THE EFFICACY OF

PERSONAL KNOWLEDGE BASES FOR

MATERIALIZING MENTAL IMPRESSIONS

by

STEPHEN C. DAVIES

B.S. Rice University, 1992

M.S. University of Colorado, 1996

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirement for the degree of

Doctor of Philosophy

Department of Computer Science

2005

This thesis entitled:
The Efficacy of Personal Knowledge Bases for Materializing Mental Impressions
written by Stephen C. Davies
has been approved for the Department of Computer Science

_____
Dr. Roger A. King, Committee Chairman

_____
Dr. Clayton H. Lewis, Committee Member

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

HRC protocol # 0705.13

Davies, Stephen C. (Ph D., Department of Computer Science)

The Efficacy of Personal Knowledge Bases for Materializing Mental Impressions

Thesis directed by Professor Roger A. King

Software tools abound for managing documents and other information sources, but are rarely used to store the mental knowledge readers glean from reading them. Hence our conceptual understanding – perhaps our most precious commodity in the so-called "information age" – is normally left subject to the whims of our erratic memories.

This thesis explores the concept of a *personal knowledge base*: an experimental database and interface designed to store and retrieve a user's accumulated personal knowledge. It aims to let the user represent information in a way that corresponds more naturally to their mental conceptions than textual notes would. People naturally form mental models of the domains they explore and learn about, and a personal knowledge base allows these to be expressed and archived directly. They need not be converted first to text, a representation which is actually alien to much of the human thought process. A personal knowledge base reflects a user's own subjective understanding of their world, permitting alternate views of the same objective data. And just as each person has many ideas from different domains tied together through perceived associations, it supports an integrated web of knowledge, potentially mirroring the entire contents of its owner's mind.

After defining the notion of a personal knowledge base precisely, and outlining its role in a user's life and expected benefits, this thesis provides the design

rationale for a prototype application. It then presents the results of deploying it to twenty volunteers who used it in real-world settings for extended periods of time. The results suggest that such a tool can be an tremendous asset for a variety of knowledge-related tasks, although commitment, discipline, and a willingness to alter one's habits are prerequisites for maximum success. It also sheds some light on the ways people typically work with knowledge, and suggests that unless these patterns are changed, they may ultimately form limitations on the effectiveness of such a tool, and indeed on our use of knowledge in general.

DEDICATION

This thesis is dedicated to my Lord and Savior Jesus Christ, Who was its inspiration, power, and hope; and to my devoted wife, Rae, whose patience and encouragement throughout these long, dark years was invaluable.

ACKNOWLEDGMENTS

I am immeasurably grateful for the contributions of Scotty Allen, Jon Raphaelson, Emil Meng, Jake Engleman, Javier Velez-Morales, Edwin Eng, and Nick Terkay to many aspects of this thesis. Their formative feedback, design insights, and just plain hard work were amazingly beneficial to me.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

It has become cliché to refer to our era as "the information age" – cliché because it is so obvious. Personally and professionally, modern man is simply bombarded with information on all sides, whether physical books, magazines, and newspapers, or electronic documents, Web pages, and e-mail messages. The flow is constant and unremitting, partially a product of the astonishing rate of new truths being discovered, and partially of the ever-growing number of opinions that are expressed. It is one of the individual's primary tasks, it seems, to cope with this onslaught; to filter it and to leverage it to its maximum utility.

For decades, the field of computer science – or "information science," a closely related discipline that now often passes as an exact synonym – has been largely focused on better equipping users to deal with the information they must manage. Information retrieval techniques help researchers find relevant needles of knowledge in the exponentially-growing haystacks. The advent of the World Wide Web has put billions of documents at our fingertips, and also empowered the individual to easily broadcast their own personal voice. The twin fields of data mining and machine learning seek to ferret out the essential truths from large quantities of otherwise impenetrable information. Even the operating systems of personal computers have been optimized to store more and more information and to selectively retrieve it faster. Keeping pace with the rate of information growth and

trying desperately to keep the user in control of it: these are the threads that tie together the majority of our diverse technologies.

It is worthwhile to pause and reflect, however, on what the ultimate purpose of all this information actually is. We evidently care a great deal about these sources of information, but what do we ultimately *do* with them?

Granted, in a few cases, they serve as inputs to automated processes that do useful work for us: for example, portal Web pages that "screen scrape" various commercial sites in search of the best price for a particular item. But because the bulk of the information that we encounter daily is expressed in natural language, rather than a rigidly coded form amenable to machine processing, such efforts are fairly rare. Indeed, despite all of the intervening technologies for improving access to information, the vast majority of documents are still authored by humans, and intended solely for human consumption. And therein lies the motivation for this thesis.

Information vs. Knowledge

The phenomenon whereby humans encounter and process natural language text can be seen as a process in which an input – *information* – is transformed into a result – *knowledge*. The related terms "data," "information," and "knowledge" have been variously defined in an attempt to draw distinctions between them. In this thesis, I 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.

The difference between information and knowledge, then, is that the former is processible and the latter has already been processed.[1] Hence the owner of a large textbook 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. Until knowledge comes into being, the information itself is merely an assortment of tantalizing but ultimately unproductive symbols and grammatical structure. Knowledge is the only means by which it can have any value whatsoever in the end.

Numerous psychological studies have confirmed that the knowledge humans store and make use of in their heads is almost never identical to the information from which they originally acquired it. [257, 259, 325] Instead, knowledge can be seen as the formation of a mental model in one's own mind that corresponds to a subjective understanding of the information encountered.[16, 135, 174, 298]. This is easily demonstrated: if an ordinary literate individual is presented with a newspaper article describing some recent occurrence, they will normally have no trouble at all reading

[1] 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.

it, understanding it, and remembering its contents a moment later. Shortly after destroying the article, in fact, they can answer questions about it with a good deal of accuracy, recollecting many of the names and events it contained, the main thrust of the article, the flow of the narrative, etc. Yet they will usually be unable to remember even a single verbatim sentence.[259] This tells us that the reader has mentally stored not the article's external form, but rather the gist or meaning of it. Indeed, the original sentences themselves would be an awkward and restrictive medium for thought, deduction, and extrapolation to operate on. What a person needs is a fluid representation reflective of the article's true semantics, so that it can be mulled over, related to other knowledge, and applied in different settings. We will return to speculate upon the precise nature of this representation in a later chapter. For now, I intend only to establish that the result of processing information is something very different from the information itself. It is an unwinding and a decoding, often very dependent on the background knowledge and biases of the individual recipient (an important point to which we shall return later.) And this is what produces the proper fuel for the mind's machinery to be effective.

As an aside, and at the risk of overstating the matter, one could argue that acquiring and managing such internal knowledge is *a human being's primary value add in our society*. Over a century ago, perhaps, the critical ingredients for a person's success were the amount of land they owned, or else their raw materials, physical strength, or control over the means of production. But in our age it seems that an individual's most precious commodity is what they know. When one seeks out an expert for advice, it is not because that expert has a large collection of physical books

in their office. It is because they have *read* and *understood* many of those books (and other things besides), and this patient study has given rise to an accumulated, integrated arrangement of knowledge which that expert can bring to bear on new problems and situations. This is primarily what we do, especially in the white collar strata: we probe, we learn, we understand, and we apply. The knowledge we form gives us the ability to gain mastery over a domain, and hence the power, prestige, and skills necessary to achieve success by virtually any measure.

We can look at this in another way. Consider that everything a person knows or believes was originally perceived through their five senses. And everything they eventually communicate to others is either the verbatim regurgitation of this information (which is rather rare) or else the result of some process that went on internally in their mind. Clearly, then, there is an incredible amount of analysis that takes place inside a person's head, that results in something quite distinct from the isolated, unprocessed information sources to which they have been exposed. The mind compares and contrasts these sources, synthesizes them, draws conclusions, identifies patterns, and forms a foundational understanding which is imperative for the individual's effectiveness and even for survival. It is this phenomenon of personal knowledge that allows each person to make their own unique contribution to their world.

Attempts to automate the process

Recognizing that natural language is a clumsy tool with which to work, various researchers have tried to devise means for automatically turning a text into a

representation more like that which the human mind might actually use. It is difficult work. Owing partially to the inherent ambiguity in natural language, automatically deriving its semantics is fraught with peril. Great effort has been invested in constructing parsers that can operate on streams of text and produce conceptual structures of its meaning; in some cases, programs that can translate from one language to another have been written which are roughly satisfactory in limited domains. But generally speaking, attempts to pre-process natural language so that humans can avoid doing the reading themselves have not been successful.

There are several reasons for this. For one, people are accustomed to natural language, and would likely be hindered rather than assisted if offered some alternate foreign representation generated by a machine. Also, when people read a text, they do not assimilate and remember every detail, nor do they intend to; rather, some subset of the information will stand out to them, and be deemed important enough (or unusual enough) to file away for later use. And this leads to a third and most important reason: that the knowledge gleaned from a source of information depends very much upon the individual reader. There is a highly subjective element to our processing of information. Different things will stand out to different people. Claims will be accepted by some, but questioned or outright rejected by others. Distinctive parallels and comparisons will come to mind depending on the individual's background knowledge, and so forth. This must be so: otherwise, any two people reading the same text would have the same reaction to it and emerge with identical understanding. But we know this is not the case, else we could hardly account for the alternative theories and fierce debate generated as ideas are discussed in the public

sphere. We are individuals, not automatons; and as far as we can tell, each person has their own unique set of experiences, presuppositions, known facts, and outlook on life. Hence it appears that for the foreseeable future, turning information into deep and useful knowledge will remain almost exclusively a human endeavor.

The principal difficulty, however, is that this manual process of knowledge accumulation relies heavily on one thing that humans are quite poor at: retention. It hardly needs to be said that our memories fade over time, and that a subject we are intimately familiar with today will be hazy and unreliable six months from now, if not completely forgotten. This varies from person to person, of course, but it is rare indeed to find someone who can conjure up all the details of a past project on demand, without the need to re-familiarize themselves with the material.

Contrast this with electronic repositories, which can be protected, copied, backed up, and even stored on redundant disk arrays so that they are not only persistent but virtually indestructible. Computers, it seems, excel at the one part of the knowledge building process that humans do not: the permanent storage and retrieval of data over indefinite periods of time, without loss or modification.

This suggests that great gains might be realized from an environment in which each component – human and computer – can contribute what they do best. Humans peruse information, analyzing and interpreting it to produce valuable – though fragile and transient – knowledge in their minds. They then express this to a computer application in some form that is faithful to their thought process, at which point it is instantly and permanently archived. If such a process could be made natural and convenient, it could increase a user's effectiveness manyfold. They would have

instant and reliable access not only to their recent thoughts and ideas, but to any they had ever formed in the past. Over time, the application would be accumulating an ever-growing replica of the user's own memory, so that months or even years from the time it was recorded, it could be faithfully reproduced. And all this *without* requiring the user to track down the original information source(s) to re-read, re-parse, and re-interpret them.

An example scenario

The need for a computer tool like this is best illustrated by example. Consider the situation of John, a fictitious but not atypical academic researcher. He is constantly busy keeping up with a quickly-changing engineering discipline, but he is also interested in philosophy and history. In his spare time (such as it is) he likes to read all he can on these subjects, from a wide variety of authors and periods.

Lately John has been doing some reading on ancient Greek philosophy, trying to make sense of it and relate it to what else he knows. It occurs to him that even though millennia have passed, the basic ideas of the famous Greek philosophical schools are still central to much of Western thinking. They have influence, he feels, that permeates even the postmodern mind to varying degrees, and he finds it useful to trace some of today's prevalent presuppositions and value judgments back to their root.

One night he picks up his copy of a "Survey of Western Civilization" book that he's been reading before bed. The current passage is describing the prominent philosophers of the Hellenistic period, post-Aristotle. It not only names many of the

great thinkers and gives capsules of their thought, but also contains historical interludes, anecdotes, etc.

After reading this passage and considering its contents, he finds that he has formed a tenuous framework in his mind. It is a mental impression that relates the basic facts, stripped of most of the details that the passage includes. Oddly enough, it seems to be geometrical, as though it could be depicted on a two-dimensional canvas. It might look like this if materialized:

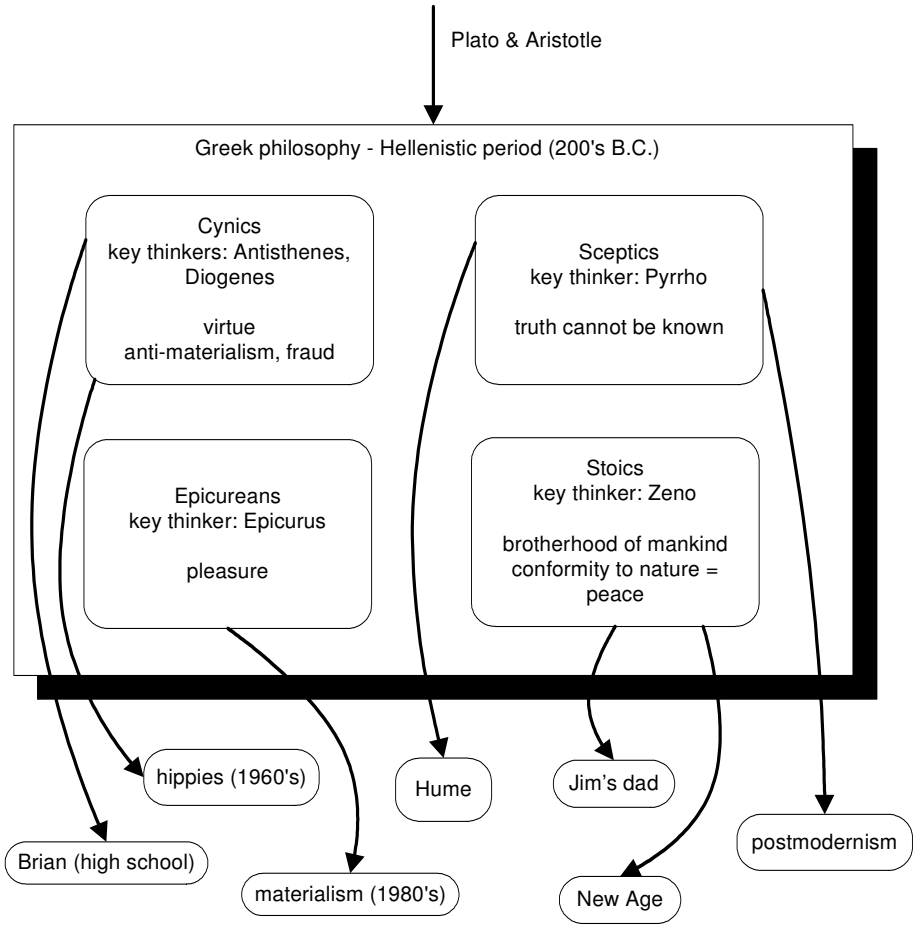

Figure 1. **A rendering of a hypothetical "mental impression" as perceived by a reader of a history book on Greek philosophy.**

Note several things about this mental impression:

1. It is a *distilled* version of what John has read. Dropped are most of the details of the narrative, including nearly all of the examples, stories, descriptions, and the writer's poetic style. This is because while those elements were enjoyable to read, they are not what John aims to retain for the long-term. One of the main reasons he chose this book was to acquire the kind of condensed view of this historical period that he can easily apply and relate to other knowledge.[174]

2. His mind seems to perceive it as *spatial*. Granted, the particulars of the boxes (shadows, rounded corners, etc.) may not be crystal clear in his mental image. They are present only because after all *some* actual shape must be used in a diagram. But what is important is that the mental conception he has is not made up of text sentences of numbered outlines. Rather, John can actually picture the "upper-left corner" of the "Hellenistic philosophy" box, and feel quite comfortable with an element called "Cynics" residing there.

3. The topic at hand *relates to other knowledge* in his mind. When he read the author's description of the Sceptics, John immediately thought of David Hume, and of the influence that he and others have had on the so-called "postmodern" worldview prevalent today. That perceived connection was so strong that it is hard for John to imagine severing it and leaving the "Hellenistic philosophy" box in isolation. Already, it has been woven into the fabric of his understanding as a whole.[23]

4. It is inherently *subjective*. This diagram represents John's own mental impression, and no one else's. He created it because it was helpful to *him*. It cannot (and should not) be replaced by an attempt to objectively catalog the "correct" view of the domain. It contains judgments and patterns that others may disagree with, and

in places it may actually be provably wrong. It includes elements that are unique to his own experience, like the arrows to "Brian" and "Jim's dad," who are acquaintances in his own personal sphere. John could articulate his mental impression to others, if they were willing to listen, and they might learn something from it. But that would be by means of adjusting and massaging his framework to intersect with and influence their own. They probably wouldn't be able (or willing) to simply copy it "as is" into their own brain.

