response, builders may decide to restrict users to annotating only structurally declared objects, which the hypertext module can locate relatively easily.

View-Oriented and Structure-Oriented Functionality

Overviews, paths and guided tours provide alternative ways of viewing an application and navigating within it.

Overviews graphically display application components as anchors, with interconnecting lines representing their interrelationships (see Thüring et al. in this issue). Alternatively, an overview may take the form of a spatially coherent display such as a map or the set-based representations Marshall and Shipman (in this issue) discuss. In computational applications, the overview portrays and provides access to computations in addition to static nodes. Users not only see the global context but can invoke a computation by selecting its anchor. For example, Figure 1 depicts a high-level overview providing access to four hypothetical South Florida storm water management districts and their interregional analyses.

Unfortunately, overviews often cannot depict adequately all interdocument and interdata structural relationships, as well as all possible computations. Spatial data sets, for example, may contain thousands of relationships, many of which exist only for certain view settings and selection criteria. (This in fact presents a formidable problem in the spatial data community.) Presenting all relationships would overwhelm a user. Thus, developers should consider pruning or allowing users to restrict the link types included. Variations on overviews could represent a specific task or process, or only the regions inferred to be relevant.

Hypertext can assist the GIS in forming abstractions of an overview's relationships and, more importantly, can depict these abstractions as link markers on a display without obscuring its content. Doing this automatically requires a spatial analysis of the overview's contents. Consider Figure 3, presenting a density analysis of a coastal area along the western region of the South Florida district. Storm water capacities appear in different colors and densities. Superimposed are three subanalyses (small bar graphs) showing the average rate of flow. At the bottom of the graphic are two annotation icons. The bar graphs are generated by the GIS computational module, but used by the hypertext module as link markers presenting encapsulated analysis information to the user. (Contrast these with

current link marker designs, which often convey little additional information.) Each link could provide access to information such as the full bar graph diagram, the raw data used in the analysis, descriptions of the data collection methods and frequency of collection periods, and a finer analysis of storm water flow by street, among other kinds.

As with structure-based query, much of the potential network often does not exist when generating overviews based upon application structure. Consider Figure 1's links to each district's storm water analysis. Each icon represents calculations and their explanations that may not have been computed yet. When the user navigates from any icon, the computational module must perform the analysis (perhaps rendered as different colored regions) and subanalyses (summarized as bar graphs) before generating the display.

Paths and *guided tours* provide a recommended "trail" through the hypertext network [22]. Paths permit users to leave the trail and return to it, while guided tours restrict users to the trail's anchors, links and annotations. Trails can guide novices or trainees, functioning as a hands-on demonstration or training tool for performing analyses. An annotated trail of all steps in performing a particular analysis can document the analysis process and justify decisions made [3, 5]. Banker et al. generate guided tours of software components matching specific criteria, which programmers then browse for reuse [2]. Isakowitz et al. discuss automatically generating guided tours as part of the RMM design methodology they describe in this issue. We also encourage builders to consult [8], in which Garzotto et al. discuss many sophisticated forms of guided tour navigation.

Consider a realtor using a hypertext-based GIS to create a video driving tour of homes. Links connect points on an electronic map to an on-line multiple listings service (MLS) and video clips of each home. The GIS uses correlation analysis to compute an ordering of the links forming a customized tour for a prospective client. During the tour, the client uses navigational and query facilities within the GIS to find out more information about the area surrounding each home (e.g., average family income, property size, home size, roadways, schools), browses the MLS listing for information on property taxes and average utility costs, and watches videos of the homes of greatest interest. Users can mark the homes to visit, allowing the application to compute a driving map that minimizes the travel distance and amount of time needed to tour the homes.

Computational Access Permissions

Static multi-user hypertext systems require builders to implement an array of new access permissions for annotations, such as create, read, modify, delete and further annotate [25]. For example, a public policy firm may permit only official comments on reports to government agencies. Computational applications require additional access permissions. The public policy firm may allow readers to reexecute an analysis, but not perform what-if analysis by altering input parameters. A consulting firm may allow its analysts—but not its clients—to see the equations within proprietary computational models. Consider also a GIS application that renders results of a household income and voting habits survey by geographic region. Confidentiality would prevent all but the analysts loading survey data to drill down below the granularity of a neighborhood summary to summary data for individual streets or detailed data from individual households. Turoff has developed 25 kinds of access permissions (as well as tickets to override them) for computer-mediated communications, many of which would apply to distributed, multi-user computational hypertext applications [23].

