Building the Memex Sixty Years Later: Trends and Directions in Personal Knowledge Bases

Stephen Davies
Javier Velez-Morales
Roger King

Department of Computer Science
University of Colorado at Boulder

Technical Report CU-CS-997-05

August 2005

Building the Memex Sixty Years Later: Trends and Directions in Personal Knowledge Bases

Stephen Davies, University of Colorado[1]
Javier Velez-Morales, University of Colorado
Roger King, University of Colorado

Abstract

Software tools abound for managing documents and other information sources, but are rarely used for managing the personal, subjective knowledge an individual gleans from them. Yet for at least sixty years there has been a thread of interest in building a system to support a personalized repository of knowledge. This survey defines such systems as "personal knowledge bases," describes a number of important historical and recent examples, and gives a taxonomy of their principal characteristics. It concludes by broadly analyzing the design choices involved and sketching out an ideal solution.

I. Motivation

Personal knowledge bases: toward Bush's Memex? Six decades have passed since the end of World War II, and also since the appearance of Vannevar Bush's ubiquitously-quoted article "As We May Think"[Bush 1945] in the Atlantic Monthly. In it, Bush, President Truman's Director of Scientific Research, surveyed the post-war landscape and laid out what he viewed as the most important forthcoming challenges to humankind, even speculating as to possible solutions. The prophetic article presaged many technological changes, including the dawning of the information age. But undoubtedly his most intriguing idea – and in the years since, the most hotly pursued – was his vision of a hypothetical information storage device called the "Memex[2]." This system was envisioned to tackle the "information overload" problem, already a formidable one in 1945. In Bush's own words:

> Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and to coin one at random, "memex" will do. A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.

As we will see, what Bush envisioned here was an early example of exactly what this survey will characterize as a *personal knowledge base.*

---

[1] Contact author: Stephen.Davies@colorado.edu

[2] The word "memex" is thought to be an abbreviation for "memory extender," though this is never explained in the article.

Goals of this paper. In this survey we define the notion of a *personal knowledge base (PKB)* and explore the technological efforts researchers and inventors have made thus far towards creating one. It is a real and pressing problem, and an elusive one, as the sheer number of attempted solutions included in this survey attests. And we will argue that although certain aspects of the problem can be adequately met by existing software tools, the essential difficulty of capturing and retrieving personal knowledge is not completely solved by any of them. Rather, it appears that despite great gains over the last sixty years in office automation, distributed information networks, and applications for enhancing interaction with computers, today we *still* don't have a viable Memex as Bush envisioned.

The Memex vision. Before delving into the survey proper, we wish to return briefly to Bush's article and note a few important points. His Memex is often cited as the precursor to today's hypertext systems ([Seyer 1991], [Nyce and Kahn 1991]) and indeed Doug Englebart and Ted Nelson, hypertext pioneers, have both acknowledged their debt to it ([Engelbart 1963], [Nelson 1987].) This is largely due to Bush's description of associative indexing, nearly identical to today's Web hyperlinks, through which the world's documents could be joined into a large associative network. Yet a careful reading of the article reveals a *personalized* dimension to the Memex that is very different than the public and collaborative focus of most hypertext efforts to date. Notice that the above quote specifies "*individual* use," and a "*private* file and library." And Bush's emphasis throughout the article was on expanding our own powers of recollection: "Man needs to mechanize his record more fully," he says, if he is not to "become bogged down…by overtaxing his limited memory."

To be sure, Bush envisioned collaborative aspects as well, and even a world-wide system that scientists could freely consult. But what is often ignored about his vision is this intensely personal aspect. With the Memex, the user can "add marginal notes and comments," and "build a trail of his interest" through the larger information space. He can share trails with friends, identify related works, and create personal annotations. A lawyer would have "at his touch the associated opinions and decisions of his whole experience," and similarly a doctor the records of his patients. In short, the Memex as Bush envisioned it would give each individual the ability to create, categorize, classify, and relate his *own* set of information corresponding to his unique personal viewpoint. Much of that information would in fact be comprised of bits and pieces from public documents, just as the majority of the knowledge inside our own heads has been imbibed from what we read and hear. But a monolithic Web of public documents is no substitute for the specialized recording of information that each individual perceives and needs to retain. The idea of "supplementing our memory" is not a one-size-fits-all proposition, since no two people have the same interests or opinions (or memories): it demands rather a subjective expression of knowledge, unique to each individual.

Pursuit of a personal knowledge base. The personal Memex is needed more today than ever before. In fact, we contend that the ability for individuals to retain and reuse the knowledge they acquire is one of *the* key technological problems of the next decade. In some ways, the whole information revolution is useless without that component. What

good are all the documents in the world if people can't figure out, manage, interrelate, and later remember what they *mean*? We all process a myriad of information sources each day, from Web pages to documents to television programs to e-mail messages. If we cannot find a way to capture each salient point and reliably store it, then how can we make use of it when it becomes relevant later? How can we relate it to what else we know, building an integrated and comprehensive understanding of ourselves and our world instead of just fragmentary bits of data whose permanence is merely a matter of chance?

Perhaps the key reason that no viable Memex yet exists is the broad spectrum of demands that could be placed on such a system. The idea of an all-purpose information assistant is alluring, but what exactly would it entail? Would such a system hold all information about all aspects of my life? Could I cross-reference this information in any imaginable way, or even in multiple, complementary ways? Could I query this information in an intuitive, semantically-rich fashion, rather than with traditional, syntactic, keyword-based techniques? Would it be accessible via a browser, or even a handheld device? Would I be guaranteed that the overhead of learning the system and storing information in it would be worthwhile, given the system's payoff in my life? Could multiple people share their Memex databases? Would the system make inferences and learn new facts as it grows? Could I easily repurpose information for new applications?

We do not attempt to identify any single or hybrid technology that could directly address all these questions. Rather, we note that the limits on what a Memex could be are effectively infinite. Our purpose is to survey potential technologies and create a viable framework for discussing and comparing them in the light of a broad definition of a personal knowledge base.

The rest of this paper is organized as follows. Following some brief definitions that will be used throughout the survey, we will describe precisely what a PKB is, and what it *isn't*, and what advantages we might expect from its use. We will outline the four main research areas that solutions have originated from, and discuss how they overlap. Then, in the main body of the paper, we will look at a large number of systems from each of these areas, and classify them along three broad lines: the data models they support, the architectures they use, and the PKB-specific features they implement. Finally, we will step back from these details and consider the larger question of what the ideal PKB system might look like, and what we can take from today's solutions to apply towards the true realization of a practical Memex.

II. Definitions

The objective and subjective realms. Before we consider the catalog of candidate Memex systems, we wish to define precisely a few terms that will enable us to better describe the relevant issues. The first is the distinction between what we term the "objective" and "subjective" realms. To wit:

the objective realm – the set of electronic documents and other information that are available to a group at large. This is often the entire public domain, as with the World Wide Web, but sometimes it may be communicated only internally with an organization. The key factor is that it consists of materials everyone within a large group has access to, and views identically (ie., a given text appears the same to everyone.)

a subjective realm – the viewpoints, interpretations, classifications, and relationships that an individual perceives when examining the objective realm. This set of elements is unique to each observer. It represents the ongoing accumulation of knowledge each person builds as they browse and learn from objective sources. It need not consist solely of elements from the objective realm, as the observer will also bring in their own background knowledge and biases, but it is most often primarily comprised of such objective elements.

Note that in most cases, a person's subjective realm is never materialized; it remains ethereal, only a product of the mind. The observer is not normally aware of its existence, in fact: it is only the lens through which they filter and interpret objective information. Examples where a user *does* materialize their subjective realm would include the activities of note-taking (electronically or on paper), giving structure to a set of documents by assigning them to filesystem folders, or arranging website bookmarks in a "favorites" hierarchy. We will argue that the primary job of a PKB is to allow a user to express their subjective realm in a tangible way so that it can be stored and retrieved later.

Subjective observations cover a wide spectrum. At one end is the freeform recording of a person's "idea"; whether that be in the form of a textual note, or some sort of graphical representation. Here the user is expressing an idea in its entirety: there is a great deal of substance to what they have personally perceived, and that idea, rather than the objective elements to which it may refer, constitutes its main value. On the other end is the simple identification of relationships between existing objective sources. Here the user does not have in mindany complex theory, but simply observes some similarity or other "link" between two elements of the objective realm. Note, however, that at any point on this continuum, the subjective knowledge is in addition to, and not merely present within, the objective sources. It expresses further interpretations, relationships, and assertions that are not present in the information itself, and hence it is real added value. And this is true whether it is materialized or not.

To avoid confusion, note that the word "objective" here does not imply "correct" or "without bias." Indeed, many of the elements in the objective realm will almost certainly express opinions or contain errors; consider the World Wide Web! By "objective" we simply mean "open and available for all to consider," rather than private.

Note also that the distinction between these realms may become blurred over time in collaborative settings, since one might express one's interpretations or organization of some objective information and then *publish* those interpretations in an objective document. The objective realm, in this case, would then contain "base-level information"

in addition to published commentary on that information, which then also becomes "objective." Our definitions will still hold, however, if we simply consider the time of publication to be a transition point between the subjective and objective realms. Once a user takes their own private perceptions, articulated in a tangible form, and intentionally publishes them, these perceptions have now been transformed out of the subjective realm and placed into the objective.

Data, information, and knowledge. The related concepts of "data," "information," and "knowledge" have been variously defined in an attempt to draw distinctions between levels of understanding or semantic content. In this paper, we will use the following definitions:

data – a series of symbols whose meaning is uninterpreted and unknown. It may be a meaningless bit stream like "1110010100" or a suggestive sequence like "(303)555-1212" or "Claims Office." At any rate, it cannot be used meaningfully in its present form. It is *potentially* information, if only there was a standard language, context, and conventions by which its semantics could be derived.

information – material that intentionally expresses something meaningful, and which, if read, could impart assertions of truth or opinion. It is *potentially* knowledge, if only someone took the time to perceive it, parse it, and comprehend it.

knowledge – true meaning that has been internalized inside one's mind. It can be analyzed, expressed, acted upon, and related to other knowledge.[3]

The difference between information and knowledge, then, is that the former is processible and the latter has already been processed.[4] If a friend plunked down a large textbook on your desk, they could rightfully claim that it contained "a lot of information." But it does not properly contain *knowledge* until someone reads it, interprets it, and grasps the points it contains and their implications. Knowledge is thus the result of assimilating and digesting an information source so that it can be understood and used. Psychologically, it is the formation of a mental model in one's own mind that corresponds to the information encountered ([Anderson 1990], [Kintsch 1970], [Sowa 1984], [Gentner and Stevens 1983].)

The personal knowledge base. We are now in a position to define a "personal knowledge base." As the term has three words, the PKB definition has three components:

personal: Like Bush's Memex, a PKB is intended for private use, and its contents are custom-tailored to the individual. It is a manifestation of a user's subjective realm. It

---

[3] Note that the objective realm is normally composed of information sources, and the subjective realm of knowledge representations. Hence we will speak often of "objective information" vs. "subjective knowledge."

[4] This distinction can also be seen in the lexical breakdown of the terms themselves: *inform*ation has the potential to *inform* whoever reads it, whereas *know*ledge is what someone actually *know*s.

contains trends, relationships, categories, and personal observations that its owner sees but which no one else may agree with. It can be shared, just as one can explain one's own opinion to a hearer, but it is not jointly *owned* by anyone else any more than explaining my opinion to you causes you to *own* my brain.

knowledge: A PKB primarily contains knowledge, not information. Its purpose is not simply to aggregate all the information sources I have seen, but to preserve the knowledge that I have *learned* from those sources. When I return to my PKB to retrieve some knowledge that I have stored, I don't want to be merely pointed back to the original documents again, so that I have to re-locate, re-read, re-parse, and re-learn the relevant passages. Instead, I want to return to the distilled version of the particular truth I am seeking, so that the mental model I once had in mind can be easily reformed.

base: A PKB is a consolidated, integrated knowledge store. It is a reflection of my own memory, which, as Bush and many others have observed, can freely associate any two thoughts together, without restriction. Hence a PKB is defeated at the outset if it attempts to partition a user's field of knowledge into multiple segments that cannot reference or interrelate with each other. Just as I have only one mind, and I can connect any of my ideas together without regard for artificial boundaries, so a PKB must act as a single, unified whole.

Not all of the solutions we will consider in this survey satisfy all three of these conditions. Some of them we mention because they handle one of them so well that their ideas are worth investigating as possible components to a true solution. But we feel that in order to be a viable implementation of the Memex, a software application (or any other tool) must meet all three criteria.

Before continuing, it is worthwhile to enumerate several classes of systems that *cannot* be classified as PKBs, and which we will not discuss in this survey. Specifically, we will *not* cover:

- collaborative efforts to build a universal objective space (as opposed to an individual's *personal* knowledge.) The World Wide Web itself is in this category, as were its predecessors HyperTIES[Schneiderman 1987] and Xanadu[Nelson 1987], Web categorization systems like the Open Directory Project[Netscape 2004], and collaborative information collections like Wikipedia.[Wales 2005] In cases where such tools could be readily used for personal purposes, however, we will discuss how they could be adapted to the task.
- search systems like Enfish[Enfish 2005], Clarity[Ralston 2005] and the Stuff I've Seen project[Dumais, *et al.* 2003] that simply index and search one's information sources on demand, rather than giving the user the ability to craft and express personal *knowledge*.
- tools whose goal is to produce a design artifact rather than to maintain knowledge for its own sake. Systems like ART[Nakakoji, *et al.* 2000] and Writing Environment[Smith, *et al.* 1987] use intermediate knowledge representations as a

means to an end, abandoning them once a final artifact has been produced, and hence are not suitable as PKBs.

- systems that focus on capturing transient information, rather than archiving knowledge that has long-term value. Examples would be Web logs[Godwin-Jones 2003] and e-diaries[Kovalainen, *et al.* 1998].
- tools whose information domain is mostly limited to time management tasks (calendars, action items, contacts, etc.) rather than "general knowledge." Blandford and Green[Blandford and Green 2001] and Palen[Palen 1999] give excellent surveys; common commercial examples would be Microsoft Outlook[Microsoft 2003], Lotus Notes[IBM 2005], and Novell Evolution[Novell 2005].
- similarly, tools developed for a specific domain, such as bibliographic research (e.g., [Intelligents 2001], [TruTamil 2005], [Oberon 2005]), rather than for "general knowledge."

In summary, this survey will cover only systems whose purpose (perhaps adapted slightly) is to store general, personal knowledge, in order to organize it and retrieve it later for its own sake.

III. Benefits

An idea of the advantages an effective PKB would bring can be gleaned from the way in which today's solutions are "pitched." From perusing both academic claims and marketing literature, we find that PKB systems aim to meet a number of related but distinct needs, including:

Knowledge generation and formulation. Here the emphasis is on procedure, not persistence; it is the act of simply using the tool to express one's knowledge that helps, rather than the ability to retrieve it later. Systems boast that they can "convert random thoughts generated while you are the most creative into the linear thoughts needed most when communicating," "help you relate and arrange random ideas," and "stimulate your brain."[Bosley 2005] In an educational setting, they "make it easier for students to grasp concepts and ideas"[SMART 2005] and "strengthen critical thinking, comprehension, and writing skills."[Inspiration 2005] In a business setting, one can "capture in detail thoughts and ideas, to explore them, and gain new understanding and insight."[Banxia 2005]

It is interesting to note that many of these systems make direct appeals to human psychology in their literature. Product names like "PersonalBrain"[TheBrain 2005], "Axon Idea Processor"[Bok 2005] and "Mind Manager"[MindJET 2005] are certainly suggestive, and one product calls itself "a trusted thought container."[HogBay 2005] Another claims to be "…an application that's actually designed to help you think. It's like having an extra brain."[Omni 2005] A research system claims to "actively model the user's own memory"[Jones 1986], and one commercial product's testimonial says flatly, "I now feel like my computer is an extension of my brain."[MicroLogic 2005] Clearly,

the connection between knowledge expression and the mechanics of the mind is one that numerous system designers are anxious to draw, a point to which we will return later.

Knowledge capture. PKBs do not merely allow one to express knowledge, but also to capture it before it elusively disappears. Often the emphasis is on a streamlined user interface, with few distractions and little encumbrance. The point is to lower the burden of jotting down one's thoughts so that neither task nor thought process is interrupted. One system speaks for many when it describes its user interface philosophy as "low threshold, high ceiling"; the interface was "designed to be nonintrusive, allowing the user to concentrate on the task…"[Canas*, et al.* 2005] Another product asserts that "it is very quick to open a…document and within seconds record the essence of that new idea without distractions, while your mind is focused on it and without disturbing the flow of your current work."[StayAtPlay 2005]

Knowledge organization. A recent short study on note-taking habits finds that "better organization" was the most commonly desired improvement in people's own information recording practices.[Hayes*, et al.* 2003] PKB systems profess to answer this need, allowing one to "organize personal information", claiming to be "a more productive way to stay organized"[AquaMinds 2005] and that "you will finally be able to organize all your random information."[MicroLogic 2005] And clearly this organization is personal and subjective; one tool brags that it "complements individual styles for capturing and organizing thoughts."[Microsoft 2003]

Knowledge management and retrieval. Perhaps the most critical aspect of a PKB is that the knowledge it stores is permanent and accessible, ready to be retrieved at any later time. Accordingly, systems support "the longer term management of…information"[Compendium 2005] and "a structure for organizing, storing, and retrieving information."[Halasz*, et al.* 1987] Products will "give you a place to stash all those stray snips of knowledge where they can be quickly recalled when you need them"[Bitsmith 2005] and let you "find any data in an instant, no matter where or how you entered it."[MicroLogic 2005]

Integrating heterogeneous sources. Finally, recognizing that the knowledge we form comes from a variety of different places, many PKB systems emphasize that the information from diverse sources and of different types can be integrated into a single database and interface.[5] This makes it possible to "capture all your information in one place"[Microsoft 2003] and "capture, analyze, and repurpose information from a variety of sources."[MindJET 2005] One system's "universal data access layer allows you to connect all your existing information into the system, giving…one interface to everything and the ability to connect all types of information"[TheBrain 2005], and another "eliminates this partition (between heterogeneous applications and data types) so that individuals can work with their information in a unified fashion."[Adar*, et al.* 1999]

[5] Note that this feature pertains primarily to the *objective* information sources that are brought in or referred to by the tool, to be then combined with the user's own subjective knowledge.

The features here identified are all tightly interrelated, of course, and most of them would be incomplete without several of the others. Perhaps this is why some designers have had trouble communicating exactly what their system *is*, resorting to blanket statements that it "can be used for everything from keeping a to-do list to writing a book"[HogBay 2005], or that "it's a notepad, outliner, scrapbook manager, information manager, freeform database, archive, bookmark manager and image database – all in one integrated application."[DEVON 2005] If nothing else, this illustrates that the potential uses of PKBs are vast. Certainly acquiring, integrating, maintaining, and using our personal knowledge is at the heart of what it means to be human, and for this reason a PKB application's power and utility could be almost unparalleled if used effectively.

