This document contains 52 pages

HYPERMEDIA NOTETAKER
FUNCTIONAL SPECIFICATION

Prepared by

Catherine C. Marshall

31 May 1990

Contract Number: 88-F665700-000

XEROX PALO ALTO RESEARCH CENTER
3333 COYOTE HILL ROAD
PALO ALTO, CALIFORNIA 94304

TABLE OF CONTENTS

TABLE OF FIGURES

REQUIREMENTS INDEX

support node and link classification	Section 7.3.2
support versioning	Section 7.4
support backup and recovery at the container level	Section 7.6

5. Hypermedia data model requirements - the notetaker should:

use a hypermedia model consisting of nodes, links, and composites	Section 2

6. Linking requirements - the notetaker should:

build and maintain source links wherever possible	Section 1.3.1
allow linking to as many outside sources as possible	Section 1.3.1
provide capabilities to attach text to the source material	Section 1.3.2
support links with more than two endpoints	Section 1.3.2
support link annotations	Section 1.3.2
support the ability to link to any object in the system	Section 1.3.2
support span-to-span links	Section 2.2
support link markers	Section 2.2
support both connection and reference links	Section 2.2.1
support link typing	Section 2.2.1
support link labelling	Section 2.2.1
support directional links	Section 2.2.1
support linking to link objects	Section 2.2.1
support n-ary links	Section 2.2.1
support links that include endpoint specifications	Section 2.2.2
support links that include properties	Section 2.2.2
support links that include user annotation	Section 2.2.2
use anchors to implement "span-to-span" links	Section 2.2.3.1
use anchors that adhere to an anchoring protocol	Section 2.2.3.1
use link markers that signal the presence of a link	Section 2.2.3.2
use link markers that signal the integration level of their destination(s)	Section 2.2.3.2
use link markers that signal the scope of the anchor's span	Section 2.2.3.2
support operations to create and delete links	Section 3.2
support operations to add, move, or remove link endpoints	Section 3.2
support operations to annotate links	Section 3.2
support operations to add, delete, or change link properties	Section 3.2
support operations to change representational types (change labels)	Section 3.2
allow users to specify the extent of link anchors	Section 3.2.1
allow users to change the extent of a link anchor	Section 3.2.1

allow users to edit material within a link anchor	Section 3.2.1
support link marker operations for link traversal	Section 3.2.2
support link marker operations for viewing link annotations	Section 3.2.2
support link marker operations for viewing and editing link properties	Section 3.2.2
support link marker operations for changing link labels	Section 3.2.2

7. Node requirements - the notetaker's nodes should:

have identification (system and user)	Section 2.1.2
have properties	Section 2.1.2
have content	Section 2.1.2
have core implementational types of text, bitmap, and graphics	Section 2.1.3
support view, edit, and remove (from the display) operations	Section 3.1.1
support create, save, and delete operations	Section 3.1.1
support naming, organizing (filing), and viewing of properties	Section 3.1.1
support read check-in and check-out (multi-user)	Section 3.1.1
support write check-in and check-out (multi-user)	Section 3.1.1
support editing of their contents with appropriate editors	Section 3.1.2

8. Composite requirements - the notetaker's composite nodes should:

allow groups of nodes to be manipulated as discrete entities	Section 2.3
support all general node operations	Section 3.3
support operations to perform composition	Section 3.1.1
support operations to add and delete new subnodes	Section 3.1.1

9. Network management requirements - the notetaker should:

implement containers and organizers	Section 4
implement containers such that each node lives in one and only one container	Section 4.1
guarantee consistency of all elements inside a container	Section 4.1
implement cross-container links	Section 4.1
support container creation and deletion	Section 4.1
support container naming and renaming	Section 4.1
support container replication	Section 4.1
support container consistency checks and repair	Section 4.1
implement organizers that provide a hierarchical filing mechanism	Section 4.2
implement organizers that disallow cycles in the hierarchy	Section 4.2
implement organizers that prevent the creation of disconnected subgraphs	Section 4.2
implement organizers that provide access to all network roots	Section 4.2

support organizer creation and deletion	Section 4.2
support organizer naming and renaming	Section 4.2
support organizer-based filing	Section 4.2
support replication of an organizer's subnetwork	Section 4.2
support moving an organizer's subnetwork	Section 4.2

10. Information access requirements - the notetaker should:

implement mechanisms for traversal-based access	Section 5
implement mechanisms for views and network overviews	Section 5
implement mechanisms for querying the database	Section 5
implement mechanisms for linearization	Section 5
provide a preview of a link's destination	Section 5.1.2
provide easy access to link annotation	Section 5.1.2
highlight the destination link anchor after the link is traversed	Section 5.1.2
provide easy access to the link's label	Section 5.1.2
provide easy access to any additional link properties	Section 5.1.2
implement a traversal history mechanism	Section 5.1.3
implement a "jump back" capability	Section 5.1.3
provide a representation of nodes visited	Section 5.1.3
implement a traversal overview	Section 5.1.3
implement indices	Section 5.2
implement outlines	Section 5.2
implement computed network diagrams	Section 5.2
implement manually laid out network diagrams	Section 5.2
generate an index based on node names	Section 5.2.1
generate an index based on node keywords	Section 5.2.1
generate an index based on representational types	Section 5.2.1
present index results as a point-and-click list	Section 5.2.1
implement outlines as indented lists for manipulating hierarchical structure	Section 5.2.2
implement a graphical browser that can be computed automatically from a simple specification	Section 5.2.3
implement a graphical browser that provides an active view of the network	Section 5.2.3
implement a graphical browser that has facilities for automatic layout	Section 5.2.3
implement a graphical browser that provides user layout tools	Section 5.2.3
implement a graphical browser that can perform various types of filtering for named views	Section 5.2.3

support interactive creation of nodes, links, and composites through a graphical browser	Section 5.2.3.1
support interactive deletion of nodes, links, and composites through a graphical browser	Section 5.2.3.1
support interactive renaming of nodes and composites through a graphical browser	Section 5.2.3.1
support interactive relabelling of links through a graphical browser	Section 5.2.3.1
support network layout operations for moving nodes and subnetworks	Section 5.2.3.2
support network layout operations for aligning nodes	Section 5.2.3.2
support operations for automatic relayout specified portions of the diagram	Section 5.2.3.2
support network layout operations for adding labels to the diagram	Section 5.2.3.2
support network layout operations for removing nodes and labels from the diagram	Section 5.2.3.2
implement a topic browser	Section 5.2.3.3
implement a traceability browser	Section 5.2.3.3
implement browser style sheets	Section 5.2.3.3
implement content search	Section 5.3
implement structure search	Section 5.3
implement filtered linearizations for different kinds of presentations	Section 5.4

11. Integration requirements - the notetaker should support:

an anchoring protocol	Section 6.1.1
a linking protocol	Section 6.1.2
a launching protocol	Section 6.1.3
an import/export format for linear documents	Section 6.2
an import/export format for hypermedia	Section 6.2

12. Extensibility requirements - the notetaker should support:

implementational and representational types mechanisms for nodes	Section 2.1.3
a programmer's interface to the implementational types mechanism	Section 2.1.3
a user interface to the representational types mechanism	Section 2.1.3
independent implementational and representational typing mechanisms	Section 2.1.3
implementational and representational types mechanisms for links	Section 2.2.1
stored specifications (for overviews and access methods)	Section 7.1
parameterized control of user interface-related variables	Section 7.1

EXECUTIVE SUMMARY

Notetaking is an activity central to the sense-making process at the core of intelligence analysis. In this specification, we outline the functional characteristics of a facility to support notetaking as practiced in the intelligence community. Hypermedia is an information management model that represents text, graphics, images, and other data forms as nodes, and the interconnections between nodes as typed links; these nodes and links form a user-defined network of information. We take hypermedia to be the fundamental underlying technology described by this specification.

In Section 1 of the specification, we abstract some functional constraints on a hypermedia notetaker by two means. First, we discuss the results of an informal work practices study conducted to determine how analysts take notes in the course of their current activities. Second, we reflect on our experiences with NoteCards, a six-year-old hypermedia idea processing system also built for the intelligence community. From these, we arrive at a set of user requirements, some that are situation-specific and others that are general design principles.

Situation-specific user requirements include the accommodation of the way analysts use their on-line and paper sources, support for the various styles of notetaking and filing that we observed in an analytic branch, assistance for the analyst in retrieving notes and files, support for the paper-writing and draft passing processes, and the accommodation of collaborative work. More general principles for implementing this kind of facility include using user interface metaphors appropriate to the task, and ensuring the requisite level of reliability and performance to make the system useful in an analytic setting.

In the second section of the specification, we provide a detailed description of the hypermedia data model. In discussing the data model, we consider the types of hypermedia objects an analyst will need in performing a notetaking task and how these objects can be broken down into their subelements. We also begin to address the concept of extensibility - how this data model can be extended and adapted to changes in the analyst's task environment.

In the notetaker, a node may be an annotation, an interpretive note, a draft in progress, a source document, a mathematical model, imagery, a cable of interest, and a variety of other kinds of information found in the analytic environment. All nodes thus have two kinds of content, structured (such as a property or a database entry) and unstructured (such as free text, drawings, or bitmaps); they also must have a means of unique identification so both the system and the user may refer to them.

In our hypermedia model, links are the component that describe relationships between nodes or portions of nodes. For example, a "source" link may connect some portion of a note with a paragraph from a document. Links consist of data describing where the link starts and ends, properties (which include a label, a specification of the link's behavior, and its direction), and some textual annotation.

We also describe a higher level abstraction for the hypermedia model, composite nodes.

Composites will give analysts the ability to define nodes that include other nodes as we might find in a complex argument structure that includes both a hypothesis and several pieces of evidence.

Basic hypermedia functionality for the notetaker is covered in Section 3 of the specification. We discuss operations over all of the basic elements in the hypermedia data model - how an analyst using the notetaker will be able to create, delete, and manipulate nodes, links, and higher-level constructs. In Section 4, we introduce the notion of containers, which allow analysts to refer to whole networks and manipulate them as files, and organizers, which support the hierarchical structuring of information as we find in it file folders. These constructs will help analysts manage their data in fairly intuitive ways.

Because we have found the structures that users build in hypermedia to be complex, we have addressed a set of issues concerning how analysts will access and view information in the notetaker; this discussion is found in Section 5. Link traversal - the primary way a user moves from node to node - is examined in some depth, starting with the point-and-click mechanics, and progressing through some facilities for helping a user maintain his or her orientation. We also look at four kinds of overviews that will be useful in the notetaker, indices, outlines and tables of content, computed network diagrams, and summary nodes and manually laid out representations of network structure. Finally, we consider more traditional search mechanisms for query-based retrieval, and linearization mechanisms, so analysts can produce the kind of reports required of them.

Analysts work in a rich, complex environment of systems and information sources. Because these systems and sources vary with time and task, we have devoted Section 6 of the specification to notetaker integration, including how an analyst can take notes in the context of a variety of applications, and how information can be imported and exported from the notetaker.

Integration will necessarily take place on three levels, depending on the willingness of applications developers to follow any or all of a set of protocols. By following a display protocol, an outside application can be launched by link traversal; the notetaker will manage the application as a window on the screen rather than a node in the network. A linking protocol will allow information from an outside application to be treated as a node. Finally, adherence to an anchoring protocol in addition to the other two protocols will allow an analyst to treat information from an application outside the notetaker in the much same manner as information developed inside the notetaker, although consistency of information outside the notetaker cannot be guaranteed.