Fortunately, builders do not have to program all hypertext functionality from scratch. Core models of hypertext such as the Dexter model [11] provide data structures and guidelines for constructing nodes, links, and anchors and for browsing. Several component systems and architectures can provide a foundation for computational applications. Builders can use *hyperbase* storage engines, such as HB3 [14] and that in the Hyperform development environment (see Wiil, this issue), as a hypertext database for storing nodes, links, and anchors. Hypertext engines, such as Microcosm [7], Multicard [18] and Bieber's [4], can be integrated with applications to provide hypertext support. Application components can be declared, scheduled, and activated using the Trellis interactive development environment (see Furuta and Stotts in this issue). Builders can employ toolkits such as HOT [17] and Andrew [21], providing libraries of hypertext function calls. Builders also could use bridge laws [3, 4, 24] to describe the internal structures of computational objects (e.g., an application's schema) and automatically identify links among them. For computational applications, builders can refer to the work of Schnase and Leggett [20] as an example of computation in a hypertext domain (in which scripts embedded in the hypertext application invoke simulation programs) and that of Bieber et al. [3, 5, 24] as examples of hypertext in computational domains.

Closing Observations

The hypertext field contains woefully little research and few tools for supplementing computational information systems with hypertext functionality. Yet this is where we believe hypertext will have its greatest impact, especially for the scientific and business worlds. The rate and complexity of data changes within a computational environment, as well as the sheer quantity of application elements, complicates the process of creating and identifying links among objects and documents. The traditional

display-oriented methods of identifying and establishing hypertext relationships are impractical. Dynamic and adaptable techniques based on structural recognition can enable builders to augment computational applications with hypertext navigation, annotation, view-oriented, and structure-oriented features. When applied to a computational domain under a philosophy of "maximum access," hypertext functionality helps users develop a greater understanding of, confidence in, and acceptance of analytical applications.

Acknowledgments

The authors wish to thank Troy Ames, Karl Mueller, and Walt Truszkowski of NASA's Goddard Space Flight Center for their support of this research. Stephen Hodge and Dean Jue of the Florida Resources and Environmental Analysis Center at Florida State University provided instrumental advice on the hypothetical GIS we portray in this article. Thanks also to Jörg Haake, Tomás Isakowitz, Paul Kahn, and Manfred Thüring, all of whose feedback was invaluable in refining this article. Both authors contributed equally to this article.