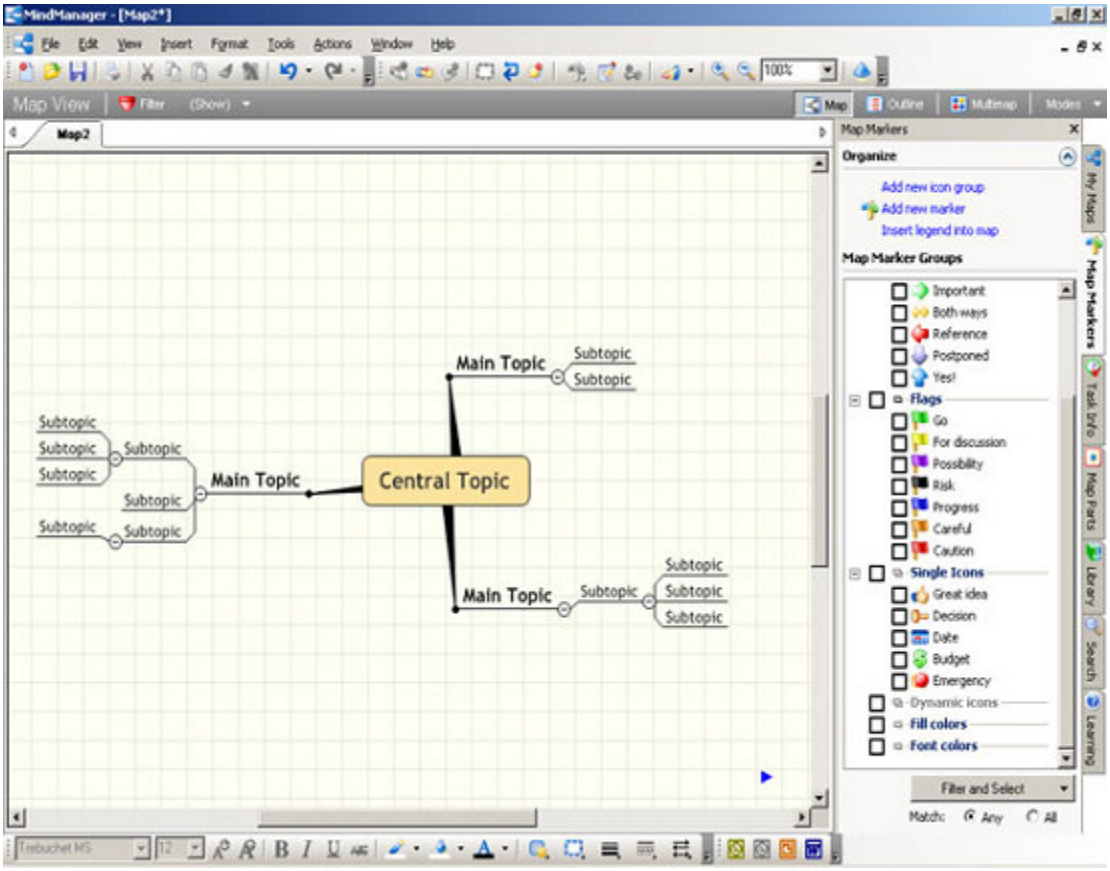
IV. Contributing bodies of research

We examine four different fields of research in our study of possible PKB systems. Many of the essential problems have been approached by multiple of them, and often even in the same way; yet the four grew up in distinct research communities, so it is worthwhile to consider each in turn.

A. Graphical knowledge capture tools

Much fanfare has been generated in the last thirty years around pictorial knowledge representations. Claims have been made, with varying degrees of scientific justification, that drawing informal diagrams to represent abstract knowledge is an excellent way to communicate complex ideas, enhance learning, and even to "unlock the potential of the brain."[Buzan and Buzan 1996] Great emphasis is placed on the pictorial nature of the knowledge diagrams; the use of spatial layout, color, and images is said to strengthen understanding and promote creativity. The three primary schools – mind mapping, concept mapping, and cognitive mapping – will be considered briefly here. Each prescribes its own data model and procedures, and each boasts a number of software applications designed specifically to create compatible diagrams.

1. Mind mapping. Mind mapping was promoted by pop psychologist Tony Buzan in the 1960's, and commands the allegiance of an impressive number of adherents worldwide. A mind map is essentially nothing more than a visual outline, in which a main idea or topic is written in the center of the diagram, and subtopics radiate outwards in increasing levels of specificity. The primary value is in the freeform, spatial layout (rather than a sequential, numbered outline), the ability for a software application to hide or reveal select levels of detail, and as mentioned above, graphical adornments. The basic data model is a tree, rather than a graph, with all links implicitly labeled "supertopic/subtopic." (This is discussed more fully in section V, below.) The number of software tools available for constructing mind maps is staggering; just a sampling would include [Bosley 2005], [FreeMind 2005], [Gael 2005], [CoCo 2005], [MindJET 2005], [NovaMind 2005], [Bignell 2005], [Odessa 2005], and [Mind 2005], and there are many more.



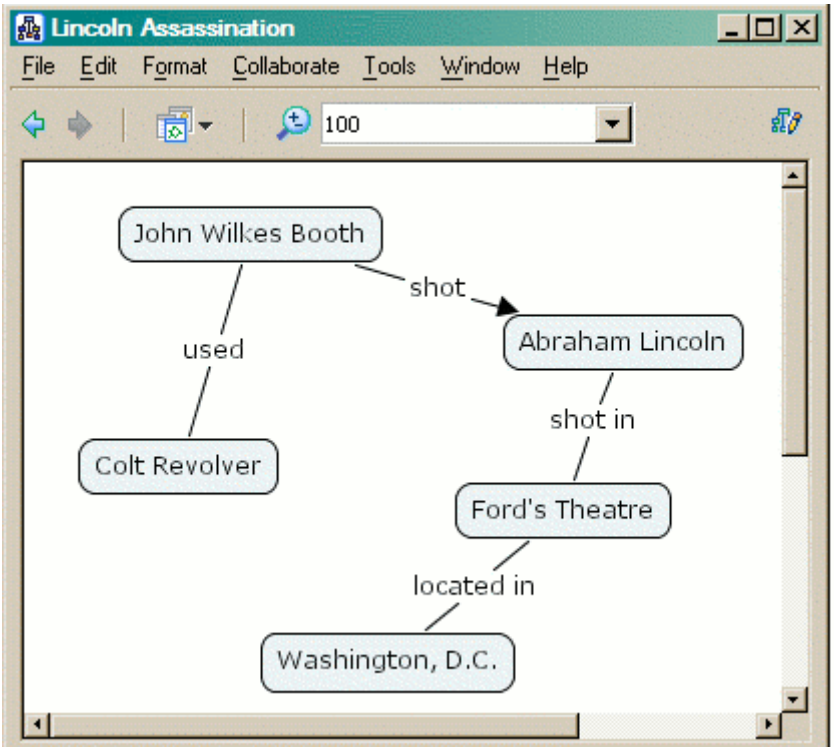
MindManager, a mind map creation tool. The structure of the diagram is inherently hierarchical, with the unlabeled branches between elements implicitly denoting "supertopic/subtopic."[MindJET 2005]

2. Concept mapping. Concept mapping is based on firmer psychological footing than is mind mapping, but it is slightly more complex, which may account for its second place in popularity. Concept maps were developed by Cornell Professor Joseph Novak[Novak 2003], and based on David Ausubel's assimilation theory of learning.[Ausubel 1968] An essential tenet is that newly encountered knowledge must be related to one's prior knowledge in order to be properly understood. Concept maps help depict such connections graphically. Like mind maps, they feature evocative words or phrases in boxes connected by lines. There are two principal differences, however: first, a concept map is properly a graph, not a tree, permitting arbitrary links between nodes rather than only parent/child relationships[6]; and second, the links are labeled to identify the nature of the inter-concept relationship, typically with a verb phrase. In this way, the links on a diagram can be read as English sentences, with the upstream node as the subject and the downstream node as the direct object of the sentence. There are many applications

[6] It should be noted that although concept maps themselves are technically arbitrary graphs, concept mapping *techniques* encourage the heavy use of hierarchy. "Place the most inclusive, most general concepts at the top…of the map," says Novak., and then "select…two, three, or four subconcepts under each general concept," and continue in this way until the most specific items are at the bottom."[Novak 2003] The non-hierarchical associations are termed "crosslinks," and are thought to be less common.

available that could be used for drawing these diagrams, not all of which directly acknowledge their support for concept maps in particular. Some which do include [Canas, *et al.* 2005], [SMART 2005], [Gaines and Shaw 1995], [AKS-Labs 2005], [Claro 2005], and [Bok 2005].

Note that a concept map is virtually identical to the notion of a "semantic network"[Woods 1985], which has served as a cornerstone for much artificial intelligence work since its inception. Semantic networks, too, are directed graphs in which the nodes represent concepts and labeled edges the relationships between them. Much psychology research has strengthened the idea that the human mind internalizes knowledge in something very like this sort of framework. This likely explains the ease with which concept mapping techniques have been adopted by the uninitiated, since concept maps and semantic networks can be considered equivalent.

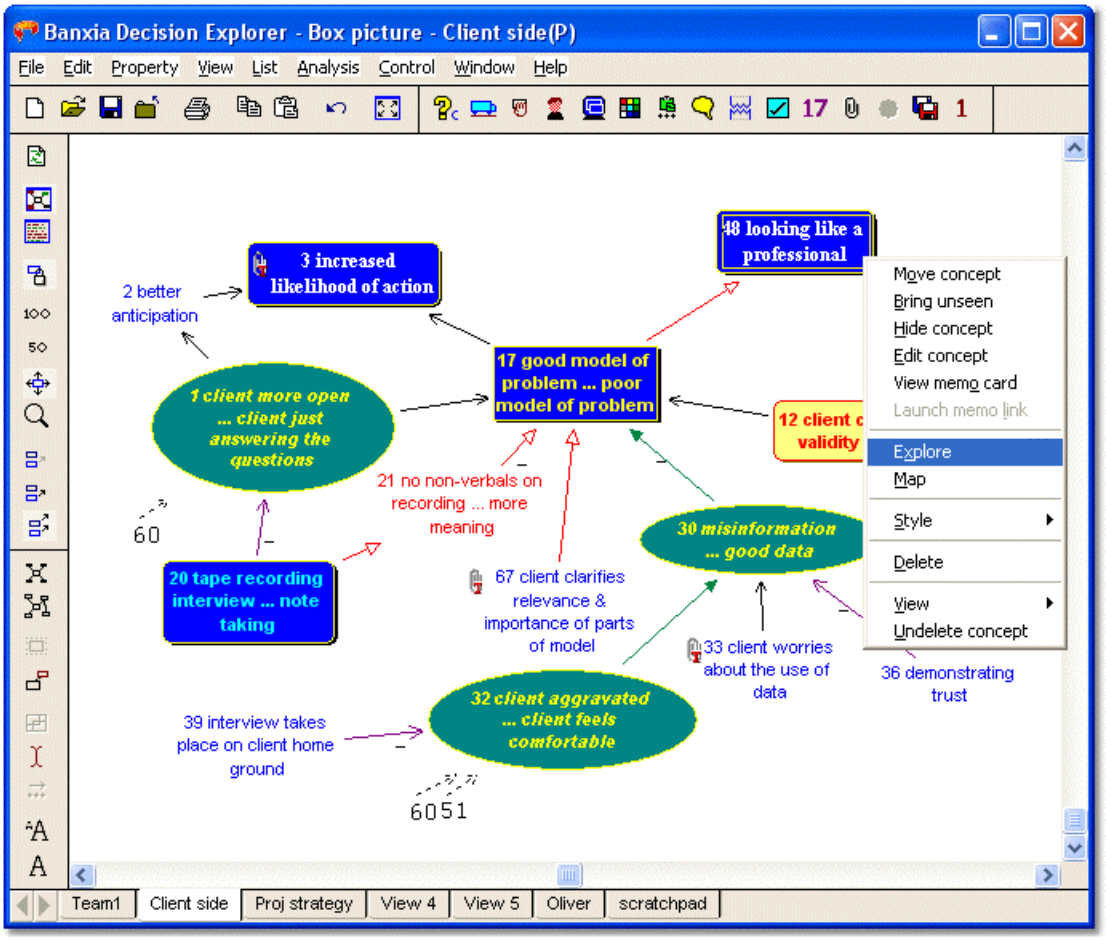


CMap, a concept map creation tool. The structure of the diagram is a general graph, with labels indicating the relationships between major concepts.[Canas, et al. 2005]

3. Cognitive mapping. Cognitive mapping, developed by Fran Ackermann and Colin Eden at the University of Strathclyde, uses the same data model as does concept mapping, but with a new set of techniques. In cognitive maps, element names have two parts, separated by an ellipsis that is read "as opposed to" in order to further clarify the semantics of the node. ("Cold…hot" is different than "cold…freezing," for example.) Links are of three types – causal, temporal, connotative – the first of which is the most common and is read as "may lead to." Generally cognitive mapping is best suited to

domains involving arguments and decision making. Cognitive mapping is not nearly as widespread as the other two paradigms; the premier design application is [Banxia 2005].

Together, these and related methods have brought into the mainstream the idea of breaking down knowledge into its fundamental elements, and representing them graphically. Students and workers from widely diverse backgrounds have experienced success in better articulating and examining their own knowledge, and in discovering how it relates to what else they know. We will see that although architectural considerations prevent any of these tools from functioning as bona fide PKBs, the ideas they have contributed to a front-end interface mechanism cannot be overestimated.

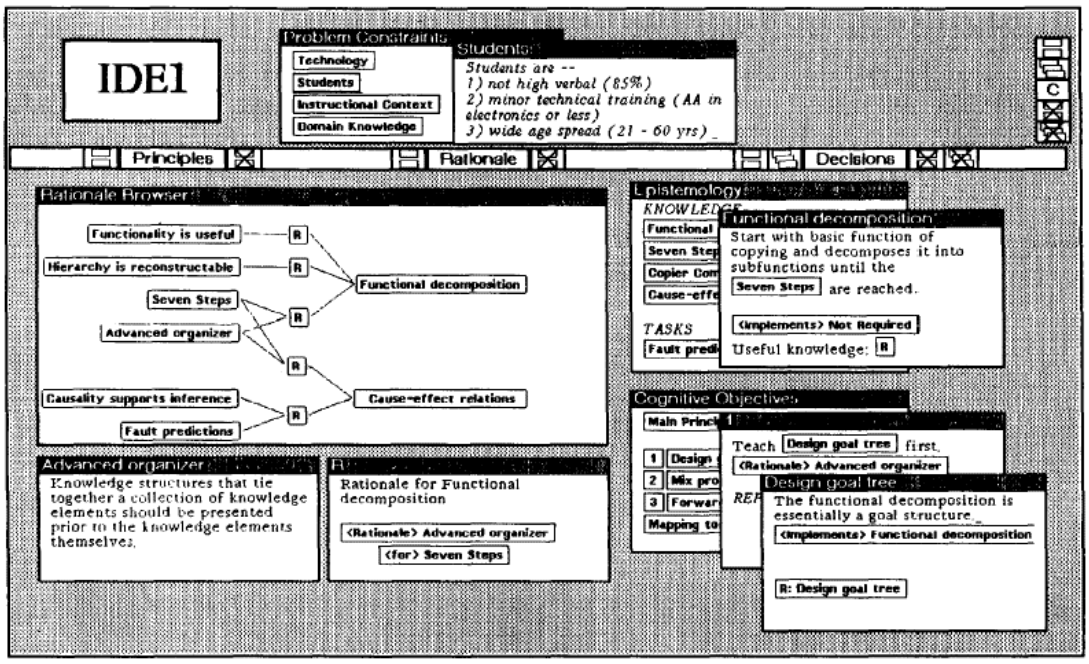


Decision Explorer, a cognitive map creation tool. Cognitive maps are mostly equivalent to concept maps, but are best suited to domains involving arguments and decision making. The ellipses inside certain elements are to be read "as opposed to" in order to make the meaning more precise.[Banxia 2005]

B. Hypertext systems

As previously stated, the hypertext community proudly points to Vannevar Bush's article as the cornerstone of their heritage. Hence the development of hypertext techniques,

while seldom applied specifically towards PKB solutions, is important. There have basically been three types of hypertext systems: those that exploit features of non-linear text to create a dynamic, but coherent "hyperdocument" (e.g., [Schneiderman 1987], [Goodman 1988]); those that prescribe ways of linking existing documents together for navigation and expression of affinities (e.g., [Garrett, *et al.* 1986], [Davis, *et al.* 1993], [Pearl 1989]); and those that use the hypertext model specifically to model abstract knowledge. Of the three, the latter is the one we are most interested in, since it relates most directly to the PKB problem space. Interestingly, though the first and especially the second category have dominated research efforts (and public enthusiasm) over the past two decades, it is this third class that is closest in spirit to the original vision of hypertext by its founders.



The NoteCards knowledge management environment. [Halasz, et al. 1987]

We have already mentioned Bush's emphasis on extending the human memory, which implicitly demands a way to model abstract knowledge. Doug Engelbart, too, who began developing the first viable hypertext system in 1959, pursued "the augmentation of man's intellect."[Engelbart 1963] Engelbart's focus has been to develop computer systems to "help people think better," and sought data models that more closely paralleled the human thought process. Though his "Augment" system underwent many evolutions and was later used for managing software engineering projects, we note that his original purpose closely aligned with a key aspect of PKBs: using hypertext as a way to represent and store abstract human knowledge.

More recently, Randall Trigg's TextNet[Trigg and Weiser 1986] and NoteCards[Halasz, *et al.* 1987] systems further explored this idea. TextNet revolved around "primitive pieces of text connected with typed links to form a network similar in many ways to a

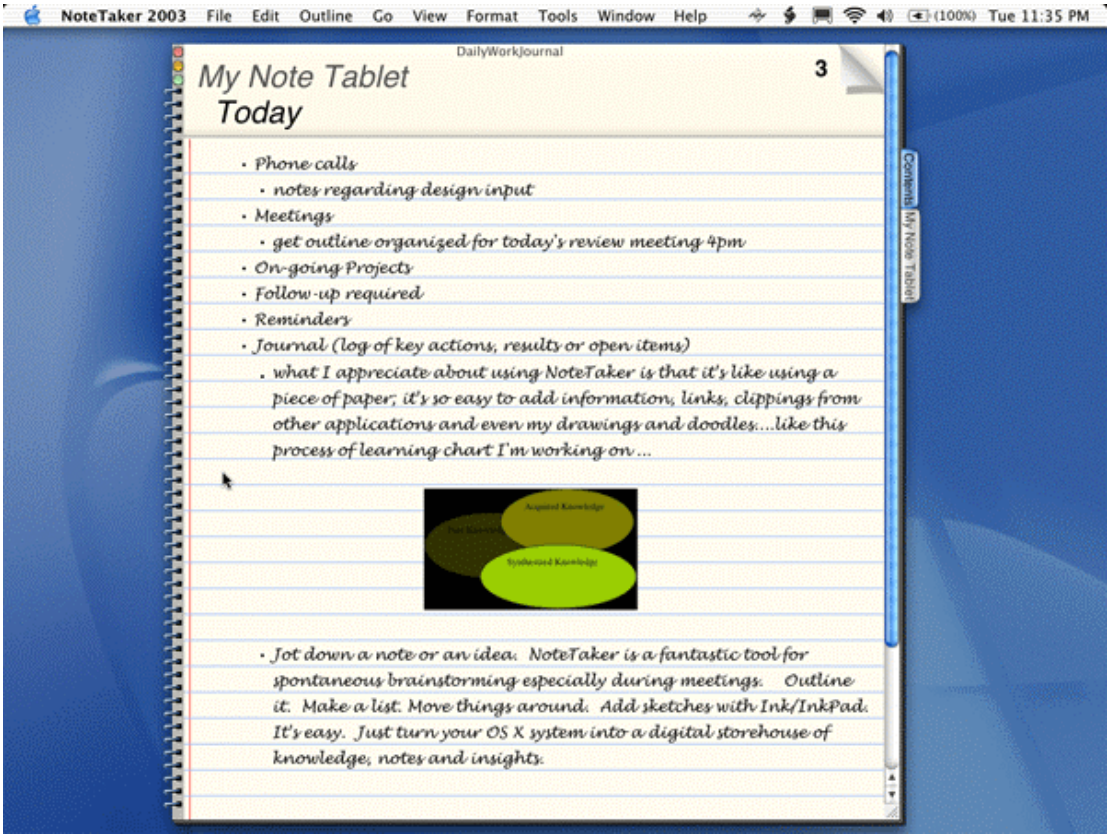
semantic network."[Conklin 1987] Though text-centric, it was clear that Trigg's goal was to model the associations between primitive ideas and hence to reflect the mind's understanding. "By using…structure, meaning can be extracted from the relationships between chunks (small pieces of text) rather than from the words making them up."[Trigg and Weiser 1986] The subsequent NoteCards effort, one of the most influential hypertext efforts in history, was similarly designed to "formulate, structure, compare, and manage ideas." It was useful for "analyzing information, constructing models, formulating arguments, designing artifacts, and generally processing ideas."