Other general issues are discussed in Section 7 of the specification. One such issue concerns tailorability and extensibility - how the notetaker will be adaptable for different branches of analysts performing different kinds of notetaking tasks, and how the notetaker will conform to the idiosyncratic requirements of the individual analyst, whose style and mode of interaction may differ from the built-in assumptions of the user interface. A second issue revolves around the requirements derived from multi-user access, both database-level needs such as notification and locking, and collaborative work-related needs such as negotiating access. We also comment briefly on security, mostly in terms of controlling access to shared information and maintaining classification labels. Finally we mention a versioning scheme, some

user interface requirements, and the need for a backup and recovery mechanism.

A requirements index has been included at the beginning of this document; it lists individual functional requirements for the notetaker and matches them with the section in which they are discussed. References are also supplied as part of this specification. We feel that existing work in hypermedia can provide a framework for the development of the notetaker. Note in particular [Halasz & Schwartz 1990], which provides an accepted reference model for hypertext abstractions.

Hypermedia Notetaker Functional Specification

1. INTRODUCTION

This specification describes the minimum capabilities for a useful notetaker in the intelligence analyst's environment; it also considers how the capabilities can be expanded either for other task environments, or for this environment as the designers and implementors learn more about the analysts' work. The notetaker is based on the concept of hypermedia, wherein a network of multimedia nodes are connected by links. This specification focuses on the functional requirements of the notetaker rather than its architecture.

To provide some background for the specification, we will first briefly describe a model of analytic activities. We will also discuss some of the shortcomings of the NoteCards system, a second generation hypermedia prototype [Halasz et al. 1987] as reported by members of its diverse user community. Finally, in the light of the informal analytic model and our experiences with NoteCards, we will outline some fundamental user requirements for the notetaker.

The specification that follows describes a hypermedia data model for the notetaker, basic notetaker functionality, how information will be accessed and viewed, and how the notetaker will be integrated into a heterogeneous applications environment. General issues are also covered. Because there is a group of successful second generation hypermedia systems available on the commercial market, we will frequently describe features in comparative terms in hopes of making the specification clearer and more concise. A list of references is supplied at the end, so more detailed descriptions of all of the reference hypermedia systems can be used if necessary. Wherever possible, we link specification details with individual user requirements.

1.1 Model of analytic activities

This specification describes a hypertext-based system to support the analysts in their notetaking and other sense-making activities. Therefore, we are basing the specification on requirements derived during the course of an informal work practices study conducted at two sites, OSWR and NPIC, coupled with our previous understanding of the idea processing task (see [Halasz et al. 1987], [Trigg et al. 1986], and [Trigg et al. 1987] for discussions of various aspects of idea processing in NoteCards).

The analysts we studied work in a rich, complex environment of systems and information sources. From these sources they gather information; they may scan the cables they receive through an institutional mail system, retrieve information from a variety of on-line resources (including outside information services like Dialog), or examine imagery. They read and interpret information they gather, manifesting their interpretation in one of several ways. Sometimes they take notes on what they read or annotate source

material before filing it in personal or shared on-line or hardcopy file systems; in other cases they reflect their understanding of the material by simply filing source material or organizing it in response to a specific assignment. The product of this interpretation process may be a formal written analytic paper, a shorter (and less formal) article, an addition to a database, or a briefing board (an image coupled with a brief analysis).

Thus, information gathering and retrieval, interpreting sources through notetaking and filing, and authoring reports are all important parts of the analytic task. These processes interact in a variety of ways; notetaking can be driven by information at hand (culled from an electronic mail inbox or film frames), or it can be driven by a requirement to produce a written report. Retrieval needs may be refined in the interpretation process as the analyst tries to make sense of the information at hand, or they may be dictated by a specific assignment. Structures to organize information may also be related to either sources or products, or to the internal models of a domain that an analyst has evolved over his or her career. Finally, presentations may be prompted by analytic requirements, or they may be driven by new interpretations that come out of the earlier processes in the flow.

Furthermore, we found that the broader categories of analytic information processing are collaborative or coordinated with people in other organizational roles. Interpretation is often collaborative, sometimes involving telephone conversations, or informal face-to-face meetings. Interpretive collaboration is initiated by three different types of questions: (1) "What do you make of it?" (2) "Do you agree with this (or can you corroborate this)?" and (3) "What are the implications of this?" If the collaboration looks to be fruitful, a draft-passing co-authorship is negotiated between the two analysts, hence starting a presentation-phase collaboration. Coordination occurs in retrieval tasks in two ways: (1) Some members of the analytic work group, in particular intelligence assistants, have specific expertise in retrieval and can help an analyst gather information he or she needs from the institutional or outside sources. (2) Some analysts have specific resources (like their own extensive files); it is a coordinated effort to locate the desired information in those files.

Figure 1.1-1 sketches the flow between the categories of analytic activities and shows how they may be conducted in a collaborative setting; this model of analytic information processing activities is described in greater detail in [Marshall 1990].

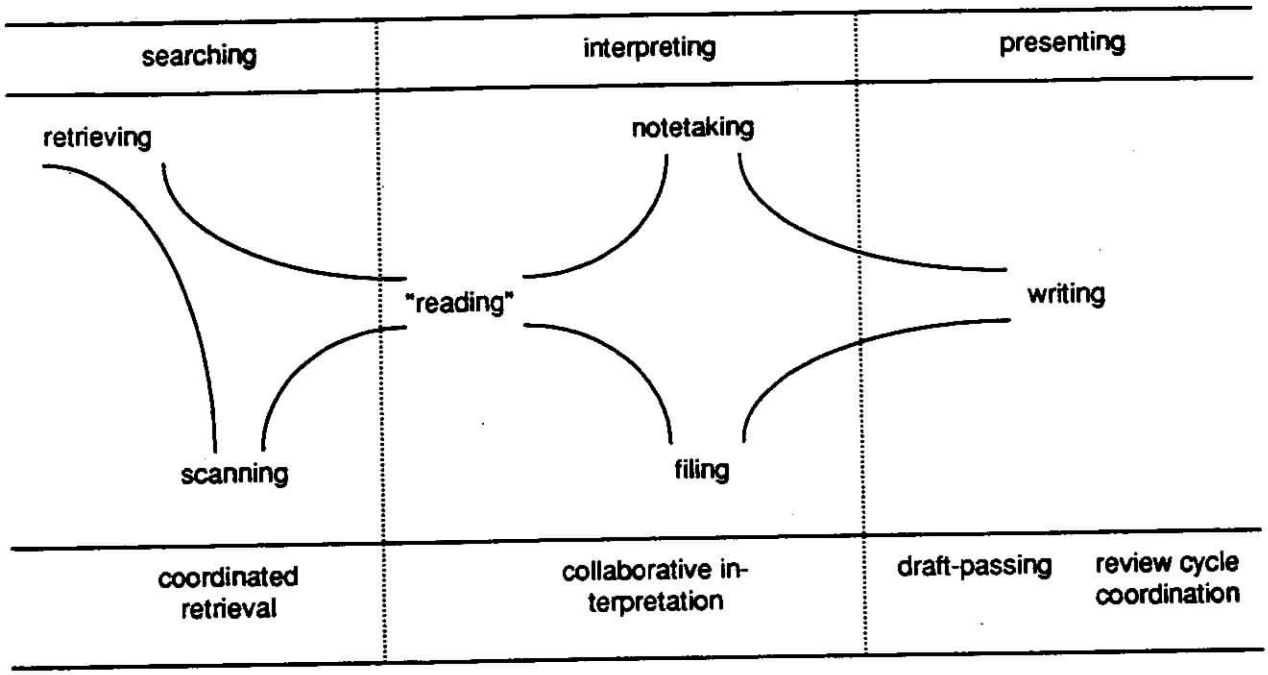

Figure 1.1-1. Analytic information processing activities

In order to determine requirements for hypertext in the context of this task environment, it is important to investigate three areas: (1) where the information comes from; (2) the relationship between the kinds of notes analysts take and the information sources; and (3) what use the information is put to after it is interpreted, including how it is structured within files. From looking at (1) and (3), we will be able to determine a strategy for integrating hypertext into an applications environment, and from (2), we will understand requirements on linking pieces of information together. From (3), we can also learn what kinds of higher level structure are useful to analysts.

1.1.1 Where information comes from

The analysts we studied use a variety of sources, some currently available on-line or destined to be on-line in the foreseeable future, and others that will continue to be available only in hardcopy forms. Frequently cited anecdotal evidence suggests that only five percent or so of the available data is ever used in analysis; therefore, analysts all feel very strongly about pulling in material from a variety of sources and processing as much of it as possible. It is a widely held belief in the intelligence community that contradictory analytic results stem from the use of different sources, rather than from different interpretations of the same information.

We have categorized the sources of on-line information that analysts use into four groups: personal files and databases, information from systems maintained by the analyst's working group, information from institutional databases and mail systems, and information maintained external to the organization such as open literature databases. These categories suggest that there are varying degrees of control that hypertext developers will have over the systems and databases supplying this information. At best - as in the case of personal files and working group databases - the hypertext substrate will be able to represent

and display the information at both ends of a link; at worst - the cases where commercial information sources are used - the hypertext substrate will only be able to represent a method for initiating the outside application.

In our study, the most important source of day-to-day on-line information for OSWR analysts is the institutional mail system, SAFE, that supplies each analyst with cable traffic, filtered by an interest profile. Each analyst described a process of going through the day's institutional mail in a linear sequence and deciding which messages are of interest. Currently, these messages are hardcopied for further processing, mainly highlighting and otherwise marking them up. Similarly, NPIC analysts describe a process of going through film frames picking out the most interesting ones and annotating and filing them. Therefore, we find that information comes from a variety of sources and media, and that much of it falls between the two extremes of being subject to personal or local control, and being maintained by an outside source, with little interest in negotiating interfaces. This will have some important consequences for integration requirements.

1.1.2 How notes are related to sources

The analysts we studied exhibited a range of notetaking styles. Many of them relied strictly on *annotative* notes; that is, they would make hardcopies of source materials and mark up the pages or mark directly on film frames. In textual sources, annotative notes are taken in two different ways. Often, a broad-tipped highlighting pen is used to go over words, sentences, or paragraphs of particular interest. Some analysts have a preference for specific colors when they are doing this type of highlighting annotation. The second annotative style of notetaking involves writing short notes in the margins of the hardcopy. For example, one of the analysts marked things he did not believe to be true, or that he found anomalous; he noted those beliefs in the margins. Similarly, film sources are annotated with marks, outlines, and marginalia. Annotative notes are therefore closely bound to selected segments of text or regions of images; in hypertext terms, they rely on access to a portion of the *content* of a node.

We found that the analysts also use *interpretive* notes to record hypotheses, conclusions they have reached, or material they have integrated from several sources. These notes are frequently taken on-line in a text editor or a database; sometimes this style of notetaking involves a significant amount of retyping to associate notes with their sources. Analysts also take interpretive notes that do not refer directly to any source, or that refer to a computational model. Interpretive notes are less tightly bound to individual words or sentences in a document. More often, they refer to a general assimilation of the document's content. Thus they frequently point to what would be represented in hypertext as a *node*.

All of the analysts in our study made some use of *reminding* notes, Post-its or other jottings on paper that serve to jog their memory about things to do (an agenda of subtasks) or portions of procedures to follow (for example, how to log on to a given outside data service, or how to retrieve a piece of information). Reminding notes may be an important way of preserving procedural knowledge. These notes often do not refer directly to a node or its content, but rather how to get to it; they can be thought of as referring to the *link*.

Figure 1.1-2 summarizes the three categories of notetaking styles we observed in the analysts' work groups.