As time goes by

This knowledge is useful to John, else he would hardly have invested the time to acquire it. Hence his desire is to retain it. He would like these facts that he has learned and the mental structure he created in order to integrate them to become a permanent part of his memory and understanding, so that he can make use of them to interpret his world.

Now if John were working full-time as a historian in this field, he would probably be constantly exercising this portion of his memory: fitting other facts into it, refining it, forming theories and drawing conclusions. But that is not the case. Philosophy for John is only a hobby, and he is very often forced to drop it for extended periods of time in order to attend to more urgent matters. When he returns to the "Survey" book after a week or two, and tries to reset the context and conjure up his mental impression so he can continue reading, he finds that it now looks like this:

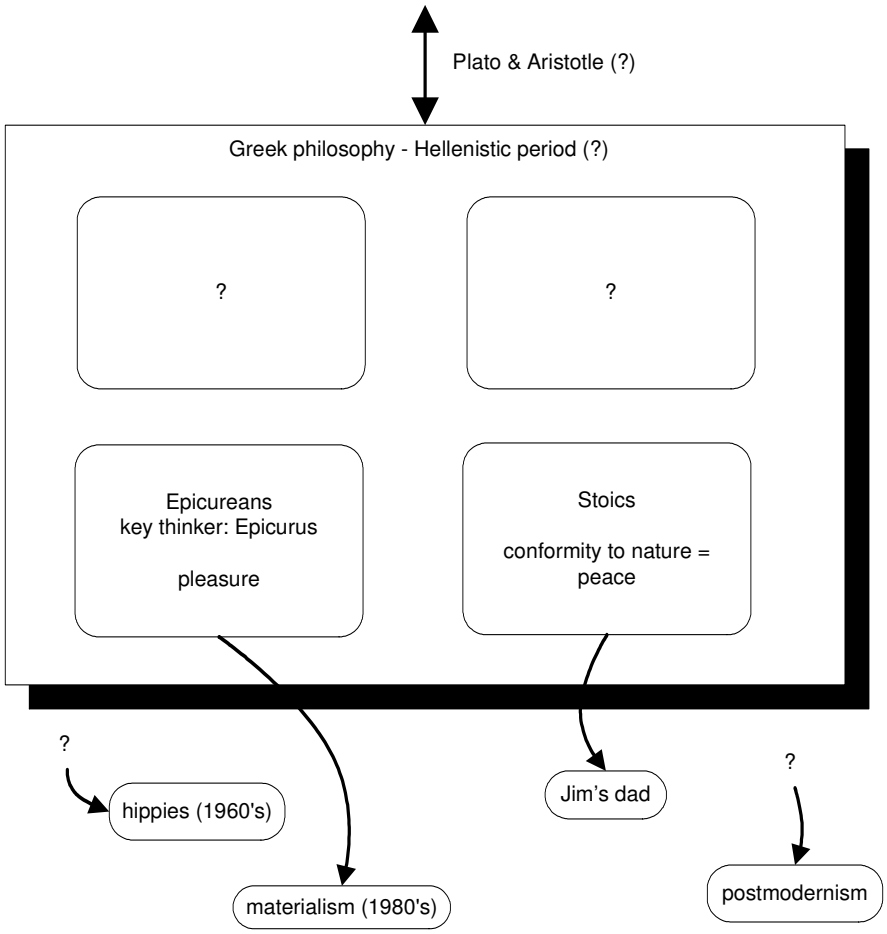

Figure 2. The same mental impression after time has passed, with many of the details forgotten or confused.

John's memory has faded considerably. He remembers that he read a passage about the Hellenistic philosophers, and he has a vague recollection that there were four important schools mentioned, but many of the details and even the very identities of some of these schools have been lost. The Epicureans and the Stoics stuck in his mind for some reason (perhaps a particularly memorable story, or the curious sound of the name itself, or that his concentration happened to have been more focused when he read those paragraphs) but that is all. The other two are simply empty boxes without even a label he could look up in an index or on the Web. He recalls only a handful of the other topics that had occurred to him when he read the passage, and of

these, the specifics of how he perceived them to be connected remain for only two. And to his embarrassment, he has even forgotten whether this entire philosophical period was before Plato and Aristotle, or after!

It appears that in order to get back to the mental frame of mind he had when he last put down the book, John will have to reread the passage and recreate the original mental impression. It will probably be quicker the second time, since he has already done some analysis and the details should spring to mind more quickly as he encounters familiar text. But that is not very encouraging. John wants knowledge that will last, not knowledge that he has to regenerate from its original sources each time he needs it. And if this is what his mental impression looks like after a week or two, what will it look like in a year or two?

The act of reading the book was entertaining and relaxing, but John will feel deflated if the only effect turns out to be a transient and experimental one. He did not pick up the book merely to pass the time, but to better his understanding of the world for the long-term. And it is beginning to look as though the intrinsic limitations of his mind will prevent him from ever accumulating the kind of focused, systematic knowledge he hopes to acquire for anything other than the field he is daily and intensely engaged in.

The state of the art

Without a tool like the one this thesis describes, John has a number of options for remedying this dilemma, none of them ideal:

- He could spend more time and effort drilling the facts into his mind in the hopes that rote memory will suffice. But this process is not enjoyable,

especially because he is aware that no matter how much time he spends "cramming," it is likely that much or most of the mental impression will inevitably crumble. This approach is also impractical for anything more than a very small number of topic areas.

- He could mark up the book with a highlighter and his personal annotations, drawing attention to the key facts and their relationships. This gives easy access to the author's elaborations should he need to delve further. And it ideally requires very little writing, if he can simply highlight select phrases already provided by the author. But his mental impression is no longer easily at hand, being stored on his bookshelf in a particular location that may not always be available to him. If the book is lost, destroyed, or loaned out, so is his access to his own mental impression. Also, more subtly, John's knowledge is now intimately tied to the book itself. This is quite artificial – presumably he read this sort of book because it contains transcendent knowledge that many experts agree on and which is applicable in a variety of settings. He wants to remember the Stoics and the Sceptics, not a particular bit of writing by the particular author William McNeill. The knowledge itself should be severed from that tome and stand on its own, yet with this solution that is impossible. Finally, there is no easy way to "link" to other knowledge he possesses. The best he could do is scribble a note like "reminds me of Jim's dad" or "see chapter 5 of Durant's 'Story of Philosophy'" which requires a good deal of anticipation, recall, and elbow grease to make profitable. And besides, these sorts of links are only "one way." They enable

him to answer the question, “now what were the things that the Stoics reminded me of?” by looking up his notes in the book, but if he wonders, “now what were the things I thought might have influenced Jim’s dad?” he is at an impasse.

- He could use pencil and paper to transcribe his mental framework into a notebook. This is fluid, adaptable, familiar, low cost, and free form. The notes will stand on their own (apart from the book), and can freely combine both text and diagrams. However, this choice gives none of the advantages that electronic representations do. Like book highlighting, the physical paper can be only in one place at a time. It cannot be backed up, copied, or searched (except by hand.) It cannot be rearranged except at the painful expense of erasing, crossing out, or copying sentences by hand to a new piece of paper. It cannot “link” to any particular document except by naming it (as above), nor can it reference another conceptual entity (like “Aristotle,” “postmodernism” or “Jim’s dad”) except by naming. This puts all the burden on the John’s own interpretation and footwork, rather than using machines to automate those kinds of tasks.
- He could create an electronic text document (with, for instance, Microsoft Notepad) to jot down notes. This would allow them to be backed up for safekeeping, mailed to a friend, copied to a laptop or PDA, and (syntactically) searched using common tools. However, as mentioned above, John’s mental impression is not most naturally representable in ASCII text, but in a nonlinear arrangement of elements. Hence it is at best awkward to reduce it to

text, and he may in fact lose valuable information that way. In any event, when he brings up his notes a year later to recollect his mental impression, he will in fact have to recreate it after interpreting his comments. And this solution has the same difficulties with "linking" that a paper document does.

- He could use Mozilla Firefox to create a page of notes with embedded markup. This solves the "link problem," since John could create an entire series of web pages on the topics he has studied, and then create links between pages representing the mental relationships he perceives among knowledge elements. However, it's still represented as text, which is not the natural medium of his mental impression. He could use structuring elements and visual cues (such as font, color, outline headings, tables, etc.) to give more richness to the words, but then he would be mixing formatting and presentation information with the true semantics of what his knowledge *means*. Also, this solution requires a fair amount of overhead that would be an obstacle to average users. The tools are more complex, editors and readers tend to be different, etc. Finally, the only thing John can link to is a page (or part of a page) which is different than a conceptual entity. This approach encourages one to use a paragraph of text as a stand-in for a real-world entity (like a person, movement, or time period), which is not the truest way to model one's understanding. (In his mind, for instance, John doesn't have "a paragraph describing Epicureanism" related to "a web page that discusses the culture of materialism in the 1980's," but rather "the philosophy of Epicureanism" related to "the phenomenon of materialism.")

- He could use a drawing tool like Microsoft Visio to create a drawing consisting of elements that represent his mental impression. This casts the knowledge in a form that is truer to its actual nature. And since "a picture is worth a thousand words," when John returns again to his diagram the richness of the visual presentation is likely to jog his memory more quickly and effectively than a paragraph of text would. However, although stored electronically, this diagram is virtually opaque to the computer. John cannot easily search for elements within a diagram, nor can he link to them from other diagrams. If he wanted to store lots of knowledge in this way, he would have no way to search his knowledge base except by the filenames of the Visio documents. Also, each element in a diagram is a separate entity. Though it may have the name "Stoics" in it, it does not really represent "the Stoic philosophers" except by one's interpretation. In actuality, it is simply a rectangle. If John were to create another diagram and add another rectangle to it with the word "Stoics" in it, this would simply be another rectangle on another diagram. It would have no semantic connection to the other rectangle except by his own inference. This is error-prone, since a box labeled "Brian" could certainly mean two different things on two different diagrams. And there is no easy way to ask, "show me all Visio diagrams that have a 'Brian' rectangle on them." The major problem here is that pictorial representations are being used to imply semantics for human viewers, without the unambiguous foundation behind them that would permit machines to leverage that same information.

As is readily apparent, these "solutions" are far from adequate.[2] They do not address the real problem, which is to externalize and make permanent the abstract web of interrelated knowledge that John has in his mind. They use various tools, built for entirely different purposes, to only crudely approximate what John needs them to do. And so the alternative he will most likely choose is to rely on his own capricious and leaky memory. He will probably be frustrated that so much knowledge that he has gained seems to disappear so quickly, and that what he learns today he can be almost certain he will not know tomorrow. But such is the status quo today. He will resign himself to an unreliable and randomly vanishing knowledge collection. The result is that as he moves through life, instead of gaining more and more understanding, he simply acquires *different* understanding: he normally does not succeed in adding to, but rather in *replacing*, his previous knowledge. John's mind seems to only retain what is fresh and active, with all else being relegated to the trash heap of "things he once knew."

Towards a solution

If our greatest asset is in fact the knowledge we have worked so hard to build, why would we be content to manage it with inadequate tools, or worse yet, to never record it at all? Surely there is more in our heads worthwhile to manage than just contact information and a calendar. What about all the books we've read, all the lectures we've attended, all the Web pages we've browsed? We once spent

---

[2] I omit commercial note-taking applications from this "state of the art" list because this body of software is analyzed in great detail in chapter 3. There, the advantages and limitations will be discussed, and it will be shown why even these kinds of tools fall short of what John needs.

significant time attending to each of them, ferreting out the knowledge therein and incorporating it into our own understanding. And the reason we did it was because that hard work was the only way to make use of the material we absorbed. Are we really content to let all of that pass away according to the whims of time?

The purpose of this thesis is to investigate the feasibility of designing a "personal knowledge base" (PKB); that is, a computer system that will supplement a user's own memory. Such an application would allow John to quickly and easily record his thoughts in a representation that is faithful to what he actually "sees" in his mind. It would accommodate varying levels of precision. It would allow him to focus on a particular area of knowledge (like "Hellenistic philosophy"), but also to connect the things he learned in one area to those in other areas. The end result of all this record-keeping would be a sort of "surrogate brain," to which he could always return to refresh his erratic and unreliable biological memory. If used correctly and consistently, it would allow him to leverage all the knowledge he has ever gained to its maximum potential.

The rest of this thesis is organized as follows. In the next chapter, I will define several important terms and then describe the characteristics of a personal knowledge base, including the benefits and limitations we might expect from one. Then, I will provide a thorough review of the relevant academic literature and commercial systems that have thus far attempted at least a partial solution to this problem. In chapter 4, I will introduce "Popcorn," a prototype PKB software application that forms the cornerstone of this thesis. I will describe the philosophy behind its design, a summary of its architecture and implementation, and a description

of how it is intended to be used in practice. The following chapter will contain a broad summary of results from testing Popcorn on a number of real-world users, identifying both strengths and weaknesses of the approach. Finally, I will conclude with a big-picture analysis of this data, draw conclusions about Popcorn's efficacy, and speculate about the possible future of PKB systems in light of what this study has revealed.

CHAPTER 2

DEFINING THE PERSONAL KNOWLEDGE BASE

The scenario from the previous chapter mentioned that today's user is often forced to adapt existing tools, which were designed for very different tasks, to the job of maintaining a store of personal knowledge. Word processing documents, spreadsheets, drawing applications, and personal websites all store information of sorts, and so they could theoretically be used as makeshift personal knowledge bases by the courageous. I have already alluded to the limitations of these approaches, however, which are so obvious they are hardly worth mentioning. But this gives rise to another question. What exactly *is* a PKB? What are its key characteristics that set it apart from other applications? How do we know one when we see it? And what are the benefits we should expect them to provide?

Before I offer a precise definition, I need to introduce a few other terms to help frame the discussion. The first is the distinction between data, information, and knowledge, which was covered in the introduction. The second is the notion of what I will 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, organized and arranged according to the user's understanding.

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. I 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 mind any complex theory, but simply observes some similarity or "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" I 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 transferred out of the subjective realm and placed into the objective.

Finally, these terms have an important correspondence to those of "information" and "knowledge" as previously defined. Simply put, the objective realm is normally composed of information sources, while the subjective realm is made up of knowledge representations. This is because the objective realm contains material, nearly always expressed in natural language, that can be browsed and examined. Only when someone takes the time to read some of it, understand it, and

incorporate it into their understanding does knowledge arise, which is always subjective because it is personally perceived.

Three essential components of a PKB

I am now in a position to offer a definition for a "personal knowledge base." As the term has three words, so the definition has three components: a PKB must be *personal*, it must contain *knowledge*, and it must be a *base*, or integrated foundation.

> **personal**: Like the system that John yearned for in the previous chapter, 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 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 the user has seen, but to preserve the knowledge that they have *learned* from those sources. When John returns to his PKB to retrieve some knowledge that he has stored, he doesn't want to be merely pointed back to the original documents again, so that he has to re-locate, re-read, re-parse, and re-learn the relevant passages. Instead, he wants to return to the distilled version of the particular truth he is seeking, so that the mental model he once had in mind can be easily reformed.
>
> **base**: A PKB is a consolidated, integrated knowledge store. It is a reflection of the user's own memory, which, as has been observed from antiquity, 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 (for instance, isolated files or directories) that cannot reference or interrelate with each other. Just as John has only one mind, and he can connect any of his ideas together without regard for artificial boundaries, so a PKB must act as a single, unified whole.