Conklin and Begeman's gIBIS system was another early effort into true knowledge representation, specifically for the field of design deliberations and arguments.[Conklin and Begeman 1988] The project lived on in the later project QuestMap[Selvin 1999] and the more modern Compendium[Compendium 2005], which has been primarily used for capturing group knowledge expressed in face-to-face meetings. In all cases, these systems use semantic hypertext in an attempt to capture shared knowledge in its most basic form. Other examples of knowledge-based hypertext tools include Mental Link[Dede and Jayaram 1990], Aquanet[Marshall*, et al.* 1991], and SPRINT[Carlson and Ram 1990], as well as a few current commercial tools such as PersonalBrain[TheBrain 2005] and Tinderbox[Bernstein 2003].

C. Note-taking applications

The most explicit attempt to create a PKB as we have defined it comes from the area of note-taking applications. These software tools allow a user to create snippets of text (often called "notes") and then organize or categorize them in some way. They draw heavily on the "note-taking" metaphor since it is a familiar operation for users to carry over from their experiences with pen and paper. A surprising number of tools even incorporate visual depictions of a ruled, spiral-bound notebook into their user interfaces. (e.g., [Microsoft 2003], [AquaMinds 2005], [CircusPonies 2005].)

We would expect applications like this to be more easily grasped by novices, since the process of simply "taking notes" in English text involves no learning curve such as that required for creating alien graphical diagrams. And this is indeed the case: note-taking tools have a tremendous number of aficionados, who often defend their choice of tool with almost religious intensity. Most of these tools are based on a tree hierarchy, in which the user can write pages of notes and then organize them into sections and subsections. (e.g., [HogBay 2005], [Microsoft 2003], [CircusPonies 2005], [AquaMinds 2005].) The higher level sections or chapters often receive a colored tab exactly as a physical three-ring notebook might. Others eschew the tree model for a more flexible category-based approach ([Kaplan*, et al.* 1990], [Bitsmith 2005], [Zoot 2005]), as we will consider in depth when we examine data models. The primary purpose of all these tools is to offer the benefits of freeform note-taking with none of the deficiencies: users are free to brainstorm and jot down anything from bullet points to polished text, while still being able to search, rearrange, and restructure the entire notebook easily.

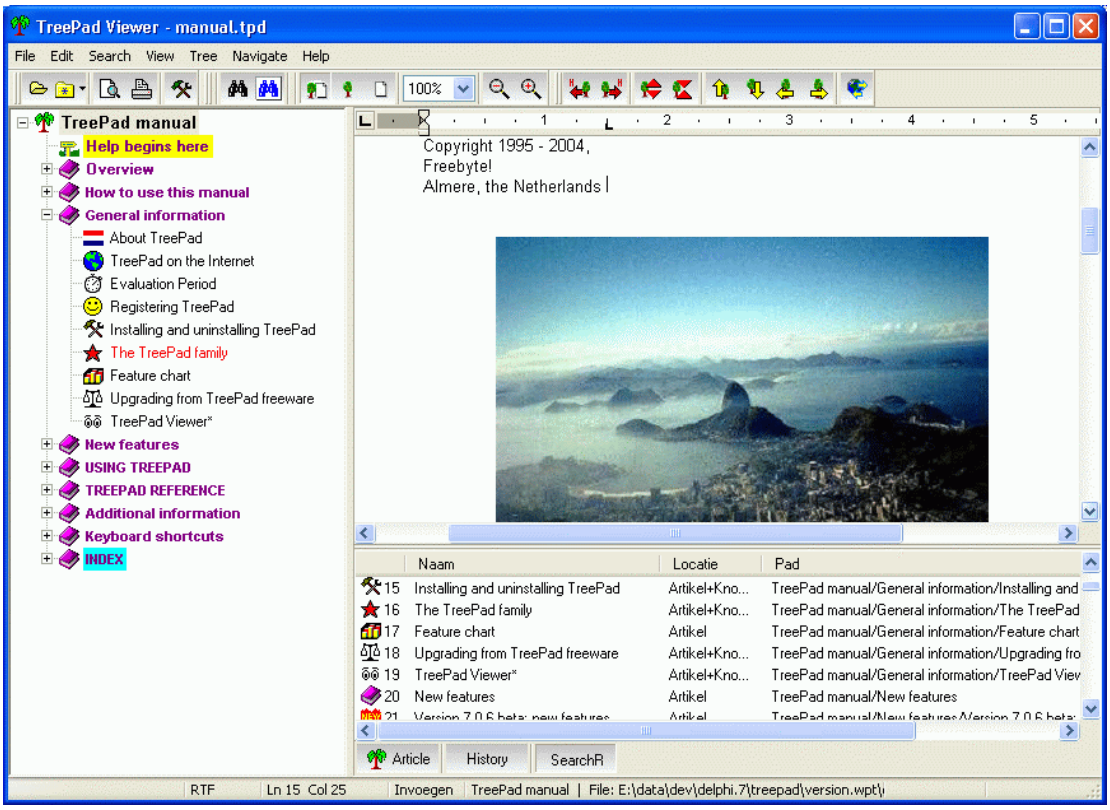


AquaMinds NoteTaker, a hierarchically-based note-taking application. [AquaMinds 2005]

An important subcategory of note-taking tools is outliners (e.g., [Omni 2005]), or applications specifically designed to organize ideas in a hierarchy. These tools typically show a two-pane display with a tree-like navigation widget in the left-pane and a list of items in the right-pane. Topics and subtopics can be rearranged, and each outline stored in its own file. Among the first applications of this kind were Dave Winer's ThinkTank and MORE programs[Winer 2005]; more modern products feature the ability to add graphics and other formatting to an item, and even hyperlinks to external websites or documents.[Freebyte 2005] The once abandoned (but now resurrected) Ecco system[NetManage 1997] was among the first to allow items to have typed attributes, displayed in columns. This gives the effect of a custom spreadsheet per topic, with the topic's items as rows and the columns as attributes. It allows the user to gracefully introduce structure to their information as it is identified.

Of particular interest are applications optimized for subsuming portions of the objective realm into a subjective view, where they can be clustered and arranged. The Virtual Notebook System (VNS)[Burger, *et al.* 1991] was one of the first to emphasize this. VNS was designed for sharing information among scientists at the Baylor College of Medicine; a user's "personal notebook" could make references to specific sections of a "community notebook," and even include arbitrary segments of other documents through a cut-and-paste mechanism. More recently, YellowPen[YellowPen 2005],

Cartagio[Missiontrek 2005], and Hunter-Gatherer[Schraefel, *et al.* 2002] are tools that allow one to easily grab snippets of Web pages and then organize them subjectively. This is a crucial feature in designing a PKB, because as we have mentioned, a user's subjective realm is primarily made up of bits and pieces from the objective realm. It seems natural to model this in the knowledge creation process.



The TreePad outliner, which allows users to organize ideas in a hierarchy and then expand on each one with text, graphics, etc. [Freebyte 2005]

D. Document management systems

Lastly, we consider systems whose primary purpose is to help users organize *documents* in the objective space. Such systems do not encode subjective knowledge *per se*, but they do create a personal knowledge base of sorts by allowing users to organize and cross-reference their information artifacts.

These efforts provide alternative indexing mechanisms to the clumsy "directory path and file name" approach. Presto[Dourish, *et al.* 1999] replaces the directory hierarchy entirely with attributes that users assign to files. These key-value pairs represent user-perceived properties of the documents, and are used as a flexible means for retrieval and organization. William Jones' Memory Extender[Jones 1986] was similar in spirit, but it dynamically varied the "weight" of a file's keywords according to the user's context and perceived access patterns. In Haystack[Adar, *et al.* 1999], users – in conjunction with automated software agents – build a graph-based network of associative links through

which documents can be retrieved. Metadata and multiple categorization can also be applied to provide multiple retrieval paths customized to the way the individual thinks and works with their information sources. WebTop[Wolber, *et al.* 2002] allows the user to create explicit links between documents, but then also merges these user-defined relationships with other types of associations. These include the hyperlinks contained in the documents, associations implied by structural relationships, and content similarities discovered by text analysis. The idea is that any way in which items can be considered "related" should be made available to the user for help with retrieval.

A subclass of these systems integrate the user's personal workspace with a search facility, blurring the distinction between information retrieval and information organization. SketchTrieve[Hendry and Harper 1997], DLITE[Cousins, *et al.* 1997], and Garnet[Buchanan, *et al.* 2004] each materialize elements from the retrieval domain (repositories, queries, search results) into tangible, manipulatable screen objects. These are introduced directly into a spatial layout that also includes the information sources themselves. These systems can be seen as combining a spatial hypertext interface as in VIKI[Marshall and Shipman 1995] with direct access to digital library search facilities. NaviQue[Furnas and Rauch 1998] is largely in the same vein, though it incorporates a powerful similarity engine to proactively aid the user in organization. CYCLADES[Renda and Straccia 2005] lets users organize Web pages into folders, and then attempts to infer what each folder "means" to that user, based on a statistical textual analysis of its contents. This helps users locate other items similar to what's already in a folder, learn what other users have found interesting and have grouped together, etc.

All of these document management systems are principally concerned with organizing objective information sources rather than the expression of subjective knowledge. Yet their methods are useful to consider with respect to PKB systems, because such a large part of our knowledge is *comprised* of things we remember, assimilate, and repurpose from objective sources. Search environments like SketchTrieve, as well as snippet gatherers like YellowPen, address an important need in knowledge management: bridging the divide between the subjective and objective realms, so that the former can make reference to and bring structure to the latter.

V. Data models

Candidate PKB systems can be compared along a number of different axes, the most important of which is the underlying data model they support. This is what prescribes and constrains the nature of the knowledge they can contain: what types of knowledge elements are allowed, how they can be structured, and how the user perceives them and can interact with them. Next to the data model, all other aspects of a system are merely ancillary features and nuances.

A few definitions are in order. First, we will use the term "knowledge element" to refer to the basic building blocks of information that a user creates and works with. There is some variation across systems as to how granular these are and precisely what they

contain, but every system has some notion of a fundamental unit that the user generates, manipulates, and rearranges to reflect their mental model. Second, the term "structural framework" will cover the rules about how these knowledge elements can be structured and interrelated. Whether the user places the elements in categories, builds a top-down hierarchy out of them, spatially arranges them on the screen, or creates arbitrary links between them is determined by the system's structural framework. Finally, by "schema" we mean the introduction of formal semantics into the data model. If the knowledge elements are, say, typeless, arbitrary words or phrases that the user creates, then there is no notion of knowledge element "schema." But if, for example, the system allows knowledge elements to be declared as being of a specific type, or if there are formal semantic properties that can be assigned to them, then the system has effectively introduced some sort of schema to both guide and constrain the user.

This rather lengthy section is organized around these three broad dimensions of data models. In the first subsection below, we will discuss the five principal PKB structural frameworks (tree, graph, tree plus graph, spatial, and category), address the key characteristic of transclusion and its influence on the various frameworks, and then touch on several alternate approaches to the traditional five that have been proposed. In the two sections following, we will address the differing knowledge element types these systems support, and the ways in which schema has been introduced.

A. Structural Frameworks

Kaplan et al. stated it well when they observed in 1990 that "dominant database management paradigms are not well suited for managing personal data," since "personal information is too ad hoc and poorly structured to warrant putting it into a record-oriented online database."[Kaplan*, et al.* 1990] Clearly this is the case; when we want to jot down and preserve a book recommendation, directions to a restaurant, or scattered lecture notes, a rigidly structured relational database table is exactly the wrong prescription. The random information we collect defies categorization and quantization, and yet it demands some sort of structure, both to match the organized fashion in which we naturally think and to facilitate later retrieval. The question is, what sort of structural framework should a PKB provide?

1. The five traditional structural frameworks

a. Tree. Systems that support a tree model allow knowledge elements to be organized into a containment hierarchy, in which each element has one and only one "parent." This takes advantage of the mind's natural tendency to classify objects into groups, and to further break up each classification into subclassifications. It also mimics the way that a document can be broken up into chapters, sections, and subsections. The tree is the basis for most modern filesystem organization (whether "directories" in Unix or "folders" on a Windows platform) and is a popular organization mechanism for Web browser bookmarks and e-mail management. It tends to be natural for users to understand.

All of the applications for creating Buzan mind maps are based on a tree model, because a mind map *is* a tree. Each mind map has a "root" element in the center of the diagram (often called a "main topic") from which all other elements emanate as descendents. Every knowledge element has one and only one place in this structure. Some tools, such as Mind Manager[MindJET 2005], extend this paradigm by introducing "floating topics," which are not anchored to the hierarchy, and permitting "crosslinks" to arbitrary topics, similar to those in concept maps. The fact that such features are included betrays the inherent limitations of the mind map as a modeling technique. A strict tree is inadequate for representing much complex information, as we will discuss later.

Other examples of tree-based systems are most personalized search interfaces ([Di Giacomo, *et al.* 2001], [Reyes-Farfan and Sanchez 2003], [Renda and Straccia 2005]), outliners ([Omni 2005], [Freebyte 2005]), and most of the "notebook-based" note-taking systems ([AquaMinds 2005], [Microsoft 2003].) By allowing them to partition their notes into sections and subsections, note-taking tools channel users into a tree hierarchy. In recognition of this confining limitation, many of these tools also permit a kind of "crosslink" between items ([MicroLogic 2005], [WJJ 2005]), and/or employ some form of transclusion (see below) to allow items to co-exist in several places ([Zoot 2005], [Chronos 2005]). The dominant paradigm in such tools, however, remains the simple parent-child hierarchy.

Generally speaking, trees still remain by far the most pervasive user interface model in computer applications today. Their allure derives from humans' natural tendency to form classification hierarchies in order to make sense of their world[Collins and Quillian 1969], and from their need to focus on a desired level of abstraction, which trees enable by hiding and concealing levels of detail.

b. Graph. Graph-based systems allow users to create knowledge elements and then to interconnect them in arbitrary ways. The elements of a graph are traditionally called "vertices," and connected by "arcs," though the terminology used by graph-based systems varies widely (see Table 1) and the hypertext community normally uses the terms "nodes" and "links." There are no restrictions on how many arcs one vertex can have with others, no notion of a "parent/child" relationship between vertices (unless the user chooses to label an arc with those semantics), and normally no "root" vertex. In many systems, arcs can optionally be labeled with a word or phrase indicating the nature of the relationship, and adorned with arrowheads on one or both ends to indicate navigability. (Neither of these adornments is necessary with a tree, since all relationships are implicitly labeled "parent/child" and are navigable from parent to child.) Note that a graph is a more general form of a tree. By using only arcs that are navigable in one direction (a "directed graph"), electing one vertex to be the "root," and ensuring that all non-root vertices have exactly one incoming arrowhead, a graph can represent everything a tree can. Hence it is a strictly more powerful form of expression.

This model is the defining characteristic of hypertext systems ([Halasz and Schwartz 1994]) including many of those used for document management ([Adar, *et al.* 1999], [Wolber, *et al.* 2002]). It is also the underpinning of all concept-mapping tools, whether

they actually acknowledge the name "concept maps" ([Canas, *et al.* 2005], [Gaines and Shaw 1995]) or advertise themselves simply as tools to draw knowledge diagrams ([Cognitive-Tools 2005], [Omni 2005]) As mentioned previously, graphs draw their power from the fact that humans are thought to model knowledge as graphs (or equivalently, semantic networks) internally. In fact, it could be argued that all human knowledge can be ultimately reduced to a graph of some kind, which argues strongly for its sufficiency as a structural framework. (See also [Quillian 1968], [Nosek and Roth 1990].) An interesting aspect of graph-based systems is whether or not they require a *fully-connected* graph. A fully-connected graph is one in which every vertex can be reached from any other by simply performing enough arc traversals. There are no "islands" of vertices that are severed from each other. Most graph-based tools allow non-fully-connected graphs: knowledge elements are added to the system, and connected arbitrarily to each other, without constraint. But a few tools, such as PersonalBrain[TheBrain 2005] and Compendium[Compendium 2005], actually require a single network of information in which every knowledge element must be indirectly connected to every other. If one attempts to remove the last link that connects a body of nodes to the original root, the severed elements are either "forgotten" or else moved to a deleted objects heap where they can only be accessed by restoring a connection to the rest of the graph.

	Vertex	**Arc**	**Graph**
Axon Idea Processor	object	link	diagram
Banxia Decision Explorer	concept	link	view
Compendium	node	link	view
Haystack	needle	tie	bale
Idea Graph	idea	connection	ideagraph
Knowledge Manager	concept	relation	map
MyLifeBits	resource	link/annotation	story
NoteCards	note card	link	browser
PersonalBrain	thought	link	brain
RecallPlus	idea	association	diagram
SMART Ideas	symbol	connector	level

Terminology employed by a sampling of graph-based knowledge tools. (Interestingly, the standard mathematical terms "vertex," "arc," and "graph" are used by none of them.)

We also note in passing that some hypertext systems (e.g., [Delisle and Schwartz 1986], [Garrett, *et al.* 1986]) add precision to the basic linking mechanism by allowing nodes to reference not only other nodes, but sections within nodes. (see [Halasz and Schwartz 1994].) This ability is especially useful if the nodes themselves contain sizeable content, and also for PKB elements making reference to fragments of objective sources.

c. Tree plus graph. Although graphs are a strict superset of trees, trees offer some important advantages in their own right: simplicity, familiarity, ease of navigation, and the ability to conceal details at any level of abstraction. Indeed, the problem of "disorientation" in hypertext navigation ([Conklin 1987], [Mantei 1982]) largely disappears with the tree model; one is never confused about "where one is" in the larger

structure, because traversing the parent hierarchy gives the context of the larger surroundings. For this reason, several graph-based systems have incorporated special support for trees as well, to combine the advantages of both approaches.

We have already seen one example of this with concept mapping techniques: a generally hierarchical paradigm is prescribed, after which users are encouraged to identify "crosslinks" between distant concepts. When systems using the mind mapping paradigm permit arbitrary relationships between nodes, they are taking the same path.

One of the earliest systems to combine tree and graph primitives was TEXTNET[Trigg and Weiser 1986], which featured two types of nodes: "chunks" (which contained content to be browsed and organized) and "table of contents" nodes (or "tocs".) Any node could freely link to any other, permitting an unrestricted graph. But a group of tocs could be combined to form a tree-like hierarchy that bottomed out in various chunk nodes. In this way, any number of trees could be superimposed upon an arbitrary graph, allowing it to be viewed and browsed as a tree, with all the requisite advantages.[7] NoteCards[Halasz, *et al.* 1987] offered a similar mechanism, using "FileBoxes" as the tree component that was overlaid upon the semantic network of notecards.

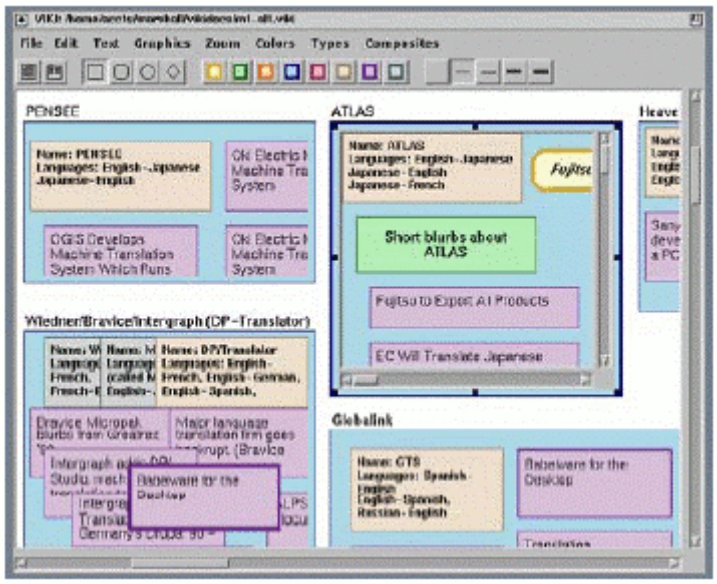
Brown University's IGD project explored various ways to combine and display unrestricted graphs with hierarchy, and used a visual metaphor of spatial containment to convey both graph and tree structure.[Feiner 1988] Their notion of "link inheritance" simplifies the way in which complex dual structures are displayed while still faithfully depicting their overall trends. Commercially, both PersonalBrain[TheBrain 2005] and Multicentrix[Koy 1997] provide explicit support for parent/child relationships in addition to arbitrary connections between elements, allowing tree and graph notions to coexist. Some note-taking tools, while essentially tree-based, also permit crosslinks between notes (e.g., [MicroLogic 2005], [WJJ 2005].)

d. Spatial. In the opposite direction, some designers have shunned links between elements altogether, favoring instead spatial positioning as the sole organizational paradigm. Capitalizing on the human's tendency to implicitly organize through clustering, making piles, and spatially arranging, some tools offer a 2D workspace for placing and grouping items. This provides a less formal (and perhaps less intimidating) way for a user to gradually introduce structure into a set of items as it is discovered.