Text highlighting (keywords, and regions); outlines, marks, and labels on images	Interpretive notes referring to one or more sources (may be in the form of text, db entries, or tables)	Auxiliary notes documenting a systematic process
Annotative notes	**Interpretive notes**	**Reminding notes**
Marginalia (marks or short comments in the margins)	Text notes not referring to any source directly	Auxiliary notes listing an agenda of subtasks

Figure 1.1-2. Analysis of notetaking styles

1.1.3 How information is used

Information is used two ways: (1) analysts build up files and (2) they write analytic reports, short articles, and other artifacts recognized by the community. The organization of notes and collected source materials varies greatly from analyst to analyst. Some of the filing structures we saw were based on chronology, either of when the material was received, when the article was published, or when an event took place. Other filing structures were based on coarse-grained categorizations of material, mostly to facilitate re-retrieval from personal or shared files; such a structure might be by general topic or world area, and might be used in combination with some other filing strategy such as chronology. The analysts' complex filing schemes may also reflect their models of the domain or their model of the topic they were reporting on. In actual practice, most of the analysts used a hybridization of the filing strategies listed above. We also saw "institutional filing schemes" - agreed upon ways of organizing materials - mainly where analysts were maintaining communal files.

Variance in filing strategies may thus depend on whether the analyst is organizing his materials by task at hand, by what he finds in his source material, by an internal domain model, or by constraints introduced by cooperative work. We saw filing practices that suggest that filing structures are dictated by all phases and aspects of the analytic process.

The analysts' on-line organizations of material do not necessarily reflect their organizations in the paper world; analysts use different filing structures on-line than they do for their file folders. This difference may be accounted for by three factors: (1) the differences in on-line and paper retrieval mechanisms; (2) the homogenization of on-line information (the loss of certain important visual cues as documented in [Monty & Moran 1986]); and (3) the differences in information volume (most of the analysts' notes and papers are still in hardcopy).

The sharing of files is necessarily a collaborative process; in many cases, one analyst's filing structure is opaque to other analysts unless there are agreed upon filing conventions. Furthermore, once an analyst leaves the organization, his or her personal files quickly deteriorate in value. Thus, in order to make the information useful to anyone else, the analyst must either document this structure, make the material accessible in a shared system, or publish any interesting analytic results.

Many kinds of analytic products are supported by the institutional system, ranging all the way from formal publications developed over a period of time to short commentaries and descriptions. These analytic products are created by integrating on-line sources and notes, and collections of annotated hardcopy material. Most of the analysts pull out their collection of materials on the desired subject to create a context for writing and to maintain traceability, which is universally cited as an important requirement on (and role for) hypertext. In all cases, the publication of an analytic product, and the subsequent usefulness of the document or article is directly related to the ability to, in hypertext terms, follow its links back to the sources.

Once an analytic product has gone through the coordination cycle, it may be used by low level policy-makers, by various staff members, and by other analysts (sometimes affiliated with different agencies). Analysts expressed a desire for a "lighter weight" analytic product in order to share smaller chunks of analytic results with their community and receive credit for coming up with these results; in hypertext terms, we might think of this as sharing an interpretive layer over a heterogeneous collection of databases.

1.2 NoteCards as a prototype

The NoteCards system represents many of the major concepts of second generation hypermedia research. It implements the concept of a node-link network to handle various applications hinging on the management semi-structured information. The system provides the user with electronic notecards interconnected by typed links. The system includes facilities for displaying, modifying, manipulating, and navigating through this network [Halasz et al. 1987].

Because its development has been driven by a large voluntary user community, NoteCards provides a valuable starting point for exploring functional specifications for a next generation hypermedia system. Our experiences with NoteCards and other hypermedia systems, coupled with the study of a particular task environment have driven the development of this functional specification for a hypermedia notetaker prototype.

As a research prototype, NoteCards exhibits some shortcomings which we address in this specification. First, NoteCards began with a "closed world" assumption; this assumption has given rise to a variety of connectivity and user interface issues throughout its life in the user community. Second, NoteCards was directed toward supporting a single user, isolated from his or her collaborators; however, because analysts (and other workers performing intellectual tasks) do not tend to work in isolation, this specification looks forward to hypermedia's role in a collaborative setting. Third, system performance and

complexity have presented problems in the analytic environment, both in terms of the learning curve and its ultimate usefulness for particular tasks. Finally, because NoteCards has been a research prototype and was built on top of the Interlisp-D prototyping environment, this specification considers pragmatic issues related to implementation strategies.

1.2.1 NoteCards as a closed world system

NoteCards is essentially a closed world system; that is, it only allows a user to perform NoteCards operations on objects it has created. What having a closed world system means is that if a user wants to create a link involving a document outside the system, he or she must include the document (or the relevant portion) in the network as a card (or use the File Card library package). Sometimes this entails losing formatting, context, reliability (for now the user is responsible for maintaining the new version), or even information (if the file is not interpretable by the system). Furthermore, if a user wants to link to an item in a database, and leave it in the context provided by the database, the user is out of luck.

However, a closed world system gives the user reliable connections between elements in the network; references specified by links are always resolvable, and the hypermedia network can guarantee the integrity of its links. Thus, traceability (within the network) is supported to a far greater extent.

Because the analyst's task environment includes a complex fabric of systems and sources, we need to balance the need for increased connectivity with other applications against the necessity of maintaining a consistent network.

1.2.2 NoteCards as a single-user system

NoteCards was designed as a single-user system, but in after-the-fact analysis, we found that many idea-processing tasks are collaborative. Similarly, in the informal analytic work practices study we found that many of the activities that the notetaker is designed to support have collaborative elements. So the notetaker's architecture must take multi-user (synchronous access) requirements into account.

Furthermore, in our vision of the future, as a logical extension to the personal information management function, the notetaker will be useful in building up a shared interpretive layer to many on-line sources of information. This extension is particularly important in creating and maintaining an institutional knowledge base that will persist after individual analysts move on. Thus this specification considers the minimum back-end (dbms-level) support for cooperative work.

1.2.3 NoteCards as "just" a prototype

At the time NoteCards was built, the Interlisp-D environment provided the developers with an ideal setting for rapid prototyping. But as the system grew, and its user community began to include people who were not familiar with the underlying programming environment, we began to notice that users ran into difficulties interacting with Interlisp-D. For example, the break windows that supported quick debugging were mystifying to the non-programming user. Furthermore, while NoteCards stabilized and

became progressively more robust, the underlying programming environment still relied on the idea of an "informed user."

Characteristics inherited from the development environment will greatly influence the notetaker's reliability and how easy it is to use. Therefore, in developing a hypermedia prototype to meet the analysts' needs, the implementor should take advantage of existing reliable software (that is more or less standard) to provide a robust framework for the notetaker. Such a framework will also make the implementation effort easier and ensure the resulting system's future extensibility. Several elements that should be present in the notetaker's environment:

- a standard windowing system and windowing toolkit
- user interface-building toolkit
- a commercial database management system to handle back-end functions

1.3 User requirements

This specification is derived from two kinds of user requirements. One stems from general knowledge of the analytic activities the notetaker is trying to support; the other stems from the notetaker's computational setting. Thus, we must not only consider the tool's suitability to task and how easy it is to learn and use; we must also consider compatibility with existing computation resources, including (but not limited to) databases, applications programs, and mail systems.

1.3.1 Accommodating the way sources are used

Analysts generally stress the importance of being able to trace information in an analytic product back to its sources. This implies that information or interpretations that appear in a document should be linked to the notes and sources necessary to retrieve the text's origins. Therefore, wherever it is possible (as in the case of annotative notetaking), the notetaker will build and maintain this type of link for the analyst.

Furthermore, analysts use a variety of sources. Which sources are important will change over time and with the task. The functional requirements should take the diversity of sources into account by specifying a partially open world, where the hypermedia network can access a universe of documents stored in a variety of places including some subset that are maintained by the notetaker itself. Thus the notetaker will control all of the linking information and some of the nodes.

By allowing sources to be annotated by analysts' notes, a community of analysts should eventually be able to build layers of public and private interpretation over institutional databases. Notes that are public can become an internal resource, provided that there is some kind of incentive for contributing to this shared interpretive layer. Augment's Journal System is a good example of support for this kind of collaborative work [Englebart & English 1968].

1.3.2 Supporting a range of notetaking styles

Analysts use three distinct styles of notetaking, annotative, integrative, and reminding, all of which

appear to be important (although they function in different ways); Figure 1.1-2 provides a summary of the distinctions among these styles. This range of styles suggests some important kinds of notetaking operations: (1) Highlighting source text or marking on images or graphics; (2) Attaching marginalia to source text; (3) Taking notes that integrate material from several sources; and (4) Taking reminding notes which can tag any object in the system. Both interpretive and reminding notes can also be sourceless - that is, the notes produced may not refer directly to any source or any object in the system.

To support notetaking that involves delimiting regions of text, images, and graphics, the system should provide highlighting and other on-line mark-up capabilities. Sources should be protected from modification, but the analyst should be able to view his or her marks with the source.

To support marginalia, the notetaker should provide capabilities to attach notes to the source material. The notetaking system should maintain a link from the comment or annotation to the original source. [Brown 1985] discusses functionality for supporting this style of annotative notetaking for text in a system called Annoland.

The current NoteCards system is designed with interpretive notetaking in mind; the notes are not seen as so tightly tied to the original text as in annotative notetaking. To support this style of notetaking, the notetaker should help the analyst maintain links to sources, and should provide NoteCards-like functionality for the written interpretation that is usually a precursor to the authoring process (see [Trigg & Irish 1988] for a discussion of hypertext habitats for writing). Sometimes an interpretive note integrates material from more than one source; in this case, the notetaker should support links with more than two endpoints.

Reminding notes, the product of the third kind of notetaking identified by our work practices study, result from a substantially different mode of working. Reminding notes may not refer directly to any source, but may document an analyst's procedure for retrieving some piece of information, or it may set his or her agenda for a given period of time, and thus refer to other elements in the system like calendar programs or agenda systems. Therefore, to support reminding notes, a notetaker should support link annotations (for procedural documentation) and the ability to link reminding notes to any other element in the system, including applications programs.

1.3.3 Supporting a range of filing strategies

Analysts use a variety of filing schemes, some transient, some long-term, depending in part on the ultimate goal of filing. Some filing is done in support of a particular task, like organizing materials in preparation for writing a paper. Other filing is performed with the goal of building up a personal or shared information resource; filing in this case must provide easy access to materials.

Therefore, filing operations must be flexible enough to support complex handmade organizing schemes, and extensible to support automatic and system-assisted filing schemes. Basic filing operations include (1) grouping items together; (2) organizing items within a group; and (3) organizing groups within

a structure. Content analysis technology should eventually be incorporated in a notetaker to provide a automated method of clustering and filing notes; filing schemes like chronological organizations can be system-assisted, but may be user initiated. On-line filing should provide some obvious advantages over physical filing systems such as allowing items to be filed in multiple places and providing advanced methods for retrieving filed items (see Section 1.3.4).

1.3.4 Assisting in the retrieval of notes and files

It is important for a notetaker to help analysts with both retrieval (to find specific items) and browsing (to help them understand the shape of the space they've built). Thus the notetaking facility should provide an overview of an analyst's personal information space (that is better constrained and easier to use than the NoteCards browser) along with good retrieval facilities (incorporating more advanced strategies than keyword-in-context searches). It is essential that notes and filed information not get lost or be perceived as lost.

1.3.5 Supporting the reformulation of notes into drafts

Written products are the significant artifacts of the analysis process. Therefore, facilities to support their production from notes are essential to making the notetaker useful. Most analysts do some reorganization of materials (notes and filed sources) to support their authoring task; they may re-read them in a new organization to help create coherence from amassed information. Some analysts also use outlines to lend task-specific coherence to their notes. Thus the notetaker should support the re-organization and composition of notes.

Furthermore, it is critical for the analysts to be able to trace backward from the draft analytic product to the notes and sources used to produce it. The notetaker should provide a specialized browser showing the ultimate sources of information included in any given product, and the notetaker's interface should make it easy to traverse links in either direction to get back to sources.