Before continuing, it is worthwhile to enumerate several classes of systems that *cannot* be classified as PKBs because they fail to meet one or more of the above criteria:

- Collaborative efforts to build a universal objective space. The World Wide Web itself is in this category, as were its predecessors HyperTIES[278] and Xanadu[229], Web categorization systems like the Open Directory Project[232], and collaborative information collections like Wikipedia.[324] Such systems distribute public information to be shared; they are ill-equipped to deal with the fluid needs of the *personal*.
- Search systems like Enfish[106], Clarity[264] and the Stuff I've Seen project[103] that simply index and search one's information sources on demand, rather than giving the user the ability to craft and express *knowledge*.
- Tools whose information domain is mostly limited to time management tasks (calendars, action items, contacts, etc.) rather than to "general knowledge." Blandford and Green[43] and Palen[249] give excellent surveys; common commercial examples would be Microsoft Outlook[216], Lotus Notes[161], and Novell Evolution[238]. These tools are custom-tailored to specific kinds of structured information, not to *knowledge*.
- Similarly, tools developed for a specific domain, such as bibliographic research (e.g., [163], [317],[240]), rather than for "general *knowledge*."
- Systems that focus on capturing transient information, rather than archiving knowledge that has long-term and enduring value. Examples would be Web logs[139] and e-diaries [178]. Hence, although they accumulate a

chronological transcript of observations, they are not designed to be easily consulted as a *knowledge base* in the way I am describing.

- Tools whose goal is to produce a design artifact rather than to maintain knowledge for its own sake. Systems like ART[226] and Writing Environment[297] use intermediate knowledge representations as a means to an end, abandoning them once a final artifact has been produced. Hence they do not concern themselves with maintaining on ongoing knowledge *base*.

Again, by bending these tools to new purposes and adapting supportive processes to cover their deficiencies, one could use some of them to approximate crude PKBs. But since personal knowledge management was not their objective, they will be lacking several crucial features that I will discuss in later chapters. For purposes of this thesis, they lay outside the bounds of candidate PKB systems.

Benefits of PKBs

In the following chapter I will provide a lengthy review of both academic and commercial efforts to create personal knowledge bases. But before classifying and dissecting them, it is interesting to simply examine the different ways such systems are "pitched" to the public. This sheds some light on the advantages that PKB's should supposedly provide. And we can see 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."[48] In an educational setting, they "make it easier for students to grasp concepts and ideas" [296] and "strengthen critical thinking, comprehension, and writing skills."[162] In a business setting, one can "capture in detail thoughts and ideas, to explore them, and gain new understanding and insight."[28]

It is interesting to note that many of these systems make direct appeals to human psychology in their literature. Product names like "PersonalBrain"[311], "Axon Idea Processor"[45] and "Mind Manager"[220] are certainly suggestive, and one product calls itself "a trusted thought container."[157] Another claims to be "…an application that's actually designed to help you think. It's like having an extra brain."[244] A research system claims to "actively model the user's own memory"[169], and one commercial product's testimonial says flatly, "I now feel like my computer is an extension of my brain."[215] 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…"[60] 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."[305]

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.[151] PKB systems profess to answer this need, allowing one to "organize personal information", claiming to be "a more productive way to stay organized"[20] and that "you will finally be able to organize all your random information."[215] And clearly this organization is personal and subjective; one tool brags that it "complements individual styles for capturing and organizing thoughts."[217]

Parenthetically, it is easy to see why organizing knowledge is helpful: it greatly enhances our ability to process and remember it. Indeed, studies have shown that imposing structure on even random information makes later recall attempts more successful and efficient.[49], [174] p.243. And even though objectively organized material is more easily learned and remembered, when a subject encounters unorganized material they will naturally impose their own organization upon it to facilitate learning. Long-term recall, in fact, presupposes a personalized organization of the material.[174] pp. 254-261, p. 266.

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"[77] and "a structure for organizing, storing, and retrieving information."[146] Products will "give you a place to stash all those stray snips of knowledge where they can be quickly recalled when you need them"[42] and let you "find any data in an instant, no matter where or how you entered it."[215]

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.[3] This makes it possible to "capture all your information in one place"[217] and "capture, analyze, and repurpose information from a variety of sources."[220] 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"[311], and another "eliminates this partition (between heterogeneous applications and data types) so that individuals can work with their information in a unified fashion."[9]

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"[157], or that "it's a notepad, outliner, scrapbook manager, information manager, freeform database, archive, bookmark manager and image database – all in one integrated application."[97] If nothing else, this illustrates that the potential uses of PKBs are vast. Certainly acquiring, integrating, maintaining, and

[3] 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.

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.

We can see already, therefore, that personal knowledge management is a real and pressing problem, and that many development efforts have arisen that make bold claims. The purpose of this thesis is to investigate the truths behind those claims, to pursue a promising alternative approach to PKB design, and to determine the extent of its utility. But before I present this new paradigm, I will examine the work that has been done in this area to date so that it will become apparent where a system like Popcorn stands in the stream of personal knowledge management research.

CHAPTER 3

LITERATURE REVIEW

Bush's dream

Although not explicitly described as such, the idea of a personal knowledge base dates back as least as far as Vannevar Bush's oft-quoted article "As We May Think"[57] in the July 1945 Atlantic Monthly. In it, Bush, President Truman's Director of Scientific Research, surveyed the post-World-War-II 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[4]." 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.

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

The Memex is often cited as the precursor to today's hypertext systems ([277], [239]) and indeed Doug Engelbart and Ted Nelson, hypertext pioneers, have both acknowledged their debt to it ([107], [229]). 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.

Four contributing bodies of research

In the sixty years since Bush published his vision, systems which might qualify as PKBs have sprung primarily from four different fields of research. 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.

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."[58] 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.

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 central idea or topic is written in the center of the diagram, and subtopics radiate outwards in increasing levels of specificity. (See Figure 3.) 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 below.) The number of software tools available for constructing mind maps is staggering; just a sampling would include [48], [127], [130], [72], [220], [237], [41], [241], and [219], and there are many more.

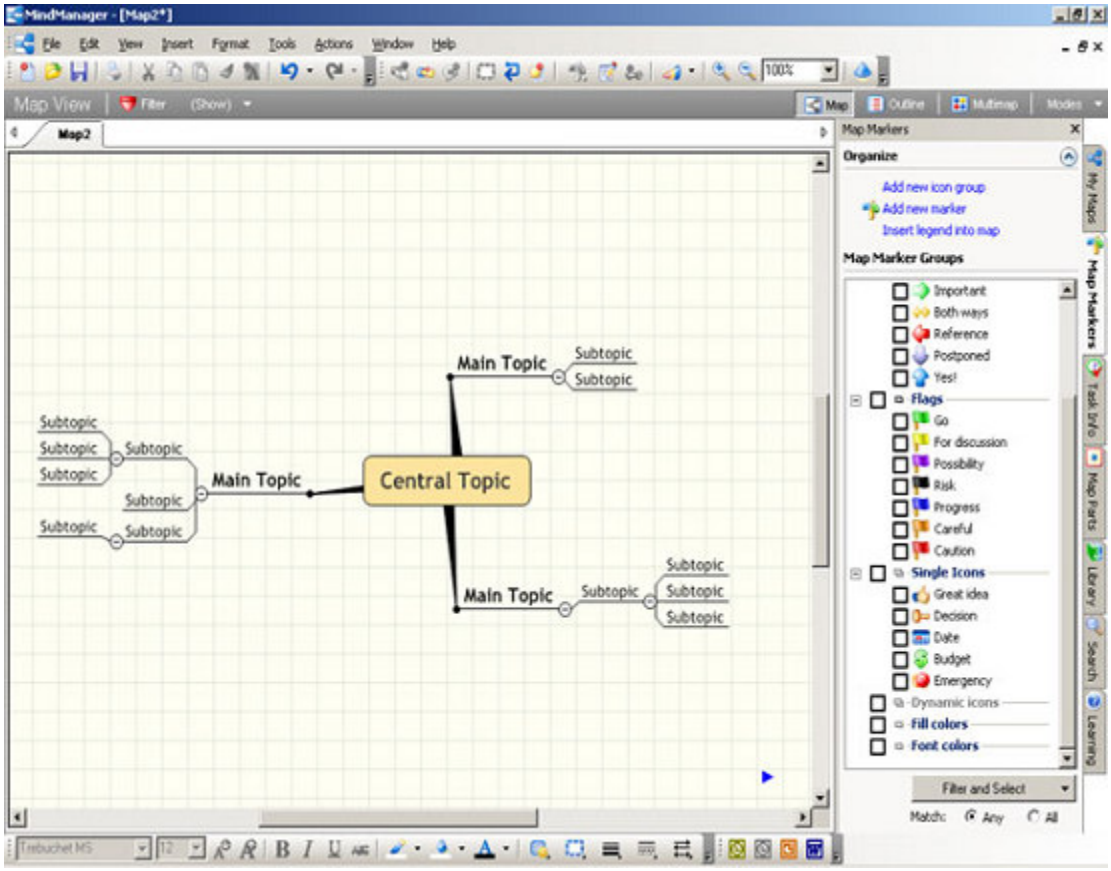

Figure 3. MindManager, a mind map creation tool. The structure of the diagram is inherently hierarchical, with the unlabeled branches between elements implicitly denoting "supertopic/ subtopic." [221]

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[235, 236], and based on David Ausubel's assimilation theory of learning.[23] 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[5]; and second, the links are labeled to identify the nature of the inter-concept relationship, typically with a verb phrase. (See Figure 4.) 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 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 [60], [296], [132], [13], [71], [45].

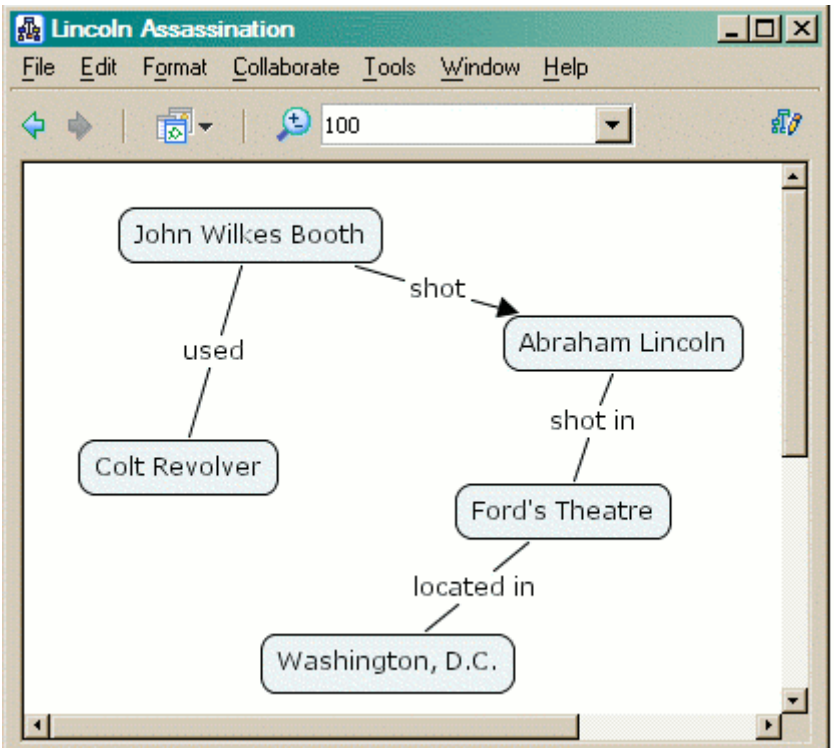

Figure 4. CMap, a concept map creation tool. The structure of the diagram is a general graph, with labels indicating the relationships between major concepts.[60]

[5] 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.

Note that a concept map is virtually identical to the notion of a "semantic network"[339], 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.

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 [28]. (See Figure 5.)

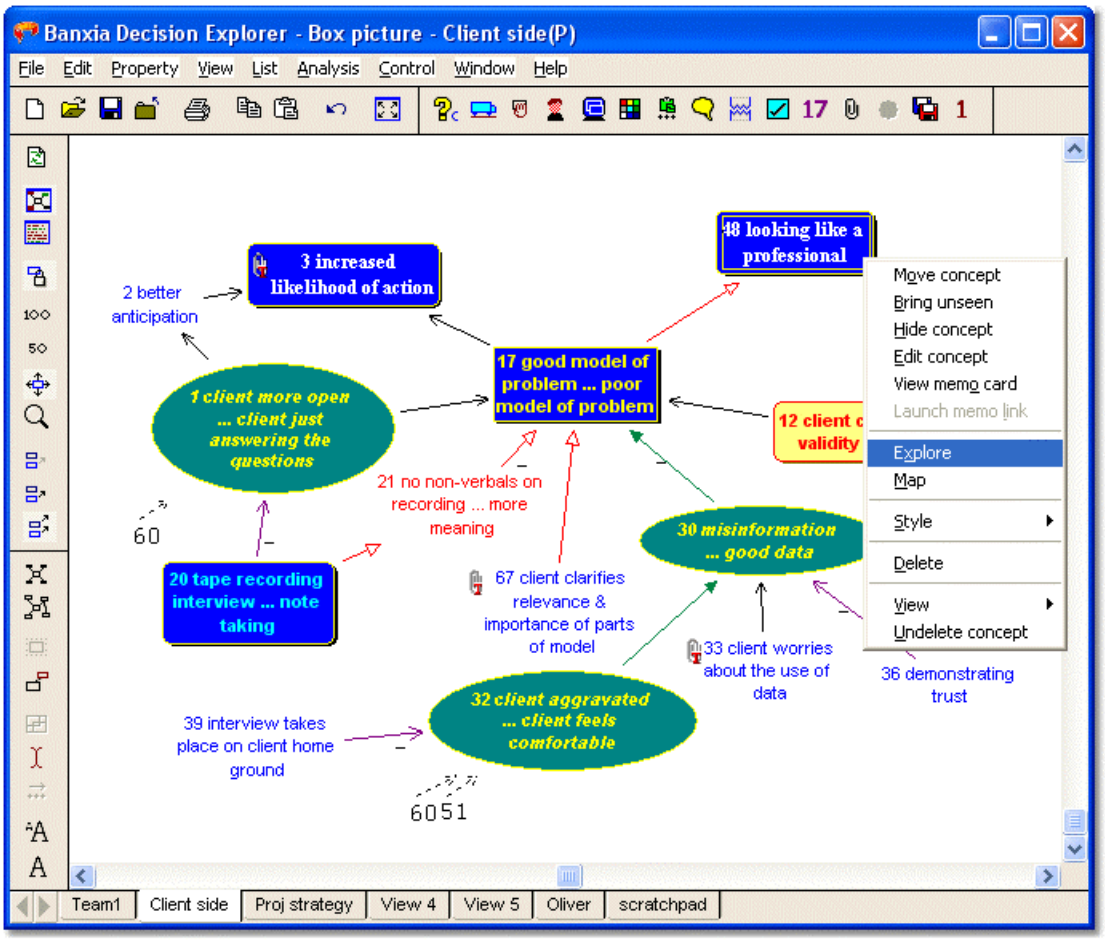

Figure 5. Decision Explorer, a cognitive map creation tool. Cognitive maps are mostly equivalent to concept maps, but are best suited to domains involving arguments and decision making. The ellipses inside certain elements are to be read "as opposed to" in order to make the meaning more precise.[28]

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.

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., [278], [141]); those that prescribe ways of linking existing documents together for navigation and expression of affinities (e.g., [133], [92], [251]); and those that use the hypertext model specifically to model abstract knowledge. Of the three, the last is the most relevant for this study, 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.

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."[107] 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, I will point out 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[316] and NoteCards[146] 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."[78] 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."[316] 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." (See Figure 6.)

