Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems

Frank G. Halasz*

Microelectronics and Computer Technology Corp. (MCC)
3500 West Balcones Center Dr.
Austin, TX 78759-6509

ABSTRACT

NoteCards is a general hypermedia environment designed to help people work with ideas. Its intended users are authors, designers, and other intellectual laborers engaged in analyzing information, designing artifacts, and generally processing ideas. The system provides these users with a variety of hypermedia-based tools for collecting, representing, managing, interrelating, and communicating ideas.

This paper presents the NoteCards system as a foil against which to explore some of the major limitations of the current generation of hypermedia systems. In doing so, this paper highlights seven of the major issues that must be addressed in the next generation of hypermedia systems. These seven issues are: search and query, composite nodes, virtual structures, computational engines, versioning, collaborative work, and tailorability. For each of these issues, the papers describes the limitations inherent in NoteCards and the prospects for doing improving the situation in future systems.

INTRODUCTION

NoteCards is a general hypermedia environment designed to help people work with ideas. Its intended users are authors, researchers, designers, and other intellectual laborers engaged in analyzing information, constructing models, formulating arguments, designing artifacts, and generally processing ideas. The system provides these users with a variety of hypermedia-based tools for collecting, representing, managing, interrelating, and communicating ideas.

NoteCards is in many ways typical of the generation of workstation-based hypermedia systems that is currently moving from the research lab into widespread use (e.g., Intermedia [Garr86b,Meyr86] and Neptune [Deli86a]). These systems are proving to be extremely useful in application domains ranging from educational courseware through computer-aided engineering to legal argumentation. At the same time, as these systems move out of the lab into the real-world, their limitations and design flaws are becoming increasingly apparent.

---

* I would like to thank Randy Trigg and Tom Moran of Xerox PARC and Jeff Conklin, Michael Begeman, and Eric Gullichsen of MCC. The ideas presented in this paper were developed during many discussions with these folks over the last year. I would like to especially thank Michael Begeman for pointing out the importance of what I've called virtual structures (and what he calls views).

This paper presents the NoteCards system as a foil against which to explore some of the major limitations of this current generation of hypermedia systems. In doing so, this paper will highlight some of the major issues that must be addressed in designing the next generation of hypermedia systems.

NOTECARDS IN BRIEF

NoteCards was designed to support the task of transforming a chaotic collection of unrelated ideas to an integrated, orderly interpretation of the ideas and their interconnections. Analyzing one's business competitors is a prototypical example. The task begins with the analyst extracting scraps of information about competitors from available sources. The collected information must be organized and filed away for subsequent use. More importantly, the information needs to be analyzed, i.e., the relationships between the various ideas have to be discovered and represented. Once these analyses are complete, the analyst composes and writes a document or presentation that communicates the discovered information and its significance.

NoteCards provides the user with a 'semantic' network of electronic notecards interconnected by typed links. This network serves as a medium in which the user can represent collections of related ideas. It also functions as a structure for organizing, storing, and retrieving information. The system provides the user with tools for displaying, modifying, manipulating, and navigating through this network. It also includes a set of methods and protocols for creating programs to manipulate the information in the network.

NoteCards was developed by Randall Trigg, Thomas Moran and the present author at Xerox PARC. A more detailed discussion of the system, its design goals, and our experiences with its use can be found in [Hala87].

Four basic constructs

NoteCards is implemented within the Xerox Lisp programming environment. The system is designed around two primitive constructs, *notecards* and *links*. In the basic system, these two primitive are augmented by two specialized types of cards, *browsers* and *fileboxes*, that help the user to manage large networks of cards and links.

Notecards. A notecard is an electronic generalization of the 3x5 paper notecard. Each notecard contains an arbitrary amount of some editable substance such as a piece of text, a structured drawing, or a bitmap image. Each card also has a title. On the screen, cards are displayed using standard Xerox Lisp windows as shown in Figure 1.

Every notecard can be 'edited', i.e., retrieved from the database and displayed on the screen in an editor window that provides the user with an opportunity to modify the card's substance. There are various types of notecards, differentiated (in part) by the nature of the substance (e.g., text or graphics) that they contain. In addition to a set of standard card types, NoteCards includes a facility for adding new card types, ranging from small modifications to existing card types (e.g., text–based forms) to cards based on entirely different substances (e.g., animation cards).

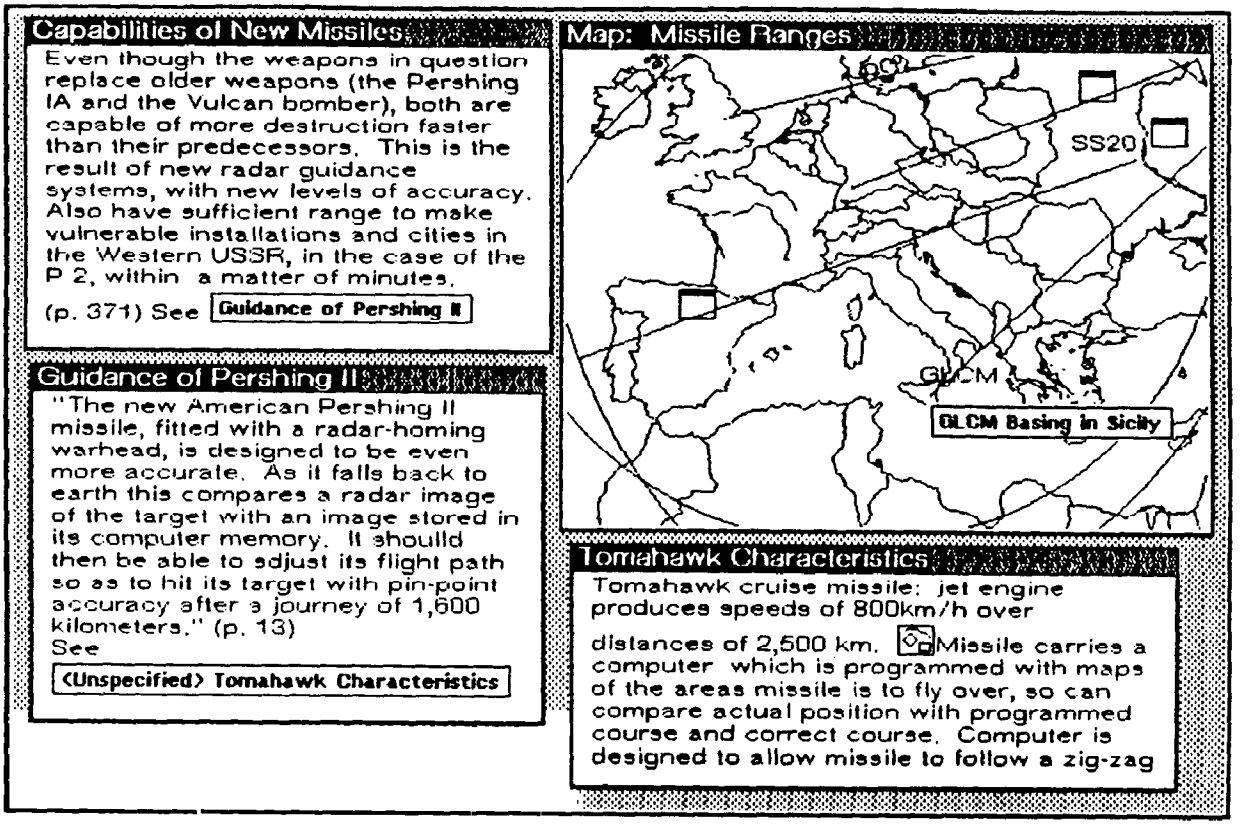

Figure 1: Example notecards with embedded link icons.