1.3.6 Supporting the draft-passing process

Informal draft-passing is part of the writing process. Analysts assimilate comments from co-authors and from colleagues before a report is completed. Comments are produced in a manner akin to annotative notes - comments are attached to specific segments of text. Support for the annotative style of notetaking (both commenting and mark-up) should be adapted into a draft-passing mechanism that makes the comment and revision process easier. A good example of hypermedia support for draft-passing is provided by Intermedia's InterNote application [Catlin et al. 1989]. In particular, InterNote's concept of warm linking should be adopted to make the revision process easier for the analyst.

In supporting draft-passing boundary issues - how interconnected text can be passed around, in and out of the notetaker - come into play. Thus, the notetaker should adhere to integration strategies that allow a draft to exist in a heterogeneous environment. These integration strategies will also be important in passing the final draft on to the publishing system.

1.3.7 Accommodating both personal and shared notes

Because some aspects of searching, interpreting, and writing are collaborative activities, it is important to provide a task-sensitive shared workspace as well as a secure personal notetaking environment. Not only are notes and documents important elements of a shared workspace; links also play a major role in information sharing. An analyst should be able to specify who can follow (or indeed see) a link he or she has made.

1.3.8 Using user interface metaphors appropriate to the task

Metaphors are powerful devices in making information accessible (or inaccessible) and easy to manipulate (or intractable). For example, a pervasive desktop metaphor may cause a user to think that small fonts make his document shorter, thus taking up less disk space. In a notetaker, interface analogies to physical methods for taking notes and spatial filing systems should make the notetaker easy to learn and use.

1.3.9 Ensuring reliability and preventing data loss

One of the strongest criticisms of the current NoteCards system is its lack of reliability. The notetaker should not only be built on a reliable storage substrate to ensure consistent completion of database transactions; it should also, in the rare event of a media failure, provide mechanisms for returning the database to a consistent state. Use of proper backup procedures should also help guard against significant data loss.

A second requirement for preventing data loss concerns unintentional deletion of information. A "trash can" model of two-step deletion should be included in the notetaker, so that analysts have the opportunity to recover unintentionally deleted material.

2. HYPERMEDIA DATA MODEL

The hypermedia data model will contain abstractions to implement the conceptual framework for creating, manipulating, and traversing information in a network. The two basic constructs in this network will be nodes that hold the information content, and links that represent the relationships between the nodes. Sections 2.1 and 2.2 provide detailed descriptions of what the elements of these components are and what roles they will play in the notetaker.

The hypermedia data model will also include a third abstraction to implement composite nodes. Composites will enable a user to build and refer to complex aggregates of nodes. Section 2.3 describes this higher level abstraction.

2.1 Nodes

In our notetaker, a hypermedia node will be the component of the system that maintains the content of the network (as opposed to the links, which define the structure of the network). The nodes must therefore be containers for the types of information elements that an analyst uses. From the work practices study, we have evidence that these information elements must include: source documents (which may include page images of papers as well as their text), source images, drawings, interpretive notes, annotative notes on documents and imagery, reminders and to do lists (informal and procedural notes), drafts of analytic products in progress, and mail messages and cables (as well as other "structured text"). More diverse uses of the notetaker in the analytic environment will include information elements such as complex graphics, expressions of mathematical models (where a node may be a simulation), and other specialized types of source and interpretive material.

Thus we must consider aspects of nodes that require uniform specification such as what they are, how their components function, what kinds of operations the user can perform on them, and how the user interacts with the node's contents.

2.1.1 Role of nodes

As elements of a notetaker's hypermedia network, nodes are the primary containers of information (text, images, etc.) in the network and the source and destination points for links.

2.1.2 Parts of nodes

A node should include three different types of information:

- identification
- properties
- content

At least two kinds of node identification are necessary, a unique identifier for the system's use, and a name by which the user can refer to the node; this name can be either user-supplied or system generated

by a method negotiated between system and user. For example, the user may wish for nodes to be identified by his or her initials and a time stamp. Node identification may also be supplemented by keywords (which again may be user-supplied or computed).

Each node should have the provision for node properties, information either supplied by the user or the system that enables regular programmatic interaction. Properties should be in the form of attribute-value pairs or some other appropriate data structure. Some of the node properties such as author or creation date should be protected and only used by the system for its own bookkeeping; the analyst should be able to name other node properties and supply their values; for example, some analysts may want to assign a confidence level to their notes. Properties will also be the means by which an applications programmer can specialize implementational node types (see Section 2.1.3) with structured information; this structured information will be helpful in extending the system to respond to new requirements. For example, a property list mechanism can help the notetaker respond to security requirements such as node classification. See Section 7.3.2 for a discussion of a classification property.

Finally, each node will contain some kind of unstructured or semi-structured substance (for example, text, a page image, or one of the other kinds of information mentioned briefly in Section 2.1).

2.1.3 Node types

The notetaker will provide two node typing mechanisms. One mechanism will support *implementational* node types; the other will support *representational* node types. An implementational node type is characterized by how the system stores and displays the node's contents. A representational node type specifies the node's semantic role in the network. Usually, the work of specializing an implementational node type or creating a new one is done by an applications programmer. On the other hand, representational node types will usually be defined or extended by an analyst in response to some aspect of a task.

For example, in NoteCards some implementational node types are *text*, *sketch*, and *graph*. Each of those node types requires a different kind of editor and different data structures. These three implementational types have been specialized into subtypes such as document cards, idea sketches, and browsers, respectively. Occasionally a NoteCards application demanded a new implementational type like an animation card or a Toulmin card; a programmer performed this kind of tailoring.

In the notetaker, a core set of implementational types should be provided to handle the types of information that an analyst uses most frequently. In particular, there should be node types that will allow an analyst to view and edit text, bitmaps, and simple graphical objects. Because there is a need for computed overviews (see Section 5.2.3), an implementational type that supports graphs or other varieties of structure browsers is also necessary. A mechanism for creating and specializing implementational types will allow the notetaker to be extended to handle new forms of information (such as videotape) or specific forms of more general types (like PINNACLE bitmaps or imagery).

Representational node types categorize a node according to the role it plays in the network; a representational type may provide some kind of embedded structure like fields or properties to be filled in, or it may constrain the kinds of relationships the node can have to other nodes in the network. For example, in a hypermedia system for analysis, a representational node type might be a *note*. This representational type might have a special property like relevance, and be constrained to require a source link. This representational type might then be subtyped to include specialized types of notes like hypothesis, annotation, and list.

The two typing mechanisms should be independent; thus each node will have both a representational and implementational type. In the example above, one note might be based on the text implementation type, and another note might be based on a bitmap. It is important to include representational and implementational node typing in the notetaker, both for specifying the types needed for the prototype, and in planning mechanisms for extensibility.

Because representational types are generally created by users, the representational typing mechanism should have an interface that allows analysts to interact with the mechanism in a simple, non-programmatic way. On the other hand, the implementational types mechanism should provide a programmer's interface (similar to the NoteCards types mechanism) so an applications programmer can extend the system.

2.2 Links

Links are the hypermedia component that encode relationships between nodes or portions of nodes. Links are represented as objects with a defined set of elements that perform a role in the notetaker - to connect notes with their sources, drafts with the notes they were written from, notes with related notes, and so on. Links should be anchored or attached to a span within the nodes and their presence indicated with markers so a user can better ascertain their existence and purpose.

2.2.1 Role of links

There are two different notions of linking in hypermedia systems. One is called a *reference*; it implements a link as a component within a node that contains a name or address that refers to another node (or a region within another node), or a procedure for retrieving that node. Figure 2.2-1 schematizes this notion of linking. In the case of a reference link, the node being referred to does not keep a record of the link. On the other hand, a *connection* implements a link as a component that connects a node or region within a node with another node or a region within it. Figure 2.2-2 shows an example of the connection style of linking; the objects at both ends of the link "know" about the link. For the purposes of the notetaker, it is useful to think of both kinds of links, since we wish to keep the system open to applications that support the linking protocol. Section 2.2.3 discusses how connection links will be used in the notetaker and Section 6.1 discusses how reference links will be used in conjunction with a layered linking protocol.

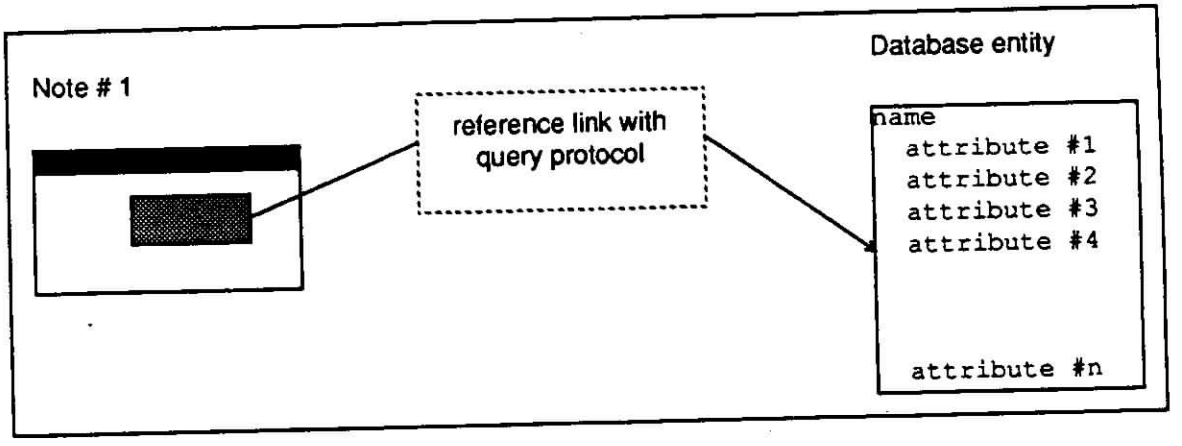

Figure 2.2-1 Example of linking by reference; database entity doesn't "know" about the link.

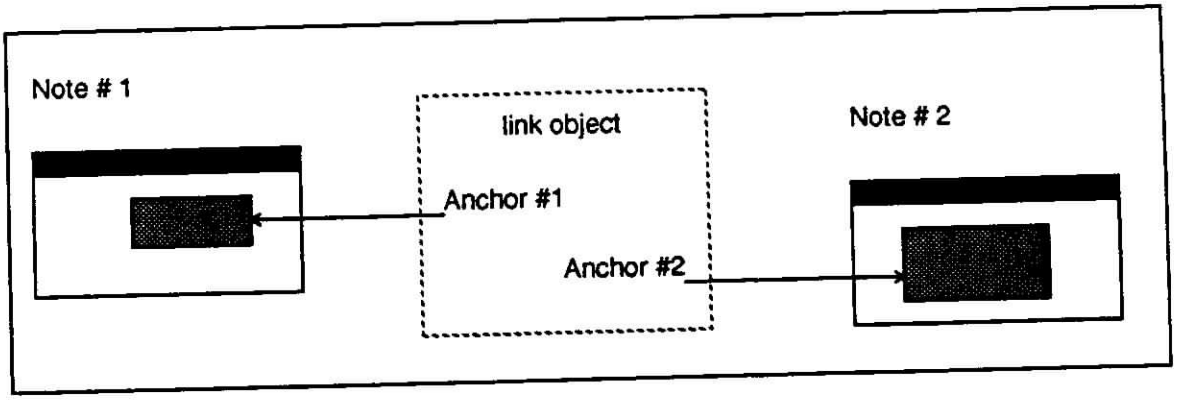

Figure 2.2-2 Example of connection-style linking; both ends "know" about the link.

Because we expect a variety of relationships between nodes (for example, an analyst might want to specify relationships like *source*, *supports*, or *refutes*), links will need to be *labelled*. Furthermore, since we expect links to have different characteristics, links will also have *types*, so that a behavior can be associated with the named link. In NoteCards, we have found that the ability to specify the *directionality* of a relationship to be somewhat difficult for users; however, we still feel that representation of the direction of a link is useful for certain types of relationships.