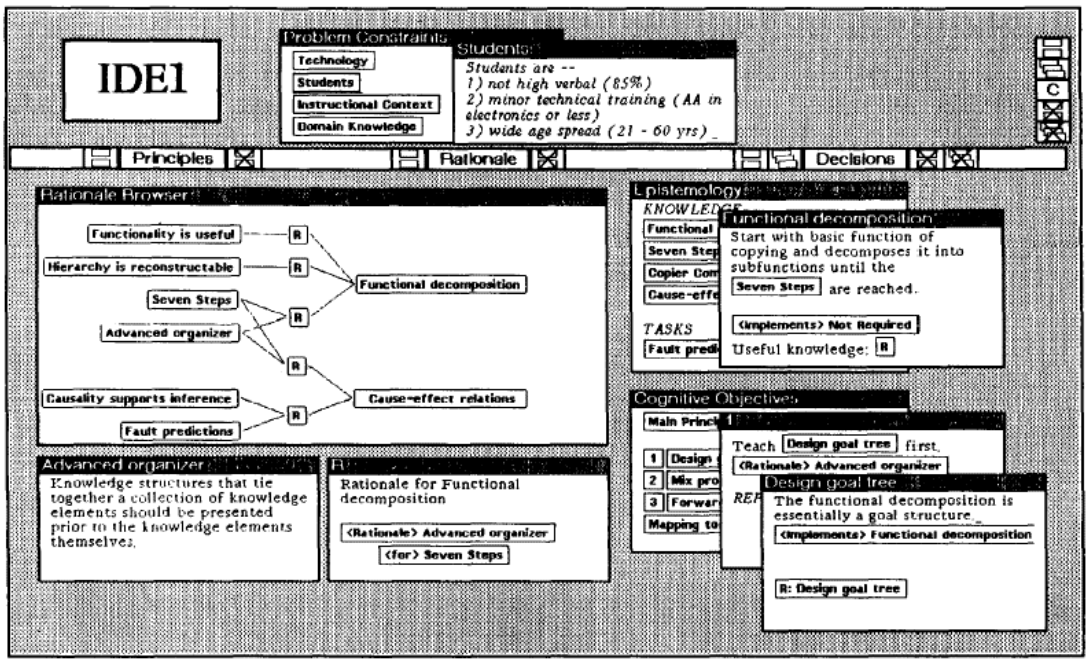

Figure 6. The NoteCards knowledge management environment.[146]

Conklin and Begeman's gIBIS system was another early effort into true knowledge representation, specifically for the field of design deliberations and

arguments.[79] The project lived on in the later project QuestMap[282] and the more modern Compendium[77, 282], 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[94], Aquanet[208], and SPRINT[63], as well as a few current commercial tools such as PersonalBrain[311] and Tinderbox[39].

Note-taking applications

The most explicit attempt to create a PKB as I 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., [217], [20], [70]). (See Figure 7.)

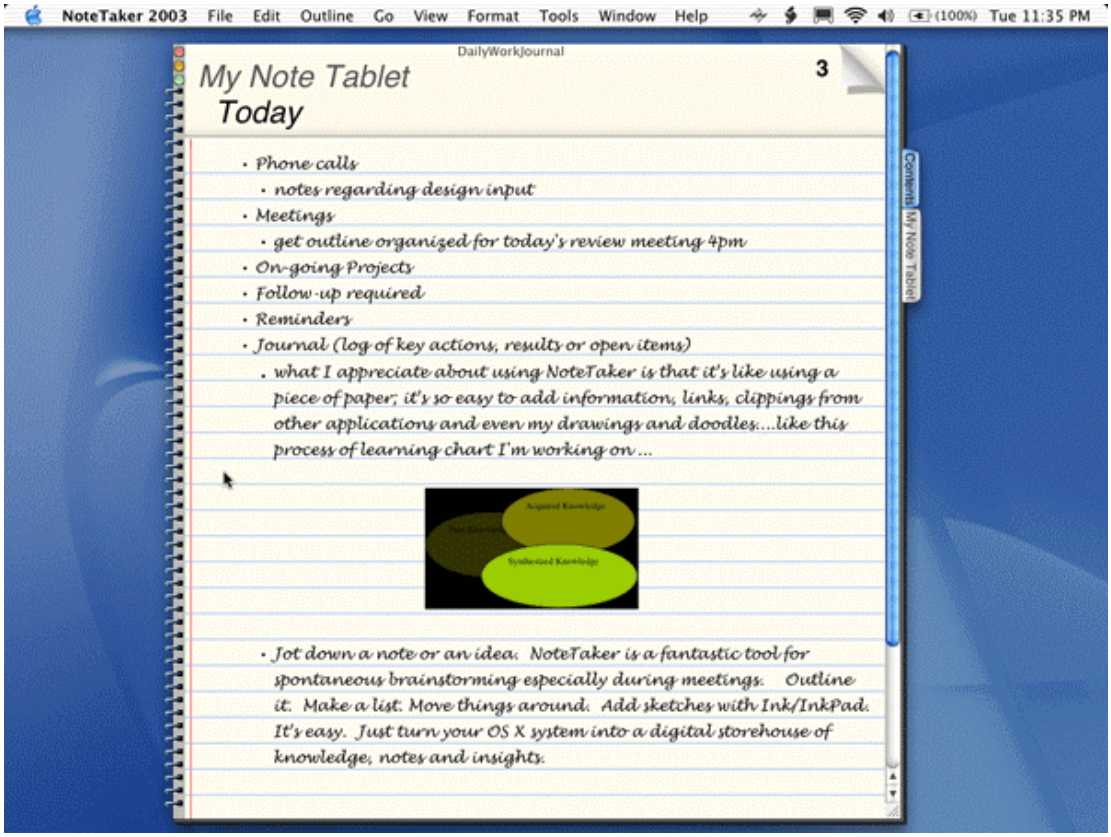

Figure 7. AquaMinds NoteTaker, a hierarchically-based note-taking application. [20]

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 application 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., [157], [217], [70], [20]). 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 ([171], [42], [344]), as I will discuss in depth

below in the section on 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.

An important subcategory of note-taking tools is outliners (e.g., [244]), 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 (see Figure 8.) 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[333]; more modern products feature the ability to add graphics and other formatting to an item, and even hyperlinks to external websites or documents.[125] The once abandoned (but now resurrected) Ecco system[231] 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.

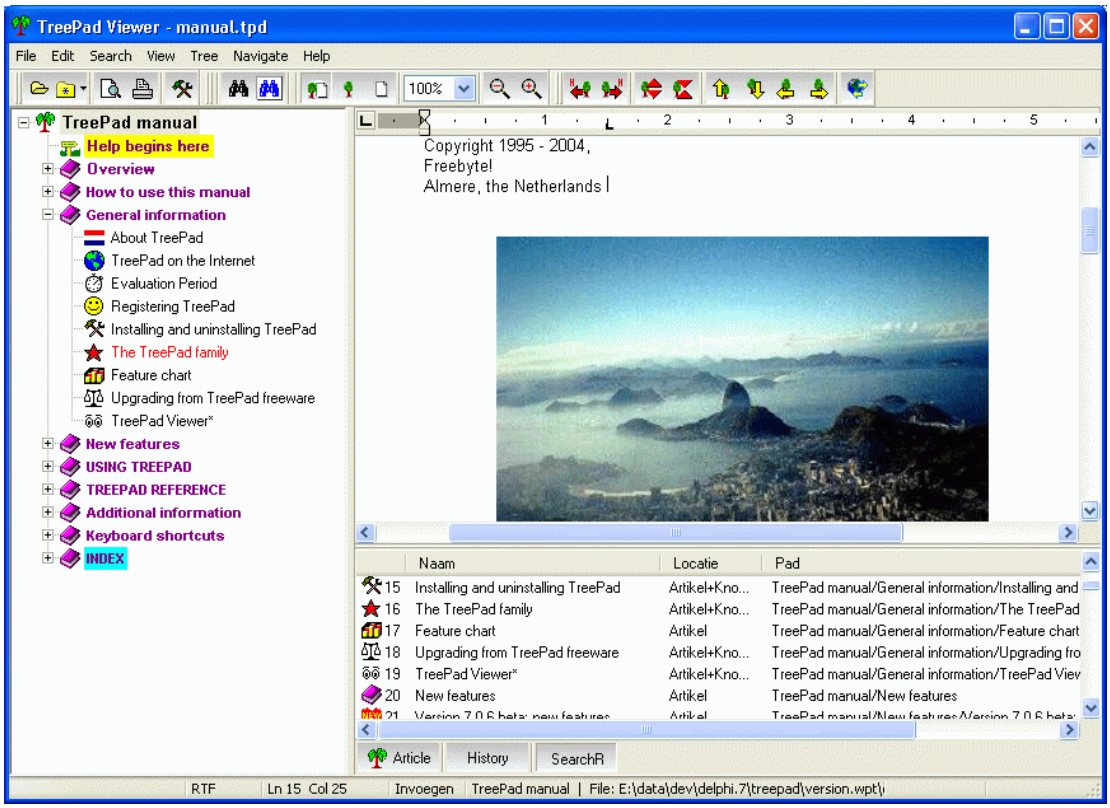

Figure 8. The TreePad outliner, which allows users to organize ideas in a hierarchy and then expand on each one with text, graphics, etc. [125]

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)[56] 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[342], Cartagio[222], and Hunter-Gatherer[280] 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 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.

Document management systems

Lastly, I 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 in personalized ways.

These efforts generally seek to provide alternative indexing mechanisms to the clumsy "directory path and file name" approach. Presto[101] 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[169] 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[9], 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[338] 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 themselves, associations implied by structural relationships, and content similarities discovered by text analysis agents.

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[155], DLITE[83], and Garnet[54] 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[209] with direct access to digital library search facilities. NaviQue[129] is largely in the same vein, though it incorporates a powerful similarity engine to proactively aid the user in organization. CYCLADES[266] 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.

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, I 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 "schemata" (or the singular form "schema") I 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 "schemata." 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, I 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, I will address the differing knowledge element types these systems support, and the ways in which schema has been introduced.

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."[171] 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?

The five traditional structural frameworks

It turns out that among the multitude of existing systems, only five basic structural frameworks have won wide acceptance, namely:

1. 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 very easy and 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 "central 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, 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 unfortunately inadequate for representing much complex information, as I will discuss later.

Other examples of tree-based systems are most personalized search interfaces ([98], [268], [266]), outliners ([244], [125]), and most of the "notebook-based" note-taking systems ([20], [217]). 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 ([215], [337]), and/or employ some form of transclusion (see below) to allow items to co-exist in several places ([344], [68]). The dominant paradigm in such tools, however, remains the simple parent-child hierarchy.

Trees 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[76], and from their need to focus on a desired level of abstraction, which trees enable by hiding and concealing levels of detail.

2. 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 directional from parent to child.) Note that a graph is a more general form of a tree. By using only unidirectional arcs (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.

	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

Table 1. Terminology employed by a sampling of graph-based knowledge management tools. (Interestingly, the standard mathematical terms "vertex," "arc," and "graph" are used by none of them.)

This model is the defining characteristic of hypertext systems ([147]) including many of those used for document management ([147],[338]). It is also the underpinning of all concept-mapping tools, whether they actually acknowledge the name "concept maps" ([60],[131]) or advertise themselves simply as tools to draw knowledge diagrams ([73], [243]). 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 purely cognitive structures can be ultimately reduced to a graph of some kind, which may point to sufficiency as a structural framework. (See also [262], [233].)

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. In other words, there are no "islands" of vertices that are severed from each other. Most graph-based

tools allow non-fully-connected graphs: knowledge elements are simply added to the system, and connected arbitrarily to each other, without constraint. But a few tools, such as PersonalBrain[311] and Compendium[77], 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.

For completeness, note that some hypertext systems (e.g., [96], [133]) add further precision to the basic linking mechanism by allowing nodes to reference not only other nodes, but sections within nodes (see [147]). This ability is especially useful if the nodes themselves contain sizeable content, and also for PKB elements making reference to fragments of objective sources.

3. 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 ([78], [205]) largely disappears with the tree model; one is rarely 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.

Concept mapping techniques are an example of this: a generally hierarchical paradigm is prescribed, after which users are encouraged to identify "crosslinks"

between distant concepts. And indeed when systems loyal to the mind mapping paradigm break its confines to permit arbitrary relationships between nodes, they are taking the same path.

One of the earliest systems to combine tree and graph primitives was TEXTNET[316], 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.[6] NoteCards 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.[114] 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[311] and Multicentrix[179] provide explicit support for parent/child relationships in addition to arbitrary connections between elements, allowing tree and graph notions

[6] 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.

to coexist. Some note-taking tools, while essentially tree-based, also permit crosslinks between notes (e.g., [215], [337]).

4. 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 two-dimensional 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[100] and VIKI/VKB ([209], [289], see Figure 9.) 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, 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.

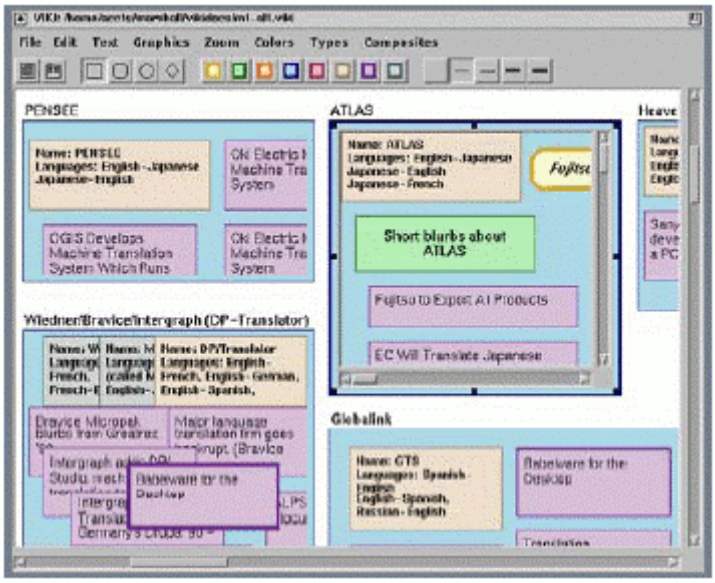

Figure 9. VIKI, one of the first spatial hypertext systems. Rather than links between elements, the primary way organizational information is conveyed is through spatial clustering. An automated spatial parser deduces the structure that the user has informally specified. [209]

Pad[252] 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 collaboratively.

Certain note-taking tools (e.g., [56], [217]) 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."[12]

Most of the integrated search workspaces (e.g., NaviQue[129]) are also in this category. And many of the 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 between items, and less formal expression through creative use of screen real estate.

5. Category. The fifth 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[101] where icons standing for documents are enclosed within ovals that represent categories (see Figure 10.) 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. (Notice that this is somewhat of a problem in Figure 10.) 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.

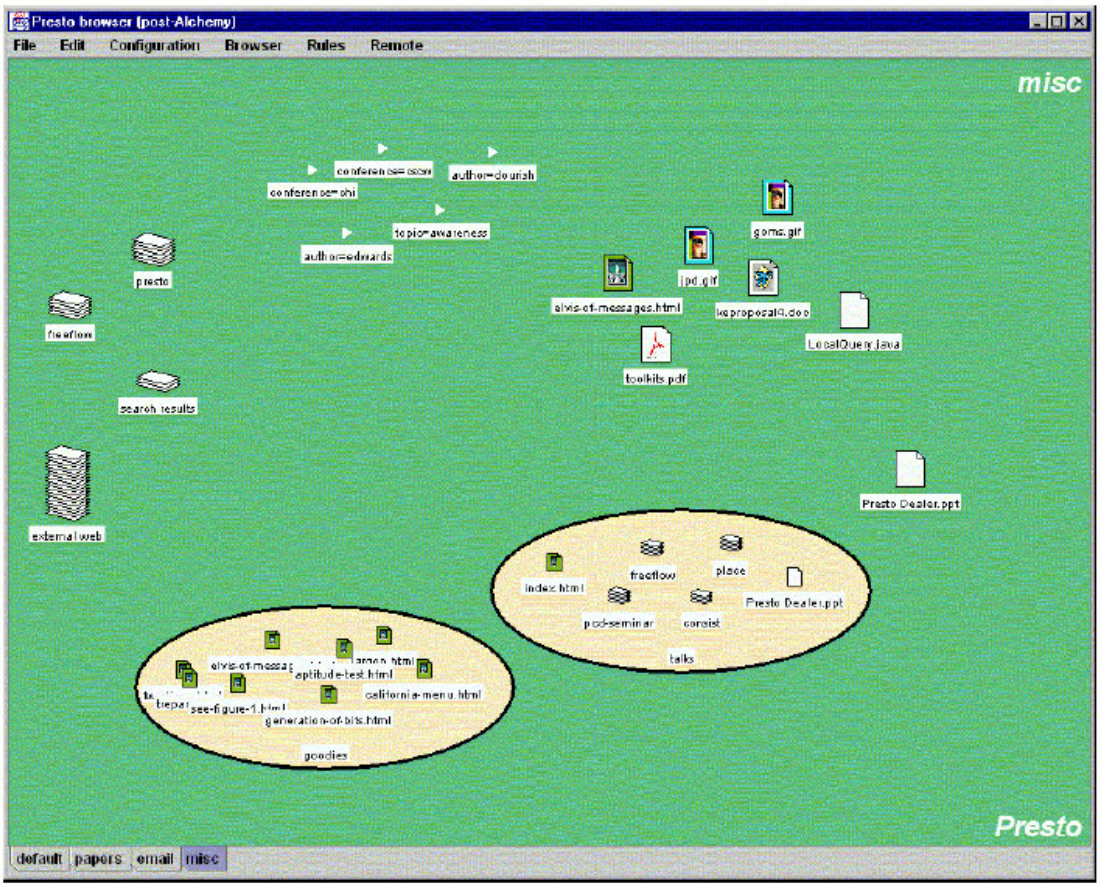

Figure 10. The Presto category-based browser, also known as "Vista." Although not shown very clearly in the diagram, items in the information space can be grouped in multiple categories at the same time, which is the cardinal feature of the category-based structural framework.