This approach originated from the spatial hypertext community, demonstrated in projects like Boxer[diSessa and Abelson 1986] and VIKI/VKB ([Marshall and Shipman 1995], [Shipman, *et al.* 2000]). With these programs, users place information items on a canvas and can manipulate them to convey organization imprecisely. VIKI and VKB are especially notable for their ability to automatically infer the structure from a user's freeform layout: a spatial parser examines which items have been clustered together,

[7] Strictly speaking, a network of tocs formed a DAG (directed acyclic graph) rather than a tree. This simply means that a "chunk" could be represented in multiple places in the tree, if two different traversal paths ended up referring to the same chunk. We will revisit this when discussing transclusion, below; a DAG is essentially the result of applying transclusion to the tree model. This is also true of NoteCards.

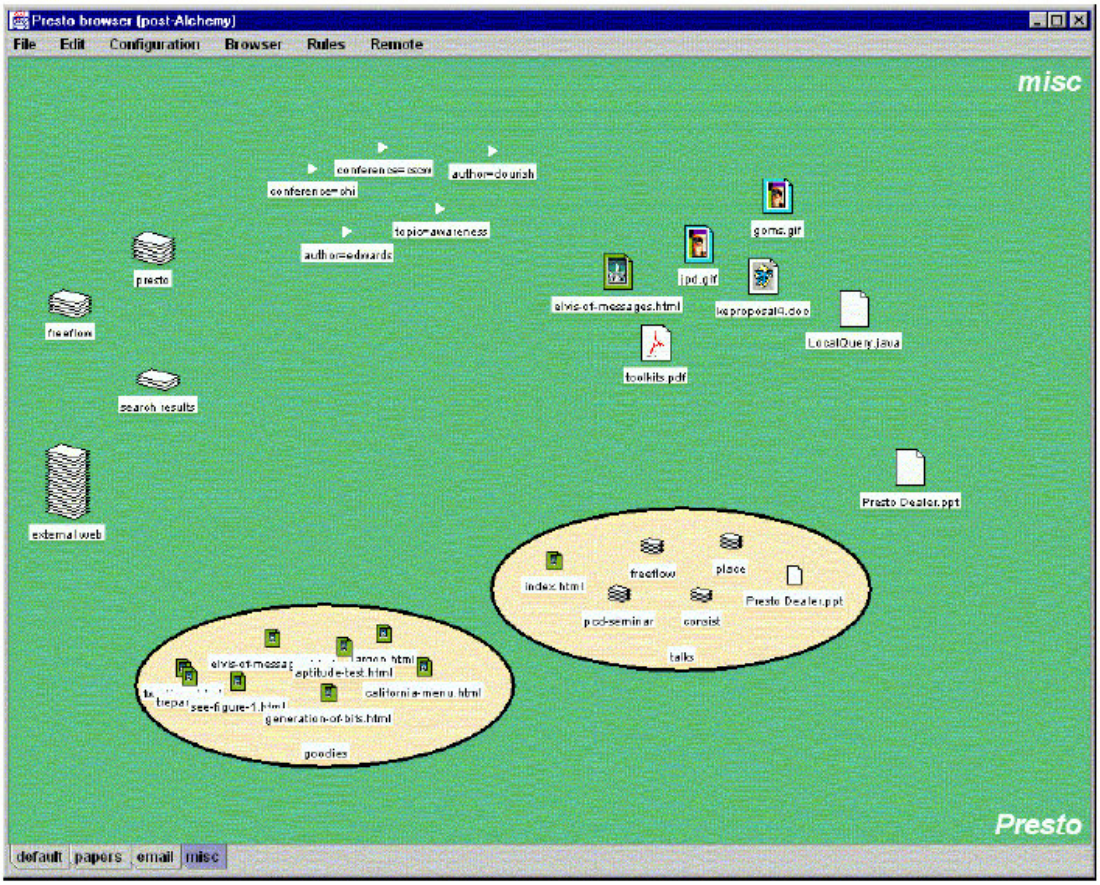
colored or otherwise adorned similarly, etc., and makes judgments about how to turn these observations into machine-processible assertions. A containment hierarchy in which one workspace subsumes another adds tree-like features to these approaches, although the user cannot typically see an overview of the entire hierarchy and thus navigate it at a glance. This is partially offset by the ability to "peer inside" a child workspace from its parent and peek at the nested structure.



VIKI, one of the first spatial hypertext systems. Rather than links between elements, the primary way organizational information is conveyed is through spatial clustering. [Marshall and Shipman 1995]

Pad[Perlin and Fox 1993] uses spatial positioning to organize items on a single, flat, gigantic canvas. The workspace is navigated by means of "portals" – instruments that give a magnifying glass effect to explore different parts of the information plane. Users can view different objects in varying levels of detail as they share the workspace.

Certain note-taking tools (e.g., [Burger, *et al.* 1991], [Microsoft 2003]) combine an overarching tree structure with spatial freedom on each "page." Users can access a particular page of the notebook with basic search or tree navigation facilities, and then lay out notes and images on the page as desired. The KMS hypermedia system offered the same spatial freedom on each of its "frames."[Akscyn, *et al.* 1987] Most of the integrated search workspaces (e.g., NaviQue[Furnas and Rauch 1998]) are also in this category. Many graph-based approaches (such as concept mapping tools) also allow for arbitrary spatial positioning of elements. This allows both kinds of relationships to be expressed: explicit links and less formal expression through creative use of the screen.

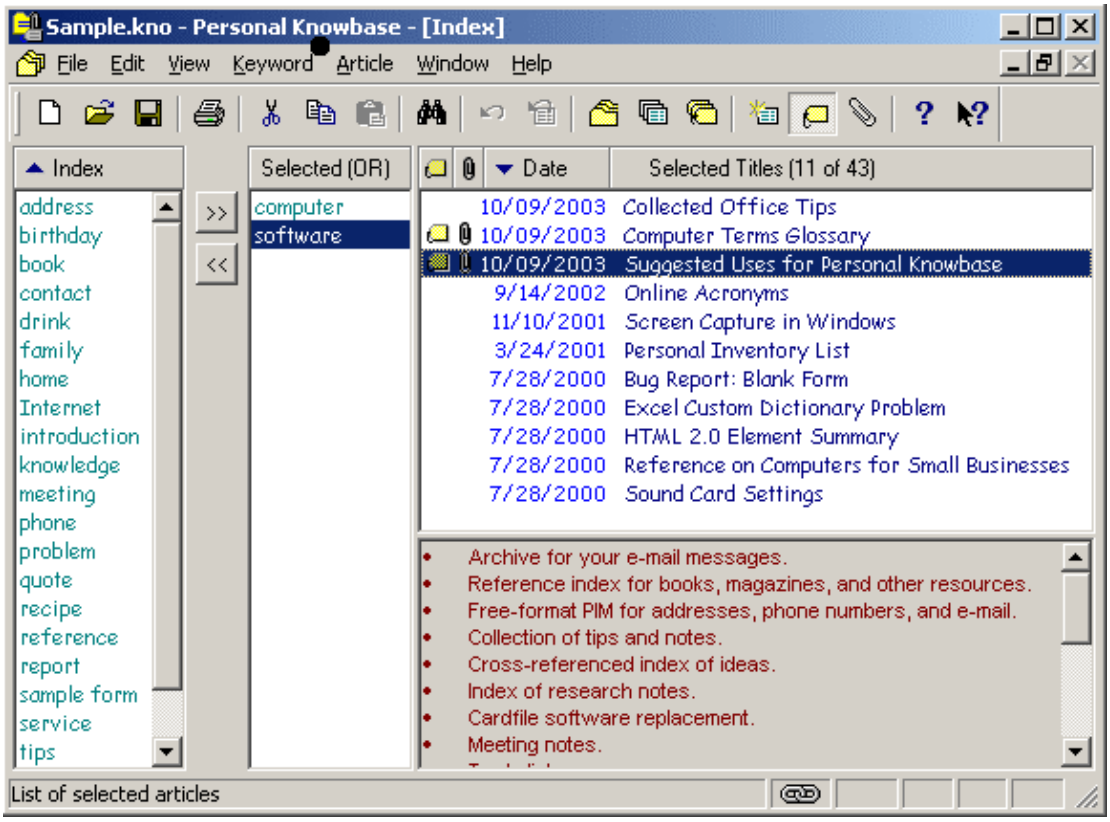


The Presto category-based browser, also known as "Vista." Although not shown very clearly in the diagram, items in the information space can be grouped in multiple categories at the same time, which is the cardinal feature of the category-based structural framework.[Dourish, et al. 1999]

e. Category. The fifth fundamental structural framework that candidate PKB systems use is that of categories. Rather than being described in terms of their relationships to other elements (as with a tree or graph), items are simply grouped together in one or more categories, indicating that they have something in common. Though seldom acknowledged, this scheme is based on the branch of pure mathematics called "set theory," in which each of a body of objects either has, or does not have, membership in each of some number of sets. There is normally no restriction as to how many different categories a given item can belong to, as is the case with mathematical sets.

Users may think of categories as collections, in which the category somehow encloses or "owns" the items within it. Indeed, some systems depict categories in this fashion, such as the Vista interface[Dourish, *et al.* 1999] where icons standing for documents are enclosed within ovals that represent categories. This is merely a convention of display, however, and it is important to note that fundamentally, categories are the same as simple keywords. Stating that a given piece of information is in the "to do today" category and also in the "need to inform Nancy" category is identical to annotating the item with both

the “to do today” and the “need to inform Nancy” keywords (or keyphrases.) It might be argued that it is wiser to avoid a graphical “containment” metaphor altogether for categories, as it may mislead users into thinking that a given item can only be in a single category at a time. To the contrary, the category approach derives its power from the fact that this is *not* the case. Permitting a single item to be in multiple “places” at a time (ie., associated with multiple groups) relieves the tree model’s most restrictive constraint.



Personal Knowbase, a note-taking system based on the category structural framework. Multiple customized user keywords (listed in far left pane) can be assigned to each note (which is comprised of a small text document and a title.) These keywords can be combined with boolean operations in order to retrieve notes.[Bitsmith 2005]

The most popular application to embrace the category approach was the original Agenda[Kaplan, *et al.* 1990], which later became a commercial product and spawned many imitations. Kaplan et al. describe Agenda as an “item/category database,” where categories are the fundamental organizational and retrieval construct. They claim that this is more powerful than the hypertext (graph-based) model, because relationships can be identified among *groups* of items, rather than simply among individual items. All information retrieval in Agenda was performed in terms of category membership. Users specified queries that were lists of categories to include (or exclude), and only items that satisfied those criteria were displayed. The user could then filter, sort, and group the results in various ways, again with categories as the control mechanism. Agenda was particularly sophisticated in that the categories themselves formed a tree hierarchy, rather

than a flat namespace. Assigning an item to a category also implicitly assigned it to all ancestors in the hierarchy, so that searches could be performed at varying levels of granularity. Agenda also provided more advanced forms of expression, allowing a user to specify certain categories as mutually exclusive, or even construct arbitrarily complex logical conditions for chained category assignment.

Personal Knowbase[Bitsmith 2005] is a more modern commercial product based solely on a keyword (category) paradigm, though it uses a simple flat keyword structure rather than an inheritance hierarchy like Agenda. Haystack[Adar, *et al.* 1999] and Chandler[OSAF 2005] are other information management tools which use categorization in important ways. William Jones' Memory Extender[Jones 1986] took an artificial intelligence twist on the whole notion of keywords/categories, by allowing an item's keywords to be weighted, and adjusted over time by both the user and the system. This allowed the strength of category membership to vary dynamically for each of an item's assignments, in an attempt to yield more precise retrieval.

2. The role of transclusion

Before we take a brief look at a few other structural frameworks that have been proposed, we wish to consider the important property of "transclusion." This term, first coined by Ted Nelson[Nelson 1995], signifies an embedded reference in one document that points to a portion of another document. This allows any updates to the referred-to document to be instantly be seen by the referring one, and avoids having to copy and store the relevant passage in multiple locations. In computer programming parlance, it is essentially nothing more than a "by reference" as opposed to a "by value" inclusion of external material. Nelson's original motive for this was to guarantee the payment of royalties to quoted authors in his vision for a worldwide information network. But when we consider this idea in the context of PKB data models, we find that it is of fundamental importance.

For us, transclusion means the ability to view the same knowledge element (not a copy) in multiple contexts. It is so pivotal because it is central to how the human mind processes information. Indeed, the whole idea of associative memory demands it: I may think of Bob and Joe as related because they both attend my seminar, but I may associate Joe and Sue because they share an interest in cross-country skiing. Here, the same person "Joe" is linked to two other elements because I think of him in two different contexts, and there are potentially dozens of others. I certainly do not relate Sue to "a copy of Joe," but to *Joe*, and if I acquire new information about Joe, this would instantly be available in all the contexts I associate with him. Without delving into psychological research to examine exactly how the mind encodes such associations, it seems clear that if we are to build a comprehensive personal knowledge base containing potentially everything a person knows, it must have the ability to transclude knowledge elements. The ability to repurpose knowledge is vital if our system is to be faithful to the way the mind operates.

This strikes at the heart of why the tree model is so limiting. In a tree, a given piece of information – whether a file, a Web bookmark, an e-mail message, or a knowledge element – is filed away in a single location. It has only a single ancestral path to the root

of the tree, and hence, only appears in a single context. Much research into human-computer interaction has investigated various ways of relaxing this paralyzing constraint.

Let us now consider how each of the structural frameworks we have explored is affected by transclusion. It should be noted that the category model inherently *is* a transclusive model, and that is what gives it its power: a given piece of information can be simultaneously "in" many different categories, and these categories are independent of one another. To use Kaplan et al.'s example[Kaplan*, et al.* 1990], this is what allows a user to assign an item "Call Fred next Tuesday about pricing policy plans" to several categories at once: "Phone calls," "Fred Smith," and "Pricing policy committee." This fits naturally with the user's conception, and allows the item to be retrieved later along multiple paths.

Adding transclusion to the tree model effectively turns it into a directed acyclic graph (DAG), in which an item can have multiple parents. This is what Trigg[Trigg and Weiser 1986] and Halasz[Halasz*, et al.* 1987] achieved with their extensions to the tree model. In TEXTNET, for example, a primitive information "chunk" can be pointed to by multiple "tocs" nodes, and hence present in multiple places in the same table-of-contents hierarchy. Altering the "chunk" changes its appearance in all contexts. From a PKB perspective, this is a great improvement over the strict tree, since it permits the same sort of flexibility as the category-based model. In fact, a category *hierarchy* (as in Agenda) is virtually equivalent to the DAG model: non-leaf nodes in the DAG represent categories, and the leaves (or "chunks" in TEXTNET's nomenclature) are the items within those categories.[8] Zoot[Zoot 2005] and Hog Bay[HogBay 2005] are tree-based commercial products that also let the user transclude items into multiple containers.

Boxer[diSessa and Abelson 1986] incorporates transclusion into the spatial model by its "port" construct. Normally, an information element (a "box" in Boxer's terminology) is accessible inside its immediate container, but not outside. To overcome this, however, a user can create "a port, which is simply a view of a box at some other place in the system. A port behaves in most respects identically to the box it views – any change in one will automatically cause the same change in the other." A single box can thus virtually appear on multiple container boxes.

A similar mechanism can be applied to graph models, as with Tinderbox's "alias" feature.[Bernstein 2003] In Tinderbox, information is broken up into "notes," which can appear on the screen as spatially laid out rectangles with links between them. By creating an "alias" for a note, one can summon its appearance on a different graph layout than the note originally appeared. An alias thus functions in the same way as Boxer's ports do. Note one limitation with both Boxer's and Tinderbox's approaches, however: the "port" or "alias" is still inherently *secondary* to the original box or note to which it refers. The port/alias is what points to the original note, not vice versa, and hence if the latter disappears or is renamed or repositioned, difficulties may arise.

[8] We say "virtually" equivalent because there is a subtle difference: with a DAG, a category *itself* could be a subcategory of more than one category. This is not possible in Agenda, where the categories form a strict tree, and only the items have transclusive properties.

These issues are solved by Compendium[Conklin, *et al.* 2001], the most thorough implementation of transclusion we have seen for a graph-based tool. In Compendium, it is the actual node (rather than an alias) that is present on multiple views, without any limitations. If the user creates a node A on view 1, then adds A to view 2, and deletes it from view 1, A will exist solely on view 2 without any dangling references. This is because in Compendium's relational database scheme, the node A has its own existence, quite separate from any view in which it might appear. This seems closer to how the mind operates: we associate ideas with contexts, but we do not embed ideas irreversibly into the first context we happened to place them in, forcing other contexts to "point" to the original location. Rather, the mind is fluid, freely associating and disassociating ideas from others so that our mental model can naturally evolve. Tightly binding an element to its original context, therefore, seems like the wrong approach.

In summary, transclusion is a property that can be advantageously combined with any of the models we have seen. It permits an item to appear in multiple contexts, just as the human mind considers ideas in multiple contexts. Thus the presence of transclusion in some form seems essential in order for a diverse PKB to grow coherently.