The need for both link typing and link labelling presents a requirement for types mechanisms parallel to the node typing mechanisms. Thus the notetaker should provide a representational and implementational types mechanism for links. A representational link typing mechanism will allow an analyst to specify the role a link plays with respect to the nodes it connects - for example, a user may decide that an evidence node can only be connected to a hypothesis node by a supports or refutes link. An implementational types mechanism for links will allow an applications programmer to extend the system by specifying different link behaviors.

Current hypermedia systems do not give their users the ability to link to links; however in the

notetaker it may be a valuable way of performing a kind of procedural annotation, functioning much the same as reminding notes. A limited facility for linking to links would allow an analyst to remind him or herself how or why a given link was made, or it might explain a little bit more about the relationship between the linked objects.

Current hypermedia systems also do not give their users the ability to specify n-ary links - a link with two or more endpoints. However, in the notetaker, n-ary links are important to representing the relationships implied by what we've called integrative notes. An integrative note synthesizes the information in more than one source (see Section 1.1.2); hence, the link from the note to the source would require multiple endpoints to accurately represent what is going on in the process. Figure 2.2-3 illustrates an n-ary link example.

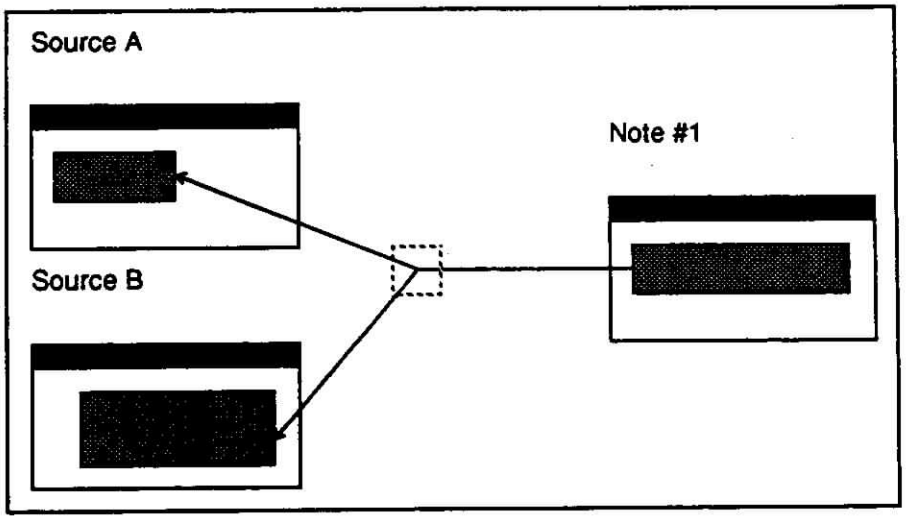

Figure 2.2-3 Example of how n-ary links will be used in the notetaker

2.2.2 Link elements

Links - as we have described them in this specification - have several main elements:

- specifications of the link's endpoints
- link properties (e.g. implementational and representational type, direction, and arbitrary attribute/value pairs)
- user annotation

Because endpoint specifications will be either explicit connections or unresolved references (as in the case of a link to an item or table in another database), the link must adhere to a protocol specifying anchor and display information, as well as information about which objects are at the endpoints.

Information related to the link typing mechanisms and link properties must be stored on the link as well. A user-designated label, specifications of methods for implementing the link's behavior (such as a method for resolving what's at the endpoint of a computed link, or special display methods associated with

implementational types), and an indication of the link's direction (if any) are basic subparts of a link. A provision for additional link properties should be provided to allow the analyst or applications programmer to associate arbitrary attribute/value pairs to the link. Attribute/value pairs will support programmatic extensions (through the implementational typing mechanism) of link information; for example, the classification level of the relationship represented by a link can be made a link property. (See Section 6.3.2 for a discussion of assigning classification levels to hypermedia elements.)

Simple textual annotations should be supported to provide a lightweight method of explaining or commenting on links; the user should be able to request these explanations in the process of traversal (like Intermedia's link explainers [Garrett et al. 1986]). Link annotations will be a first cut on supporting reminding notes on links. The system should ultimately support the idea of links to links, where a link's endpoint specification can be another link.

2.2.3 Link anchors and link markers

A link anchor is the span within a node corresponding to the endpoint of a link. In some hypermedia systems the span may be limited to a single point (as in NoteCards) or to the entire node (e.g. gIBIS [Conklin & Begeman 1988]). Other anchoring schemes may allow anchors to encompass arbitrary extents of text (or graphics) within a node. A link marker is used to signal the existence of the anchor.

2.2.3.1 Link anchors

In the notetaker, linking will primarily be implemented with the notion of connection (see Section 2.2.1). The link anchors should implement "span-to-span" links, where an arbitrary region or collection of objects can be connected with another arbitrary region or collection of objects in another node. Intermedia provides a good example of this kind of linking [Garrett et al. 1986]; Figure 2.2-4 illustrates span-to-span links. Span-to-span linking is important to the notetaker because the notes that analysts take generally refer to a subregion of the text (or an area of an image) within a node. Furthermore, it is important to identify which parts of a multi-source note or a document refer to which sources.

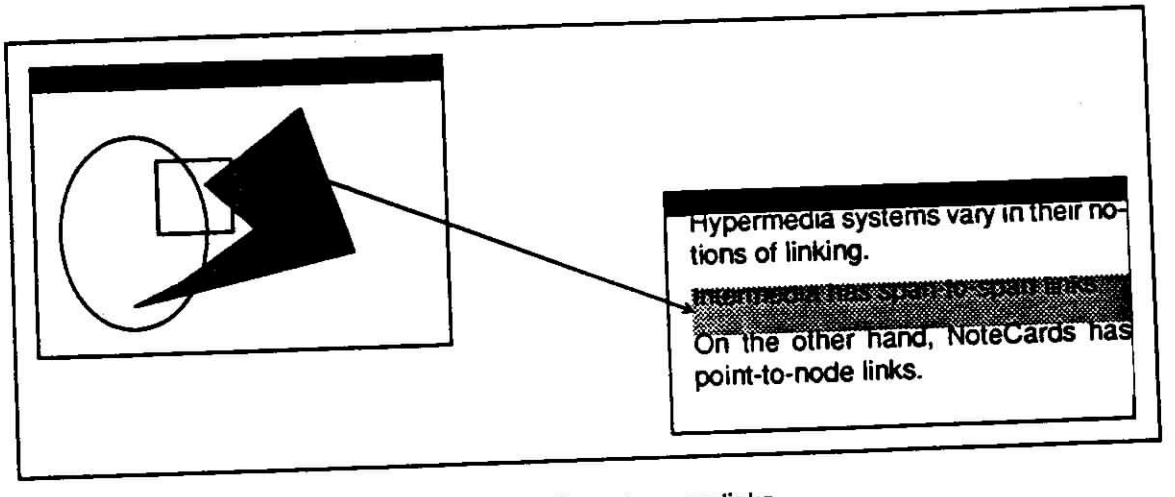

Figure 2.2-4 Span-to-span links.

Span-to-span linking will support many of the kinds of annotative notetaking that we've observed. The anchoring and marking process is similar to the highlighting that analysts use to set apart a region of text. In this case, it is the delimiting of text that is important; a special link type can support this span-to-null link. The ability to include marginalia as annotations may also initially be simulated using a span-to-node link. See [Catlin et al. 1989] for an example of how span-to-span linking can support annotation.

In order for anchoring to be implemented across applications outside the notetaker, the applications must follow an anchoring protocol. See Section 5.1.1 for a discussion of how outside applications can participate in connection-style linking.

2.2.3.2 Link markers

Link markers are the method by which the system indicates the presence of a link anchor to the user. In the current implementation of NoteCards, markers are link icons; many of the IBM PC-based hypermedia systems use highlighted text to show the link anchor (and its extent). What information a link marker displays should reflect its function. Link markers in the notetaker should allow an analyst to

- detect the presence of a link without requiring extra action
- distinguish the level of integration of the link's destination (see Section 6.1)
- determine the scope of the anchor's span

It is particularly important that the analyst be able to distinguish the level of integration of the link's destination (see Section 5.1), since it signals whether traversal involves another application. Other information that might be given by the link marker (such as destination node type, link type, or destination node title, as in NoteCards, or even the link's direction) can be presented to the user during traversal (see Section 5.1).

2.3 Composites

A composite structure is a node that *includes* other nodes from the network; composites are a mechanism for manipulating groups of nodes as discrete entities within the hypermedia abstraction. A composite node can be linked to any other node, for example, and abstracted from what may be a complex internal structure. Figure 2.3-1 shows schematizations of two hypothetical composite nodes, one called an *interpretation*, and a second called a *Briefing Board*. The first includes a hypothesis node and several notes serving as evidence, and the second has a fixed length text node called a Description and an image.

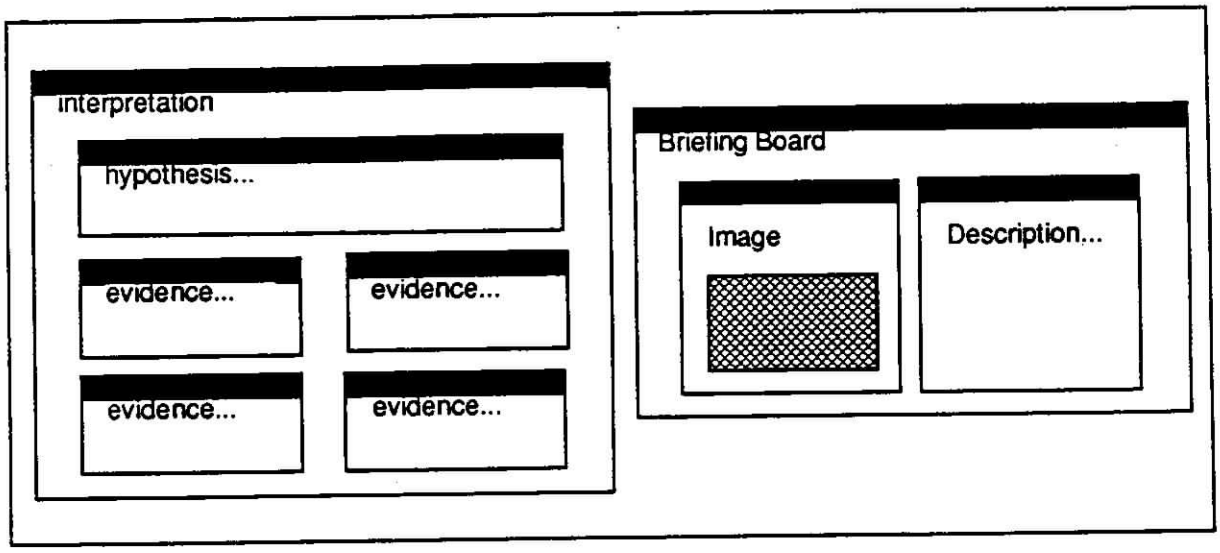

Figure 2.3-1 Two examples of composite nodes

3. BASIC HYPERMEDIA FUNCTIONALITY

Basic hypermedia functionality will enable a user to create, delete, and manipulate nodes and links. It will also support the creation, deletion, and manipulation of higher-level objects including containers, organizers, and composites.

3.1 Operations on nodes

Given an individual node in the hypermedia network, a user will need to be able to perform operations on its various parts. Some of these operations are general - that is, a user should be able to apply them to any type of node in the network. Section 3.1.1 discusses operations that should be available on all nodes to which a user has appropriate access. Others operations are specific, according to the type of content that node has or the role it plays in the network. These will vary from node to node, but should include the operation necessary to edit it or otherwise modify its content (such as recomputing it or, more generally, respecifying the way it is computed).