The most popular application to embrace the category approach was the original Agenda[171], 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[42] 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. (See Figure 11.) Haystack[9] and Chandler[246] are other information management tools which use categorization in important ways. William Jones' Memory Extender[169] 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.

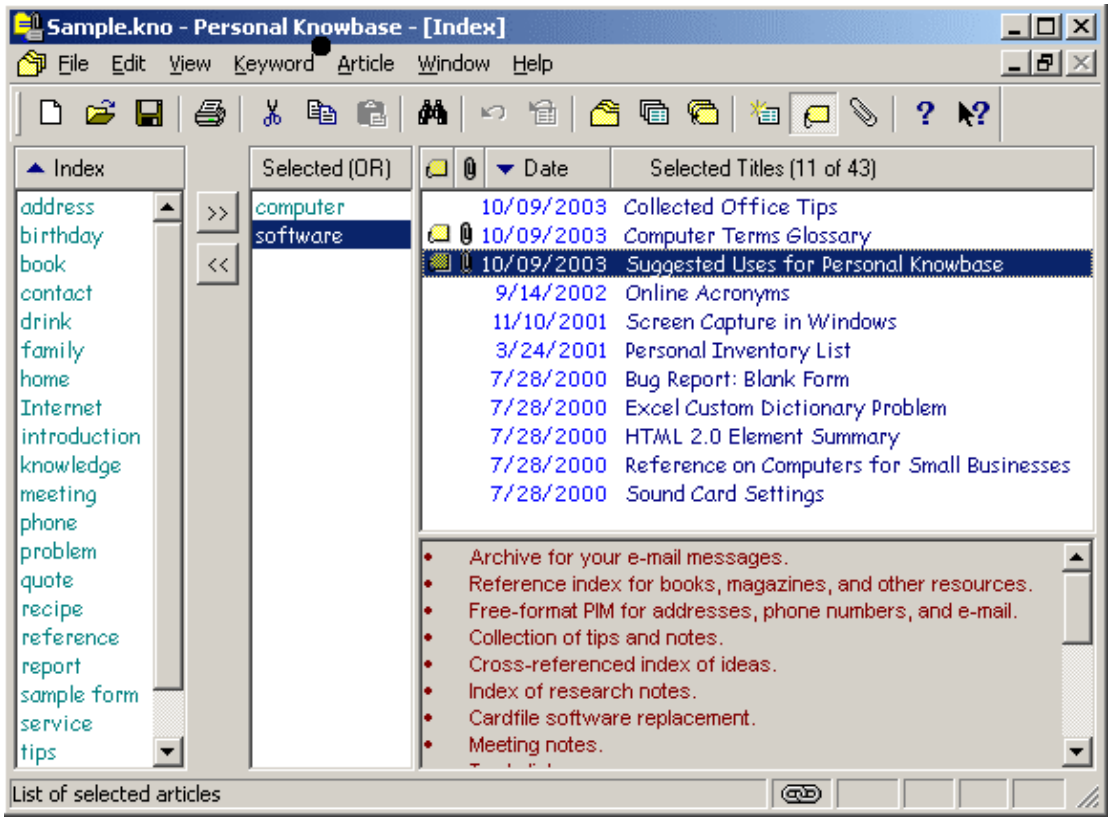

Figure 11. Personal Knowbase, a note-taking system based on the category structural framework. Multiple customized user keywords (listed in far left pane) can be assigned to each note (which is comprised of a small text document and a title.) These keywords can be combined with boolean operations in order to retrieve notes. [42]

The role of transclusion

Before taking a brief look at a few other structural frameworks that have been proposed, I wish to draw attention to the important property of "transclusion." This term, first coined by Ted Nelson[228], 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 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.

In this context, 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 together 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 one were 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 a 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.

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[171], 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[316] and Halasz[146] 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.[7] Zoot[344] and Hog Bay[157]

[7] I 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.

are tree-based commercial products that also let the user transclude items into multiple containers.

Boxer[100] 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.[39] 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.

These issues are solved by Compendium[77, 282], the most thorough known implementation of transclusion 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, then, transclusion is a property that can be advantageously combined with any of the structural frameworks above. 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.

Alternative frameworks

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

Chronological. Yale University's Lifestreams project[121] used timestamps as the principal means of organization and retrieval of personal documents. (See Figure 12.) 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…

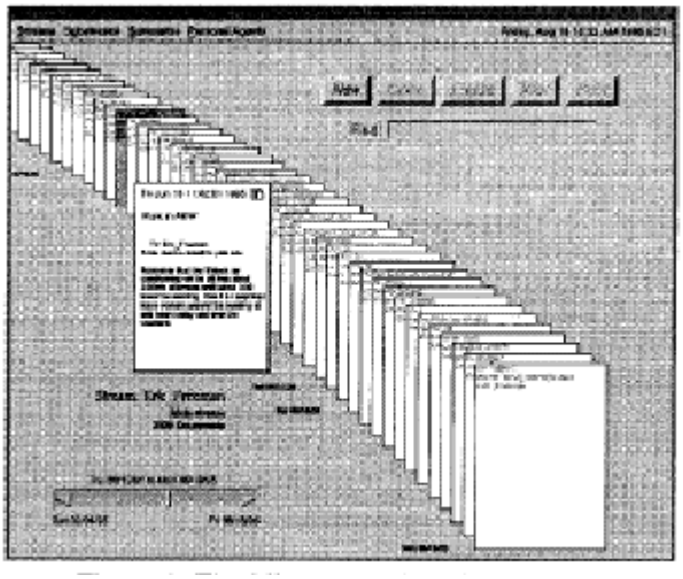

Figure 12. Lifestreams, an exclusively chronologically-based information management system. [121]

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."[126] 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[124], based on the Franklin-Covey planner system, is also chronologically modeled, and a number of

products based on other data models (e.g., [171], [70]) offer chronological indexing in addition to their core paradigm.

Aquanet. Though advertised as a hypertext system, Marshall et. al's Aquanet[208] 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. (See Figure 13.) 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.[8]

[8] 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.

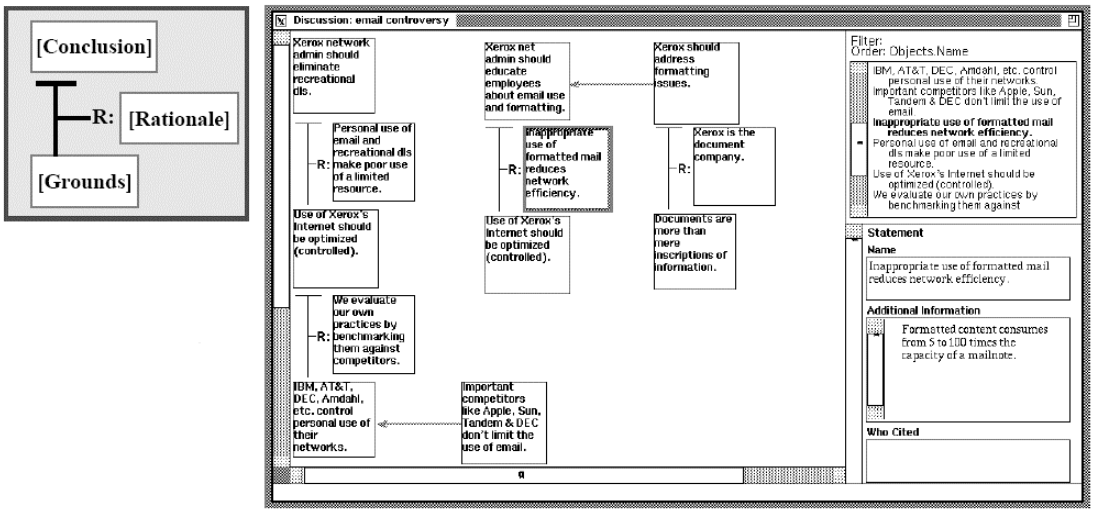

Figure 13. The Aquanet data model. The user can define "relations," or templates for semantic n-ary based links between objects. (See the upper-left corner for an example.) Knowledge elements can then be linked together with these relations into structures of arbitrary complexity (see right-hand side of the diagram.)[208]

Zigzag. Finally, it is worth mentioning the data model of Zigzag[230], 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.

Knowledge elements

Having surveyed the various options for structuring knowledge elements together in a PKB, it is now time to consider 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:

Word/phrase/concept. Most systems engineered specifically 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.[28] Note that if a user were to break up all of the knowledge they intended to capture into a semantic network, individual words and phrases are 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 in the system.

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[145, 146], Hypercard[141], TreePad[125]) or attached to elements as separate, supplementary pages (Agenda[171], Zoot[344], Hog Bay[157]). 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.

Links to the objective space. I have mentioned 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:

1. The knowledge element actually *represents* an item in the objective space. This is the case for document management systems (WebTop[338], MyLifeBits[134], Haystack[9]), integrated search facilities (NaviQue[129], CYCLADES[266]), and VIKI/VKB([209], [289]). Tinderbox[39] 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" so as to receive constant updates from the Web.

2. 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[127], PersonalBrain[311], Inspiration[162]). VNS[56], which allows users to point to a community notebook page from within their personal notebook, offers similar functionality.

3. 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[222], Hunter-Gatherer[280], and YellowPen[342] (see Figure 14) 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[97] and MyBase's WebCollect plug-in[337] 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[70] and Sticky Brain[68] have been heavily optimized towards grabbing select bits of information from other applications and bringing them into the PKB without disturbing the user's workflow.

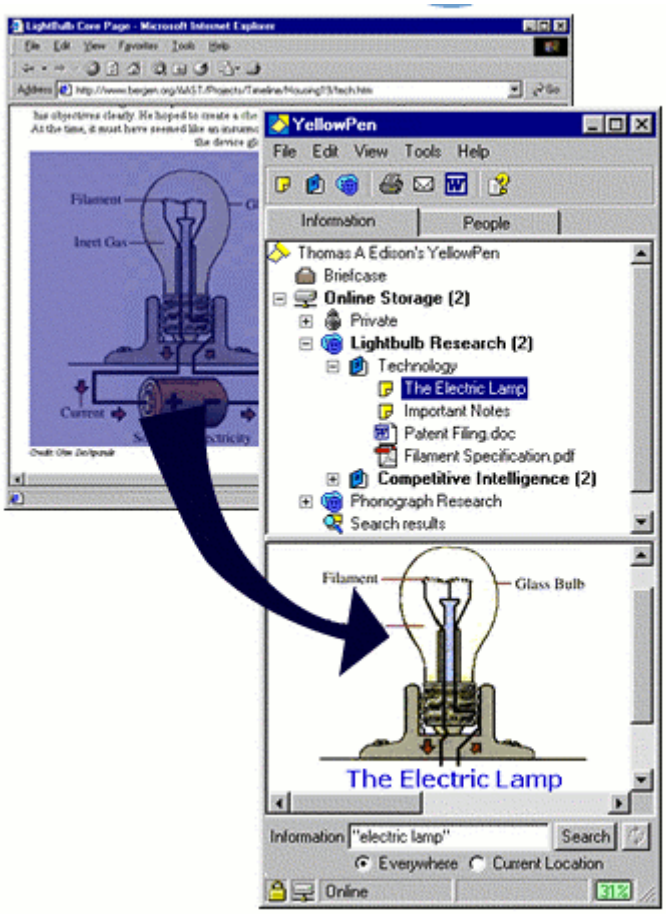

Figure 14. YellowPen, which allows snippets of textual or graphical content to be captured from the web and repurposed in the user's own personal (hierarchical) structure. [342]

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[208], 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[100]'s paradigm is similar. Tinderbox is a graph-based tool that supports hierarchical composite structures, and Compendium[77] 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.

Schemata

Finally, let us consider the notion of "schema" in various systems' data models. By schema I 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 discussed later on. I consider first schemata for knowledge elements, and then for links.

Schemata for knowledge elements

Types, and related schemata

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.

Notice the distinction 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" over time, but its core type of "company" would probably be constant.) 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 I will describe later. Note that some systems that do not allow typing offer the ability to approximate this function through categories. (e.g., OneNote[217], Mind Manager[221]).

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[219]). 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[63] 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[131, 132] 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[20] 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[289], 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 discussed 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[145] 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, I refer the reader to Klas, et al[176] and Hatzopoulos[150].

Other forms of schemata

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 I 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. I have already covered the category data model, and seen that systems like Agenda use keywords for the latter purpose. 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.[70] OneNote[217], Mind Manager[221], 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[13] 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[231] and OmniOutliner[244]. 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[68]).

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[296] and VisualMind[219] let the user freely choose each element's icon from a variety of graphics, while KMap[131, 132] ties the icon directly to its underlying type. Other graphical aspects that can be modified include color (VIKI[209]), the set of attributes shown in a particular context (VKB[289]), and the spatial positioning of objects in a relation (Aquanet[208]).

Schemata 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[60] and Inspiration[162].

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[311], 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[79], DecisionExplorer[28]), 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[9] 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[96] and Intermedia[133] 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[147] covers this issue in detail. For PKB purposes, this seems to be most relevant as regards links to the objective space, as 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 I 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.

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 I have defined it above.

In this section I examine some of the architectural choices that are prevalent among candidate systems, and comment upon their ramifications.

File-based

The vast majority of solutions discussed in this chapter use a simple storage mechanism based on flat files in a filesystem. This is true of virtually all of the mind mapping tools (e.g., Mind Manager[221]), concept mapping tools (e.g., Cmap[60], Axon[45], Inspiration[162]), outliners (e.g., TreePad[125], OmniOutliner[244]), and note-taking tools (e.g., OneNote[217], Hog Bay[157], Zoot[344]), and even a number of hypertext tools (e.g., NoteCards[146], Hypercard[141], Tinderbox[39]). 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. Only knowledge elements that reside in the same file can be meaningfully connected to one another; those in other files are outside its scope.

This is a very serious constraint which I will argue 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 that Betty, a user of such a system, were to store the details of a particular work procedure in her knowledge file for the XYZ Project she

is working on at her place of business. Then she could not link this procedure to anything in the ABC Project, since that knowledge is in a separate file. Nor could she associate it with Jeff (who once discussed the procedure with her at a cocktail party), since he is a friend of hers, and knowledge about him is stored in Betty's "social" knowledge file. Nor can she refer to it in her notes from last summer's seminar (during which a speaker suggested an alternative to such a procedure), since those notes are stored in her "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.

I have dwelt on this point at some length 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.

Database-based

A small number of systems 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 they 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 they remember 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 ([16], [195], [81]). The database architecture enables a PKB to truly form an integrated knowledge *base*, and contain the full range of relationships.

Agenda[171] and gIBIS[79] were two early tools that subsumed a database backend in their architecture. More recently, the MyLifeBits project[134] uses Microsoft SQL Server as its storage layer, and Compendium[77] interfaces with the open source MySQL database. A few note-taking applications such as StickyBrain[68] 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 approaches.

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.[96] 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 ([320], [283]) 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[337] 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[9] 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[101] exhibits similar features.

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[132] 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[280], Cartagio[222], and Notestar[15] 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.

Handheld devices

Lastly, I 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![105] 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[65], which merely supplements a feature-rich desktop information

management tool (Microsoft Outlook) by providing access to that information on the mobile device. InfoSelect[215], a tree-based note-taking application, also offers a mobile product. There are no prominent handheld companion applications for any bona fide knowledge representation tools, however. Idea Pad[234], 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, satellite handheld software will presumably emerge to support them.

Supplementary features

Before I conclude with an overall analysis of the systems presented in this chapter, I 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.

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. I have already mentioned the spatial parser of VIKI/VKB([209], [289]) 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[28] 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[179] and Knowledge Manager[160] 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[63] had an active inference engine built in to assist users in drawing logical conclusions from the knowledge assertions they have expressed.