Links. Links are used to interconnect individual notecards into networks or structures of related cards. Each link is a typed, directional connection between a source card and a destination card. The type of a link is a user-chosen label specifying the nature of the relationship being represented.

The links are anchored at a particular location in the substance of their source card by a link icon but point to their destination card as a whole. Clicking in the link icon with the mouse traverses the link, i.e., retrieves the destination card and displays it on the screen. In Figure 1, each of the two cards contains two link icons.

Browsers. A browser is a notecard that contains a structural diagram of a network of notecards. Figure 2 shows a Browser card for a network composed of 8 cards and 8 links. The cards from this network are represented in the browser by their title displayed in a box. The links in the network are represented by edges between the boxed titles. Different dashing styles distinguish different types of links.

The diagrams in Browser cards are computed for the user by the system. Once created, browsers function like standard notecards. The boxed titles in the browser are in fact icons for traversable links between the browser and the referenced card.

Browsers support two levels of editing. First, the user can edit the underlying structure of a network of notecards by carrying out operations on the nodes and edges in the browser. Second,

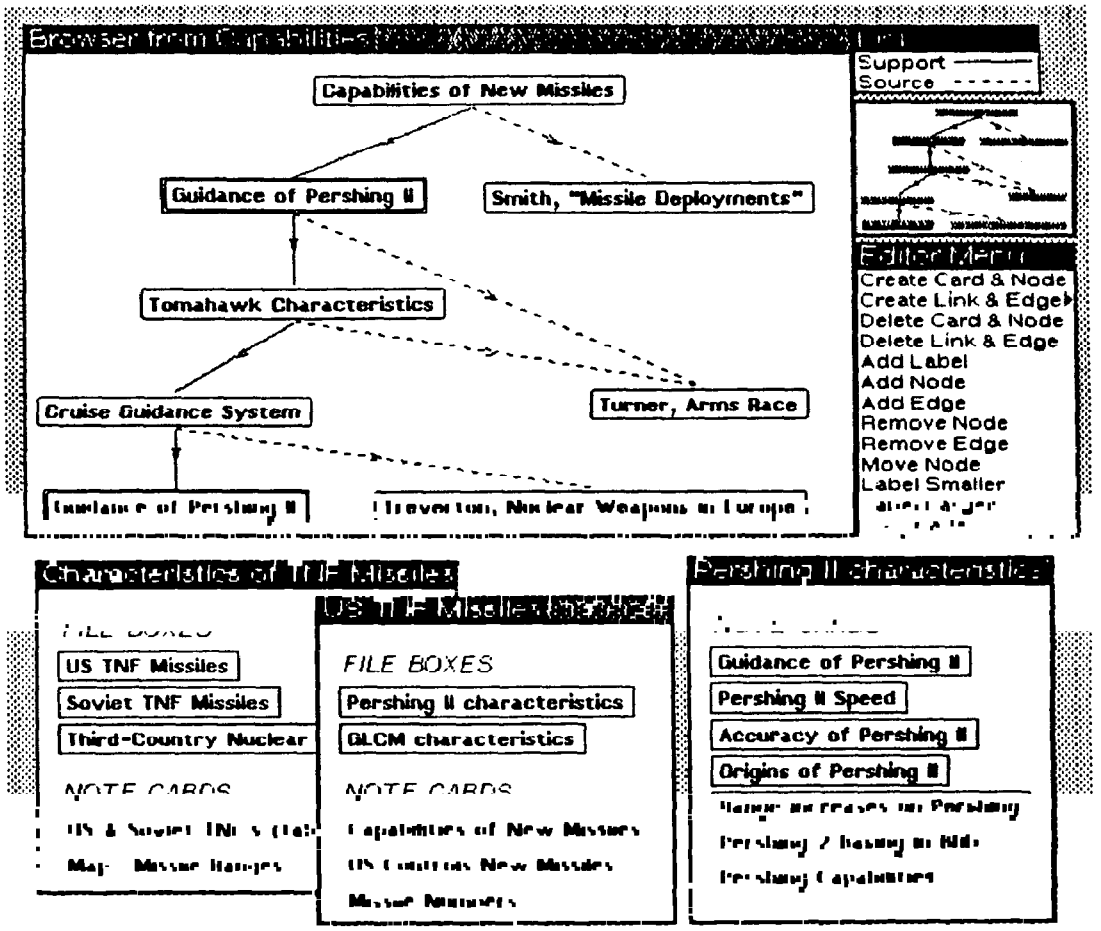

Figure 2: Example browser card (top) and filebox cards.

the user can add and delete nodes and edges in the browser diagram without making corresponding changes to the underlying NoteCards structures.

Fileboxes. Fileboxes are specialized cards that can be used to organize or categorize large collections of notecards. A filebox is a card in which other cards, including other fileboxes, can be filed. NoteCards requires that every notecard (including fileboxes) must be filed in one or more fileboxes. Figure 2 shows 3 fileboxes in addition to the browser.

Fileboxes were designed to help users manage large networks of interlinked notecards by encouraging them to use an additional hierarchical category structure for storage and retrieval purposes.

Interacting with NoteCards

Accessing information. Navigation is the primary means for accessing information in NoteCards. The user moves through the network by following links from card to card. Alternatively, the user can create an overview Browser for some sub-network and traverse the links from the Browser to the referenced cards. NoteCards also provides a limited search facility that can locate all cards matching some user-supplied specification (e.g., a particular string in the card's title or text).

User interface. The NoteCards user interface is mouse- and menu-based. Operations are initiated either by direct manipulation or by choosing commands from menus associated with the various icons and windows on the screen. Whenever possible specification of objects is done by the user pointing to the referent on the screen. Beyond these generalities, the user interface incorporates a diversity of specific interaction styles due to the great variation in user interface styles among the many preexisting Lisp packages incorporated into the system.

Tailorability

NoteCards is fully integrated into the Xerox Lisp programming environment. It includes a widely used programmer's interface with over 100 Lisp functions that allow the user to create new types of cards, develop programs that monitor or process a network, integrate Lisp programs (e.g., an animation editor) into the NoteCards environment, and/or integrate NoteCards into another Lisp-based environment (e.g., an expert system).

There is some degree of (non-programming) user tailorability in NoteCards as well. The system includes a large set of parameters that users can set to tune the exact behavior of the system (e.g., how links are displayed or the default size of notecards). In addition, users often create template cards or structures of cards that can be copied to create instances of the template. See **[Trig87]** for further details about tailorability in NoteCards.

NoteCards in use

From its inception, the design and development of NoteCards has been driven by the needs of its user community. Currently there are over 70 'registered' users within Xerox. At least 25 of these are 'everyday' users. There are, in addition, an undetermined number of users at various university, government, and industrial sites outside Xerox. This user community has provided invaluable feedback on the strengths and weaknesses of NoteCards as applied to a variety of tasks including document authoring, legal, scientific, and public-policy argumentation, design and development of instructional materials, design and analysis of copier parts, and competitive market analysis. Perhaps the most common use of NoteCards is a database for storing miscellaneous personal information.