3.1.1 General operations on nodes

General operations on nodes will apply to all types of nodes in the network, regardless of their content or type. A user should always be able to view, edit, and remove from the display all nodes to which he or she has access rights (see Section 7.3.1) in the network. A user should also be able to create, save, and delete nodes except those nodes reserved by the system for special purposes (for example, the NoteCards Table of Contents, Orphans, and To Be Filed fileboxes). Operations on nodes will give a user the ability to name nodes, specify node composites (i.e. group nodes together into a composite object which is also a node), specify organization (see Section 4.2), link some or all of its contents of a node to another node, see any of the node's "hidden" properties or structured information and edit them if appropriate, and be able to find all the to and from links for that node. The representational types mechanism should also make it easy for a user to change a node's representational type (i.e. respecify what role it plays in the network). In a multi-user situation, an individual should be able to check nodes in and out of the network for read or write access.

3.1.2 Content-specific operations

Content or type specific operations will vary, depending on how the user is expected to interact with the content of a node. In many cases, this means that the operations are those inherited from the editor that makes sense given the node's implementational type (see Section 2.1.3). For example, a bibliography card may have an alphabetization operation.

3.2 Operations on links

Operations on links will apply to all non-protected link types in the network. The user should be able to:

- create links and specify associated anchor spans (see Section 3.3)

- delete links
- add, move, or remove link endpoints
- annotate links
- add, delete, or change link properties
- change representational types (change labels)

3.2.1 Operations on anchors

Users should be able to specify the extent of link anchors (if an anchoring protocol is supported by the nodes at either end) and change the anchor's scope without deleting and respecifying the link. It should also be possible to edit material within the anchor.

3.2.2 Operations on markers

Link markers should provide an interface to support the following user operations:

- link traversal
- viewing link annotations (see Section 2.2.2)
- viewing and editing link properties
- changing the link's label

3.3 Operations on composites

Once they are formed, composites should behave like a node; that is, they should be subject to all of the general node operations. In addition, a user should be able to form composites by specifying what nodes are to be included, and change composite nodes by adding new subnodes and deleting subnodes from the composite. A delete of a composite node should not affect the included nodes.

4. MANAGING THE NETWORK: CONTAINERS AND ORGANIZERS

Filing and other activities to organize notes and sources (putting papers in stacks or writing the outline for a paper, for example) are an important part of the analytic task (see Section 1.3.3). Information is managed in larger units as well - files are stored in drawers and desktops sometimes organize all the information needed for a particular assignment. Thus, in order for a hypermedia notetaker to be useful in this environment, it should implement some higher level constructs for the management, organization, and structuring of information. The notetaker will include two such constructs, containers and organizers.

4.1 Containers and their function

Containers are a construct for storing networks or meaningful subnetworks of nodes and links in the system's operating environment; they are the interface between the hypermedia system and its underlying file system. In NoteCards, containment is implemented as notefiles, in KMS as framesets, and in HyperCard as stacks. Each provides the user with a mechanism for partitioning large collections of nodes and links into useful, named subsets. For example, an analyst might want to define and name a container that holds all her information about a particular technology.

Each node and link should belong to one and only one container; this restriction will help the notetaker provide a consistency guarantee within a given container. The notetaker should also implement the notion of cross-container links, so that a node in one container can be linked to nodes in other containers. Similarly, composites might refer to nodes from several containers.

Operations on containers should correspond to file management operations; they are a convenient unit of hypermedia for performing backup, for scavenging, for compacting or compressing networks for efficient storage, for copying and renaming networks, and for removing old networks from the storage medium.

Operations on containers should give an analyst the ability to:

- Create a new container
- Delete a container
- Replicate a container (as for backup)
- Name or rename a container
- Perform consistency checks and repairs on a container

4.2 Organizers and their function

Organizers will provide analysts with a second level of information management; they are structures that are used within containers to hierarchically organize a collection of nodes and links. Organizers fit very naturally into the analyst's way of structuring information in file folders and shoeboxes.

NoteCards implements organizers as fileboxes. In our experiences with NoteCards, we have found

that fileboxes incorporate three basic kinds of functionality that are important for implementing organizers. The notetaker's organizers should:

- provide a hierarchical filing mechanism
- disallow cycles in the hierarchy
- prevent the creation of disconnected subgraphs and provide access to all roots

Organizer operations should allow analysts to perform actions on filing hierarchies. An analyst should be able to:

- create an organizer
- delete an organizer (without losing or deleting nodes)
- name (or rename) an organizer
- specify what nodes belong in the organizer
- replicate the subnetwork specified by the organizer
- move the subnetwork specified by the organizer

5. ACCESSING AND VIEWING HYPERTEXT INFORMATION

It is vital that the notetaker provide adequate access to the information it contains; otherwise, analysts will not have the required level of control over their notes and may lose information simply by not being able to retrieve it from the network.

The notetaker should include mechanisms for traversal-based access that help the analyst maintain his or her orientation in the network (see Section 5.1). It should supplement basic traversal mechanisms with ways of producing alternate views and network overviews (see Section 5.2). It should also provide a means of querying the database of nodes and links (see Section 5.3). Linearization capabilities will give an analyst a way of viewing a non-linear network in a form closer to its presentational form (see Section 5.4).

5.1 Link traversal

Link traversal is the primary way a user moves from node to node. We consider several different aspects of link traversal that are important in specifying a mechanism for the notetaker. First, the mechanics of traversal must make it easy to navigate through the network. Second, rhetorical information must be available to the user so she can determine where she is going and why. Finally, mechanisms for maintaining orientation must be provided to supplement the basic traversal mechanism.

5.1.1 Mechanics of link traversal

In most of the systems we have examined, the primary traversal means is simple - the user clicks on a link marker with some kind of pointing device, and the link's destination appears on the screen or some region of the screen. The notetaker should incorporate this point-and-click style of link traversal.

In the case of a connection style link, link traversal is simple, and should be transparent to the user. However, if the link is a reference to an node external to the notetaker, resolution will sometimes be difficult (if for example the database the node is stored in has moved), or impossible (if for example the whole database or particular database entry has been deleted). In those cases where the system cannot resolve the reference, additional user interaction should supplement normal link traversal mechanisms.

5.1.2 Rhetoric of link traversal

The rhetorics of departure and arrival are an important way a user can establish his orientation in a multi-dimensional space like a hypermedia network [Landow 1987]. To perform multiple step link traversal, a user must be able to find the answers to several basic questions: (1) Where am I going if I follow this link; (2) Why should I follow this link; and (3) What is the relationship between this node and the one I just left. Although rhetorical devices of this sort may only seem important for traversing someone else's web, in practice, we have found that informed traversal is necessary for finding one's way even in a personal information space to interpret work done in the past.

The notetaker's link marker display and operations should help a user answer these questions. To answer question (1), it is important to give the user information about the link's destination(s). In the event

that the user has supplied node identification, the notetaker should use it to provide a preview of the destination. The link marker should also indicate whether the destination is (a) in another container, or (b) maintained in an application outside the notetaker. This kind of information will enable a user to bail out of high overhead traversals (such as those involving remote database access). Viewing the link annotation (see Section 3.2.2) will help the user answer question (2). The notetaker should provide several answers to question (3). First, the notetaker should highlight the destination link anchor after the link is traversed (as is done in Intermedia); second, the user should be able to view the link's label, indicating its representational type; third, the user should be able to browse any additional link properties that are available (for example, one assessing relationship strength).

5.1.3 Link traversal and orientation

It is well documented in the hypertext literature that users traversing through a large or unfamiliar network can become disoriented [Conklin 1987]. Mechanisms in the notetaker should help the user become reoriented when this occurs. Thus the notetaker should include a *traversal history*; a traversal history will enable a user backtrack quickly and easily. Implementations of a backtracking mechanism include HyperCard's iconization of nodes recently visited (so a user can jump back to that place in the stack), and KMS's 'Back" button at the bottom of the screen that allows the user to go back one link. Hyperdoc also maintains a network representation of where the user has been to enable her to backtrack to any point in her traversal history. The notetaker's traversal history mechanism should use incorporate a "jump back" capability, and a representation of nodes visited.

The notetaker should also include a *traversal overview* - a map that shows the 'neighbors' of the current node *ala* Intermedia's local tracking map. Figure 5.1-1 shows an example of such a map. In a multi-window system, the notion of current node is a little hard to pin down, but it can be thought of as the most recent destination card, which is presumably the focus. For a more extensive specification of the notetaker's overview facilities, see Section 5.2.

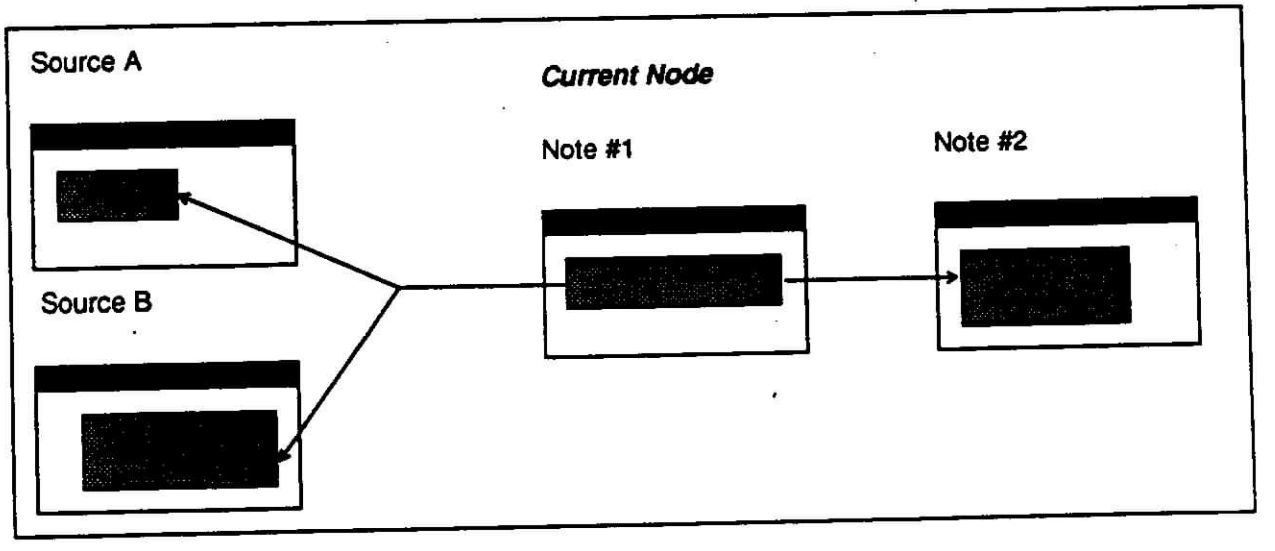

Figure 5.1-1 Example of a traversal overview

5.2 Views and overviews

A view is a specific rendering of a node or link. The current generation of hypermedia systems does not support multiple views, but rather displays nodes (and links) the same way regardless of how the user arrived there; given fixed contents, the node always looks the same. In NoteCards, for example, text nodes are always displayed using the Interlisp text editor. However, we expect views to be a useful part of the notetaker. For instance, a Read-only view could display protected text; a Secret view could display only the contents of a node that were classified at the Secret level or below; a Brief view could display only the first paragraph of a note.

An overview of the network provides a presentation of structure within a document or network. Overviews have multiple functions. They can provide a means of searching the network; they can help a user get reoriented; and they can give the user a spatial/relational description of a complex structure. The notetaker should provide several types of overviews including:

- Indices
- Outlines and tables of content
- Computed network diagrams
- Manually laid out network diagrams

Indexing (*ala* Hyperties [Schneiderman 1987]) takes little advantage of the spatial or relational structure of a network, but it is a useful access mechanism. Outlines and tables of content provide a good means of examining a topic hierarchy and show some of the relational structure of a network. Computed network diagrams (*ala* the NoteCards browser) represent more of the semantic (relational) content of the network through graphic devices such as horizontal and vertical positioning, proximity, and spatial annotations (lines, boxes, arrows, and so on). Named specializations of styles of computed network diagrams will help users request useful overviews from the system. Through the use of manually laid out network diagrams (*ala* gIBIS), the spatial and relational structure of a network can be represented in a way meaningful to the user. Because each form of overview has its uses, we believe that each form should be supported in the notetaker.