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[171] and DEVONThink[97] 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[338], CYCLADES[266] and Horse[89] discover similarity relationships in much the same way. Haystack[9] and Presto[101] both feature background services that examine content and auto-annotate it with metadata for future retrieval.

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.[60] This is designed to assist the user in brainstorming and

to help them flesh out an "incomplete" map. Similarly, NovaMind[237], 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[131] 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 probe 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. In the present context, 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.

Auto source capture. I 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[217] and YellowPen[342] achieve this with the Internet Explorer browser. DEVONThink provides its own browser as part of its integrated environment, and so has access to the URL of every page viewed and stored. StickyBrain[68] 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., [280])

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[100] 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[13], Axon[45], and Omnigraffle[243] 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.

Search services. As mentioned earlier, several tools effectively integrate a search environment with a personal information management workspace (e.g., [83], [155], [54], [129]). 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) achieve a similar result.

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[60], KMap, and even the original Augment/NLS[107] 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. One would expect it to become much more widespread after personal knowledge bases become the norm.

Three-dimensional rendering. All of the spatial tools I 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[45] and HeadCase[41] 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.

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., [160], [127], [289], [39]). 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, one can only hope that as PKBs become embraced by the public, industry consortia would arise to agree upon standards for knowledge interchange between systems.

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[94] was designed to support this: its stated purpose was to use knowledge models as a communication vehicle from educators to teachers. RecallPlus[111] 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.[160]

Semantic Web support. The Semantic Web initiative is a collaborative effort sponsored by the World Wide Web Consortium[323] to extend today's Web by adding machine processible information.[37] 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)[182] 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)[36]. 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[123] which defines a standard schema for describing relationships between people, and the Dublin Core Metadata Initiative[102] 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[104]. 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[25] adds similar Semantic Web support to concept maps.

Compendium has also been used for a similar purpose[282], 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[159] 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 if accepted on a global scale, the potential it would bring for integrating private and public knowledge would be a tremendous asset to PKBs.

Summary and critique

The problem of knowledge management is well-appreciated, as the sheer number of attempted solutions attests. The systems presented in this chapter are intended for a variety of overlapping purposes – some to help formulate abstract knowledge ([127], [162]), others to store it safely ([215], [97]), still others to repurposing objective snippets ([209], [280]) – but they all share a common, overarching goal: helping the user cope with the multitudinous bits of knowledge they encounter and generate.

And yet for John – the fictional, amateur philosopher presented in the introduction – the alternatives are unfortunately slim. He wanted a way to archive his memories as he conceived them, potentially materializing his entire subjective view of the world so that it did not fade. And although a myriad of design efforts have

produced some very good tools, none of them are a very good fit for *his* problem. Let us look broadly at the choices here presented, and consider how they would apply to John's situation:

1. Document management systems (like [134], [9], and [169]) would allow him to better organize and retrieve the *documents* he encounters, but they do nothing to help him store the knowledge he gleans *from* those documents.
2. Text-based tools (like [244], [217], [20]) let him record subjective knowledge, but he would have to do so with traditional English sentences. These cannot capture interrelated concepts very well, since the noun phrases that describe concepts would invariably be inconsistent, besides being buried in the middle of paragraphs. In any case, English sentences do not adequately capture the fluid, nonlinear nature of his mental impression. Additionally, the tools in this category most often force the text to be structured in a tree hierarchy, which is not flexible enough to capture arbitrary relationships.
3. The category model (as in [344], [42]) improves upon the tree-centric approach of most of these text-based tools, but it is still insufficient to encompass John's mental impressions. It allows him to represent a thought like "The Epicureans, Stoics, and Cynics were all Hellenistic philosophers," but not "Hellenistic philosophy followed Plato and Aristotle" or "The Sceptics influenced postmodernism, which influenced Bobby's older sister." Mental impressions like these demand a nonlinear set of arbitrary relationships, preferably with spatial layout.

4. Mind mapping tools (like [237], [241]) give a partial solution to this, since concepts can be arranged spatially and related in ways other than categories. But their tree model is far too restrictive, as we have seen.
5. Concept mapping tools (like [60], [39]) go a step further and permit arbitrary relationships between items. This suffices on a small scale, since John could use such a tool to sketch the ideas and the relationships he perceives. The trouble is that these tools cannot serve as a true PKB, since they are file-based and offer limited or no transclusion. Therefore, the files that they produce are isolated subsets of knowledge that cannot refer to each other. A label in a box on one diagram has no direct connection to an identical label in another box on another diagram: to the tool, they are just boxes, not coherent concepts that can appear in multiple contexts. And to even retrieve an individual concept map, the user is forced to navigate the same path-and-filename filesystem paradigm that so frustrates users today.

The best candidates available today for John's application appear to be PersonalBrain[311], NoteCards[146], and Compendium[77]. Yet each of these also presents difficulties:

- Although PersonalBrain supports an arbitrary graph, and implements its own database and search facility capable of supporting many thousands of items, it offers no true transclusion. Each "thought" can be connected to any number of others, and these connections labeled, but a given thought always appears in the same context, showing all of its relationships. Thus there is no way to

construct a picture of a mental impression that combines four or five concepts in a particular relation to each other. One cannot place John Wilkes Booth and Abraham Lincoln in a view, for example, and say anything about them without simultaneously bringing in all the *other* knowledge one may have acquired about Lincoln. Partly because of this, PersonalBrain does not give the user spatial control over the display: rather, it automatically lays out all the associated thoughts to the currently focused thought. Finally, PersonalBrain requires a fully-connected graph, as explained above, which makes it awkward at best to enter knowledge from a new domain. The user must either create meaningful connections to the trunk at the time the knowledge is entered, or else put the new knowledge into a separate file, which promotes isolation and partitioning.

- The NoteCards system (and its precedent, TEXTNET) had much to recommend it. Yet some unsolved problems remain that would make it less than ideal for John's particular application. For one, its transclusion was somewhat incomplete. Although multiple "browser cards" could be created that referred to the same "notecard," which effectively placed the notecard in multiple contexts, there was apparently no easy way to see all of the browser cards that referred to a particular notecard. Hence the containment could only be traversed (and seen) in one direction: from owner to owned. This restriction prevents a user from navigating from context to context via shared ideas, as the human mind habitually does. Equally problematic is the strict division of elements into notecards and browser cards, forcing the user to

choose at each stage between content and relationships.[9] More subtly, the notecards themselves were intended to function as coherent chunks of text or graphics, similar to a physical 3x5 card. Hence they would not normally represent conceptual entities, but rather paragraphs. One could imagine a NoteCard user operating *only* with browser cards, and except for the aforementioned transclusion limitation this would give an approximate solution to John's needs. Finally, NoteCards was developed before public file exchange was commonplace (as it is today with the Web), and so there was no ability to incorporate objective snippets into the knowledge base.

- Of all the solutions here presented, Compendium boasts a feature set closest to John's needs. It supports full transclusion, and allows all the contexts of a particular node to be viewed and accessed. It features a robust database-driven backend, and permits arbitrary spatial positioning of elements. However, it has three main drawbacks. First, it is unfortunately hardwired for a particular domain (decision rationale expressed in group meetings) and all knowledge elements must be declared as conforming to one of a few types specific to that domain ("argument," "decision," "question," etc.) It is thus unsuitable for general-purpose use. Second, it requires a fully-connected graph, as does PersonalBrain, and this gives it the same limitations described above. And third, it offers no facility at all for bridging the subjective-

[9] One could not, for instance, create a notecard to contain some textual information, and then decide to place other notecards on it in a particular configuration. Linking from within one notecard to another was permitted (similar to today's Web hyperlinks, with link types added), but not displaying particular relationships between notecards on another notecard. The latter could only be accomplished on a browser card, which could then not hold any content. The basic limitation here is that notecards were entirely separate from browser cards, and the features of the two could not be easily combined.

objective divide (even simple, hardcoded references to objective sources are not supported, let alone automatic drag and drop source capture.)

There is therefore no true solution to the kind of personal knowledge base that John desires. This also implies, of course, that there have been no studies of how such a system would operate in practice. The purpose of this thesis was to identify the design goals for such a system, implement a workable prototype, and then deploy it to real-world users who would use it for real-world tasks. Only then would it become clear how it would be used, whether it would be valuable, and what its benefits and liabilities would be. In the next chapter, I will discuss what was learned about these matters.

CHAPTER 4

POPCORN: A PROTOTYPE PERSONAL KNOWLEDGE BASE

Design goals

In considering how to best architect an effective personal knowledge base, I identified and pursued a number of design goals. Together, these goals form an analysis of the task of personal knowledge management.

1. Recording knowledge should be quick and painless. Building knowledge from information is difficult enough, and requires focusing the powers of concentration. Hence it is imperative that the tool itself not hinder this task by presenting a cumbersome or non-intuitive interface, forcing the user to navigate numerous options, etc.

2. Returning to previously recorded knowledge should be quick and painless. This is especially true with regards to rapidly changing contexts. The human mind has the remarkable ability to abruptly transition from one domain to another. One can be hard at work on a project, for example, and then suddenly interrupt their thoughts when the phone rings to discuss a completely different topic. After a brief conversation, it is relatively easy to return to the previous task. A personal knowledge base should support the same pattern. From deep inside one domain, it should not be difficult to wrestle the tool away from its surroundings and direct it to another. Rather, context switches should be rapid and facile.

3. Reorganizing knowledge should be easy. As we learn more about a new domain, our perceptions about it change. The ideal tool should encourage rather than inhibit the restructuring of our knowledge to reflect our current understanding. Indeed, people are somewhat reluctant to abandon their mental organization patterns, even when these prove to be suboptimal[318]; the last thing a tool should do is further enforce this rigidity.

4. Users should be able to express knowledge both formally and informally. We sometimes wish to take the time to specify knowledge very meticulously, especially in a difficult domain where we had to work hard to discern its precise meaning. Other times, however, being forced to spell out details would only get in our way; we do not know, care, or want to take the time to be so strict. Both cases are important, and so an effective PKB should allow the user to operate at any point along this continuum.

Similarly, for maximum effectiveness a PKB should also allow a user to operate freely along the information-knowledge continuum. Fully transforming information into knowledge is a time-consuming process, and users may sometimes wish to simply record the information either in its raw form, or only partially digested. Then later they could return to it and complete the difficult work of expressing it as a knowledge structure.

5. Public content should be easy to assimilate. One's private knowledge is primarily *comprised of* bits and pieces of public information. The facts and opinions that we read are the raw material for the learning process; it is only our own organization of those facts that represents our personal point of view. Since this is

true, a PKB should make it easy to incorporate snippets of information from the public realm (say, the Web) into the knowledge base, and to maintain the links between them so they can be revisited.

6. The tool should work naturally with human memory. If the goal is to archive memories and return to them later, then the tool's data model and user interface paradigm must take into consideration the processes humans naturally undergo, in order to exploit strengths and fortify weaknesses. This is an immense task, of course, given the breadth of psychological findings (and controversies) about the nature of memory. But the following seven points seemed worth incorporating:

Semantic network. Memories are often encoded in something very like a concept map, or semantic network[10].[339] This is by no means a unanimous conclusion among psychologists, but convincing evidence has been found to suggest that the mind stores at least some knowledge in the form of key ideas and their relationships. ([16], ch.4; [174], p.267; [298], p.39, pp.76-77; [166], pp.249-251.) A PKB should therefore allow users to express knowledge in this form.

Categories. It should also be easy to organize knowledge based on sets of categories. Both categories and associations are fundamental organizational constructs, and neither is explainable in terms of the other. ([174], pp.243-253.)

Unlimited total memory, limited working memory. There are at least two important components to human memory: our working, or short-term memory (STM), and our long-term memory (LTM.) The latter is our permanent storage

[10] There are subtle differences between concept maps and semantic networks (as well as the related notions of propositional networks, frame-based representations, and object-oriented models.) But for our purposes, we consider them equivalent. They all ultimately represent knowledge as a graph: a set of named concepts connected by typed relationships.

repository, containing everything we have ever learned, even if some of those items are difficult or impossible to retrieve. Our STM contains what our mind is actively thinking about: in fact, cognitive processes cannot function on knowledge that is not in STM. ([16], ch.6.) One interesting relation between the two is that our STM normally contains a (very small) subset of items *from* our LTM. It is as though knowledge from our LTM has to be "loaded" into our STM in order for our mind to operate on it. The size of the STM is extremely limited: a famous finding estimated its capacity as from five to nine items of information, depending on the individual and their circumstances.[218] Interestingly, however, these "items" were often found to be multifaceted, as if a great deal of underlying complexity could be collapsed into a single element and held compactly in the STM.

Following these findings, then, it seems prudent for a PKB to store unlimited amounts of information (like the human LTM), but to allow the user to work with only a coherent, manageable subset of items at any one time, each of which can hide unlimited complexity (as the human STM does.) This should hopefully interface well with the way the human mind naturally stores, retrieves, and works with information. Users will not be overwhelmed by encountering too much information at a time, and yet the items they do work with will be implicitly linked to many others by virtue of their presence in LTM.

Traversal between contexts. The human mind is associative: one idea reminds us of another, and we often follow a meandering train of thought flowing across many contexts. This idea has been pinned down experimentally and denoted as "spreading activation"[75]: loading one item into STM partially triggers the other items we

associate with it and renders them more readily available. A PKB, it seems, could explicitly support this tendency to great advantage. First, the multiple contexts that a particular item appears in could all be readily available when that item is considered. Second, it could be made easy to traverse from context to context via the ideas that are shared between them. And thirdly, the general user experience could be one of fluidity: as the user navigates from idea to related idea, the display would be continuous and free-flowing, not disjoint or jarring. All of this would allow the user to traverse their memories along the paths that they would naturally travel.

Multiple retrieval paths. Remembering is a reconstructive process, and we often make use of multiple, sometimes redundant retrieval paths in order to locate and bring information back into our STM. ([16], p.194.) Hence a PKB should encourage the creation of many multiple associative connections to any given knowledge element. It should not, for instance, lodge each element into only one "correct" place with a single access path, as this may inhibit retrieval later on.

Extension of prior knowledge. A critical part of the learning process is relating new information to the background knowledge that we already understand.[24, 235] Indeed, some claim that this is the *only* way that learning is possible: we merely interpret new knowledge in terms of the old.[23] Whether or not this is always the case, it seems that an effective PKB interface should encourage users to extend their existing knowledge with new findings, and to incorporate old elements into the new knowledge structures they create. This allows one's knowledge representation to grow seamlessly over time instead of being a discontinuous sequence of isolated, perhaps incompatible snapshots.

Visio-spatial support. Finally, evidence suggests that our minds make use of visio-spatial cues to help us process information[19], and that providing users with spatial flexibility can be a great aid to organizing information[209]. A basic design premise, therefore, was to take advantage of the inherently spatial aspects of human cognition by giving users control over a two-dimensional canvas.

The Popcorn data model

The following data model attempts to support these goals, and forms the underlying structure of a Popcorn knowledge base. In Popcorn, the basic building block of knowledge is the *kernel*: an entity corresponding to a real-world concept in the user's mind. A kernel can be viewed "from the outside" as simply a named item, and it can also be expanded to show its inner contents, which is called the *kernel view*. Each kernel view is a two-dimensional canvas on which can be placed other kernels and also free-text snippets of varying size called *notes*. Notes are atomic in the sense that they contain only verbatim text and cannot be further expanded; they also have local scope in the sense that no note is visible anywhere except on the kernel view that owns it. Be careful to observe, however, that this is *not* the case with kernels. Many kernels can appear on a kernel view, but this does not preclude their simultaneous appearance elsewhere (ie., on other kernel views.) Indeed, all kernels have global scope in a Popcorn knowledge base, and can appear in any spatial position on any number of other kernel views. At any point in time, exactly one kernel view is the "active kernel"; that is, the one that the user is currently focusing on.

It is worth remarking that although kernels can be named (and usually are) they actually can be left unnamed if the user desires. This sometimes occurs when a kernel is used only for its kernel view; ie., it is always viewed from the inside, never from the outside. A user can simply create a new, unnamed kernel view and represent some snapshot of knowledge within it. In this case the kernel view can only be retrieved by means of its contained items, not by its name (since it has none.)