A typical example of how the system can be used to support idea structuring and generic authoring is the network created by a history graduate student who used the system to research and write a 25-page paper. Figure 3 shows a browser of the filebox hierarchy created during this project. The author made a habit of keeping this browser on his screen at all times as a way of speeding up the process of filing and accessing cards. This hierarchy was made up of 40 fileboxes and contained 268 (non-filebox) cards.

The cards in Figure 1 are taken from this hierarchy. In general, cards stored in the hierarchy contain a short (average of about 100 words) quote or paraphrase taken from an article or book. About half of the cards have links embedded in their substance. As a rule, these were 'See' or 'Unspecified' links and were placed at the end of the card's text preceded by the word 'See' There are also a few dozen inter-card links of other types.

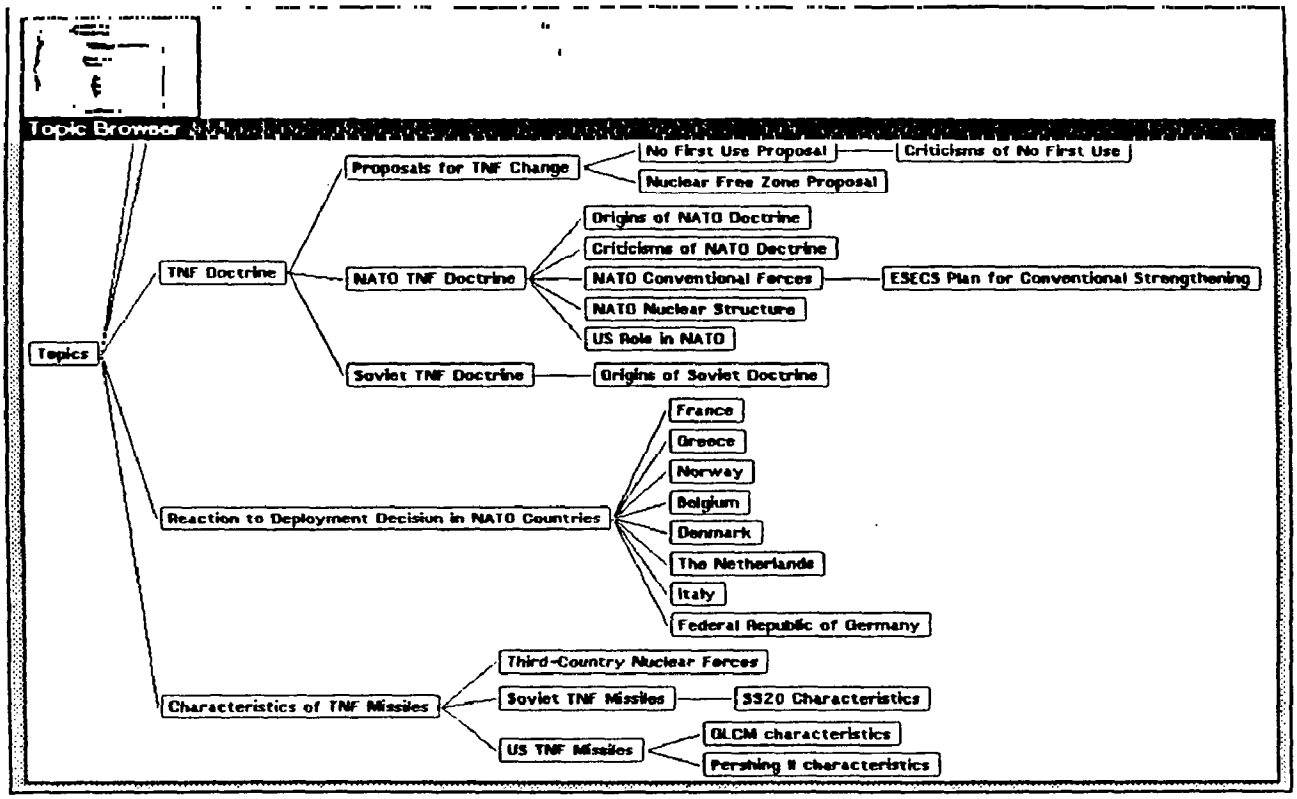

Figure 3: Browser of the filebox hierarchy from the public policy (nato-missiles) notefile.

Further description of this network and the process involved in using to author a paper can be found in [Mont86a] and [Hala87].

NoteCards' niche in the space of hypermedia systems

The term 'hypermedia' has been used to describe a wide variety of different systems and concepts ranging from outline processors through artificial video worlds. In this paper, 'hypermedia' will be used more narrowly to refer to information representation and management systems that organize information into networks of multi-media nodes interconnected by links. Each node generally contains a large chunk of 'content' such as a document, a drawing, or a voice annotation. The links are used to represent interrelations among these nodes.

Even with this restricted definition, there is wide variety in the nature and function of existing systems. NoteCards' niche in this varied space of hypermedia systems can be clarified by partitioning the space along three fundamental dimensions: *scope*, *browsing vs. authoring*, and *target task domain*.

Scope: Hypermedia has been proposed as *the* mechanism for storing and distributing the world's entire literary output [Nels81]. Hypermedia has been proposed as an common information space for teams of programmer's on large software projects [Garg87]. Hypermedia has been proposed as a tool for individuals and small work groups engaged in authoring and idea processing [Hala87]. Although all of these proposals share the notion that information should be organized

into network of nodes and links, they differ radically in *scale*, e.g., in the sizes of their expected information bases and user populations. This extreme variation in scale implies there will be differences throughout these systems, ranging from underlying storage mechanisms through the user interface to conventions for their use.

Browsing versus Authoring: In systems designed primarily for browsing, the hypermedia network is carefully created by a relatively small number of specialized authors in order to provide an information space to be explored by a large number of more or less casual users. These browsing systems are generally characterized by relatively well-developed tools for information presentation and exploratory browsing. Tools for creating and modifying the network tend to be less evolved. Hypermedia instructional delivery systems **[Garr86b]** and interactive museum exhibits **[Shne87]**.

In systems designed primarily for authoring, the hypermedia network serves as a information structure that users create and continuously modify as part of their ongoing task activities. Hypermedia systems for idea processing **[Hala87]**, document authoring **[Smit86]**, and software development **[Deli86a]** are primary examples. In such systems, the tools for creation and modification of the network are well-developed. Tools for easy browsing and sophisticated information display tend to be less evolved.

Target task domain: Many hypermedia systems have been designed to support a specific task. For example, WE **[Smi86]** is an environment designed specifically to support the professional writer. Other hypermedia systems are designed to provide general hypermedia facilities to be used in a variety of applications. Even these generic systems, however, are usually designed with a target task domain in mind. The features and capabilities emphasized in the system often reflect the requirements of this target. Contrast, for example, Intermedia **[Garr86b]** and Neptune **[Deli86a]**. Both are general hypermedia systems. But Neptune was designed to support software engineering and thus emphasizes versioning **[Deli86b]** and node/link attributes. In contrast, Intermedia was designed for multi-user interactive educational applications and thus emphasizes novel interactive displays **[Garr86a]** and 'annotation' facilities.