3. Alternative frameworks

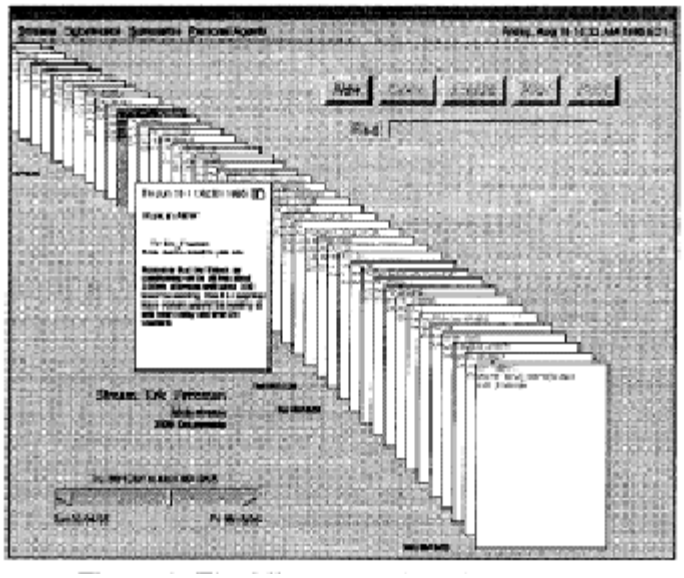
The vast majority of candidate PKB tools are based on one of the five principal frameworks, but we will mention three notable others that researchers have experimented with:

a. Chronological. Yale University's Lifestreams project[Fertig, *et al.* 1996] used timestamps as the principal means of organization and retrieval of personal documents. In Fertig et al.'s own words:

> A *lifestream* is a time-ordered stream of documents that functions as a diary of your electronic life; every document you create is stored in your lifestream, as are the documents other people send you. The tail of your stream contains documents from the past, perhaps starting with your electronic birth certificate. Moving away from the tail and toward the present, your stream contains more recent documents such as papers in progress or the latest electronic mail you've received…

Documents are thus always ordered and accessed chronologically. Metadata-based queries on the collection produce "substreams," or chronologically-ordered subsets of the original documents. The rationale for time-based ordering is that "time is a natural guide to experience; it is the attribute that comes closest to a universal skeleton-key for stored experience."[Freeman and Gelernter 1996] Whether chronology is our principal or even a common natural coding mechanism psychologically can be debated. But since any PKB system can create such an index "for free" (it is a simple matter to record the time of any change to the knowledge base), it seems worthwhile to follow Lifestreams' lead and allow the user to sort and retrieve based on time. If nothing else, it relieves the user from having to create names for knowledge elements, since the timestamp is always an implicit

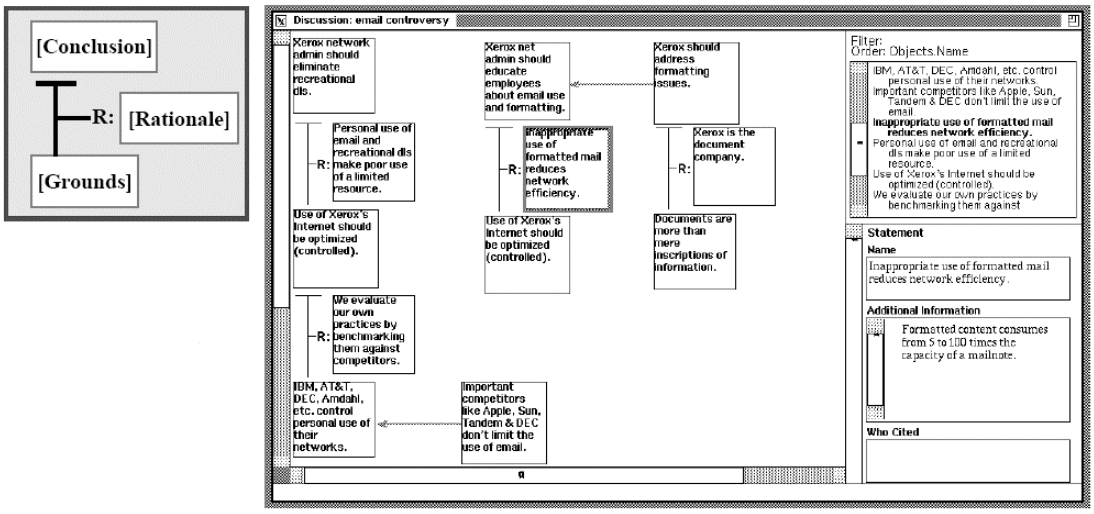
identifying mark. PlanPlus[FranklinCovey 2005], based on the Franklin-Covey planner system, is also chronologically modeled, and a number of products based on other data models (e.g., [Kaplan*, et al.* 1990], [CircusPonies 2005]) offer chronological indexing in addition to their core paradigm.



Lifestreams, an exclusively chronologically-based information management system. [Fertig, et al. 1996]

b. Aquanet. Though advertised as a hypertext system, Marshall et. al's Aquanet[Marshall*, et al.* 1991] went far beyond the traditional node-link graph model. Knowledge expressed in Aquanet is centered around "relations," or n-ary links between objects in which the semantics of each participant in the relation is specified by the relation type. Each type of relation specifies a physical display (ie., how it will be drawn on the screen, and the spatial positioning of each of its participants), and a number of "slots" into which participants can be plugged in. Each participant in a relation can be either a base object, or another relation. Users can thus define a schema of relation types, and then build a complex semantic model out of relations and objects. Since relation types can be specified to associate any number of nodes (instead of just two, as in the graph model), this potentially allows more complex relationships to be expressed.[9]

[9] It should be noted, however, that the same effect can be achieved in the basic graph model by simply taking the n-ary relations and "reifying" them (ie., turning them into nodes in their own right.) For instance, suppose we define a relation type "assassination," with slot types of "assassin," "victim," "location," and "weapon." We could then create a relation based on this type where the participants are "John Wilkes Booth," "Abraham Lincoln," "Ford's Theatre," and "derringer." This allows us to express a complex relationship between multiple objects in Aquanet. But we can express the same knowledge with the basic graph model by simply creating a node called "Lincoln's assassination" and then creating typed links between that node and the other four labeled "assassin," "victim," etc. Aquanet's biggest achievement in this area is the ability to express the schema of relation types, so that the types of objects an "assassination" relation can connect are consistent and enforced.



The Aquanet data model. The user can define "relations," or templates for semantic n-ary based links between objects. (See the upper-left corner for an example.) Knowledge elements can then be linked together with these relations into structures of arbitrary complexity (see right-hand side of the diagram.)[Marshall, et al. 1991]

c. Zigzag. Finally, we mention the data model of Zigzag[Nelson 1999], the successor to Ted Nelson's original Xanadu project. Zigzag is a flexible augmentation of the original hypertext model, in which information "cells" (nodes) are connected together along any number of linear "dimensions." Each dimension has an upstream and a downstream direction, and multiple dimensions allow a cell to be simultaneously in many different linear contexts. In programming terms, this pattern is equivalent to having a number of objects, each of which is a participant in an arbitrary number of independent, doubly-linked lists. By carefully arranging the cells and their dimensions, many standard data structures (lists, spreadsheets, even trees) can be represented in the Zigzag model. This scheme may eventually prove to be the elegant generalization of all the structural frameworks we have presented here; currently, however, it is not in widespread use.

B. Knowledge elements

Having surveyed the various options for structuring knowledge elements together in a PKB, we now turn our attention to the elements themselves: of what do they consist, and what kind of internal structure (if any) do they possess?

There are several options here, most of which can be combined:

1. **Word/phrase/concept**. Most systems engineered for knowledge representation encourage structures to be composed of very simple elements, usually words or phrases. This is in the spirit of both mind mapping and concept mapping, where users are encouraged to use simple phrases to stand for mental concepts. Decision Explorer, too, restricts the nodes in its graph to be phrases, and does not allow anything else.[Banxia 2005] Note that if a user were to break up all of the knowledge they intended to capture

into a semantic network, this is presumably all the result would consist of. One could argue that if any significant free text remains (see #2, below) then the user has not completed the process of converting their serialized information into a proper conceptual framework. As a practical matter, of course, users may not always wish to invest the time to do that, which makes it advantageous to permit free text to be stored.

2. **Free text notes**. For this reason, nearly all systems permit large amounts of free text to exist in the PKB, either as the contents of the elements themselves (NoteCards[Halasz, *et al.* 1987], Hypercard[Goodman 1988], TreePad[Freebyte 2005]) or attached to elements as separate, supplementary pages (Agenda[Kaplan, *et al.* 1990], Zoot[Zoot 2005], Hog Bay[HogBay 2005]). There is a danger here, since if a system encourages free text to proliferate, then the user's knowledge base is prone to becoming a dumping ground for unprocessed information, rather than distilled and encoded knowledge. Nevertheless, the majority of system designers have found this useful.

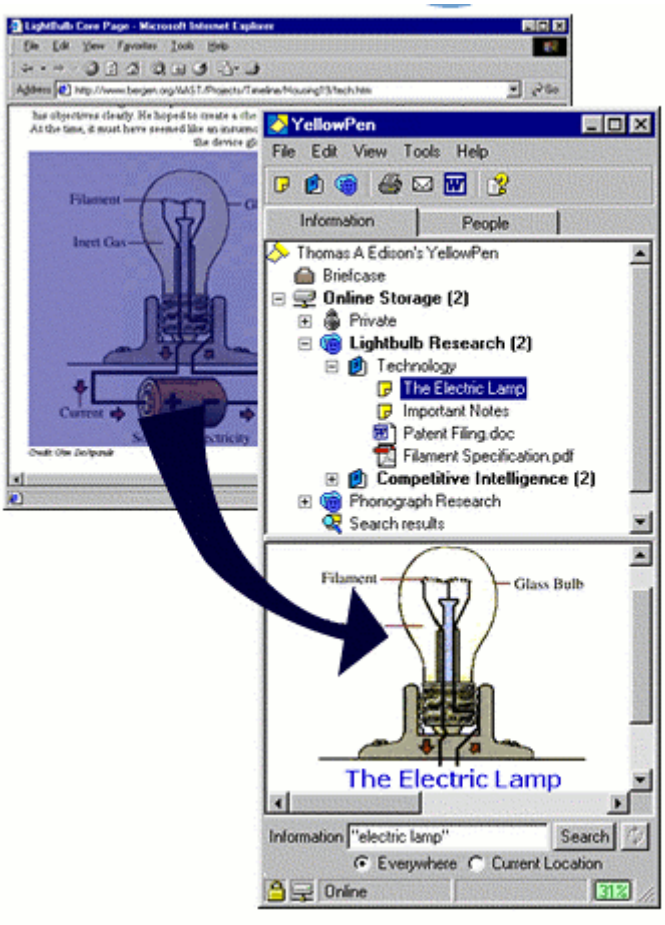
3. **Links to the objective space**. We have spoken about the "subjective-objective divide," and how if a user's knowledge base is to correspond to their mental perceptions, it should be possible for the PKB to point to entities in the objective realm. Many systems do in fact allow their knowledge elements to point to the objective space in some way. There are three common techniques:

a. The knowledge element actually *represents* an item in the objective space. This is the case for document management systems (WebTop[Wolber, *et al.* 2002], MyLifeBits[Gemmell, *et al.* 2002], Haystack[Adar, *et al.* 1999]), integrated search facilities (NaviQue[Furnas and Rauch 1998], CYCLADES[Renda and Straccia 2005]), VIKI/VKB ([Marshall and Shipman 1995], [Shipman, *et al.* 2000]). Tinderbox[Bernstein 2003] will also allow one of its notes to *be* a URL, and the user can control whether its contents should be captured once, or "auto-fetched" as to receive constant web updates.

b. The knowledge element *contains a link* to the objective space. Many systems, in addition to storing a page of free text for each knowledge element, also permit any number of hyperlinks to be attached to a knowledge element (e.g., Freemind[FreeMind 2005], PersonalBrain[TheBrain 2005], Inspiration[Inspiration 2005]). VNS[Burger, *et al.* 1991], which allows users to point to a community notebook page from within their personal notebook, gives similar functionality.

c. The knowledge element is a *repurposed snippet* from the objective space. This is the most powerful form of subjective-objective bridging, but is sorely lacking from most fully-featured PKB systems. Cartagio[Missiontrek 2005], Hunter-Gatherer[Schraefel, *et al.* 2002], and YellowPen[YellowPen 2005] all allow Web page excerpts to be assimilated and organized, but they primarily *only* do that, without allowing them to easily be combined with other subjective knowledge. DEVONThink[DEVON 2005] and MyBase's WebCollect plug-in[WJJ 2005] add similar functionality to their more general-purpose, tree-based information managers. Both of these systems, when a snippet is captured, archive the entire Web page locally so it can be returned to later. The user interfaces of Circus Ponies Notebook[CircusPonies 2005] and Sticky Brain[Chronos

2005] have been heavily optimized towards grabbing information from other applications and bringing them into the PKB without disturbing the user's workflow.



YellowPen, which allows snippets of textual or graphical content to be captured from the web and repurposed in the user's own personal (hierarchical) structure.[YellowPen 2005]

4. **Composites**. Finally, some programs allow a user to embed knowledge elements (and perhaps other information as well) *inside* a knowledge element to form an implicit hierarchy. Trees by themselves fall into this category, of course, since each node in the tree can be considered a "composite" of its content and children. But a few graph-based tools offer composite functionality as well. In Aquanet[Marshall, *et al.* 1991], as we have already seen, "relations" form the fundamental means of connection, and the units that are plugged into a relation can be not only objects, but other relations as well. This lends a recursive quality to a user's modeling. VIKI/VKB's spatial environment offers "subspaces" which let a user partition their visual workspace into subregions, whose internal contents can be viewed at a glance from the parent. Boxer[diSessa and Abelson 1986]'s paradigm is similar. Tinderbox is a graph-based tool that supports hierarchical composite structures, and Compendium[Compendium 2005] extends this even further by allowing transclusion of "views" as well as of nodes. Unlike the other tools, in

Compendium the composite hierarchy does not form a DAG, but rather an arbitrary graph: view A can appear on view B, and B can in turn appear on A. The user's intuitive notion of "inside" must be adapted somewhat in this case.

C. Schema

Finally, we consider the notion of "schema" in various systems' data models. By schema we mean the ability for a user to specify types and introduce structure to aspects of the data model. It is a form of metadata whereby more precise semantics can be applied to the elements of the system. This allows more formal knowledge expression, ensures consistency across items of the same kind, and better allows automated agents to process the information, as we will see later. We consider first knowledge elements, then links.

1. Schema for knowledge elements

a. Types, and related schema. Generally speaking, systems can make knowledge elements untyped, rigidly typed, or flexibly typed. In addition, they can incorporate some notion of inheritance among elements and their types.

Note that we distinguish between types and categories here. A category-based scheme, as previously discussed, typically allows any number of categories/keywords to be assigned to an item. There are two differences between this and the notion of type. First, items are normally restricted to being of a *single* type, and this usually indicates a more intrinsic, permanent property of an item than simply its presence in a category collection. (For example, we could imagine an item called "XYZ Corporation" shifting into and out of categories like "competitors, " "overseas distributors," or "delinquent debtors," but its core type of "company" would probably be static for all time.) Second, types often carry structural specifications with them: if an item is of a given type, this means it will have values for certain attributes appropriate to that type, as we will see later. Note that some systems that do not allow typing offer the ability to approximate this function through categories. (e.g., OneNote[Microsoft 2003], Mind Manager[MindJET 2005]).

Untyped elements are typical among informal knowledge capture tools, since they are designed to stimulate brainstorming and help users discover their nascent mental models. These tools normally want to avoid forcing the user to commit to structure prematurely. Most mind mapping and many concept mapping tools are in this category: a concept is simply a word or phrase, with no other semantic information (e.g., VisualMind[Mind 2005]). Note-taking tools also usually take this approach, with all units of information being of the same type "note."

At the other extreme are tools which, like older relational database technology, require all items to be declared as of a specific type when they are created. Often this type dictates the internal structure of the element. NoteCards took this approach; each "card" was declared to be of a particular type (text card, sketch card, query card, etc.) that determined what sort of information could appear on the card. In this case, typing was used simply to control the kind of media the item contained, not the semantic category of

the conceptual entity. Frame-based systems such as SPRINT[Carlson and Ram 1990] and Aquanet add semantics to this scheme: as in object-oriented technology, each element has a declared type, which fixes the "slots" (fields) of information that it contains, and their meanings. These tools are better suited to domains in which the structure of knowledge to be captured is predictable, well-understood, and known in advance. For PKB systems, they are probably overly restrictive. KMap[Gaines and Shaw 1995] and Compendium are examples of tools that allow (and require) each item to be typed; in their case, the type controls the visual appearance of the item, rather than any internal structure. In KMap these types are invented by the user; in Compendium, they are hardcoded to a particular domain (organizational decision-making.)

In between these two poles are systems that permit typed and untyped elements to co-exist. AquaMinds NoteTaker[AquaMinds 2005] is such a product; it holds simple free-text pages of notes, without any structure, but also lets the user define "templates" with predefined fields that can be used to instantiate uniformly structured forms. TreePad has a similar feature. Other systems blur the distinction between typed and untyped, allowing the graceful introduction of structure as it is discovered. VKB[Shipman*, et al.* 2000], for example, supports an elegant, flexible typing scheme, well suited to PKBs. Items in general consist of an arbitrary number of attribute/value pairs. But when consistent patterns emerge across a set of objects, the user can create a *type* for that group, and with it a list of expected attributes and default values. This structure can be selectively overridden by individual objects, however, which means that even objects assigned to a particular type have flexible customization available to them. Tinderbox offers an alternate way of achieving this flexibility, as we will see below.

Finally, the object-oriented notion of type inheritance is available in a few solutions. The different card types in NoteCards are arranged into an inheritance hierarchy, so that new types can be created as extensions of old. Aquanet extends this to *multiple* inheritance among types; the "slots" that an object contains are those of its type, plus those of all supertypes. SPRINT and Tinderbox also use a frame-based approach, and allow default values for attributes to be inherited from supertypes. This way, an item need not define values for all its attributes explicitly: unless overridden, an item's slot will have the shared, default value for all items of that type.

In general, inheritance is not widely implemented in PKB systems. Perhaps this is because non-technical users find the concept foreign, or perhaps because its only real selling point is more detailed structure in a realm where unstructured data is the rule. Nearly two decades ago, Halasz[Halasz 1988] suggested that incorporating inheritance as well as other object-oriented constructs into hypertext systems would be beneficial, but to this point these ideas have not made their way into the mainstream, at least for the PKB domain. For other investigations in this area, we refer the reader to Klas, et al[Klas*, et al.* 1993] and Hatzopoulos[Hatzopoulos*, et al.* 1990].

b. Other forms of schema. In addition to the structure that is controlled by an item's type, other forms of metadata and schema can be applied to knowledge elements. Keywords and attribute/value pairs are the two we consider here.

Many systems let users annotate items with user-defined keywords. Here the distinction between an item's contents and the overall knowledge structure becomes blurred, since an item keyword could be considered either a property of the item, or an organizational mechanism that groups it into a category with like items. We have already covered the category data model, and seen that systems like Agenda use keywords for the latter purpose. We note here that systems based on other data models also use keywords to achieve category-like functionality. Circus Ponies, a tree-based note-taking application, allows "keywords, stickers, and highlighting" to adorn its notes, all of which are forms of category annotations. OneNote[Microsoft 2003], Mind Manager[MindJET 2005], and other tools offer similar features.

Arbitrary attribute/value pairs can also be attached to elements in many systems, which gives a PKB the ability to define semantic structure that can be queried. We have already seen examples of this with frame-based systems like SPRINT and Aquanet, as well as NoteTaker, VKB, and Tinderbox. MindPad[AKS-Labs 2005] is notable for taking the basic concept mapping paradigm and introducing schema to it via its "model editor." As mentioned earlier, adding user-defined attribute/value pairs to the items in an outliner yields spreadsheet-like functionality, as in Ecco[NetManage 1997] and OmniOutliner[Omni 2005]. Note that some systems feature attribute/value pairs, but only in the form of system-defined attributes, not user-defined ones. (e.g., Mind Manager, StickyBrain[Chronos 2005]).

c. Knowledge element appearance. Finally, some tools modify a knowledge element's visual appearance on the screen in order to convey meaning to the user. SMART Ideas[SMART 2005] and VisualMind[Mind 2005] let the user freely choose each element's icon from a variety of graphics, while KMap[Gaines and Shaw 1995] ties the icon directly to its underlying type. Other graphical aspects that can be modified include color ([Marshall and Shipman 1995]), the set of attributes shown in a particular context (VKB[Shipman*, et al.* 2000]), and the spatial positioning of objects in a relation (Aquanet[Marshall*, et al.* 1991]).

2. Schema for links

In addition to prescribing schema for knowledge elements, many systems allow some form of information to be attached to the links that connect them.

In most of the early hypertext systems, links were unnamed and untyped, their function being merely to associate two items in an unspecified manner. The mind mapping paradigm also does not name links, but for a different reason: the implicit type of every link is one of generalization/specialization, associating a topic with a subtopic. Hence specifying types for the links would be redundant, and labeling them would clutter the diagram.

Concept mapping prescribes the naming of links, such that the precise nature of the relationship between two concepts is made clear. As mentioned above, portions of a

concept map are meant to be read as English sentences, with the name of the link serving as a verb phrase connecting the two concepts. Numerous systems thus allow a word or phrase to decorate the links connecting elements, for instance Cmap[Canas, *et al.* 2005] and Inspiration[Inspiration 2005].

Named links can be distinguished from *typed* links, however. If the text attached to a link is an arbitrary string of characters, unrelated to that of any other link, we consider this the link name. Some systems, however, encourage the re-use of link names that the user has defined previously. In PersonalBrain[TheBrain 2005], for instance, before specifying the nature of a link, the user must create an appropriate "link type" (associated with a color to be used in presentation) in the system-wide database, and then assign that type to the link in question. This promotes consistency among the names chosen for links, so that the same logical relationship types will hopefully have the same tags throughout the knowledge base. This feature also facilitates searches based on link type, among other things. Other systems, especially those suited for specific domains such as decision modeling (gIBIS[Conklin and Begeman 1988], DecisionExplorer[Banxia 2005]), predefine a set of link types that can be assigned (but not altered) by the user.