Kernels and notes can have named *relationships* with each other. Each pair of objects (kernels and notes are the two kinds of Popcorn "objects") can optionally have a single relationship between them. This can be named with a free-text tag describing the nature of the relationship, though it need not be. It can also be specified as navigable in one or both directions. Relationships between two kernels are global, just as the kernels themselves are global: they are not restricted to any kernel view. In other words, if a kernel "Antony" has a relationship of type "married" to another kernel named "Cleopatra," then that relationship holds (and will automatically appear) on *any* kernel view that contains both Antony and Cleopatra. Relationships between a kernel and a note, or between two notes, on the other hand, are visible only on a single kernel view (since the notes themselves are not visible outside of it.)

Finally, notes can have *source* information associated with them. This is useful for snippets of text that were excerpted from information sources, so that the original source of the excerpt can be maintained. (Popcorn automatically captures the source URL of snippets that were dragged into it from a Web browser, as explained below.)

Formally, if Σ is the set of alphanumeric characters, we can denote a Popcorn knowledge base as a triple <*K, R, a*> where:

K is the set of kernels (defined below),
$R \subset \{K \times K \times types \times f \times b\}$ is the set of relationships between kernels, where $types = \Sigma^*$ is the set of possible relationship types (strings), and $f, b \in \{0,1\}$ indicate whether the relationship is navigable in the forward and/or backward directions, and
$a \in K$ is a special kernel called the "active kernel."

Each kernel $k \in K$ is defined as a 5-tuple <*n, C, N, P, L*>, where:

$n \in \Sigma^*$ is the name of the kernel,
$C \subset K - \{k\}$ is the set of "child kernels" that appear on *k*'s kernel view
N is the set of notes (defined below),
$P : C \to \{(x, y, w, h) \mid x, y, w, h \in \Re \wedge 0 \le x, y, w, h \le 1\}$ is a function that specifies the spatial position (*x* and *y*) and size (*w*idth and *h*eight) of each child kernel on k's kernel view (as a fraction of the kernel view's overall width and height), and
$L \subset \{(C \cup N) \times (C \cup N) \times types \times f \times b\}$ is the set of "local" relationships between notes and kernels, or notes and other notes.

Finally, each note $n \in N$ is defined as a triple <*c, s,* (*x,y,w,h*)>, where:

$c \in \Sigma^*$ is the *content* of the note,
$s \in \Sigma^*$ describes the *source* of the note (often a URL), and
x, y, w, and *h* specify the position and size of the note on the kernel view that owns it.

Notice that there are fundamentally two ways that kernels can be related to each other. One is explicitly through *relationships*: the user can create a relationship from "Brutus" to "Caesar" of type "conspired against." The other is implicitly through *containment*: the user can specify that the kernel "Bach" appears *on* the view for the kernel "Baroque composers." Both techniques are available, and it is interesting to observe how different individuals make use of them (see chapter 5.)

It is also worth emphasizing that although Popcorn includes the notion of containment (one kernel can appear "on" another), the kernels in a knowledge base do *not* form a tree hierarchy. This is because placing a kernel A on a kernel view B does not prevent A from also appearing on kernel view C. "Bach" may well appear on both "Baroque composers" and on "Influences on Mozart," as well as others. It is also very possible (and common) for cycles to exist in the knowledge base: A may appear on B *and* B on A, for instance. For example, consider a professor teaching a small graduate seminar. When her mind is on the seminar, she thinks of each of the students who are enrolled in it. Thus she might have a kernel called "CSCI 6800 seminar" on whose view appears a kernel called "Bill Smith," representing one of the students. When she turns her mind specifically to Bill Smith himself, however (perhaps to recall her experiences with him, write a letter for him, or consider him for work on a future project), she naturally reflects on all of the things she knows about him. Since he was present in her seminar class, one of the kernels on Bill Smith's view is likely to be "CSCI 6800 seminar." So a cycle appears: "Bill Smith" appears on "CSCI 6800 seminar," and vice versa, since considering each topic naturally conjures up the other. The professor is actually encoding two related but distinct facts: "Bill was one of the students in my seminar," and "the seminar was one of the ways I got to know Bill."

This does not present a problem. The strategy here is to allow the user to work with small amounts of information at a time, just as the limitations of their working memory force them to. Each kernel view thus contains a small subset of items from their knowledge base, chosen and arranged according to some particular

purpose. A kernel view can be seen as a kind of "snapshot" of working memory: a record of what the user was thinking about at some point in time.

Kernels have global scope because the ideas in our minds have global scope. True, it may sometimes be convenient to model knowledge as a hierarchy: "Joe DiMaggio" might be contained within "New York Yankees," which in turn is contained under "American League" and "Major League Baseball." But there is certainly nothing preventing our minds from seeing a modern ballplayer and remarking, "Barry Bonds reminds me of Joe DiMaggio," or learning that "Marilyn Monroe was married to Joe DiMaggio." Here we have taken an idea embedded deep within a hierarchy (DiMaggio) and placed it in a completely different context *outside* its immediate container. It would be unfortunate indeed if DiMaggio's scope was restricted to only the "New York Yankees" kernel view, preventing his appearance elsewhere. One never faces this trouble in real life: we are never unable to repurpose an individual fact because we once classified it a certain way. On the contrary, our minds can maintain alternate, complementary organizations of the same concepts, depending on the relationships we perceive. Popcorn was designed explicitly to support this phenomenon.

I will sometimes refer to the kernels whose views contain a particular kernel *A* as the *parents* of *A*. "Baroque composers" and "Influences on Mozart," then, would both be *parents* of the "Bach" kernel. Formally, the parents of *A* are the set of kernels $\{k \mid A \in k(C)\}$. Popcorn prevents a kernel from being its *own* parent, but again note that any two kernels may very well be parents (and children) of each other.

The Popcorn interface

Viewing and recording knowledge

Popcorn's user interface follows this data model exactly. (See Figure 15.) The active kernel view appears in the application's *viewport*, a rectangular area that normally occupies most of the bottom-left corner of the window. The child kernels and notes of this active kernel can be positioned arbitrarily within the viewport. The top of the window contains the *parents panel*: a scrollable list of thumbnails depicting each of the active kernel's parents. The bottom-right corner of the window contains a special kernel called the "sandbox." It is designed as a convenient placeholder or scratch pad for the user to work with as they manipulate items in the viewport. Since it is always visible, it can also be used to hold "bookmarks" to commonly used kernels.

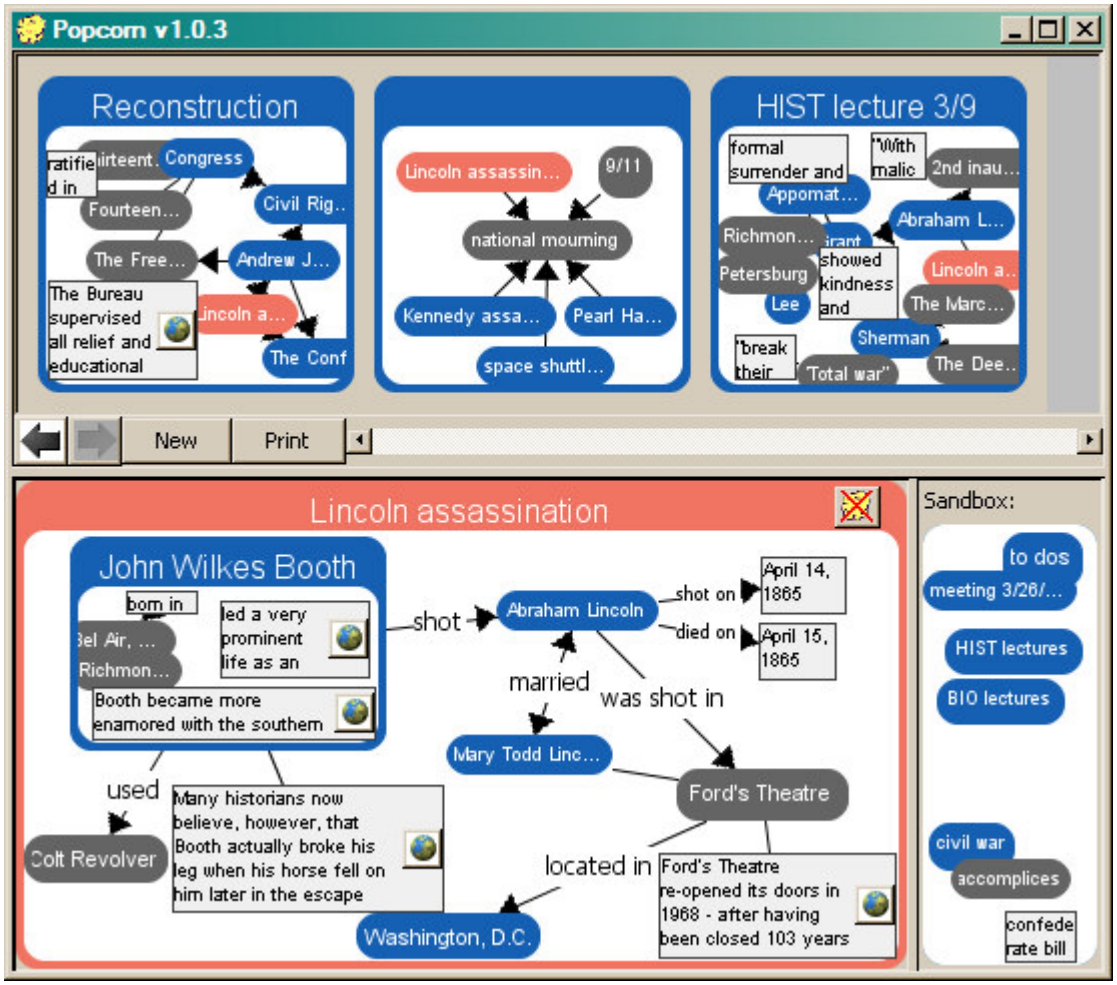

Figure 15. The Popcorn user interface. The view of the active kernel (here called "Lincoln assassination") occupies the viewport, while all of its parents (other kernel views in which it appears) are displayed immediately above it. The active kernel is displayed in a highlighted color wherever it appears, in order to draw attention to it. Kernels within a kernel view may be expanded, as is the case with the "John Wilkes Booth" kernel here. Popcorn's layout engine attempts to show as much meaningful information for each component as possible, subject to the space available, while still preserving the spatial relationships between components. Note, for instance, that in the "Reconstruction" kernel (upper left), the kernel named "The Freedman's Bureau" has been shortened to "The Free…", and that in many places only the first few words of each note are visible. The user can hover over notes or kernels with the mouse pointer to summon a popup "tool tip" with the complete contents.

The active kernel itself is shown in a different color (salmon) than other kernels wherever it appears in a parent view. This allows the user to see at a glance all of the contexts in which a particular kernel appears. Other kernels appear in one of two colors: blue, if they have inner contents (ie., one or more child kernels or

notes), and grey if they do not. This helps users identify which kernels contain further information so that they do not waste time navigating to kernels only to find them empty.

A kernel can also be expanded while inside the viewport so that its inner contents are revealed in miniature. When the mouse pointer hovers over a kernel, two buttons appear next to it: one to expand/collapse it, and the other to remove it from the view. The user may also resize an expanded kernel to see more or less detail. This allows users to "peek inside" the kernels in the viewport without changing contexts.

It is important to realize that removing a kernel from a particular view does *not* delete it from the database. This is because kernels represent generic ideas and have global scope within the knowledge base. A given kernel may appear on one view, many views, or no views at all; in any of these cases, it is still a bona fide entity whose existence is not tied to its appearance in any particular context. A kernel can only be permanently deleted by means of a special, explicit operation: namely, navigating to that kernel (making its view the active view), and then pressing the delete button in the upper-right-hand corner of the viewport. (This button looks like a popcorn kernel with a red "X" through it.) If the user presses this button, the system presents a dialog box explaining the consequences of the action, and asking the user to confirm it (see Figure 16.) These consequences include altering each of the kernel's parent views, and permanently deleting all of its notes (since they only have scope within the kernel.) This operation does *not* delete its child kernels, of course, since they, too, are generic entities with global scope, and may (or may not) appear

elsewhere. This "permanent delete" operation was created as a concession to the fact that users may wish to undo mistakes, and may wish to periodically "clean up" their knowledge base by purging old data.

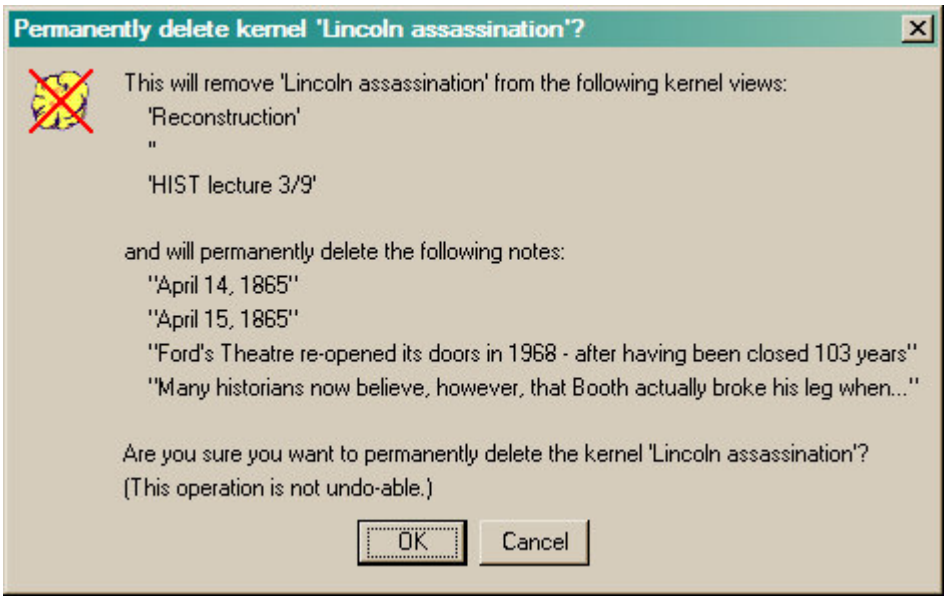

Figure 16. Confirmation dialog box for permanent kernel deletion.

Kernels and notes can be moved about the viewport using the familiar drag and drop paradigm. Creating a relationship is accomplished by positioning the mouse pointer near the *edge* of an object (at which point the cursor changes to a crosshair) and then dragging to the related object. The arrowheads on either side of the line (indicating navigability) can be toggled on and off by clicking on them. New kernels and notes can be added to the view by double-clicking or right-double-clicking, respectively.

When the user double-clicks in the background to add a kernel to the view, they may type a name for it. At this point an autocomplete search pane appears immediately below the kernel name, offering to match existing kernel names – or the contents of notes within them – as the user types. (This list of kernel names is a

smaller version of the "quicksearch" panel shown in Figure 17.) Pressing the return key selects a kernel within this list. This allows users to place an existing kernel on the view without typing its entire name, and also encourages the re-use of kernels in multiple contexts. It is also intended to alert users to the presence of kernels they have previously created: if the user attempts to type the name of a new kernel, and it matches that of a previous kernel, this will be readily apparent.

The same name can be given to multiple kernels, however. This relieves the user's burden of having to generate unique names, since humans often use the same identifier in different contexts to refer to different entities. Creating a new kernel with the same name as an existing one must be a deliberate operation on the part of the user. When they type the name, the autocomplete search pane will, by default, highlight the name of the existing kernel, so that if the user simply presses the return key, the existing kernel will be retrieved (and placed on the current view) rather than a new one being created. To create a new kernel, the "new…" option at the top of the list must be intentionally chosen (by pressing the up arrow), which again alerts the user to the fact that they are about to create a duplicately-named kernel. The two (or more) kernels with duplicate names can be distinguished during retrieval as explained in the next section.

Finally, if any kernel appears in the search results pane that is *already* present on the view, its listing will appear "grayed out" and be unselectable. This is because according to the data model, each kernel can only appear once on a given view. Showing the search result anyway should avoid disorienting the user (by not hiding a

result that they know should be there), and they gray color should alert them that the kernel is already present somewhere on the current view.

Note that all of the basic operations – adding and changing kernels, notes, and relationships – are accomplished without any menus, toolbars, or palettes. This is possible since Popcorn permits only a small number of operations, and it speeds up the interface experience considerably. Also, note that all of the operations here described are applicable not only to the viewport, but to *any* expanded kernel, including the ones in the parents panel or sandbox. A user can, for instance, expand a kernel within the viewport and then manipulate *its* children by dragging and dropping them, or double-click in the background of a parent kernel view to create a new child of that parent. All expanded kernels thus behave identically, promoting consistency throughout the interface.