5.2.1 Indices

The notetaker should generate at least the following kinds of indices upon request:

- Index by node name
- Index by node keywords
- Index by representational type (either link or node)

These indices will provide a quick access mechanism to get to any node in a large network. It should be possible to do some very simple filtering on the indices as well. For example, the notetaker

should be able to provide an alphabetical index of all authors (where "author" is a representational node type). The notetaker should present the results as a point-and-click list of nodes, except in the case of links indexed by representational link type, where the presentation will be organized from sets of nodes at the link's endpoints.

5.2.2 Outlines and tables of content

In the NoteCards user community, users have found topic browsers, representations of high-level content, and outlines to be useful constructs in working with a complex network. Even though these kinds of structures offer limited views of the full relational content of the network, they support certain kinds of analytic activities - organizing notes into a paper, for example, or setting up a hypertext draft for collaborative authoring. Thus the notetaker should have facilities to support overviews that are indented lists that can be used to manipulate hierarchical structure (*ala* Guide [Brown 1987]).

5.2.3 Browsers and network diagrams

Network diagrams *ala* the NoteCards browser show a graphical representation of a hypermedia network, using spatial relations and visual links to reflect its relational structure. Network diagrams are good aids for visualization - they allow a user to construct a spatial model of what's in her network. They provide a mechanism for analyzing dependencies and for looking at data interrelationships that are difficult to capture other ways. The notetaker should support a graphical browser that:

- can be computed automatically from a simple specification
- provides an active view of the network
- has facilities for automatic layout of the graphical view
- provides user layout tools
- can perform various types of filtering for named views.

5.2.3.1 Support for active views of a hypermedia network

In a passive view of a hypermedia network, users are not allowed to manipulate objects in the network diagram; in an active view, manipulation of the diagram produces the corresponding modifications on the network represented by the diagram. Supporting direct interaction with the representation is a good way of enabling analysts to work top-down, as well as facilitating restructuring activities that occur in the course of a lengthy analysis. In an active view of the network, operations should allow an analyst to:

- create nodes
- create links
- aggregate nodes (as a composite)
- delete nodes
- delete links

- rename nodes
- change the label of links

5.2.3.2 Layout of a network.

When users request a computed diagram of a network, they expect layout to be automatic. However, when users manipulate the view, they often want to rearrange the layout to fit their visual images of how the network should look. Browser layout operations should include the means to:

- move a node
- move subnetworks
- align nodes
- automatically relayout specified portions of the diagram
- add labels to the diagram
- remove nodes from the diagram
- remove labels from the diagram

5.2.3.3 Filtering a view of the network

Our experiences with NoteCards have shown us that naive users are confounded by the prospect of constraining a browser. Therefore several types of filtered views should be available to the analyst to match their work practices. For example, a topic browser should show the organizing structure of the network; a traceability browser should follow notes to their sources.

More sophisticated users should be given the ability to intervene and create her own filtered view of the network by specifying path constraints. The notetaker should use an idea akin to a document editor's property sheet to allow these users to name and save these specialized filters so they can be reused.

5.3 Search

The notetaker should provide search mechanisms so the analyst has a way of finding information that may be spread throughout the network. A query facility that enables an analyst to match text patterns is the simplest basis for searching.

In later versions, the notetaker should include mechanisms for querying against a representational structure. For example, an analyst may want to ask the network whether it contains any evidence nodes without source links linked to hypotheses nodes about VLSI technology. See the discussion of structure search in [Halasz 1988a] for further description of the issues.

5.4 Linearization

The final product of an analysis is usually some kind of linear form, be it a formal analytic report, a short article, or a briefing. These forms may vary in length or intended audience, but they all have a relationship to the original notes that an analyst has organized.

Traceability back to original sources has been uniformly cited by the analysts as an important part of the bookkeeping that must be done during linearization.

Linearization should only include the capabilities needed to render a non-linear network linear; we anticipate that some of the formatting and other text and graphical layout design - desktop publishing - will be performed external to the hypermedia system. Other formatting will occur within the notetaker so that some preliminary draft-passing can be performed with links intact.

6. NOTETAKER INTEGRATION

Two different issues must be examined to ensure the notetaker's integration into its human and electronic environment. One has to do with connectivity - how an analyst will access different applications and sources of information from the notetaker. The other has to do with the necessity of importing information into and exporting information from the notetaker, both as linear files (to, say, be used by a desktop publisher), or as linearized hypertext to be reformulated into a network within another hypertext environment (say, a commercial system like KMS, or a custom application like the Analyst).

6.1 Levels of integration

Linking requirements coupled with the analysts' need to trace notes and finished intelligence back to its sources and their use of a variety of tools in the sense-making process, leads us to a multi-tiered integration scheme for the notetaker. Of the different tools and applications available in the analysts' environment, some will be more amenable to deep integration than others. Furthermore, we have found that the various kinds of notes that analysts take require greater or lesser connection to outside information, and that in some situations, the payoff for deeper integration is large, while in others, shallow integration is all that is necessary. Thus the notetaker should adhere to a system of integration protocols.

We have divided integration into three levels, listed in order of depth: (1) data or content based integration (see Section 6.1.1); (2) tool or node based integration (see Section 6.1.2); and (3) display or window based integration (see Section 6.1.3). This list suggests a need for three protocols, which we feel are general to embedding hypertext in a heterogeneous application environment: an anchoring protocol, a linking protocol, and a launching protocol. Figure 6.1-1 summarizes the relationship between the protocols and decreasing levels of integration.

DEPTH	PROTOCOLS		
	anchoring	linking	launching
data/content	●	●	●
tool/node		●	●
display/window			●

Figure 6.1-1 Relationship between protocols and depth of integration

The difference between the levels of integration should be as easy to discern (from the link marker) as possible. That way, there will be less confusion about what kinds of operations are available, and how deeply the node being displayed is tied to the network.

Figure 6.1-2 shows a hypothetical notetaking situation, where an analyst has taken a note referring

to three outside sources, one at each level of integration. The first text span of the note is integrative, and refers to the first two outside nodes; protocols tell the notetaker how to launch each application and retrieve the appropriate node. Because the node from the first application supports anchoring, the extent of the anchor's span is also marked. The note's second span of text refers to the entirety a node in the second application; linking is supported, but anchoring is not, so only the node can be retrieved and displayed. The third span of text in the notetaker's node refers to some portion of the application launched in the third window. Since neither linking nor anchoring is supported, the application can only be brought up in a window. The annotation on the third link object is the user's procedural note describing how to get the proper information from the third application.

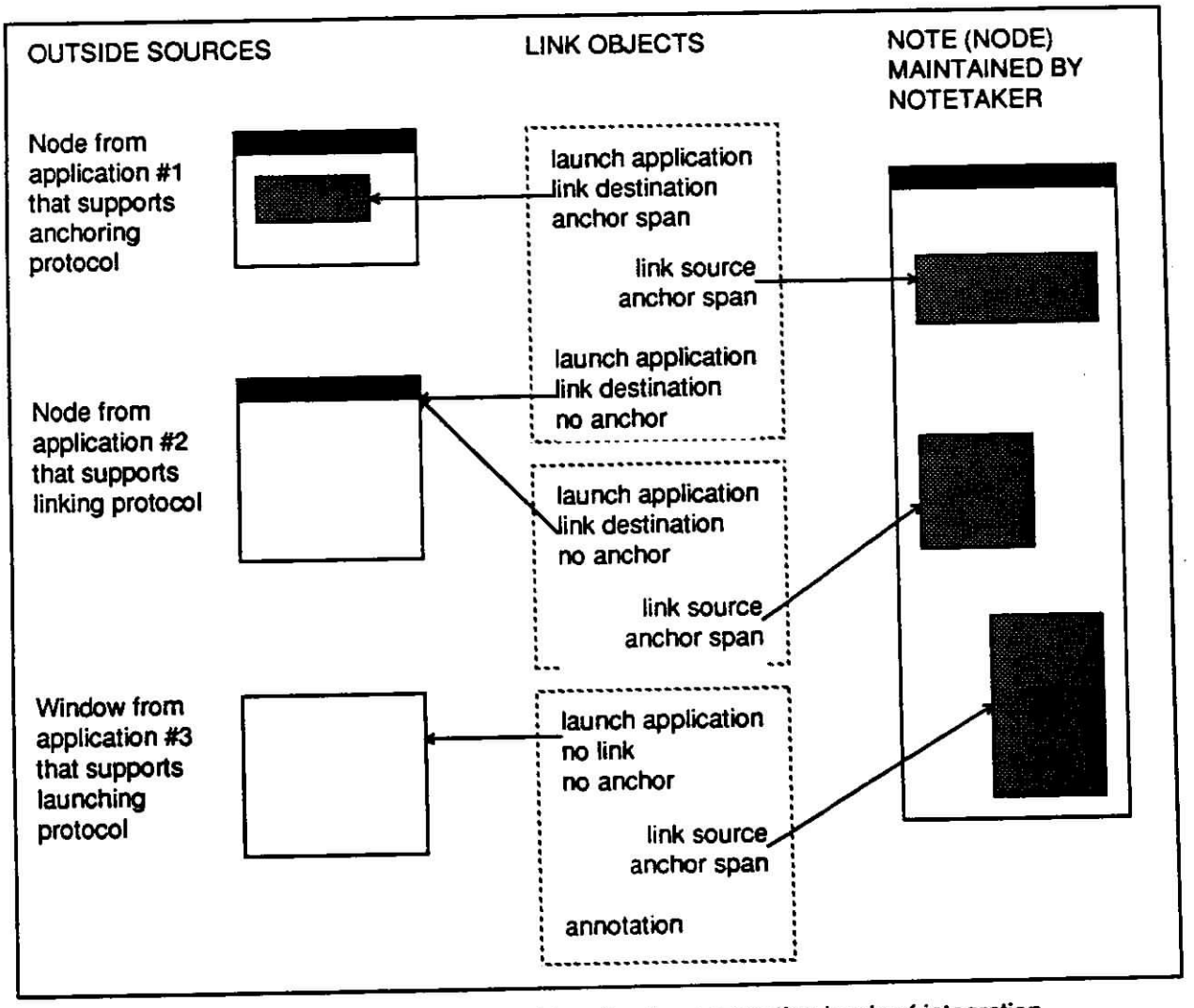

Figure 6.1-2. Hypothetical notetaking situation contrasting levels of integration

6.1.1 Data based integration

Applications that support protocols for data based integration will enable information to be fully incorporated in the hypermedia network, although it will not retain the container-level consistency guarantees (see Section 4.1). Integration of elements outside the notetaker at the data-item level will require

that their parent applications conform to three protocols:

- an anchoring protocol
- a linking protocol
- a display protocol

The anchoring protocol will describe the extent of the anchor within the node; the linking protocol will describe how to retrieve nodes outside the network; and the display protocol will tell the notetaker how to display the node it has retrieved in a window. This type of full reference will support the same kind of traversal that takes place between nodes actually in the hypermedia network.

6.1.2 Tool based integration

Tool based integration requires that the second and third types of protocols listed in 6.1.1 be supported by the application. These two protocols will be sufficient to implement simple reference links. In this case, we can think of traversing a link as a retrieval of a piece of information outside the notetaker.