Some more advanced systems allow links to bear attribute/value pairs themselves, and even embedded structure, similar to those of the items they connect. In Haystack[Adar, *et al.* 1999] this is the case, since links ("ties") and nodes ("needles") are actually defined as subtypes of a common type ("straw.") KMap similarly defines a link as a subclass of node, which allows links to represent n-ary relationships between nodes, and enables recursive structure within a link itself. It is unclear how much value this adds in knowledge modeling, or how often users would take advantage of such a feature. Neptune[Delisle and Schwartz 1986] and Intermedia[Garrett, *et al.* 1986] are two older systems that also support attributes for links, albeit in a simpler manner.

Another aspect of links that generated much fervor in the early hypertext systems was that of link *precision*: rather than merely connecting one element to another, systems like Intermedia defined anchors within documents, so that a particular snippet within a larger element could be linked to another snippet. The Dexter model[Halasz and Schwartz 1994] covers this issue in detail. For PKB purposes, this seems to be most relevant as regards links to the objective space, as we have discussed previously. If the PKB truly contains *knowledge*, expressed in appropriately fine-grained parts, then link precision between elements in the knowledge base is much less of a consideration.

Finally, note that throughout this discussion on links we have only been considering connections between knowledge elements in the system, where the system has total control over both ends of the connection. As described in the previous section, numerous systems provide the ability to "link" from a knowledge element inside the system to some external resource: a file or a URL, say. These external links typically cannot be enhanced with any additional information, and serve only as convenient retrieval paths, rather than as aspects of knowledge representation.

VI. Architecture

The idea of a PKB gives rise to some important architectural considerations. While not constraining the nature of what knowledge can be expressed, the architecture nevertheless affects more mundane matters such as availability and workflow. But even more importantly, the system's architecture determines whether it can truly function as a lifelong, integrated knowledge store – the "base" aspect of the personal knowledge base as we have defined it.

A. File-based

The vast majority of solutions presented in this survey use a simple storage mechanism based on flat files in a filesystem. This is true of virtually all of the mind mapping tools (Mind Manager[MindJET 2005]), concept mapping tools (Cmap[Canas*, et al.* 2005], Axon[Bok 2005], Inspiration[Inspiration 2005]), outliners (TreePad[Freebyte 2005], OmniOutliner[Omni 2005]), and note-taking tools (OneNote[Microsoft 2003], Hog Bay[HogBay 2005], Zoot[Zoot 2005]), and even a number of hypertext tools (NoteCards[Halasz*, et al.* 1987], Hypercard[Goodman 1988], Tinderbox[Bernstein 2003]). Typically, the main "unit" of a user's knowledge design – whether that be a mind map, a concept map, an outline, or a "notebook" – is stored in its own file somewhere in the filesystem. The application can find and load such files via the familiar "File | Open…" paradigm, at which point it typically maintains the entire knowledge structure in memory.

This is a very serious constraint and in our mind ultimately rules out any such system from truly serving as a PKB. This conclusion may sound too sweeping, but consider the ramifications of a file-based architecture on an individual's ongoing knowledge accumulation. The user must choose one of two basic strategies: either store all of their knowledge in a single file; or else break up their knowledge and store it across a number of different files, presumably according to subject matter and/or time period. The first choice results in insurmountable scalability problems – consider how much knowledge a user might collect over a decade, if they stored things related to their personal life, hobbies, relationships, reading materials, vacations, academic course notes, multiple work-related projects, future planning, etc. Surely it is unrealistic to keep adding this kind of volume to a single, bloated, ever-growing multi-gigabyte file! Just the time it would take the application to load it and save it is enough to render this untenable, to say nothing of backup concerns.

But the user's other choice is equally flawed: each bit of knowledge can be stored in only *one* of the files (or else redundantly, which leads to synchronization problems), and the user is forced to choose this at knowledge capture time. We have already spoken of the importance of flexibility in linking: the human mind can freely associate any two items together, so a PKB *must* support such unrestricted links. If we introduce artificial boundaries into the PKB, we have given up the game: the basic limitation of the tree model has ensnared us, in which an item is bound to a single context. To illustrate, suppose Phillip, a user of such a system, were to store the details of a particular work

procedure in his knowledge file for the XYZ Project he is working on at his place of business. Then he could not link this procedure to anything in the ABC Project, since that knowledge is in a separate file. Nor could he associate it with Julie (who suggested it to him at a cocktail party), since she is a friend of his, and knowledge about her is stored in Phillip's "social" knowledge file. Nor can he refer to it in his notes from last summer's seminar (during which a speaker suggested an alternative to such a procedure), since those notes are stored in his "seminars" knowledge file. The situation is hopeless. The user has lost most of the benefit of fluid knowledge capture, and has returned to the world of isolated, hierarchical, name-based storage.

We have dwelt on this point at some length both because it is so crucial and so commonly overlooked. Exceptions to the file-based paradigm are rare, which may be because of difficulty in implementation, or simply because of user familiarity with "open" and "save" operations. But ultimately, any system that takes such an approach is doomed in its efforts to serve as a PKB. The human mind is not rigidly partitioned into separate compartments, and neither must be any system that attempts to capture the knowledge the mind contains.

B. Database-based

We will not conceal our approbation for the small number of systems that have rejected the file paradigm and have embraced a relational database for their storage mechanism. This choice yields all the advantages that the file-based approach did not: scalability, reliability, and seamless uniformity throughout the knowledge base. Knowledge elements reside in a global space, which allows any idea to relate to any other: now a user can relate a book he read on productivity not only to other books on productivity, but also to "that hotel in Orlando that our family stayed in last spring," because that is where he remembers having read the book. Though such a relationship may seem "out of bounds" in traditional knowledge organization, it is exactly the kind of retrieval path that humans often employ in retrieving memories ([Anderson 1990], [Lorayne and Lucas 1974], [Conway, *et al.* 1991].) The database architecture enables a PKB to truly form an integrated knowledge *base*, and contain the full range of relationships.

Agenda[Kaplan, *et al.* 1990] and gIBIS[Conklin and Begeman 1988] were two early tools that subsumed a database backend in their architecture. More recently, the MyLifeBits project[Gemmell, *et al.* 2002] uses Microsoft SQL Server as its storage layer, and Compendium[Compendium 2005] interfaces with the open source MySQL database. A few note-taking applications such as StickyBrain[Chronos 2005] also store information in an integrated database rather than in user-named files. The only significant drawback to this architectural choice (other than the modest footprint of the database management system) is that data is more difficult to copy and share across systems. This is one true advantage of files: it is a simple matter to copy them across a network, or include them as an e-mail attachment, where they can be read by the same application on a different machine. This problem is solved by some of the following architectural choices.

C. Client-server

Decoupling the actual knowledge store from the PKB user interface can achieve architectural flexibility. As with all client-server architectures, the benefits include load distribution, platform interoperability, data sharing, and ubiquitous availability. Increased complexity and latency are among the liabilities, which can indeed be considerable factors in PKB design.

One of the earliest and best examples of a client-server knowledge base was the Neptune hypertext system.[Delisle and Schwartz 1986] Neptune was tailored to the task of maintaining shared information within software engineering teams, rather than to personal knowledge storage, but the elegant implementation of its "Hypertext Abstract Machine" (HAM) was a significant and relevant achievement. The HAM was a generic hypertext storage layer that provided node and link storage and maintained version history of all changes. Application layers and user interfaces were to be built on top of the HAM. Architecturally, the HAM provided distributed network access so that client applications could run from remote locations and still access the central store. Another, more recent example, is the Scholarly Ontologies Project ([Uren*, et al.* 2004], [Sereno*, et al.* 2005]) whose ClaiMapper and ClaiMaker components form a similar distributed solution in order to support collaboration.

These systems implemented a distributed architecture primarily in order to share data among colleagues. For PKBs, the prime motive is rather user mobility. This is a key consideration, since if a user is to store all of their knowledge into a single integrated store, they will certainly need access to it in a variety of settings. MyBase Networking Edition[WJJ 2005] is one example of how this might be achieved. A central server hosts the user's data, and allows network access from any client machine. Clients can view the knowledge base from within the MyBase application, or through a Web browser (with limited functionality.)

The Haystack project[Adar*, et al.* 1999] outlines a three-tiered architecture, which allows the persistent store, the Haystack data model itself, and the clients that access it to reside on separate machines. The interface to the middle tier is flexible enough that a number of different persistent storage models can be used, including relational databases, semistructured databases, and object-oriented databases. Presto's architecture[Dourish*, et al.* 1999] exhibits similar features.

D. Web-based

A variation of the client-server approach is of course Web-based systems, in which the client system consists of nothing but a (possibly enhanced) browser. This gives the same ubiquitous availability that client-server approaches do, while minimizing (or eliminating) the setup and installation required on each client machine.

KMap[Gaines and Shaw 1995] was one of the first knowledge systems to integrate with the World Wide Web. It allowed concept maps to be shared, edited, and remotely stored

using the HTTP protocol. Concept maps were still created using a standalone client application for the Macintosh, but they could be uploaded to a central server, and then rendered in browsers as "clickable GIFs." Clicking on a concept within the map image in the browser window would have the same navigation effect as clicking on it locally inside the client application. Hunter-Gatherer[Schraefel, *et al.* 2002], Cartagio[Missiontrek 2005], and Notestar[ALTEC 2005] are more recent browser-based systems that use proxies or browser plugins to achieve a knowledge building workspace. The user's knowledge expressions are stored on a central server in nearly all cases, rather than locally on the browser's machine.

E. Handheld devices

Lastly, we mention mobile devices as a possible PKB architecture. Storing all of one's personal knowledge on a palmtop computer would solve the availability problem, of course, and even more completely than would a client-server or web-based architecture. The safety of the information is an issue, since if the palmtop were to be lost or destroyed, the user could face irrevocable data loss; this is easily remedied, however, by periodically synchronizing the handheld device's contents with a host computer. More problematic is simply the limitations of the hardware. Screen real estate, processing power, and storage capacity are of course much more limited, and this hampers their overall effectiveness.

Most handheld applications are simple note-taking software, with far fewer features than their desktop counterparts. BugMe![Electric 2005] is an immensely popular note-taking tool that simply lets users enter text or scribble onto "notes" (screenfulls of space) and then organize them in primitive ways. Screen shots can be captured and included as graphics, and the tool features an array of drawing tools, clip art libraries, etc. The value add for this and similar tools is purely the size and convenience of the handheld device, not the ability to manage large amounts of information.

Perhaps the most effective use of a handheld architecture would be as a satellite data capture and retrieval utility. A user would normally employ a fully-functional desktop application for personal knowledge management, but when "on the go," they could capture knowledge into a compatible handheld application and upload it to their PKB at a later convenient time. To enable mobile knowledge retrieval, either select information would need to be downloaded to the device before the user needed it, or else a wireless client-server solution could deliver any part of the PKB on demand. This is essentially the approach taken by software like KeySuite[Chapura 2005], which merely supplements a feature-rich desktop information management tool (Microsoft Outlook) by providing access to that information on the mobile device. InfoSelect[MicroLogic 2005], a tree-based note-taking application, also offers a mobile product. We are not aware of handheld companion software for any bona fide knowledge representation tools, however. Idea Pad[Nosleep 2005], a drawing program for the Palm OS platform, is the closest, supporting simple mind maps and concept maps and capable of exporting the results to any desktop graphics application. The drawings remain just drawings, however, with no ability to integrate or connect them to form a full-fledged knowledge

base. As desktop PKB solutions become more viable and widely accepted, we would expect satellite handheld software to emerge to support them.

VII. Supplementary features

Before we conclude with an overall analysis of the systems in this survey, we wish to identify several key features that some of them have implemented, that may prove especially beneficial in the realization of a true PKB. Some are fundamental, others merely peripheral to the tool's basic operation.

1. **Analysis tools**. Knowledge bases that humans define can become quite large over time, of course, and some systems provide automated means to analyze them for general patterns. We have already mentioned the spatial parser of VIKI/VKB([Marshall and Shipman 1995], [Shipman, *et al.* 2000]) which can analyze how a user has visually arranged and adorned elements and draw conclusions about their implicit structure. In a somewhat different vein, Decision Explorer[Banxia 2005] provides "analysis functions" that show trends in the overall knowledge graph. Clusters, cycles, and highly influential concepts can be discovered and brought to the user's attention. Multicentrix[Koy 1997] and Knowledge Manager[Hypersoft 2005] can each compute the paths that connect any two elements in a knowledge network, however distant. This helps a user see how two concepts are related. SPRINT[Carlson and Ram 1990] had an active inference engine built in to assist users in drawing logical conclusions from the knowledge assertions they have expressed.

2. **Auto-classification**. To assist the user in organizing their data, some systems examine incoming information and either suggest or automatically perform a categorization for it. Agenda[Kaplan, *et al.* 1990] and DEVONThink[DEVON 2005] demonstrate two alternate ways of doing this. In Agenda, the user specifies an explicit set of rules for evaluating items – keying on whether certain text strings are present in the content, for instance. The system then executes these rules whenever items are changed or introduced into the database, and then automatically assigns them to the appropriate categories. DEVONThink takes an artificial intelligence approach, comparing the contents of new information units with the items in the folders the user has already established, in order to auto-file similar items together. WebTop[Wolber, *et al.* 2002] and CYCLADES[Renda and Straccia 2005] discover similarity relationships in much the same way. Haystack[Adar, *et al.* 1999] and Presto[Dourish, *et al.* 1999] both feature background services that examine content and auto-annotate it with metadata for future retrieval.

3. **Auto-suggest**. PKBs are intended to capture human knowledge, and interestingly, some systems actually attempt to *suggest* knowledge for the user to consider. The CmapTools program includes a "concept suggester" module that takes a concept map in the process of being designed and searches the Web for concepts that may be relevant to it.[Canas, *et al.* 2005] This is designed to assist the user in brainstorming and to help them flesh out an "incomplete" map. Similarly, NovaMind[NovaMind 2005], a mind mapping tool, includes a "branch proposal system," which suggests words or phrases that

relate conceptually or linguistically to the selected node. These enhancements are primarily suited to the knowledge generation process, of course, and do not pertain to long-term storage and retrieval.

KMap[Gaines and Shaw 1995] took a considerably more ambitious approach by integrating with tools that auto-generate knowledge from text sources. Programs that perform text analysis (by determining the linguistic relationship between phrases, for example, or by examining the co-occurrence of words in sentences) can give a preliminary attempt at a graphical knowledge representation of the text. KMap can import these results to generate concept maps that can be perused and refined by the average user. These sorts of techniques bring us to the boundary of the definition of personal knowledge, since they indeed produce a knowledge representation (rather than raw, unprocessed information) yet they were not generated by the user's own understanding. For our purposes, such efforts are best seen as methods to expedite the knowledge generation process, which must then be examined and approved by the user before admission into the PKB.

4. **Auto source capture**. We have pointed out that an individual's subjective knowledge is in large part comprised of elements from the objective realm, and mentioned several tools that tap into this phenomenon by allowing bits of objective sources to be easily subsumed into the knowledge base. An extension of this is the ability to automatically preserve the *source* location of the objective information. Some tools, for instance, will allow a user to highlight a snippet of a Web page in a browser and then drag that information into the tool, automatically capturing the source URL so that the original Web page can be easily referred to later. This is of considerable value, since it lets users concentrate on reading and assimilating information, without having to bother explicitly copying every source address in case it is needed later on. OneNote[Microsoft 2003] and YellowPen[YellowPen 2005] achieve this with the Internet Explorer browser. DEVONThink[DEVON 2005] provides its own browser as part of its integrated environment, and so has access to the URL of every page viewed and stored. StickyBrain[Chronos 2005] boasts a similar feature. And completely browser-based tools offer this sort of feature by definition, of course, since all they *do* is archive and organize Web content (e.g., [Schraefel*, et al.* 2002])

5. **Actions**. Some tools extrapolate from the idea of a passive knowledge base and allow executable actions to be attached to knowledge elements. This affords the user customization and automation of the knowledge management process. Boxer[diSessa and Abelson 1986] was an extreme example of this, since it *was* a bona fide programming environment: the structure a user created was composed largely of executable programming elements. KMap allowed the user to customize the behavior of the interface on a per-concept-map basis by attaching AppleScript macros to the maps. As Gaines and Shaw describe, "each concept map can have its own script which receives messages triggered by user interaction with the concept map. This enables KMap to be integrated with other applications, and user interaction with graphical structures in the visual language to be used to control any activity supported on the host computer or network." MindPad[AKS-Labs 2005], Axon[Bok 2005], and Omnigraffle[Omni 2005]

are tools that allow various kinds of scripts to be attached to knowledge elements, and automatically executed in response to user navigation. This would allow a user to track how often items have been viewed or changed in the knowledge base, integrate with related information from external applications, etc. These features typically require significant expertise on the user's part to take advantage of them, however.

6. **Search services**. As mentioned earlier, several tools effectively integrate a search environment with a personal information management workspace. (e.g., [Cousins, *et al.* 1997], [Hendry and Harper 1997], [Buchanan, *et al.* 2004], [Furnas and Rauch 1998].) This helps smooth the divide between the objective and subjective realms by giving easy access to the one from the other. Applications that integrate browsers into a knowledge storage paradigm (e.g., DEVONThink[DEVON 2005]) achieve a similar result.

7. **Collaboration support**. Though outside the sphere of personal knowledge management, many tools facilitate *sharing* portions of a knowledge base with others, and/or integrating knowledge with a public repository. Cmap[Canas, *et al.* 2005], KMap, and even the original Augment/NLS[Engelbart 1963] had this goal in view. Clearly this is a desirable addition to a PKB, since ultimately all public information begins as private knowledge, before it is identified as of general interest and is published to an accessible location. The prospect of easily integrating, sharing, and selectively publishing subjective knowledge is an exciting one, and is really the next logical step in the process. We would expect it to become much more widespread after personal knowledge bases become the norm.

8. **Three-dimensional rendering**. All of the spatial tools we have mentioned thus far have been based on a flat canvas on which the user can arrange knowledge elements. To try and improve visualization of large knowledge bases, however, tools such as Axon[Bok 2005] and HeadCase[Bignell 2005] provide three-dimensional views of a knowledge network. Taking advantage of the extra dimension permits more freedom in the laying out of nodes and connections. This has the potential of increasing comprehensibility, but comes at the expense of more awkward (or at least less familiar) forms of navigation. Placing items in a three-dimensional space is not a common task with today's popular computer applications, and until such a paradigm becomes widespread this feature may be of limited value.

9. **XML export**. Several tools offer the ability to save or export knowledge representations in XML format, in an attempt to facilitate interchange between PKB applications (e.g., [Hypersoft 2005], [FreeMind 2005], [Shipman, *et al.* 2000], [Bernstein 2003].) Standards are needed here, of course, in order to ensure proper migration, and this is a great challenge because of the great variations in PKB data models. Currently, one's only option is to handcraft XSL transformation templates to convert the format of one tool's output to another, which needless to say is quite a daunting proposition. For now, we simply hope that as PKBs become embraced by the public, industry consortia would arise to agree upon standards for knowledge interchange between systems.

10. **Memorization aid**. Though their basic purpose is to store knowledge electronically so it can be later retrieved, a few tools also emphasize the ability to strengthen *biological* memory. The unspoken premise here is that for at least some areas of knowledge, users need immediate recall from their own minds, rather than simply easy access to archived records. Mental Link[Dede and Jayaram 1990] was designed to support this: its stated purpose was to use knowledge models as a communication vehicle from educators to teachers. RecallPlus[Evolution 2005] allows the same sort of knowledge expression as many of the systems in this survey, but it is advertised as a study tool. Students input the knowledge they wish to retain for an exam, for instance, and the tool iteratively quizzes them, repeating material at optimal intervals and focusing on problem areas. KnowledgeManager's "assessment questions" feature is in a similar spirit.[Hypersoft 2005]

11. **Semantic Web support**. The Semantic Web initiative is a collaborative effort sponsored by the World Wide Web Consortium[W3C 2004] to extend today's Web by adding machine processible information.[Berners-Lee*, et al.* 2001] As it stands today, the Web is an enterprise almost strictly for human viewers: the emphasis is on free text content and decorative markup, the meaning of which is virtually impenetrable to agent software. The Semantic Web proposes to incorporate technologies such as the Resource Description Framework (RDF)[Lassila and Swick 1999] for annotating pages with formal expressions of their content. The authors, topics, and institutions involved, the details of the services advertised, and even the assertions a page makes will be coded in such a way that automated programs can reason about them and draw deductive conclusions. Though primarily intended to enable machine processibility, the Semantic Web also promises global consistency of terminology and unambiguous identifiers for shared concepts. This latter capability is what makes Semantic Web technologies an enticing component to consider for a PKB system.