NoteCards is designed for use by individuals or small work groups. In this respect, NoteCards is similar to PC-based systems like Guide **[Guid86]** and to single-user workstation systems like WE **[Smit86]**. Conversely, NoteCards differs significantly along this dimension from 'global' systems like Xanadu **[Nels81]**. as well as systems designed to support larger groups such as ZOG **[Robe81]** and NLS/Augment **[Engl68]**. Although its original design places less-emphasis on multi-user access, NoteCards is very similar in scope to Neptune **[Deli86a]**, Intermedia **[Meyr86,Garr86b]** and Textnet **[Trig83]**.

NoteCards is first and foremost an authoring system designed to provide its users with facilities for creating and modifying hypermedia structures. In this respect, NoteCards is similar to many of the aforementioned systems (i.e., Augment, Guide, Intermedia, Neptune, WE) and different from online presentation systems such as TIES **[Shne86]** and interfaces to CD/ROM databases **[Lamb86]**. ZOG has a slightly different flavor, being simultaneously browser- and author-oriented.

NoteCards was originally designed as a tool to support idea processing and authoring in a research environment. Its original goals were very similar to those of NLS/Augment, although the actual implementations of the two systems are (on the surface) quite different. Systems such as WE and Guide were also designed to support authoring tasks. Intermedia (education) and Neptune (software engineering) were designed with very different application domains in mind, although both systems were designed in part to support document authoring tasks.

Overall, NoteCards is most similar to Intermedia and Neptune despite the differences in their target application domains. The three are very similar in scope and in the type of facilities they provide. This similarity is reinforced by several factors including a common emphasis on extensibility, similar underlying platforms (i.e., workstations), and a contemporaneous development schedule. Although the present paper specifically discusses NoteCards, most of the issues raised are equally relevant to these two systems as well as to most of the current generation of hypermedia systems.

SEVEN ISSUES FOR THE NEXT GENERATION OF HYPERMEDIA SYSTEMS

In the three years since its first release, we have been able to observe NoteCards in use and misuse in a wide range of situations and applications. These observations have provided significant insight into the system's particular strengths as well as its weak points. A brief but balanced assessment of the system is contained in **[Hala87]**. In contrast, for expository reasons the present paper focuses only on NoteCards shortcomings, i.e., on the ways in which the system falls short in meeting the needs and preferences of its users.

Many of NoteCards' shortcomings are specific to its current implementation and could be corrected by limited redesign or reimplementation of the existing system. However, some of NoteCards' shortcomings reflect fundamental weaknesses in the hypermedia model on which the system is built. It is precisely these fundamental shortcomings and the mechanisms for their correction that will form the basis for designing the next generation of hypermedia systems beyond NoteCards, Neptune, Intermedia, and their cohorts. The following sections describe seven such limitations that have become evident through our observations of the NoteCards user community. Each of these limitations raises a variety of issues for the design of future hypermedia systems.

Issue 1: Search and query in a hypermedia network

The primary method for accessing information in NoteCards is 'navigation' through the network by following links from card to card. Typically, the NoteCards user brings a card onto the screen, examines its content and links, and then traverses the link that is most likely to move closer to the target information. Fileboxes support such localized link traversal by providing a hierarchical structure in which information is located by recursive descent through an increasingly specific category structure. In addition, localized link following is augmented by browsers (and other user-invented overview displays) that provide global maps of the network. Such global overviews allow the user to visually scan for and then directly move to areas of the network in which the target information is likely to be found.

Navigational access to information has been adequate, and occasionally even ideal, for a large number of NoteCards applications. These applications can be divided into three basic classes. First, the navigational access has proven sufficient for the small authoring, note taking, and informal knowledge representation tasks that NoteCards was originally designed to support. In these tasks, an individual or a small (2 to 3 person) workgroup is creating and intensively using a relatively small network (50 to 250 cards). Because the network is small and familiar, users have little problem locating information.

A second class of 'navigational' applications are the display-oriented representation tasks in which the network is centered around a single display, usually a browser, used to represent a structure being designed or studied. The goal of these tasks is to create and manipulate this display. In some sense, the network is secondary to the display and is used only to create the structure to be displayed and to "hide" unimportant details. In these tasks, information access occurs through the central display, with little direct card-to-card navigation. An example of such a network is described in **[VanL85]**.

The third class of navigationally-oriented applications is online interactive presentations. In these applications, the network's author often includes in each card navigational instructions to be used by readers of the network. Such 'guided' online presentations are discussed in **[Trig86]**. If no such navigational instructions are included, then the network is generally designed to be explored by the user in a non-directed manner.

In contrast to these navigationally-oriented applications, there are a variety of applications for which NoteCards' reliance on navigational access is problematic. These applications are generally characterized by large, unfamiliar, heterogeneously structured networks. Even in a 500 node single-user network, navigational access can be difficult as the network changes and its structure becomes heterogeneous. In these cases, navigational access is problematic because users tend to get lost while 'wandering around' in the network looking for some target information. Often these users can describe exactly what information they are looking for, but simply cannot find it in the network.

An incremental solution to the navigational problems encountered in NoteCards would be to improve and augment the existing navigation tools. For example, browsers could be made significantly more effective by applying techniques such as fish-eye views **[Furn86]** and graph 'fly-overs' **[Fair87]**. In addition, new tools such as a voting scheme similar to Synview **[Lowe86]** could be implemented in NoteCards. Although these changes would alleviate some of the navigation problems, they would not eliminate them entirely.

A more fundamental solution to the navigation problem is to augment navigation by a query-based access mechanism. With query-based access, the user could formulate a query encapsulating a description of the target information and then rely on the system to locate that information in the network. NoteCards already provides some limited query/search facilities. In particular, a user may search for all of the cards with some distinct property, e.g., that contain a particular text string. This search is extremely simple with no boolean expressions and no regular expressions provided. For NoteCards to be a useful in managing large heterogeneous networks, search and

query needs to be elevated to a primary access mechanism on par with navigation. Providing a search and query mechanism in the context of a hypermedia system is an interesting challenge.

There are two broad classes of query/search mechanisms needed in a hypermedia system. The first mechanism is *content search*. In content search, all cards and links in the network are considered as independent entities and are examined individually for those entities whose content or properties match a given query. For example, *all the cards containing the string 'hyper* system'* would be a content query. Content search is more or less standard information retrieval applied to a hypermedia information base. Although many techniques for such searches are well known, there are many innovative approaches that could be explored in a hypermedia environment.

Content search basically ignores the structure of a hypermedia network. In contrast, *structure search* examines the hypermedia network for sub–networks that match a given pattern. For example, a simple structure query might be: *all sub–networks containing two cards connected by a 'supports' links where the target card contains the word 'hypertext'*. This query contains a description of node content (i.e., *contains the word 'hypertext'*). It also contains a structural description of a sub–network (i.e, *two cards connected by a 'supports' link*). A more complicated structure query, involving an indefinite sequence of links, might be something like: *a circular structure containing a card that is indirectly linked to itself via an unbroken sequence of 'supports' links*. This query could be used, for example, to find circular arguments.