References

1. Akscyn, R., McCracken, D., and Yoder, E. KMS: A distributed hypermedia system for managing knowledge in organizations. *Commun, ACM 31* 7 (July 1988), 820–835.
2. Banker, R., Isakowitz, T., Kauffman, R., Kumar, R,. and Zweig, D. Tools for managing repository objects. In *Software Engineering Economics and Declining Budgets.* T. Geriner, T. Gulledge, and W. Hutzler Eds. Springer-Verlag, New York 1993, 125–146.
3. Bieber, M. Automating hypermedia for decision support. *Hypermedia 4* 2 (1992) 83–110.
4. Bieber, M. On integrating hypermedia into decision support and other information systems. *Decision Support Systems.* North-Holland (forthcoming).
6. Bieber, M., and Wan, J. Backtracking in a multiple window hypertext environment. In *European Conference on Hypermedia Technologies 1994 Proceedings.* (Edinburgh, Sept. 1994) 158–166.
5. Bieber, M., and Kimbrough, S.O. On generalizing the concept of hypertext. *Manage. Info. Syst. Q., 16* 1 (1992), 77–93.
7. Davis, H., Knight, S., and Hall, W. Light hypermedia link services: A study of third-party application integration. In *European Conference on Hypermedia Technologies 1994 Proceedings.* (Edinburgh, Sept. 1994), 41–50.
8. Garzotto, F., Mainetti, L., and Paolini, P. Navigation in hypermedia applications: Modelling and semantics. *J. Organ. Comput.* 1995 (forthcoming).
9. Garzotto, F., Paolini, P., and Schwabe, D. HDM: A model-based approach to hypertext application design. *ACM Trans. Info. Syst. 11,* 1 (Jan. 1993), 1–26.
10. Halasz, F. Reflections on NoteCards: Seven issues for the next generation of hypermedia systems. *Commun. ACM 31,* 7 (July 1988),836–855.
11. Halasz, F., and Schwartz, M. The Dexter hypertext reference model. *Commun. ACM 37* 2 (Feb 1994), 30–39.
12. Kacmar, C. Supporting hypermedia services in the user interface. *Hypermedia 5,* 2 (1993), 85–101.
13. Kacmar, C. and Leggett, J. Proxy: A process-oriented extensible hypertext architecture. *ACM Trans. Info. Syst . 9,* 4 (1991), 399-419.
14. Leggett, J. and Schnase, J. Viewing Dexter with open eyes. *Commun. ACM 37,* 2 (Feb. 1994), 77–86.
15. Mueller, K. and Truszkowski, W. Reuse in the large: Utilizing the evolutionary domain life cycle model with a knowledge-based approach. In *Research and Technology Goddard Space Flight Center.* NASA Goddard Space Flight Center Tech. Rep., 150–152.
16. Nielsen, J. The art of navigating through hypertext. *Commun. ACM 33,* 3 (March 1990), 296–310.
17. Puttress, J., and Guimaraes, N. The toolkit approach to hypermedia. In *Hypertext: Concepts, Systems and Applications, Proceedings of the European Conference on Hypertext.* (Nov. 1990) A. Rizk, N. Streitz, and J. André, Eds. Cambridge University Press, Cambridge.
18. Rizk, A. and Sauter, L. Multicard: An open hypermedia system. In *Proceedings of the ACM Conference on Hypertext* (Milan, Nov. 1992) 4–10.
19. Salton, G., Alan, J. and Buckley, C. Automatic structuring and retrieval of large text files. *Commun ACM 37,* 2 (Feb. 1994), 97-108.
20. Schnase, J., and Leggett, J. Computational hypertext in biological modelling. In *Hypertext'89 Proceedings.* (Pittsburgh, Nov. 1989) 181–198.
21. Sherman, M., Hansen, W., McInerny, M., and Neuendorfer, T. Building hypertext on a multimedia toolkit: An overview of Andrew toolkit hypermedia facilities. In *Hypertext: Concepts, Systems and Applications, Proceedings of the European Conference on Hypertext.* (Nov. 1990) A Rizk, N. Streitz, and J. André, Eds. Cambridge University Press, Cambridge. 13–24.
22. Trigg, R. Guided tours and tabletops: Tools for communicating in a hypertext environment. *ACM Trans. Off. Info. Syst. 6,* 4 (Oct. 1988), 398–414.
23. Turoff, M. Computer-mediated communication requirements for group support. *J. Organ. Comput. 1,* 1 (1991) 85–113.
24. Wan, J., Bieber, M., Wang, J. and Ng, P. Document management through hypertext: A logic modeling approach. In *Proceedings of the 27 Annual Hawaii International Conference on System Science.* (Maui, HI, Jan. 1994), Vol. III, 558–568..
25. Yankelovich, N., Haan, B. Meyrowitz, N. and Drucker, S. Intermedia: The concept and the construction of a seamless information environment. *IEEE Computer 2,* 1 (Jan. 1988) 81–96.

About the Authors:

MICHAEL BIEBER is assistant professor of Information Systems at the New Jersey Institute of Technology's CIS Department, where he directs the Hypermedia Information Systems Lab. He is active in the ACM Hypertext conferences and co-chairs the Hypermedia in Information Systems minitrack at the annual Hawaii International Conference on System Sciences. He directed the development of ACM SIGLINK's LINKBase system. **Author's Present Address:** Institute of Integrated Systems Research, New Jersey Institute of Technology, University Heights, Newark, NJ 07102-1982 email: bieber@cis.njit.edu. URL: http//hertz.njit.edu/~bieber/bieber.html.

CHARLES KACMAR is an assistant professor in the Department of Computer Science at Florida State University, Tallahassee. His current research interests include digital libraries, hypertext/media, human-computer interaction, and cooperative systems. **Author's Present Address:** Department of Computer Science, Florida State University, 203 Love Building, Mail Stop 4019, Tallahassee, FL 32306-4019 email: kacmar@cs.fsu.edu.

This research is supported in part, by the New Jersey Institute of Technology under Grant #421200, a grant from the AT&T Foundation, and the NASA Jove faculty fellowship program.