The RDF data model is essentially a graph, in which the vertices represent real-world concepts denoted by Uniform Resource Indicators (URIs)[Berners-Lee*, et al.* 1998]. Each URI is a globally unique identifier, potentially shared by an entire community of users to refer to the same concept. The idea is that a distributed community will jointly agree on a formal description of their domain – including the kinds of relevant entities and how they relate to each other – called an ontology. This standard terminology is then used by all members of the community to describe the information they work with in a consistent manner. Simple examples include the FOAF project[FOAF 2005] which defines a standard schema for describing relationships between people, and the Dublin Core Metadata Initiative[DublinCore 2005] for describing properties of electronic resources.

Since RDF is used to describe abstract knowledge in a graph model, and brings with it standardization, deduction engines, and the promise of numerous community-authored ontologies stating useful facts, the prospect of integrating it into a PKB becomes attractive. It could be useful, for instance, to associate knowledge elements in a PKB with public URIs, and to link them together with relationships that conform to those defined in a standard ontology. This would allow the plethora of "common sense" facts

that ontologies embody to be made immediately available to the tool, and would better facilitate the sharing of PKBs.

The Semantic Web is still being defined, so existing applications are quite rare. One PKB system to delve into this area is the Mind Raider open source project[Dvorak 2005]. Mind Raider is a tree-based outliner, but in addition to basic outliner functionality (and a graphical view that portrays the outline as a mind map), URIs can be assigned to nodes in the tree. Nodes can then be annotated with semantic metadata according to standard ontologies. For instance, nodes in a mind map that represent people can be annotated with FOAF information. The tool automates the process of importing the ontology and defining compatible data. Idea Graph[Ayers 2005] adds similar Semantic Web support to concept maps.

Compendium has also been used for a similar purpose[Selvin 1999], though strictly for an ontology pertaining to organizational meetings and discussions. (It is not intended for general purpose RDF.) Recent extensions to the original Haystack project[Huynh*, et al.* 2002] have incorporated Semantic Web technologies for personal document management, but not abstract knowledge expression. Documents, messages, and other information units can be annotated according to a personal ontology, and custom metadata added.

Since the Semantic Web itself is still in its infancy, this area has not yet been widely explored. But we anticipate that if accepted on a global scale, the potential it would bring for integrating private and public knowledge would be a tremendous asset to PKBs.

VII. Putting it all together: an ideal Personal Knowledge Base

We conclude this survey by briefly looking back at the systems we have studied and offering our own vision for an "ideal" PKB solution. The system we describe does not yet exist. It is best seen as an amalgamation of some of the best and brightest aspects of what we have covered in this survey, plus one or two original notions. In the spirit of Bush's original "As We May Think" article, we simply describe the system as though it already existed. Our position is different than Bush's, however, in that the supporting technology required for such a system *is* available today. We no longer need to be content to dream: the kind of personal knowledge base we describe could be constructed tomorrow by taking and learning from what has gone before.

We begin by noting some brief design goals, then present the design choices that we feel would best achieve those goals.

A. Design goals

As we have mentioned, many of the systems in this survey were not specifically designed to be PKBs at all, but to enable some other related task. Some were designed to manage information, not knowledge; some to serve communities, not individuals; some to draw

knowledge diagrams, but not to maintain them for the long term. Hence their problem space and corresponding design tradeoffs are somewhat different from ours, and while some aspect of a system may form a key contribution to a PKB solution, the system overall would not serve well as one.

Here, then, are some of the design goals of our ideal PKB, dubbed the "Memex II" for simplicity:

- *Works "like the brain thinks."* The Memex II preserves memories; therefore, it represents them faithfully and in all ways attempts to operate "as the brain works." For us, this means principally three things:

    1. The knowledge base consists largely of semantic knowledge representations, not English sentences. Psychologists have established that what we normally remember is the "gist" of the information we process, not the syntactic form in which it was originally expressed. ([Anderson 1990], [Kintsch 1970], [Sowa 1984].) The Memex II, therefore, captures knowledge in this processed form so it can be later retrieved, integrated, and repurposed in a variety of ways.
    2. As the human brain can associate any concepts together, without constraints or boundaries, so any knowledge elements in the Memex II can be linked together, with no restrictions.
    3. Just as the mind often considers the same concept in many different contexts, so the Memex II supports multiple views in which the same element can appear in different surroundings and with different sets of neighbors.

- *Contains* all *types of knowledge*. The Memex II is a general-purpose knowledge representation tool, not custom-tailored to a particular domain. It does not appeal to one aspect of a user's life more than others (e.g., to business functions rather than personal) and does not encourage (or even allow) the user to partition their knowledge into different constrained realms.

- *Supports informality*. The ideas a user wishes to preserve are often embryonic, and the tool allows them to express these as such. Users are not forced to commit to structure prematurely. In the Memex II, therefore, all knowledge elements can be typeless. Connections can be made between them that have no specific semantics attached. Elements can be grouped or clustered together in a less formal manner than drawing explicit relationships. The user is thus encouraged to import their knowledge freely into the tool in whatever state it exists, even if it has not yet been precisely formulated.

- *Supports formality*. As the user acquires more firm notions about a domain, and becomes more certain and more detailed in their perceptions, the tools allows them to express these ideas formally. Users can assign types to concepts, create hierarchies of types, add coded attributes to concepts, label links with semantics, etc. In all cases, the tool supports a graceful continuum of precision. Searches and any automated

processing functions can take advantage of formal specifications when they are present, but still cope with imprecise data at a more basic level.

- *Reversible and flexible refactoring*. Users often acquire knowledge at a breathtaking pace, but only later discover how to organize it. And as they learn more, they frequently revise previous categorizations and structures in favor of new understandings. The Memex II permits the quick assimilation of knowledge and information bits as a user encounters them, without regard to how they should be systematized. Users can return to these elements later to rearrange and restructure them. And any decisions they make about how to organize their information can be easily reversed, which inspires a willingness to experiment.

- *Streamlined user interface*. In contrast with many other tools, the purpose of the Memex II is not to draw pretty diagrams, but to record knowledge. Its user interface, therefore, is optimized for this purpose. The operations for creating new knowledge elements, connecting them together, viewing them in different contexts, etc. are streamlined, free from the clutter of most drawing tools' numerous options. The user specifically does *not* have the flexibility of adding lots of color, icons, or other fancy formatting to the knowledge representation: such functions would actually mislead the user into thinking that the purpose of the tool was something other than what it was.

- *Ubiquitous availability*. Users absorb new information and draw upon existing knowledge in all facets of their lives, often in remote settings. To as great an extent as possible, then, the Memex II, like our own brains, is available everywhere. A user can tap into their personal knowledge base from any network-enabled computer, both accessing previously recorded thoughts and adding newly discovered ones while they are fresh in the user's mind. Where network connection is not possible, a mobile device can serve as a substitute, letting the user download a snapshot of certain knowledge areas before going on the road, and uploading any newly stored knowledge when the user again returns.

- *Bridges the objective-subjective divide*. In addition to storing knowledge that the user manually enters, the tool makes it easy to drag snippets and quotes from electronic sources into the knowledge base. This lets the user repurpose and reorganize what they have read, and integrate it with their background knowledge. Links back to the objective sources are automatically captured, so that the user does not need to bother worrying about returning to the source later.

- *Immediate retrieval*. Returning to previously stored knowledge is normally effortless, since people tend to strongly associate certain words and names with concepts. In cases where the user has a well-named entity, then, not even a navigation or query operation is necessary to retrieve it: the user can simply type a word or phrase, and any concepts in the PKB that match will instantly appear in a pick list. Items that are not well-named (or whose names the user forgets) are likely to be associated to items that *are* well-named, either directly or indirectly. In these cases, the user will be able

to first recall the well-named entity, and then navigate from there to the desired knowledge via the links they previously established. When both of these approaches fail, the Memex II also supports a more traditional query facility. Here, the user can specify multiple criteria, using a combination of free text and any coded attributes they have defined, in an attempt to narrow down the knowledge elements in the system to the one they wish to retrieve.

B. Design choices

And finally, in terms of the issues considered in this survey, we feel that an ideal PKB should have the following characteristics:

1. Data model – a spatial, transclusive graph of knowledge elements

If we really want to capture knowledge "in the way the mind thinks of it," then we want it to be expressed in its processed form, not in the English sentences through which it would normally be discovered and communicated. Hence, the Memex II rejects the natural language approach of most note-taking tools in favor of a graphical knowledge representation. This allows the key, distilled components of a user's knowledge to be easily expressed, uncluttered by the superficial adornments that prose expression inevitably involves. It also makes it easier for the user to integrate new knowledge with old, since all key concepts are reified as objects with formal identities that can be referred to in multiple contexts. This is much more difficult to achieve if the main concepts that a user wishes to relate together are buried within paragraphs of text as a succession of mostly equivalent noun phrases.

Among graphical knowledge representations, we choose a graph rather than a tree model, simply because it is more accommodating. As previously discussed, the tree, though easy to understand and therefore attractive, is far too restrictive a model to represent the vast array of knowledge that humans possess. People interrelate the concepts that they perceive in countless, unpredictable ways, and the strict parent-child hierarchy is much too simplistic to support this. The data model for the Memex II, then, is a semantic network, in which any concept can be related to any other. The associations between concepts can be, but need not be, labeled to denote their semantics.

Concept mapping tools have demonstrated that the graph and spatial models can peacefully coexist, and in fact complement one another. Grouping and arranging items coarsely on the screen allows the user to gradually introduce structure and to express relationships less formally; it also takes advantage of the inherent spatial paradigm in which we mentally conceive of information.[Marshall and Shipman 1995] Explicit links between concepts can then be added to a spatial workspace when the user has processed the knowledge completely enough to identify the semantic relationships.

Full transclusion is supported by the Memex II, in which a concept in the system can be present in any number of "views" that show its relationships to selected other concepts. To reduce the mental load associated with trying to view a gigantic semantic network in

its entirety, the user's primary interface to the knowledge base will be through these views. Each one represents a single, modest-sized snapshot of the knowledge base's contents, depicting a handful of concepts and their interrelationships. This correlates well with the user's short-term memory, since humans can only simultaneously consider about seven elements at once anyway.[Miller 1956] The user can navigate from view to view through the concepts that are shared between them, much as our train of thought meanders from topic to topic through the associations we have formed. In this way the cohesiveness of the entire knowledge base can be explored even though only small portions are visible at once, just as our own memories form an integrated mass even though we only consider a very small subset of knowledge at a time.

The Memex II does *not* require the graph to be fully connected, however. Unlike PersonalBrain[TheBrain 2005] and Compendium[Compendium 2005], whose data models are otherwise quite similar to ours, islands of information can freely exist in the knowledge base. The user can of course associate any two concepts they wish, but the system does not force all elements to be ultimately connected to a "root" or threaten deletion if this is violated. This is because our memories have no "root" element to which all else is connected: we often experience new perceptions and form islands of knowledge that are not yet connected with previous learning. Some may dispute this, and argue that all new knowledge must be related to a user's existing knowledge in order to be properly understood ([Wittrock 1974], [Ausubel 1968]); even if this is the case, however, we do not feel it is the software's job to enforce this.

The notion of "views" through which arbitrary subsets of knowledge can be seen and manipulated also serves well as a category function. A user can group several items together on a view, and without even forming any relationships between them, these items can be seen as implicitly in a category. As with Agenda[Kaplan*, et al.* 1990] and other category-based tools, an item's presence in one category/view does not prevent its simultaneous membership in another, because of the transclusion principle. In some cases, however, a category may grow so large that it is infeasible to position all of its members within a single view. For this reason, the Memex II also supports a keyword tagging function whereby concepts can be labeled with user-defined keywords apart from their presence on any particular view.

Like Agenda and Circus Ponies[CircusPonies 2005], the Memex II also supports a chronological mode that can be used to locate elements by their creation or modification dates. This complements other forms of access and takes advantage of the commonly temporal nature of information organization cited by Lifestreams[Fertig*, et al.* 1996].

As mentioned above, the Memex II's knowledge elements are primitive concepts, not documents. Hence the tool does not encourage large amounts of free text to be included. There are at least two situations, however, when the user would find such a capability convenient: (1) when information has to be stored quickly, and time does not permit its breakdown into conceptual form until a later point, and (2) when the information *itself* is what is of value. An example of the second scenario is a quoted passage of text: in this case, the knowledge acquired actually *is* a verbatim passage that must later be retrieved in

its entirety, not merely the concepts it contains. For this reason, the tool allows snippets of free text (and possibly graphical and other elements) to be subsumed into the knowledge base. A drag and drop or copy/paste mechanism accomplishes this, in which the filename or URL of the source document is automatically included so the user can return to the context of the passage on demand.

The Memex II anticipates Semantic Web technologies. It can process RDF ontologies and hence allows knowledge elements to be declared as of particular types and have particular coded attributes. This promotes uniformity and ease of integration in collaborative environments, and allows a casual user to take advantage of the painstaking efforts that experts in a community have made to establish "correct" domain models. Users can create their own local ontologies too, if they wish, and even publish them for public use directly through the tool. Finally, and perhaps most importantly, knowledge elements can be tagged with globally accepted URIs that guarantee that everyone who uses them will be referring to the same conceptual entity. When the URIs of the concepts that a Web page deals with are someday manifested in the browser, users will be able to drag icons representing semantic concepts into their PKB and implicitly bring all objective assertions about those entities into the tool.

2. User interface – streamlined knowledge manipulation

The Memex II provides very modest controls on the appearance of views and the knowledge elements they contain. It does not purport to be a drawing tool, and in fact discourages the user from superfluous adornments. The only things the user can control are qualities vital to the representation of the knowledge itself. This includes spatial positioning, relationship arrows between elements, and perhaps color. The fact that so few operations are possible makes the user interface very straightforward and minimizes the number of mouse clicks and keystrokes necessary to express the knowledge in a user's mind. This in turn encourages the user to enter a great deal of knowledge into the tool, since the perceived cost of doing so will be so low.

3. Architecture – Integrated database with multiple clients

Not surprisingly, the Memex II employs a full database storage facility, which can scale to a large number of elements interconnected in an unconstrained manner. Nothing is stored in files, which frees the user both from having to commit to a premature set of information "boundaries," and from remembering the "location" of anything. There are no "locations" in the Memex II: there are just lots and lots of concepts, which can be retrieved by their names, by any free text information they contain, or via any concepts they are associated with. This mimics the associative aspects of biological memory and its property of "spreading activation." ([Anderson 1990], [Kintsch 1970].)

Clients that access this database, however, are distributed. In today's world of microelectronics and wireless communication, we can do much better than Bush's original dream of a physical "desk" to house an individual's knowledge. Instead, a user's PKB will be available anywhere that Internet access is present. From their home or work

desktop machine, portable laptop, or colleague's computer, users can access their personal knowledge store over an HTTP protocol. When on the road, a palmtop computer equipped with wireless service gives access anywhere. And before road trips to remote areas, a handheld device can be synchronized with the PKB server so that important knowledge is available to be perused and gathered even offline.

4. Key features

As for the key features we identified in the previous section, we are mostly agnostic. The Memex II will accommodate whichever features users find essential and the market demands. As PKBs become practical and widely used, their key capabilities will certainly converge and new possibilities will be unveiled. The only features from our list that we feel certain about at this point are:

- *Auto-source capture*. As mentioned above, users need to be able to repurpose snippets of knowledge from the objective realm, and preserve the source of those snippets automatically (as with, e.g., YellowPen[YellowPen 2005].)
- *Integration with search services*. Similarly, since the boundary between the subjective and objective realms needs to be smooth and traversable, it seems worthwhile for the Memex II to launch Internet searches directly within it, perhaps even using local subjective knowledge to focus the search automatically. (This gives us integrated search features similar to those in DLITE[Cousins, *et al.* 1997].)
- *Collaboration support*. PKB developers need to keep this feature in mind from the outset, so as to design systems that can ultimately share, compare, and publish knowledge (as does KMap[Gaines and Shaw 1995].)
- *XML export*. When standards emerge, PKB systems must be able to import and export information in a common format, so that users are not trapped with a particular vendor for the long-term. Due to its popularity and simplicity, we favor an XML syntax (used by, e.g., Knowledge Manager[Hypersoft 2005].)
- *Semantic Web support*. The possibilities here are endless, and promise collaborative abilities and machine processibility that may signal a new frontier in knowledge management. The PKB is possibly the most important tool to play in this domain, and must anticipate this new dimension to today's information infrastructure (as do MindRaider[Dvorak 2005] and IdeaGraph[Ayers 2005].)

VIII. Conclusion

As can be seen, the system we prescribe is largely composed of elements of various applications that already exist. Yet no system that we know of implements this vision entirely. Mind mapping tools, for instance, promote structure-based rather than sentence-based representation, but their data model (a tree) is too limiting to support the breadth of human thought. PersonalBrain supports a full graph model, but not transclusion, and thus every knowledge element can be seen only in a single context. Tinderbox's transclusion overcomes this, but with a file-based architecture the limitations of which we have

already explored. Compendium features a transclusive graph-based database architecture, but it requires a fully-connected graph, is hardwired for a particular domain (decision rationale expressed in group meetings), and offers no facility at all for bridging the subjective-objective divide (even simple, hardcoded references to objective sources are not supported, let alone automatic drag and drop source capture.) And none of these solutions offer the sort of comprehensively distributed architecture that would give users the ubiquitous availability they need.

But by following the guidelines and principles of successful systems, a simple but powerful tool could be created that we believe would rival all previous "killer" software applications. Of course, there is always a tradeoff between expressiveness and complexity, and certainly, various users would find their "perfect" system in different places of the tradeoff spectrum. In particular, many people who do not like having to learn to use computing systems might strongly prefer a less powerful, but simpler system.

Through disciplined use of a personal knowledge base, a user can essentially overcome the limitations of their own memory. We have all known people who were like "walking encyclopedias," clinging effortlessly to every fact they ever encountered. The Memex II promises to raise *every* individual to that level. It helps finally realize the potential of the information age: not only can the knowledge of the world be accessed, but it can be organized, personalized, and integrated, and later retrieved and applied. The user's personal effectiveness is enhanced to the point where they can actually make proper *use* of the information that has been at their fingertips for over a decade.

The contribution of this article is not to announce PKBs as a new idea. Indeed, the notion of a "Memex" is at least sixty years old, which is a testimony to its allure. Our purpose is rather to focus attention on the key characteristics of the problem and the potential solutions, and to inspire innovation in this area that will result in real, viable personal knowledge base systems that are easy to use for everyone. We believe that such applications could radically improve the way people deal with computers, information, and their entire world.

References

1. Adar, E., Karger, D. and Stein, L.A., Haystack: per-user information environments. in *Proceedings of the Eighth International Conference on Information Knowledge Management*, (Kansas City, Missouri, 1999), 413-422.
2. Akscyn, R., McCracken, D. and Yoder, E., KMS: a distributed hypermedia system for managing knowledge in organizations. in *Proceedings of the ACM Conference on Hypertext*, (Chapel Hill, North Carolina, 1987).
3. AKS-Labs. Mind Pad v1.1. Available at: www.mind-pad.com. 2005
4. Advanced Learning Technologies in Education Consortia (ALTEC). NoteStar. Available at: http://notestar.4teachers.org. 2005
5. Anderson, J.R. *Cognitive Psychology and Its Implications, 3rd Ed.* W.H. Freeman, New York, 1990.