The development of a structural query facility is an interesting and difficult task. One major subtask is to design a query language geared towards describing hypermedia networks. This structure query language needs to be useable by the typical hypermedia user, who is unlikely to be facile with the intricacies of structure representation languages. A second major challenge is to design and implement an efficient structure search engine to processes these queries. A mechanism that searches a network for any sub–network matching some (possibly complex) pattern is difficult to implement and even more difficult to implement efficiently.

In addition to its role as a mechanism for locating information, search and query be critical to many other aspects of future hypermedia systems. For example, the search and query mechanism will be used as a filtering mechanism in the hypermedia interface. Users will specify a 'query' in order to describe the information of interest to them. The interface would then show only those aspects of the network that 'matched' this query. The NoteCards browser currently operates in this manner, but only with respect to a very limited set of structure queries. A full–blown query mechanism would allow much more interesting browsers to be constructed. More importantly, the search/query mechanism could be linked much deeper into the interface providing for a pervasive information filtering mechanism that is absent in NoteCards and its cohorts.

Search and query will also be a critical component underlying the virtual structures mechanism described in Issue 3 below. Thus, in many ways, the success of the next generation of systems will depend on a good solution for the problem of search and query in a hypermedia network.

Issue 2: Composites — augmenting the basic node and link model

There are only two primitive constructs in NoteCards, cards and links. All other mechanisms in the system, including fileboxes and browsers, are built out of these two constructs. Although, this design has been surprisingly successful, experience suggests that this basic hypermedia model is insufficient. In particular, the basic model lacks a composition mechanism, i.e., a way of representing and dealing with sets (or sub-networks) of nodes and links as unique entities separate from their components.

A typical use for composites can be seen in the task of writing an organized document (e.g., a technical report) in NoteCards (see [Trig87]). In this task, users typically put the text for each subsection and for each figure into a separate card. All of the cards for a single section are then filed in a filebox. These section fileboxes are filed in the appropriate chapter fileboxes, which in turn are filed in a single filebox representing the document. This scheme is workable. It allows the user to focus in on the text/graphics for a particular chapter, section, or subsection. Using the NoteCards document compiler, the user can linearize the network into a single document card containing all of the text and graphics for the document in the appropriate order but without any hierarchical structure. This document can then be manipulated, e.g., read or printed, as a single entity. There is a problem, however, in that the document card is a separate entity from the 'source' cards stored in the document's filebox hierarchy. It contains only copies of the text/graphics from these source cards. Changes made to the text/graphics in the document card are not (automatically) reflected in the corresponding source card.

More importantly, the user can see the entire document at only one level. Despite the elaborate filebox hierarchy, there is no way to 'zoom' in and out of the document structure, examining its contents at different levels of detail (e.g., just the chapters, or all the chapters with their subsection headings, etc). This capability is commonly found in outline processors and is a critical component in many writing and information organization tasks. As a result, a number of writers have abandoned NoteCards in favor of outline processors for their simple authoring tasks.

This example suggests that NoteCards is missing the critical notion of composition. In particular, the system lacks a mechanism by which collections of cards and links can be reified as single composite entity. If such a mechanism were available, the document filebox hierarchy could be replaced by a composition hierarchy. In this latter hierarchy, the document would be a linearly ordered composite of chapter cards. Each of these chapter cards would themselves be a linearly ordered composite of section cards, which in turn would be a linearly ordered collection of the source cards containing (finally) text/graphics. Each composite card in this hierarchy could be displayed showing varying amounts of 'detail' about its (direct and indirect) subcomponent cards. For example, the highest level composite, i.e., the document itself, could be viewed as a list of chapters or, alternatively, as a concatenation of all of the text/graphics in the source cards at the base of the composition hierarchy.

The semantics of composites implies that the components of a composite are included 'by reference' rather than 'by value' as is currently done in NoteCards' document card. Thus changes to a source card would by definition be reflected in any composite that contains that card. Con-

versely, changes to text/graphics viewed from within the context of a composite would by definition be changes to the source card actually containing the text/graphics.

The use of a composition mechanism to augment the basic node and link mechanism is a critical advancement in the hypermedia model. The filebox concept in NoteCards was designed (in part) to provide some of the characteristics of a composition mechanism in the context of encouraging hierarchical organizational structures. But the filebox concept was flawed because it failed to take into account the differences between *reference* relations and *inclusion* relations. In particular, inclusion implies a part–whole relationship in which characteristics of and operations on the whole will be true of the part as well. Reference implies a much looser relationship in which the participating entities allude to each other but remain essentially independent.

The difference between (and the confusion among) inclusions and references arises very frequently in designing hypermedia networks and applications. In NoteCards, a browser in some sense 'contains' the sub–network it displays. For example, one can edit the network by editing the browser, which implies an inclusion relation. But confusion arises because the actual implementation uses standard links (i.e., references) to connect the browser to the nodes it 'contains'. The NoteCards filebox concept was a source of great confusion among users for similar reasons.

On a less system–oriented level, NoteCards users have frequently requested the ability to refer to sub–networks or collections of unlinked cards as unique entities with names and properties of their own, separate from the names and properties of their component nodes and links. For example, a legal application might involve building a network of arguments (one argument per card) for each case. Ideally, this entire network would be 'included' in composite card, which the user could then utilize to refer to the network in its entirety. For example, the user might want to attach properties to the network as a whole marking its validity or its convincingness. In the absence of a composition mechanism, the user must invent her own mechanism for dealing with structures made up of more than one card. In NoteCards, users frequently utilize the root card of a sub–network as the representative of the entire sub–network structure.

In the next generation of hypermedia systems, composition should join nodes and links as a primitive mechanism. NoteCards, Intermedia, and Neptune all currently lack a notion of composition. Designing a composition mechanism appropriate for inclusion in these systems raises a host of interesting decisions and issues including, for example:

> Can a given node be included in more than one composite?
>
> Do links necessarily refer to a node *per se* or can they refer to a node as it exists within the context of a given composite? If the latter is possible, what does it mean to traverse a link?
>
> How does one handle versions of composite nodes? E.g., does a new version of an included node necessarily imply a new version of the composite?
>
> Should composites be implemented using specialized links or is a whole new mechanism necessary?

These issues present a challenging design problem for future hypermedia systems. However, the task is not impossible. NLS/Augment [Engl68] has (of course) already pioneered much of the

territory. The notion of composition is used heavily and effectively in the NLS/Augment system, although in ways that are not always directly relevant to NoteCards and its cohorts.

Issue 3: Virtual structures for dealing with changing information

NoteCards requires its users to segment their ideas into individual 'nuggets' to be stored away, one per card. Each of these cards then needs to be assigned a title and filed in at least one filebox. Empirical observations [Mont86b] have shown that these three seemingly trivial tasks pose significant problems for many users. In particular, a user in the very early stages of working with a particular set of information may not sufficiently understand the content and structure of that information. Knowledge about the critical dimensions of the idea space, the characteristics which distinguish one idea from another, and appropriate naming schemes develops over time as the user becomes familiar with her information. The problem arises because the segmentation, titling, and filing tasks all require the user to have such knowledge 'up-front'. As the user's knowledge of the information space evolves, previous organizational commitments (e.g., titles and filing categories) become obsolete.