6.1.3 Display based integration

In display-based integration, only the third type of protocol, a display protocol, must be supported by the application. At this level, the most superficial kind of integration, we can think of traversing a link as a launch of an application in a window.

6.2 Importing and exporting information

Two different import/export formats will be defined for the notetaker. One will specify how linear text moves in and out of the notetaker; other will specify a hypermedia interchange format.

A plain-text format for a linearized portion of the notetaker's network will be necessary for passing text on for publication, and reassimilating text that has left the notetaker during the review process. Knowledge of this format will also enable an analyst to import documents and other text into the network.

Conforming to a hypermedia interchange format will make it possible to write a parser for bringing other hypertext networks into the notetaker. It will also allow an analyst to use (at least some portion of) a text network stored by the notetaker in another hypertext system for which an appropriate parser has been written. These kind of interchange formats are important to users who put a great deal of effort in storing information in a hypermedia network; it reassures them that it can be reconstructed elsewhere or used in another situation.

7. GENERAL ISSUES

Section 7 discusses a grab-bag of issues that are raised by specifying the notetaker's functionality. Most of these issues have global implications on the notetaker's operation and, possibly, its architecture.

7.1 Tailorability and extensibility

Several kinds of tailorability and extensibility should be supported by the notetaker. One kind of extensibility involves defining the protocols for integration (see Section 6.1). Because the applications that the analysts use will change and expand, and new on-line sources of information will become available, these protocols are essential for ensuring that the notetaker will adapt to environmental changes.

A second kind of extensibility should be supported by a programmer's interface to the notetaker that provides access to user operations and objects in the system. More specifically, an applications programmer should be able to extend the system by creating new implementational types of links, nodes, and composites (see Sections 2.1.3 and 2.2.2).

The notetaker should provide tailorability to non-programming analysts in the form of a representational types mechanism, so they can name specialized types of nodes (like a "person" node) and constrain its relationships with other nodes ("persons" can only be linked to "institutions" by "membership" links, for example). Thus a user should be able to create new representational types of links, nodes, and composites (see Sections 2.1.3 and 2.2.2).

The notetaker should provide analysts with the ability to store specifications (say, for building a network overview) as data. An example of this kind of tailorability can be found in some text editors (for example Microsoft Word) which give users the opportunity to define styles; these are named and saved separately from the documents. Thus they can be re-used. The ability to name and store specifications will make complex operations (like browsing, searching, and linearization) more efficient for analysts (see Section 5.2.3.3).

The simplest kind of tailorability, parameterized control of certain aspects of the system (standard fonts and default values for system variables, for example), should provide the non-programming user with some access to the general behavior of the notetaker.

Both [Halasz 1988a] and [Trigg et al. 1987] contain a more detailed discussion of extensibility and tailorability issues for hypermedia systems.

7.2 Multi-user access

To meet the requirements of its user community, the notetaker should include mechanisms to support multi-user access. Two important mechanisms for multi-user access are locking and notification.

7.2.1 Locking

A locking mechanism will help with concurrency control - in essence, keeping two users from editing the same node at the same time.

7.2.2 Notification

A notification scheme is the other side of concurrency control; it allows users to know about each other and negotiate node access.

7.3 Security

The notetaker should support two kinds of security - standard access rights that enable users to protect and share on-line work in a multi-user setting, and classification levels that support the protection of sensitive information according to existing Government regulations.

7.3.1 Access Rights

In a multi-user situation, it is important to define levels of access at the grain-size of operations; therefore, access rights should be maintained for nodes and links. The standard Unix file system scheme of access rights (read, write, execute modes according to owner, group, and others) can serve as a model.

7.3.2 Classification

Since not only node contents can be sensitive, but also the relationships between nodes requires protection, the notetaker should support the ability to tag both nodes and links with classification level, . Since nodes and links have properties (see Sections 2.1.2 and 2.2.2), they already have a mechanism for representing this information and attaching it ubiquitously to objects in the network. Composites and network management constructs like containers and organizers should also understand how to inherit the appropriate classification levels from the nodes and links they include, structure, or contain. Operations that provide views or overviews of the network should also use classification levels in their computations.

7.4 Versioning

The purpose of versioning is to maintain a history of changes to the elements of a hypermedia network. Versioning is critical not only to analytic work, where it is important to keep a record of changes in data or interpretation, but also to the cooperative aspect of supporting a shared information space.

As an eventual goal, the PIE (Personal Information Environment) model [Goldstein & Bobrow 1980] provides a good conceptual framework for designing a solution to the versioning problem. Versioning, by that scheme, maintains branching *threads* for individual entities (including nodes and links), and allows both the versions and the deltas to be referenced. This type of versioning will address the requirements suggested by the need for record-keeping in the process of the analytic task.

Versions of sets of nodes and links are also maintained in this scheme in two ways: (1) A collection

of coordinated changes to a set of nodes is maintained as a *layer*, and (2) a composition of layers that produces a specific version of the whole network is maintained as a *context*. This type of versioning will address the needs of collaborative analytic tasks like draft-passing and cooperative review of analytic products.

As a simple first step for the notetaker, we will consider versions to be consistent layers of the network. In this scheme *every save forms a new layer*. In order to maintain consistency, users are never allowed to make a link out of the layer. Thus, versioning will initially be time-based - a user can back up to any previous consistent layer, but no time travel will be supported. That is, a user will not be able to make links between his or her time-based versions.

7.5 User interface

In a multi-window environment, each node will appear in its own window. The notetaker windows should use the same windowing environment as the windows of the existing tools and applications. The analyst should be able to cut and paste text and bitmaps between windows (and tools).

Links will be traversed by placing the cursor over a link marker and selecting the marker. Traversal will cause the link's destination to be viewed in a window (see Sections 3.2.2 and 5.1.3).

7.6 Backup and Recovery

The notetaker will include functions for backing up and recovering hypermedia networks. This is basic functionality that the notetaker should provide at the container level (see Section 4.1) to protect the analyst against significant data loss. The notetaker should include facilities for the analyst to perform her own container backups (if they are not performed regularly by a system administrator), and to restore a network from a previous backup.

8. REFERENCES

[Akscyn et al. 1988] Akscyn, R., McCracken, D.L., and Yoder, E., "KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations," *Communications of the ACM*, Vol. 31, No. 7, July 1988, pp. 820-835.

[Brown 1985] Brown, J. S., "Process Versus Product: A Perspective on Tools for Communal and Informal Electronic Learning," *Journal of Educational Computing Research*, Vol 1(2), 1985, pp. 179-201.

[Brown 1987] Brown, P.J., "Turning Ideas into Products: The Guide System," *Proceedings of Hypertext '87*, University of North Carolina at Chapel Hill, November 13-15, 1987.

[Bush 1945] Bush, V., "As We May Think," *Atlantic Monthly*, July 1945, pp.101-108.

[Catlin et al. 1989] Catlin, T., Bush, P., and Yankelovich, N., "InterNote: Extending a Hypermedia Framework to Support Annotative Collaboration," *Proceedings of Hypertext '89*, Pittsburgh, Pennsylvania, November 5-8, 1989, pp. 365-378.

[Conklin 1987] Conklin, J., "Hypertext: An Introduction and Survey," *IEEE Computer* 20, 9, 1987, pp. 17-41.

[Conklin & Begeman 1988] Conklin, J. & Begeman, M., "gIBIS: A Hypertext Tool for Exploratory Policy Discussion," *ACM Transactions On Office Information Systems*, Vol. 6, No. 4, October, 1988, pp. 303-331.

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

[Englebart & English 1968] Englebart, D. C. & English, W.K., "A Research Center for Augmenting Human Intellect," *Proceedings of the Fall Joint Computer Conference*, 1968, pp. 395-410.

[Evenson et al. 1989] Evenson, S., Rheinfrank, J., and Wulff, W., "Towards a Design Language for Representing Hypermedia Cues," *Proceedings of Hypertext '89*, Pittsburgh, Pennsylvania, November 5-8, 1989, pp. 83-92.

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

[Goldstein & Bobrow 1980] Goldstein, I.P. & Bobrow, D.G., "A Layered Approach to Software Design," Xerox Palo Alto Research Center Technical Report CSL-80-5, 1980.

[Halasz et al. 1987] Halasz, F. G., Moran, T. P., Trigg, R. H., "Notecards in a Nutshell," *Proceedings of the ACM CHI+GI Conference*, pp. 45-52, Toronto, 1987.

[Halasz 1988a] Halasz, F.G. "Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems," *Communications of the ACM*, Vol. 31, No. 7, July 1988, p. 836-852.

[Halasz 1988b] Halasz, F.G. "NoteCards: A Multimedia Idea Processing Environment." In S. Ambron & K. Hooper (Eds.) *Interactive Multimedia*, Microsoft Press, 1988.

[Halasz & Conklin 1989] Halasz, F.G., Conklin, J. "Issues in the Design and Application of Hypermedia Systems," Tutorial from CHI '89, Austin Texas, May, 1989.

[Halasz & Schwartz 1990] Halasz, F.G. & Schwartz, M. "The Dexter Hypertext Reference Model,"

Proceedings of the Hypertext Standardization Workshop, National Institute of Standards and Technology, Gaithersburg, Maryland, January 16-18, 1990.

[Leggett et al. 1988] Leggett, J., Schnase, J. L., Kacmar, C. J. "Working Definitions of Hypertext," TAMU 88-020, Hypertext Research Lab, Texas A&M University, October, 1988.

[Marshall 1990] Marshall C.C., "Work Practices Study: Analysts and Notetaking," Xerox Internal Report, May 1990.

[Marshall 1990] Marshall C.C., "A Multi-Tiered Approach to Hypertext Integration: Negotiating Standards for a Heterogeneous Application Environment," *Proceedings of the Hypertext Standardization Workshop*, National Institute of Standards and Technology, Gaithersburg, Maryland, January 16-18, 1990.

[Meyrowitz 1986] Meyrowitz, N., "Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework," *Proceedings of OOPSLA '86*, Portland, Oregon, Sept. 1986.

[Monty & Moran 1986] Monty, M. L., Moran, T. P. "A Longitudinal Study of Authoring Using NoteCards," poster presented by Monty at CHI '86 conference on Human Factors in Computing Systems, (Boston, April 13-17, 1986). Extended abstract published in ACM SIGCHI Bulletin 18(2), October, 1986.

[Nelson 1984] Nelson, T.H., *Literary Machines*, T. Nelson, P.O. Box 118, Swarthmore, PA 19081, 1984.

[Schneiderman 1987] Schneiderman, B., "User Interface Design for the Hyperties Electronic Encyclopedia," *Hypertext '87 Papers*, University of North Carolina at Chapel Hill, November, 1987.

[Trigg & Irish] Trigg, R. H., Irish, P. M. "Hypertext Habitats: Experiences of Writers in NoteCards," *Hypertext '87 Papers*, Chapel Hill, North Carolina, November 13-15, 1987.

[Trigg et al. 1986] Trigg, R. H., Suchman, L., Halasz, F. G., "Supporting Collaboration in NoteCards," *Proc. of Conference on Computer Supported Cooperative Work*, Austin, Texas, December 3-5, 1986, pp 153-162.

[Trigg et al. 1987] Trigg, R. H., Moran, T. P., Halasz, F. G., "Adaptability and Tailorability in NoteCards," *Human-Computer Interaction - INTERACT '87*, H.-J. Bullinger & B. Shackel (Eds.), Elsevier Science Publishers B.V. (North-Holland), 1987.

[Webb 1989] Webb, D., "CATALYST: Computer Aided Tools for Analysis of Science and Technology," Directorate of Intelligence Internal Report, 1989.

[Yankelovich 1988] Yankelovich, N., Hann, B., Meyrowitz, N., Drucker, S., "Intermedia: The Concept and Construction of a Seamless Information Environment," *IEEE Computer* 21, 1988.