6. AquaMinds Software Corporation. NoteTaker 1.9. Available at: www.aquaminds.com. 2005
7. Ausubel, D.P. *Educational Psychology: A Cognitive View*. Holt, Rinehart, and Winston, Inc., New York, 1968.
8. Ayers, Danny. IdeaGraph. Available at: www.ideagraph.net. 2005
9. Banxia Software Ltd. Decision Explorer. Available at: http://www.banxia.com/demain.html. 2005
10. Network Working Group: Request for Comment 2396. Uniform Resource Identifiers (URI): Generic Syntax. Available at: http://www.ietf.org/rfc/rfc2396.txt. 1998
11. Berners-Lee, T., Hendler, J. and Lassila, O. The Semantic Web. *Scientific American*, 2001.
12. Bernstein, M., Collages, Composites, Construction. in *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, (Nottingham, UK, 2003).
13. Bignell, Elliott. HeadCASE: Mind Mapping for Windows. Available at: www.bignell.de. 2005
14. Bitsmith Software. Personal Knowbase. Available at: http://www.bitsmithsoft.com/product.htm. 2005
15. Blandford, A.E. and Green, T.R.G. Group and individual time management tools: what you get is not what you need. *Personal and Ubiquitous Computing*, 5, 4.(December 2001), 213-230.
16. Axon Research. Axon Idea Processor. Available at: web.singnet.com.sg/~axon2000. 2005
17. The Bosley Group. MindMapper Professional v4.2. Available at: www.mindmapperusa.com. 2005
18. Buchanan, G., Blandford, A.E., Thimbleby, H. and Jones, M., Integrating information seeking and structuring: exploring the role of spatial hypertext in a digital library. in *Proceedings of the Fifteenth ACM Conference on Hypertext and Hypermedia*, (Santa Cruz, California, 2004), 225-234.
19. Burger, A.M., Meyer, B.D., Jung, C.P. and Long, K.B., The virtual notebook system. in *Proceedings of the Third Annual ACM Conference on Hypertext*, (San Antonio, Texas, 1991), 395-401.
20. Bush, V. As we may think. *The Atlantic Monthly*, 1945, 101-108.
21. Buzan, T. and Buzan, B. *The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential*. Plume Books, 1996.
22. Canas, A.J., Hill, G., Carff, R., Suri, N., Lott, J., Gomez, G., Eskridge, T.C., Arroyo, M. and Carvajal, R., CmapTools: a knowledge modeling and sharing environment. in *Proceedings of the First International Conference on Concept Mapping*, (Pamplona, Spain, 2005), 125-133.
23. Carlson, D.A. and Ram, S. HyperIntelligence: the next frontier. *Communications of the ACM*, 33, 3 311-321.
24. Chapura, Inc. KeySuite v3.3.2. Available at: http://www.chapura.com/keysuite.php. 2005
25. Chronos. StickyBrain 3. Available at: www.chronosnet.com. 2005
26. Circus Ponies. NoteBook 2.0. Available at: www.circusponies.com. 2005

27. Claro Software. MindFull V2. Available at: http://clarosoftware.com. 2005
28. CoCo Systems Ltd. VisiMap Professional 4.0. Available at: www.visimap.com. 2005
29. Cognitive-Tools. Easy-Mapping-Tool. Available at: www.cognitive-tools.com. 2005
30. Collins, A.M. and Quillian, M.R. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Memory*, 8 240-247.
31. The Compendium Institute. Compendium Version 1.3.04. Available at: www.compendiuminstitute.org. 2005
32. Conklin, J. Hypertext: an introduction and survey. *Computer*, 20, 9.(Sept 1987), 17-41.
33. Conklin, J. and Begeman, M.L., gIBIS: a hypertext tool for exploratory policy discussion. in *Proceedings of the 1988 ACM Conference on Computer-supported Cooperative Work*, (Portland, Oregon, 1988), 140-152.
34. Conklin, J., Selvin, A.M., Shum, S.B. and Sierhuis, M., Facilitated hypertext for collective sensemaking: 15 years on from gIBIS. in *Proceedings of the Twelfth ACM Conference on Hypertext and Hypermedia*, (2001).
35. Conway, M.A., Kahney, H., Bruce, K. and Duce, H. Imaging objects, routines, and locations. in Logie, R.H. and Denis, M. eds. *Mental Images in Human Cognition*, Elsevier Science Publishing Company, Inc., New York, 1991, 171-182.
36. Cousins, S.B., Paepcke, A., Winograd, T., Bier, E.A. and Pier, K., The digital library integrated task environment (DLITE). in *Proceedings of the Second ACM International Conference on Digital Libraries*, (Philadelphia, Pennsylvania, 1997), 142-151.
37. Davis, H., Hall, W., Heath, I., Hill, G. and Wilkins, R., MICROCOSM: an open hypermedia environment for information integration. in *Proceedings of the INTERCHI Conference on Human Factors in Computing Systems*, (1993), ACM Press.
38. Dede, C.J. and Jayaram, G. Designing a training tool for imaging mental models. Air Force Human Resources Laboratory, Brooks Air Force Base, Texas, 1990.
39. Delisle, N. and Schwartz, M., Neptune: a hypertext system for CAD applications. in *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data*, (Washington, D.C., 1986), 132-143.
40. DEVONtechnologies. DEVONthink Personal Edition. Available at: http://www.devon-technologies.com/. 2005
41. Di Giacomo, M., Mahoney, D., Bollen, J., Monroy-Hernandez, A. and Meraz, C.M.R. MyLibrary, a personalization service for digital library environments.
42. diSessa, A.A. and Abelson, H. Boxer: a reconstructible computational medium. *Communications of the ACM*, 29, 9 859-868.
43. Dourish, P., Edwards, W.K., LaMarca, A. and Salisbury, M. Presto: an experimental architecture for fluid interactive document spaces. *ACM Transactions on Computer-Human Interaction*, 6, 2 133-161.
44. The Dublin Core Metadata Initiative. Available at: www.dublincore.org. 2005
45. Dumais, S.T., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R. and Robbins, D.C., Stuff I've Seen: a system for personal information retrieval and re-use. in *Proceedings*

of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, (Toronto, Canada, 2003), 72-79.
46. Dvorak, Martin. MindRaider. Available at: mindraider.sourceforge.net. 2005
47. Electric Pocket, Inc. BugMe! Notepad. Available at: http://www.electricpocket.com/bugme-palm/. 2005
48. Enfish Software, Campbell, California. Enfish Find. Available at: http://www.enfish.com/Products_Find.asp. 2005
49. Engelbart, D.C. A conceptual framework for the augmentation of man's intellect. in Howerton, P.W. ed. *Vistas in Information Handling*, Spartan Books, Washington, D.C., 1963, 1-29.
50. EvolutionCode Pty Ltd. RecallPlus V3. Available at: www.recallplus.com. 2005
51. Feiner, S., Seeing the forest for the trees: hierarchical display of hypertext structure. in *Proceedings of the Conference on Office Information Systems*, (Palo Alto, California, 1988), ACM Press, 205-212.
52. Fertig, S., Freeman, E. and Gelernter, D., Lifestreams: An alternative to the desktop metaphor. in *Proceedings of the Conference on Human Factors in Computing Systems (CHI96)*, (Vancouver, British Columbia, 1996), 410-411.
53. The Friend of a Friend (FOAF) Project. Available at: www.foaf-project.org. 2005
54. FranklinCovey. PlanPlus 4.0 for Windows XP. Available at: www.franklincovey.com. 2005
55. Freebyte. TreePad Business Edition 7.1.7. Available at: www.treepad.com. 2005
56. Freeman, E. and Gelernter, D. Lifestreams: a storage model for personal data. *ACM SIGMOD Record*, 25, 1.(March 1996), 80-86.
57. FreeMind. Available at: http://freemind.sourceforge.net. 2005
58. Furnas, G.W. and Rauch, S.J., Considerations for information environments and the NaviQue workspace. in *Proceedings of the ACM Conference on Digital Libraries*, (1998), ACM, 79-88.
59. Gael Ltd. MindGenius Home v2.22. Available at: www.mindgenius.com. 2005
60. Gaines, B.R. and Shaw, M.L.G. Concept maps as hypermedia components. *International Journal of Human Computer Studies*, 43, 3 323-361.
61. Garrett, L.N., Smith, K.E. and Meyrowitz, N., Intermedia: Issues, strategies, and tactics in the design of a hypermedia document system. in *Proceedings of the Conference on Computer-Supported Cooperative Work*, (1986), 163-174.
62. Gemmell, J., Bell, G., Lueder, R., Drucker, S. and Wong, C., MyLifebits: Fulfilling the Memex vision. in *Proceedings of the 2002 ACM Workshops on Multimedia*, (2002), 235-238.
63. Gentner, D. and Stevens, A.L. (eds.). *Mental Models*. Lawrence Erlbaum Associates, Inc., New Jersey, 1983.
64. Godwin-Jones, B. Blogs and wikis: environments for on-line collaboration. *Language Learning and Technology*, 7, 2.(May 2003), 12-16.
65. Goodman, D. *The Complete Hypercard Handbook*. Bantam Books, New York, 1988.
66. Halasz, F.G. Reflections on NoteCards: seven issues for the next generation of hypermedia systems. *Communications of the ACM*, 31, 7 836-852.
67. Halasz, F.G., Moran, T.P. and Trigg, R.H. NoteCards in a Nutshell. *ACM SIGCHI Bulletin*, 17 45-52.

68. Halasz, F.G. and Schwartz, M. The Dexter hypertext reference model. *Communications of the ACM*, 37, 2.(February 1994), 30-39.
69. Hatzopoulos, M., Gouscos, D., Spiliopoulou, M., Vassilakis, C. and Vazirgiannis, M., An object-oriented data model for hypermedia systems. in *Proceedings of the DELTA Conference in Research and Development*, (The Hague, 1990), 483-493.
70. Hayes, G., Pierce, J.S. and Abowd, G.D., Practices for capturing short important thoughts. in *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, (Ft. Lauderdale, Florida, 2003), 904-905.
71. Hendry, D.G. and Harper, D.J. An information information-seeking environment. *Journal of the American Society for Information Science*, 48, 11 1036-1048.
72. Hog Bay Software. Hog Bay Notebook v3.5. Available at: www.hogbaysoftware.com. 2005
73. Huynh, D., Karger, D. and Quan, D., Haystack: a platform for creating, organizing, and visualizing information using RDF. in *Proceedings of the 18th National Conference on Artificial Intelligence: Workshop on Ontologies and the Semantic Web*, (Alberta, Canada, 2002).
74. Hypersoft-net. Knowledge Manager v8.4. Available at: www.knowledgemanager.us. 2005
75. IBM Lotus Software. IBM Lotus Notes 6.5. Available at: http://www.lotus.com/products/product4.nsf/wdocs/noteshomepage. 2005
76. Inspiration Software, Inc. Inspiration 7.6. Available at: www.inspiration.com. 2005
77. Intelligents, LLC. NoteWorthy. Available at: http://www.intelligents.com/noteworthy.htm. 2001
78. Jones, W.P., The Memory Extender personal filing system. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, (Boston, Massachusetts, 1986), 298-305.
79. Kaplan, S.J., Kapor, M.D., Belove, E.J., Landsman, R.A. and Drake, T.R. Agenda: a personal information manager. *Communications of the ACM*, 33, 7 105-116.
80. Kintsch, W. *Learning, Memory, and Conceptual Processes*. John Wiley & Sons Inc., 1970.
81. Klas, W., Aberer, K. and Neuhold, E. Object-oriented modeling for hypermedia systems using the VODAK modeling language (VML). in Dogac, A., Ozsu, T., Biliris, A. and Sellis, T. eds. *Advances in Object-Oriented Database Systems*, Springer-Verlag, Berlin, 1993.
82. Kovalainen, M., Robinson, M. and Auramaki, E., Diaries at work. in *Proceedings of the 1998 ACM Conference on Computer Supported Collaborative Work*, (Seattle, Washington, 1998), 49-58.
83. Koy, A.K., Computer Aided Thinking. in *Proceedings of the 7th International Conference on Thinking*, (Singapore, 1997).
84. World Wide Web Consortium. Resource Description Framework (RDF) Model and Syntax Specification. Available at: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/. 1999
85. Lorayne, H. and Lucas, J. *The Memory Book*. Stein and Day, New York, 1974.

86. Mantei, M.M. Disorientation behavior in person-computer interaction. *Communications Department*, University of Southern California, 1982.
87. Marshall, C., Halasz, F.G., Rogers, R.A. and Janssen, W.C., Aquanet: a hypertext took to hold your knowledge in place. in *Proceedings of the Third Annual ACM Conference on Hypertext*, (San Antonio, Texas, 1991), 261-275.
88. Marshall, C. and Shipman, F. Spatial hypertext: designing for change. *Communications of the ACM*, 38, 8 88-97.
89. Micro Logic. Info Select for Windows, Version 8. Available at: http://www.miclog.com. 2005
90. Microsoft Corporation. Microsoft Outlook 2003. Available at: www.microsoft.com/outlook. 2003
91. Microsoft Corporation. OneNote 2003. Available at: http://www.microsoft.com/office/onenote/prodinfo/default.mspx. 2003
92. Miller, G.A. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63 81-97.
93. Mind Technologies. Visual Mind 7, Business Edition. Available at: www.visual-mind.com. 2005
94. Mindjet. MindManager X5 Pro. Available at: www.mindjet.com. 2005
95. Missiontrek. Cartagio Home. Available at: http://www.missiontrek.com/cartagio/chome.asp. 2005
96. Nakakoji, K., Yamamoto, Y., Takada, S. and Reeves, B.N., Two-dimensional spatial positioning as a means for reflection in design. in *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, (New York, 2000), 145-154.
97. Nelson, T.H. The heart of connection: hypermedia unified by transclusion. *Communications of the ACM*, 38, 8 31-33.
98. Nelson, T.H. *Literary machines : the report on, and of, Project Xanadu concerning word processing, electronic publishing, hypertext, thinkertoys, tomorrow's intellectual revolution, and certain other topics including knowledge, education and freedom*, 1987.
99. Nelson, T.H. Xanalogical structure, needed now more than ever: parallel documents, deep links to content, deep versioning, and deep re-use. *ACM Computing Surveys*, 31, 4.
100. NetManage, Inc. Ecco Pro. Available at: www.netmanage.com. 1997
101. Netscape Communications Corporation. The Open Directory Project. Available at: dmoz.org. 2004
102. Nosek, J.T. and Roth, I. A comparison of formal knowledge representation schemes as communication tools: predicate logic vs. semantic network. *International Journal of Human Computer Studies*, 33, 2 227-239.
103. Nosleep Software. Idea Pad 3.1. Available at: www.nosleep.net. 2005
104. Novak, J.D. The theory underlying concept maps and how to construct them. Institute for Human and Machine Cognition, University of West Florida, 2003.
105. NovaMind Software Pty Ltd. NovaMind v2.4.5. Available at: www.nova-mind.com. 2005
106. Novell, Inc. Novell Evolution 2. Available at: http://www.novell.com/products/desktop/features/evolution.html. 2005

107. Nyce, J.M. and Kahn, P. *From Memex to hypertext*. Academic Press, Boston, 1991.
108. Oberon, askSam Systems. Citation. Available at: http://www.citationonline.net/brochure.asp. 2005
109. Computer Systems Odessa. ConceptDraw MINDMAP v3.5. Available at: conceptdraw.com. 2005
110. Omni Development, Inc. OmniGraffle 3. Available at: http://www.omnigroup.com/applications/omnigraffle/. 2005
111. Omni Development, Inc. OmniOutliner 3. Available at: http://www.omnigroup.com/applications/omnioutliner/. 2005
112. Open Source Applications Foundation (OSAF). Chandler. Available at: www.osafoundation.org. 2005
113. Palen, L., Social, individual and technological issues for groupware calendar systems. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, (Pittsburgh, Pennsylvania, 1999), 17-24.
114. Pearl, A., Sun's Link Service: a protocol for open linking. in *Proceedings of the Second Annual ACM Conference on Hypertext*, (Pittsburgh, Pennsylvania, 1989), 137-146.
115. Perlin, K. and Fox, D., Pad: an alternative approach to the computer interface. in *Proceedings of the 20th annual conference on computer graphics and interactive techniques*, (1993), 57-64.
116. Quillian, M.R. Semantic memory. in *Semantic Information Processing*, The MIT Press, Cambridge, Massachusetts, 1968.
117. Ralston Technology Group. Clarity 2.0. Available at: www.ralstontech.com. 2005
118. Renda, M.E. and Straccia, U. A personalized collaborative digital library environment: a model and an application. *Information Processing and Management: an International Journal*, 41, 1 5-21.
119. Reyes-Farfan, N. and Sanchez, J.A., Personal spaces in the context of OAI. in *Proceedings of the Third ACM/IEEE-CS Joint Conference on Digital Libraries*, (2003), 182-183.
120. Schneiderman, B., User interface design for the Hyperties electronic encyclopedia. in *Proceedings of the ACM Conference on Hypertext*, (Chapel Hill, North Carolina, 1987), 189-194.
121. Schraefel, M.C., Zhu, Y., Modjeska, D., Widgdor, D. and Zhao, S., Hunter Gatherer: interaction support for the creation and management of within-Web-page collections. in *Proceedings of the Eleventh International Conference on the World Wide Web*, (2002), 172-181.
122. Selvin, A.M. Supporting collaborative analysis and design with hypertext functionality. *Journal of Digital Information*, 1, 4.
123. Sereno, B., Shum, S.B. and Motta, E., ClaimSpotter: an environment to support sensemaking with knowledge triples. in *Proceedings of the International Conference on Intelligent User Interfaces*, (San Diego, California, 2005).
124. Seyer, P. *Understanding Hypertext: Concepts and Applications*. Windcrest Books, Blue Ridge Summit, Pennsylvania, 1991.

125. Shipman, F., Hsieh, H. and Airhart, R. Analytic workspaces: supporting the emergence of interpretation in the Visual Knowledge Builder. Texas A&M University, 2000.
126. SMART Technologies, Inc. SMART Ideas Concept-mapping Software. Available at: www2.smarttech.com. 2005
127. Smith, J.B., Weiss, S.F. and Ferguson, G.J., A hypertext writing environment and its cognitive basis. in *Proceedings of the ACM Conference on Hypertext*, (Chapel Hill, North Carolina, 1987), 195-214.
128. Sowa, J.F. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley Publishing Company, 1984.
129. The StayAtPlay Co. Idea Knot. Available at: www.stayatplay.com. 2005
130. TheBrain Technologies Corporation. PersonalBrain v3.02. Available at: www.thebrain.com. 2005
131. Trigg, R.H. and Weiser, M. TEXTNET: a network-based approach to text handling. *ACM Transactions on Information Systems*, 4, 1 1-23.
132. TruTamil, LLC. ndxCards. Available at: ndxcards.com. 2005
133. Uren, V., Buckingham Shum, S., Li, G. and Bachler, M. Sensemaking tools for understanding research literatures: design, implementation, and user evaluation. Knowledge Media Institute, The Open University, 2004, 1-42.
134. The World Wide Web Consortium. Available at: www.w3c.org. 2004
135. The Wikimedia Foundation. Wikipedia. Available at: en.wikipedia.org. 2005
136. UserLand Software Inc. ThinkTank 2.41NP and MORE1.1c. Available at: www.outliners.com. 2005
137. Wittrock, M.C. Learning as a generative process. *Educational Psychologist*, 11 87-95.
138. Wjj Software. MyBase Desktop Edition. Available at: wjjsoft.com. 2005
139. Wolber, D., Kepe, M. and Ranitovic, I., Exposing document context in the personal web. in *Proceedings of the 7th International Conference on Intelligent User Interfaces*, (San Francisco, California, 2002), 151-158.
140. Woods, W.A. What's in a Link: Foundations for Semantic Networks. in Brachman, R.J. and Levesque, J. eds. *Readings in Knowledge Representation*, Morgan Kaufmann, 1985.
141. YellowPen, Inc. YellowPen Professional Service. Available at: www.yellowpen.com. 2005
142. Zoot Software, Delray Beach, Florida. The Zoot Information Processor. Available at: http://www.zootsoftware.com. 2005