Experienced NoteCards users get around this problem by adopting various strategies to delay the segmentation, titling, and filing of information. To avoid premature segmentation, these users will place the entire idea stream in a single text card. They will go back and review the entire stream before segmenting into separate cards. To avoid premature filing, experienced users file all cards in a single filebox and then use a sketch card in order to organize the cards. In this sketch card, they organize links (used as representatives of the target cards) into piles based on some judgment of similarity or 'belongingness' . These piles can be easily shifted or rearranged when new information comes in. When the piles are stable, they can be transferred into a filebox structure.

In a sense, NoteCards encourages its users towards premature organization of their information. This occurs because users' conceptual structures have a tendency to change faster than their corresponding NoteCards structures. The result is that the NoteCards structures are often obsolete with respect to the user's current thinking. To some extent, this situation is unavoidable because it will always be easier, quicker, and less tedious to change one's internal conceptual structures than it will be to update the external representations of these conceptual structures.

The pressure towards premature organization also reflects limitations in the NoteCards user interface. Relaxation of the strict titling and filing requirements is an often requested NoteCards 'enhancement' that would certainly help minimize this pressure. Providing less-stringent organizational structures such as the similarity piles described above would also provide a more natural environment for some organizational tasks. Increasing the ease by which structures could be modified (e.g., improving browser-based editing) would make it easier for users to track their changing internal structures.

At a more fundamental level, however, users experience difficulty in coping with change because the basic hypermedia model in NoteCards is inherently fragmentary and static. By definition hypermedia imposes a structure in which information is encoded into a collection of more or less independent nodes interconnected into a static (although changeable) network. This encoding is static because the segmentation and linking is represented directly in the data structure. The

encoding can be changed only by altering the data structure, i.e., by altering one static encoding to make a new static encoding.

The static nature of hypermedia networks could be largely eliminated (when appropriate) by including in the hypermedia model a notion of *virtual or dynamically-determined structures*. In the current model, nodes and links are extensionally defined, i.e., nodes and links are defined by specifying the exact identity of their components. In contrast, virtual structures would be defined intensionally, i.e., by specifying a *description* of their components. The exact subcomponents of a virtual structure are determined by a query/search procedure whenever the structure is accessed or instantiated. For example, a possible virtual composite node might be defined by a specification of the form: *a sub-network containing all nodes created by someone other than me in the last 3 days*. Each time this composite was accessed, its structure and content would be recomputed.

The notion of virtual structures for hypermedia is a direct adaptation of the concept of views (a.k.a. virtual tables) in the world of relational database systems (e.g., **[Date81]**). In the relational database world, a view is a table constructed at instantiation time by applying a stored view definition to data explicitly stored in base (or non-view) tables. From a user's perspective, a view-based table is nearly identical to a base table. All the same operations apply, including updates. Although virtual structures in a hypermedia network would be significantly more complex than the views in a relational database, the same principle of non-differentiation at the interface should apply. Any operation possible on a base hypermedia entity should apply as well to virtual structures.

The notion of virtual structures in hypermedia would be possible only in a system that supported a substantial search/query mechanism over the hypermedia network. The definition of the components in virtual structures are in fact queries. Instantiating a virtual structure involves satisfying these queries and constructing a dynamic entity from the results. Although it is not a strict requirement, it would make sense if the 'query' language used for virtual structure descriptions was the same as the basic query language.

Virtual structures are a particularly powerful mechanism when combined with the notion of composites. A virtual composite allows the user to create nodes that are dynamically constructed at access time from other nodes, links, and composites that are stored in the network. Such virtual composites are true hypermedia entities and not simply a display of results from a query. Thus, the user can add links, properties or additional static descriptions to a virtual composite. Browsers, for example, could be implemented as virtual composites built from the results of a structure query.

The notion of virtual links in a hypermedia structure has already been explored in ZOG **[Robe81]**. ZOG includes a small set of navigational links that are constructed whenever a node is accessed and displayed. These links connect the displayed node to nodes that the user has recently visited, thereby allowing the user to move quickly back from whence she came. In future systems, the notion of virtual (or computed) links will extend to non-navigational situations as well. Rather than linking to a specific node, the user will have the capability to link, for example, to *the most recently created node containing the string 'hypertext'*.

Implementing virtual nodes, links, and composites will be a difficult problem for the next generation of hypermedia systems, especially when response time is a important factor. Nevertheless, virtual structures will provide these systems with an ability to adapt to changing information in a way that is simply not possible with the current static hypermedia model. Although virtual structures will not replace static structures (since not every relation can be described by a query), they will be critical components of future hypermedia networks.

Issue 4: Computation in (over) hypermedia networks

NoteCards is basically a passive storage and retrieval system. It provides tools for users to define, store, and manipulate a hypermedia network. In service of this goal, it does some processing of the network and the information it contains. For example, browsers involve computing the transitive closure defined by a root node and a set of links types. The system, however, does not *actively* direct the creation or modification of the network or the information contained therein. Unlike an expert system, for example, NoteCards does not include an inference engine that actively processes the network in order to derive new information.

Although the basic system is relatively passive, NoteCards is frequently augmented by more active computational engines for a particular application or task domain. In one case, for example, NoteCards was augmented to function as the delivery vehicle for computer–assisted instruction **[Russ87]**. In this case, the applications developers implemented a driver that retrieved from the network and interpreted special 'script' cards. These script cards orchestrated the display of other cards containing instructional and test material. Students were expected to answer the test material and their answers were stored in the appropriate cards in the network. The driver then used these answers together with the instructions in the script cards to determine what material to display next. In a more advanced version of this system, the driver was a rule–based system that examined a number of cards in the network, including the cards containing the student's previous answers, in order to determine the what material to present.

In the foregoing example there is a clear separation between NoteCards *per se* and the computational engine embodied in the driver. The computational engine is not integrated into NoteCards. Rather it serves to consume and produce cards and links through the programmer's interface in much the same sense that a (human) user of the system consumes and produces cards and links through the user interface. Aside from the programmatic access to information in the network, NoteCards contains no special support for computational engines of this sort.

NoteCards does provide some haphazard support for integrating event–driven computations into the system. In particular, a (programming) user can create new classes of nodes whose methods carry out some auxiliary computation whenever an instance of the class is created, accessed, or modified. An example is the Query card, which performs a query on an external database to refresh its contents whenever it is brought up for display. While this ability to embed auxiliary computations in the methods of a card class is useful, it is limited in a number of ways. First, since links are not first class objects in NoteCards, it is not possible to trigger computations during the access or modification of links. Second, triggers are defined at the class level with no general support for triggering computations attached to specific node instances. Third, computation trig-

gered inside an access method cannot generally access the full functionality of NoteCards. Because of these limitations, relatively few users have implemented event-driven computations within NoteCards.

It is unclear whether the level support that NoteCards provides for computational engines is appropriate for future hypermedia systems. Designers of such systems could follow the NoteCards model and continue to support computational engines as separable external entities that create, access, and modify information via the standard programmatic interface. Alternatively, one could design a hypermedia system incorporating a more active computational component that automatically processes (e.g., make inferences from) the information stored in the network. In this case, the hypermedia system would function more like an expert system, both storing information and actively processing that information.

To a great extent the choice between a passive and an active hypermedia system is an efficiency versus generality trade-off. The ultimate functionality for the approaches is approximately the same. However, the distribution of responsibility and effort is different. Computation built into the hypermedia system it is likely to be more efficient, especially when that computation involves extensive access to information in the network. In contrast, an external computational engine is less restricted because no commitment to a particular computational engine needs to be made when the system is implemented. Thus, the choice between an active and a passive hypermedia system will be determined largely by the intended applications and the performance needs of these applications.

Both the internal and external computation models will involve challenging design issues. For systems that support an external engine, the triggers and hooks provided for this engine will have to be carefully chosen. Hooks need to be provided at both the instance and the class levels for all entities in the network (i.e., nodes, links, compositions, storage compartments, etc). Triggers should also be provided for particular events (rather than entities) to make it easier to implement event-driven systems. In addition, the system might provide some basic data structures that make it easy for an external computational engine to store temporary state information associated with entities in the network. For example, a cycle-detection engine might need to store markers on nodes it has visited. Handling these markers inside the hypermedia system might result in efficiency gains.

The design issues for the internal computation model are significantly broader and less crisp. Most importantly, the nature of the hypermedia computational engine needs to be determined. One possibility is a rule-based system in which the rules specify patterns in the network (akin to structure queries) and produce new network structures or modify existing structures. The rule interpreter for such a system could run continuously and asynchronously or it could be triggered by storage and retrieval events occurring in the hypermedia network. Other computational engines are also possible (e.g., truth-maintenance engines). The choice of an appropriate engine depends on the intended application and on the degree to which the concepts inherent in the computational engine can be integrated with the concepts inherent to hypermedia systems.

Issue 5: Versioning

NoteCards has no versioning mechanism. Each card and link exists in only one version and is altered in place when modified. As a result, the range of applications that can be easily supported in NoteCards is severely limited. For example, NoteCards has never been used for maintaining software due, in part, to the overwhelming importance of versioning in configuration management. The lack of versioning has had a lesser impact on the authoring, argumentation, and idea processing tasks for which NoteCards was originally designed. Although users in these applications frequently request versioning support, they have been able to make significant progress in its absence.

NoteCards lags behind its cohort systems in the versioning arena. Both Neptune and Intermedia provide some versioning support. Neptune, for example, provides a time-based linear version thread for individual nodes and links. The system also provides a partitioning scheme called 'contexts' **[Deli86b]** that allows users to begin a independent version thread for a given set of nodes. Intermedia, like NoteCards, keeps only a single version of each node. Intermedia, does however, have a notion of alternative named versions of the 'web', i.e., the set of links that interconnect a collection of documents. This allows each user, project, or application to work with its own network structure over a common set of document nodes.

PIE **[Gold84]** included a more extensive versioning mechanism than either Neptune or Intermedia. In the PIE model, versioning in a hypermedia network occurs at two levels: the level of individual entities (nodes, links, composites) and the level of changes to the hypermedia network considered as a whole. At the lower level, each entity has its own version history. In PIE the version history was a linear thread, but in the general case it could be an arbitrary version graph. Although the issue was not specifically addressed in PIE, it is important to provide (virtual) entities corresponding to both specific versions of an entity and the differences (deltas) between successive versions. Both the versions and the deltas should be addressable within the system, i.e., be possible hits for a search. For example, in a software engineering context it should be possible to search for either *the version that implements Feature X* or *the set of changes that implements Feature X*. One possibility would be to treat the deltas between successive versions as special hypermedia nodes capable of being annotated and referenced.

Providing a branched version history for all entities in a hypermedia network raises some very difficult issues regarding the semantics of references between entities. In particular, a reference to an entity may refer to a specific version of that entity, the newest version of that entity, to the newest version of that entity along a specific branch of the version graph, or to the (latest) version of the entity that matches some particular description (i.e., query). Which of these reference types is supported is a decision that affects the entire hypermedia system, but it is an especially critical decision in the design of links and composites. Moreover, composites raise the related problem of propagating version changes from subcomponents to their composites. For example, a (significant) change in an individual software module implies the creation of a new version of the system of which that module is a part. In contrast, updating the spelling of a few words in a paragraph may not require the creation of a new version of the document(s) containing that paragraph.

Maintenance of a version tree for individual entities, however, is not sufficient support. In general, users will make coordinated changes to a number of entities in the network at one time. For example, a software developer may implement a new feature by making coordinated changes to a number of separate modules. The developer may then want to collect the resultant individual versions into a single 'version set' for future reference. This set would be a snapshot of the collection of entities at some particular point in time, what in the software domain is often called a 'release'.

An alternative to version sets, is the 'layer' mechanism used in PIE. A layer is a coordinated set of *changes* to one or more entities in the network. For example, all of the changes made to various modules in a software system could be collected into a single layer described as *the changes that implement Feature X*. The resulting layer could then be applied to the pre-change versions of the entities to get the post-change versions.

In a layer-based system, the primary issue is the mechanisms available for managing and for composing layers. In PIE, there were constructs called 'contexts' that were essentially sequences of layers. The hypermedia network for a given context was determined by applying the first layer to the base network and then applying the next layer to the result and so on through the sequence of layers in the context. New contexts could be created by mixing and matching the layers from other contexts and then producing a hypermedia network by successive applications of the layers in the context. However, not all such constructions made semantic sense. PIE aided the user in determining which contexts were sensible and which weren't.

One important application of the notion of contexts is to provide a collaborative system (see Issue 6) in which each user maintains a private context through which she interacts with the hypermedia network. Each user's context contains a base layer that is the public hypermedia network. On top of this base are one or more personal layers which alter the base network to provide a personalized view. At any time a user can make her view public by applying the layers in her view to the layers in the base system (and informing her collaborators to use this new base network in their contexts).

Although the PIE versioning scheme is not perfect, it is richer and more complete than those provided by most hypermedia systems. One of the design issues facing the next generation of systems is how to improve the level of versioning support in hypermedia networks. The PIE model seems like an appropriate place to start in resolving this issue.

Issue 6: Support for collaborative work

In its original design, NoteCards focused almost exclusively on supporting individuals working alone on idea processing and information management tasks. Collaboration was seen as occurring outside of the NoteCards environment, e.g., through face-to-face conversations, electronic mail, or hardcopy documents. In practice, however, this has rarely been the case. Most idea processing and information management tasks are inherently collaborative, with groups of varying from two to ten people working on a common topic or project.

The problems and prospects surrounding support for collaborative work in NoteCards have been discussed in detail by Trigg, Suchman, and Halasz [Trig86] and hence will not be described here. It is important to note, however, that support for collaborative work remains a critical issue for the next generation of hypermedia systems. Although many existing systems have provided for the mechanics of multi-user access to a hypermedia network, none has adequately addressed the issue of support for the social interactions involved collaboratively sharing a hypermedia network. The challenge for the next generation of hypermedia systems is to provide adequate support for these social processes. This support should include system-level features such as the automatic maintenance of change histories, the tracking of individual contributions to the network, and mechanisms for enforcing collaborative conventions. It should also include a 'rhetoric of hypermedia' that provides guidelines or conventions for creating hypermedia networks that will be useable or understandable by others who share the 'rhetoric'. Interestingly, the development of these conventions is a crucial issue that can only be solved by accumulated experience in using hypermedia systems in real-world tasks.

Issue 7: Extensibility and tailorability

NoteCards was designed to be an extensible system. The expectation was that users would tailor the generic functionality of NoteCards to better match the requirements of their application and their preferred interaction style. The major mechanism provided for this purpose was the NoteCards programmer's interface. In practice, the programmer's interface has been very successful. A number of tailored systems, both major and minor, have been built on top of NoteCards using this interface [Hala87]. Trigg, Moran, and Halasz [Trig87] contains a lengthy discussion of tailorability in NoteCards, including examples of its use, problems with it implementation, and prospects for it improvement. Therefore, no further discussion of these issues will be included here.

However, one of the issues raised by Trigg *et al.* represents a critical challenge for future hypermedia systems. NoteCards was designed to make tailorability available to the entire range of users from non-programmer's to experienced system builders. The goal was to build tailoring mechanisms having the characteristic that small changes to the system could be accomplished with a small amount of work by users without extensive knowledge of programming and the system's implementation. This design goal was simply not met. Tailoring NoteCards remains a non-trivial programming task that requires some degree of expertise. This appears to be case for other hypermedia systems as well. The challenge, then, for the next generation of hypermedia systems is to discover how to enhance the 'small' tailorability of hypermedia systems, i.e., how to make it easy for the typical (non-programming) user to extend the system to match her task requirements or interaction style.

CONCLUDING REMARKS

Hypermedia has suddenly become quite popular. There is something alluring about the concept of navigating through an information network following links from node to node until you find something of interest. But as the current crop of hypermedia systems move into more widespread use, their limitations will become quite evident to serious users. The simple node and link model

is just not rich and complete enough to support the information representation, management, and presentation tasks that these users will want to accomplish using their hypermedia system. This has been the experience with NoteCards over the last three years and there is no reason to believe that other systems will fare significantly better. The seven issues presented in this paper are an attempt to move the hypermedia model beyond simple nodes and links to a model that will better suit the needs and preferences of these users. In this sense, these seven issues represent an agenda for the next generation of hypermedia systems.

REFERENCES

[Date81] Date, C.J. *An Introduction to Database Systems Vol. 1.* Reading, MA:Addison-Wesley, 1981.

[Deli86a] Delisle, N. & Schwartz, M. Neptune: a hypertext system for CAD applications. *Proceedings of ACM SIGMOD '86*, Washington, D.C., May 28-30, 1986, 132-142.

[Deli86b] Delisle, N. & Schwartz, M. Contexts - a partitioning concept for hypertext. *Proceedings of the Conference on Computer-Supported Cooperative Work*, Austin, TX, December 3-5, 1986, 147-153.

[Engl68] Englebart, D.C. & English, W. A research center for augmenting human intellect. *Proceedings of 1968 FJCC*, 33, Part 1, Montvale, N.J.:AFIPS Press, 1968, 395-410.

[Fair87] Fairchild, K. F., Poltrock, S. E., & Furnas, G. W. SemNet: Three-dimensional graphic representations of large knowledge bases In R. Guindon (Ed.) *Cognitive Science and its Applications for Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates, in press.

[Furn86] Furnas, G.W. Generalized fish-eye views. *Proceedings of the 1986 ACM Conference of Human Factors in Computing Systems (CHI '86)*, Boston, MA, April 13-17, 1986, 16-23.

[Garg87] Garg, P.K. & Scacchi, W. A hypertext system to manage software life cycle documents. Paper submitted to the 21st Hawaii International Conference on Systems, 1987.

[Garr86a] Garrett, L. N. & Smith, K.E. Building a time-line editor from pre-fab parts: The architecture of an object-oriented application. *Proceedings of the Conference on Object-oriented Programming Systems, Languages, and Applications (OOPSLA '86)*, Portland, OR, September 29 - October 2, 1986, 202-213.

[Garr86b] Garrett, L. N., Smith, K.E., & Myrowitz, N. Intermedia: Issues, strategies, and tactics in the design of a hypermedia document system. *Proceedings of the Conference on Computer-Supported Cooperative Work*, Austin, TX, December 3-5, 1986, 163-174.

[Gold84] Goldstein, I & Bobrow, D A layered approach to software design. In D. Barstow, H. Shrobe, & E. Sandewall (Eds.) *Interactive Programming Environments*. McGraw-Hill: 1987, pp 387-413.

[Guid86] Guide Users Manual. Owl International, Inc. , Bellevue, WA., 1986.

[Hala87] Halasz, F.G., Moran, T.P., & Trigg, R.H. NoteCards in a Nutshell. *Proceedings of the 1987 ACM Conference of Human Factors in Computer Systems (CHI+GI '87)*, Toronto, Ontario, April 5–9, 1987, 45–52.

[Lamb86] Lambert, S. & Ropiequet, S (Eds) *The New Papyrus*. Redmond, WA: Microsoft Press, 1986.

[Lowe86] Lowe, D. SYNVIEW: The design of a system for cooperative structuring of information. *Proceedings of the Conference on Computer–Supported Cooperative Work*, Austin, TX, December 3–5, 1986, 376–386.

[Meyr86] Meyrowitz, N. Intermedia: The architecture and construction of an object–oriented hypermedia system and applications framework. *Proceedings of the Conference on Object–oriented Programming Systems, Languages, and Applications (OOPSLA '86)*, Portland, OR, September 29 – October 2, 1986, 186–201.

[Mont86a] Monty, M.L. & Moran, T.P. A longitudinal study of authoring using NoteCards. *ACM SIGCHI Bulletin*, Vol. 18 No. 2, October 1986, 59–60.

[Mont86b] Monty, M.L. Temporal context and memory for notes stored in the computer. *ACM SIGCHI Bulletin*, Vol. 18 No. 2, October 1986, 50–51.

[Nels81] Nelson, T.H., *Literary Machines*, T.H. Nelson, Swarthmore, PA., 1981.

[Robe81] Robertson, G, McCracken, D., & Newell, A. The ZOG approach to man–machine communication. *Int. J. of Man–Machine Studies*, 14, 1981, 461–488.

[Russ87] Russell, D.M., Moran, T.P., & Jordan, D.S. The instructional design environment. Xerox PARC working paper, April 1987.

[Shne87] Shneiderman, B. *User interface design and evaluation for an electronic encyclopedia.* Technical report CS–TR–1819, Dept. of Computer Science, University of Maryland, College Park, MD, March, 1987.

[Smit86] Smith, J.B., Weiss, S.F., Ferguson, G.J., Bolter, J.D., Lansman, M., & Bea, D.V. *WE: A writing environment for professionals.* Technical report 86–025, Dept. of Computer Science, University of North Carolina, Chapel Hill, N.C., August, 1986.

[Trig83] Trigg, R. *A Network–based Approach to Text Handling for the Online Scientific Community.* PhD thesis, Dept. of Computer Science, University of Maryland, November, 1983.

[Trig86] Trigg, R., Suchman, L. and Halasz, F. Supporting collaboration in NoteCards. *Proceedings of the Conference on Computer–Supported Cooperative Work*, Austin, TX, December 3–5, 1986, 147–153.

[Trig87] Trigg, R.H., Moran, T.P., & Halasz, F.G. Tailorability in NoteCards. Paper to be presented at Interact '87, Stuttgart, West Germany, August, 1987.

[VanL85] VanLehn, K *Theory reform caused by an argumentation tool.* Xerox Palo Alto Research Center Technical Report, ISL–11, 1985.