Retrieving knowledge

Browsing through the knowledge base can be done in two ways: navigation, and search. The user can make any visible kernel the active kernel by simply double-clicking on its name. This is true for any of the parents as well as the children of the currently active kernel or sandbox. Double-clicking initiates a "zoom" animation sequence that smoothly brings the selected kernel into the viewport. The aim here was to provide a seamless transition from context to context as the user navigates, giving a sense of continuity.

Searching is initiated simply by typing when nothing is selected. Typing immediately pops up a search pane (see Figure 17), which, like the autocomplete pane for kernel name completion, matches kernel names and contents as the user

types. This allows for very rapid switching of contexts: the user only need type the name of something they are thinking of in order to instantly bring that information to the foreground.

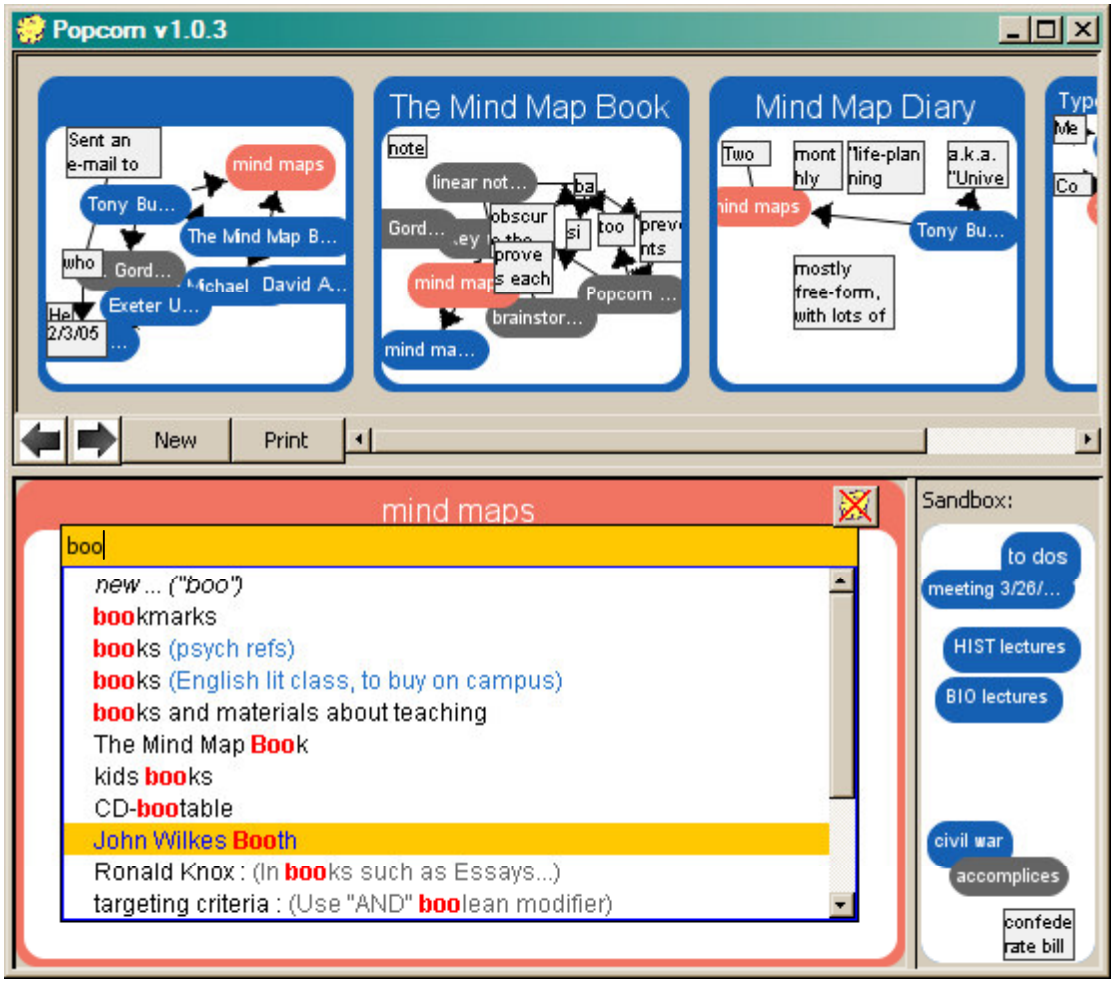

Figure 17. Popcorn's "quicksearch" facility for switching contexts to another kernel view. The user has just typed "boo" in an attempt to bring up the kernel "John Wilkes Booth," intending to switch contexts from the currently active kernel (which deals with mind maps, a completely different topic.) After typing "boo," the user has pressed the down arrow on the keyboard to move the selection to the desired kernel. (In practice, it would probably be quicker in this case for the user to continue typing "t" and "h," which would bring the target kernel to the top of the results list.) If the user were now to press the return key, the active kernel would immediately become "John Wilkes Booth," with its parent kernels replacing those in the parents panel.

Note several things about the quicksearch results pane in Figure 17. After each keystroke, the results panel is refreshed with four sets of choices:

1. The "*new… ("boo")*" entry. If selected, this will create a *new* kernel with the name the user has typed (in this case, "boo"), and make this kernel the active kernel.

2. All kernels whose names begin with the letters typed. The letters typed are shown in red to indicate which part of the name has matched.

3. All kernels with *any* word (other than the first word) matching what was typed. Again, red letters draw attention to the match.

4. All kernels that contain a *note* which matches the letters typed. In this case, the red letters appear in an excerpt of the note (trimmed to a word boundary if necessary to fit the results pane) which follows the kernel name.

(When more results are available than can be displayed, the list becomes scrollable and the user can view the additional items by simply pressing the down arrow.)

The idea is that the user can recall information by simply typing what comes to their mind. As long as the word or phrase that they type is somewhere in the name or the contents of the kernel, it can be successfully retrieved this way. And if neither the name nor any note can be remembered, the user can still navigate to the kernel if they can first browse to a kernel to which it is related (either through association or containment.)

All searches are case-insensitive, as can be seen from the figure. (The string "boo" matches both "bookmarks" and "The Mind Map Book.") This is because I

assumed users would be unlikely to remember specific capitalization, and in any case it is slightly more cumbersome to have to use the shift key. Also, note that searches only match at the *beginnings* of words, not within them. (The "boo" search would not match a kernel named "notebook," for example.) This is partially because the implementation of a within-word search would be problematic (see implementation section, below) and partially because it seems that users would normally remember words, rather than syllables. Admittedly, this issue has not been further explored, but it is worth noting that no users complained of failing to recover knowledge because they could only remember the middle of a key word!

Finally, since there is nothing preventing the user from giving two (or more) kernels the same name, the search results list attempts to distinguish between them by listing the names of the parent kernel views immediately after identically named kernels. In the figure, note that there are *two* kernels named "books" that matched the user's search text. The program therefore indicates that one of them has a single parent named "psych refs," and the other has two parents named "English lit class" and "to buy on campus." This will hopefully be enough to help the user choose the correct kernel, if they were indeed trying to retrieve one of the identically named ones. This strategy will not suffice in every case, of course, since there is nothing preventing both identically named kernels from appearing on the same set of parents (or no parents at all), and even if these sets of parents are unique there is no guarantee that they will help the user properly distinguish between them. In practice, however, such corner cases seem to be extremely rare, and so I believe this technique is sufficient.

The "jump-index-local-nav" retrieval strategy

Note that these two methods – navigating from kernel to kernel, and quicksearching to a new context – are the *only* ways to bring up previously entered data. There is no all-encompassing browsing function, for instance, that would allow the user to see everything they have entered at a glance. This omission is intentional: Popcorn is trying to model the way the mind naturally works with knowledge, and people are not known to ask themselves, "now what are all the things I know?" Instead, one always begins with a reference point and starts the recall process from there. The implication is that in Popcorn, a user can never reach a previously recorded kernel unless they can remember (a) its name, (b) any of its contents (notes), or (c) the name or contents of any kernel it is connected to, either directly or indirectly.

This may seem to be a great hindrance to retrievability, since users cannot (for instance) browse through a folder of items, but are instead required to remember the names of items. This gamble is based on the practical observation that even when one cannot recall a name, one is normally able to free associate to related information. If we cannot remember a former colleague's name, we can usually at least remember the names of a few other colleagues at that place of business, or else the names of the projects we worked on with them, or at the very least the name of the business itself. Presuming we have modeled the knowledge correctly, then, all of these other avenues will be available to us: we can simply quicksearch to the name of another colleague (or a project name, or the company name) and browse from there to very rapidly locate our target.

I term this retrieval strategy "jump-index-local-nav," meaning that whenever the user desires to retrieve a particular piece of information (kernel or note), they commence a two-step process. The first step is to "jump-index" to a related kernel, based on the free associations that come naturally to them in their mind. The second step is "local-nav": the user navigates locally within that part of the knowledge base that surrounds the kernel that they jumped to. This makes the entire knowledge base available to the user as long as they remember a relatively few starting points to jump to. The user thus is not required to remember the name of every kernel in order to access every kernel.

To take an example, suppose a user wished to recall some information about the U.S. Secretary of State in the late 1990's. Conceivably they might face some difficulty if they were required to produce the name "Madeline Albright" from scratch. But it is less likely that this user would forget the name "Bill Clinton," and presuming that a fact such as "Bill Clinton nominated Madeline Albright" were in the knowledge base, all of the information about Albright would be retrievable by first jump-indexing to Clinton, then locally navigating to Albright. The efficacy of this approach is based on the notion expressed by Lorayne and Lucas: "you can remember any piece of information if it is associated to something you already know or remember." ([195], p.7.) And as reported in the next chapter, users did indeed find this retrieval mechanism straightforward and reliable.

Miscellaneous supporting features

Browser integration. In order to help the user assimilate Web content, Popcorn features tight drag and drop integration with the Mozilla Firefox browser.[1]

Users can highlight text in *any* drag-and-drop-enabled application, and then drag the selection into Popcorn, which automatically creates a note for it. But for Firefox, a special Popcorn "plug-in" also includes the URL of the source Web page for such events. When the user drags an excerpt from a Web page into Popcorn, the created note will have a special icon attached to it. (The icon looks like a small earth, and is visible in several of the notes in Figure 15.) If the user later clicks on that icon, a new Firefox window will automatically pop up and connect to the Web page, scroll to the excerpt, and highlight it. This makes it easy to incorporate publicly-available information into one's personal knowledge store, without dealing with the hassle of maintaining links or copying URLs.

Hotkeys. It is clear that a PKB will often be used in conjunction with other applications, such as Web browsers and PDF readers (for assimilating knowledge) and word processors and e-mail clients (for using knowledge to produce artifacts.) For this reason, users can press a Popcorn "hotkey" combination to quickly toggle the application between foreground and background. This allows Popcorn to always remain "at the user's fingertips" and yet not consume screen real estate. The hotkey can be combined with the drag and drop from Firefox: users highlight text, click and move the mouse pointer to initiate the drag, and then press the hotkey combination to bring up Popcorn to receive the drop.

Back and forward buttons. Finally, a pair of browser-like navigation buttons (the large left and right arrows in Figures 15 and 17) permit users to scroll back and forth through the list of recently accessed browser views. This gives the user a better feel for where they have been and also helps "undo" false navigations. Each button

becomes “grayed out” if there is no next or previous view to navigate to. (The contents of this recently-navigated list is not stored persistently between Popcorn sessions.)

Architecture and implementation

The Popcorn prototype was written almost entirely in Java for purposes of cross-platform compatibility. (Users of Microsoft Windows, Macintosh, and Linux systems all volunteered for user testing, and hence it was essential to support all three platforms with a common code base.) It consists of 107 source files in nine packages that comprise a total of about 25,000 lines of Java code.

Only two small components were written in other languages. The hotkey event handler code needed to be compiled natively and installed into the operating system, and so a separate C++ handler had to be written for each platform. The Firefox extension was written in XUL[140], a user interface markup language that can be interpreted by various Mozilla tools to parametrically describe GUI widgets and their actions. The latter component is also cross-platform for any operating system supporting the Firefox browser, and so with the exception of the hotkey handlers complete portability was achieved.

Portability comes at the expense of speed, of course, since cross-platform code must be interpreted by a program at runtime rather than executing directly on the system processor. However, with a personal knowledge base, throughput is not a concern: only latency matters. This is because the volume of information recorded and retrieved in real time by a single human user is miniscule compared to the

amount of time available (milliseconds, seconds, or even minutes between requests.) The only performance questions involve the responsiveness of the interface to GUI interactions: for example, will the search pane come up quickly enough when the user types, how long will it take to load a single kernel view from disk when the user requests it, etc. Careful optimization around specific performance-sensitive operations was sufficient to make the interface relatively quick. (Only two out of twenty users in the user trial complained of an unresponsive interface.)

Popcorn interfaces with the open source MySQL relational database, which it uses to actually store the user's knowledge. The schema for this database is given in Appendix A. Essentially, each note, kernel, and relationship is stored as a row in a table designed for that type of entity. The containment relationships, along with the spatial positions on each kernel view, are held in a separate table called "containedObjects." The transclusion property can be observed by simply looking at this table. When one kernel appears on another's view, a row with the id of the contained kernel will appear in this table. This same id can appear in any number of other rows, which means that (1) a kernel can appear in many contexts, and (2) there is no "primary" context in which it appears; all contexts are "equal." (As an aside, the same table is used to hold information about where *notes* appear on kernel views, which means that the database schema itself does not prohibit notes from being transcluded by multiple views just as kernels can. The Popcorn interface, however, does not allow this since there is no way to search for a note itself outside the view in which it was originally created.)

The Popcorn prototype caches information from the database as it runs. When the user starts the application, and searches or navigates to a particular kernel view, that kernel view will be loaded into memory (or "hydrated") and kept up to date there until the application terminates. Changes are also flushed to the database as they occur. Measures were taken, however, to ensure that cascading hydration was limited. A user's knowledge base, after all, is normally composed of many interlinked entities, and so the danger existed that navigating to one kernel view would cause a huge segment of the database to be hydrated at once (all the children and parent kernels of the kernel view, plus all of *those* kernels' children and parents, and so on.) For this reason, a more sophisticated caching mechanism was called for, so that only the information that would actually appear on the screen would be hydrated, rather than everything the kernel was ultimately connected to. This keeps the memory footprint of the application as small as possible, while eliminating the latency associated with hydrating unnecessary data.

Finally, it is worth mentioning the specialized indexing necessary for the quicksearch facility. Quicksearch responsiveness is the most critical performance aspect of the entire system, since the application must present search results as the user types. Any sizeable latency here is aggravating, and it can disrupt the thought process of a user who may be guessing at the names of kernels created long ago. For this reason, a fairly complex search mechanism was designed to optimize performance in this area. First, MySQL supports a basic "free-text search" facility, which was extended to support searches for both kernel names and contents. When properly configured, MySQL can build word-by-word indexes of specific fields in a

table, so that the rows which match specific words (or the beginnings of words) can be identified without actually searching the entire table row by row. Second, since numerous kernels may match a search (especially short search strings, when the user first begins typing) it was impractical to return all the results for each keystroke. A paging strategy was employed, so that when a key is typed, only enough results to fill the search results pane are returned for display. The additional pages of results are only fetched as the user scrolls. Somewhat complicating matters was the fact that a given kernel might match a search in more than one way. For instance, a kernel named "The End of the Affair" with a note "The protagonist was sympathetic" on its view would match the search string "the" *three* times: twice in the kernel name, and once in the note. Care was taken to display a given kernel only once in the results list, and happily this additional processing did not appear to slow down the search responsiveness significantly.

Evaluation of design goals

Before presenting the actual user results in the next chapter, it is worthwhile to illustrate briefly how this design attempts to meet the goals originally outlined:

1. Recording knowledge should be quick and painless. Popcorn's interface is streamlined to allow one thing and one thing only: encoding knowledge. The user has no options, say, to select different colors, fonts, or graphics, and hence the entire set of operations can be mapped directly to mouse and keyboard inputs. The interface does not need to be modal, and no menu selection or similar tasks are required that might soon become tedious when large amounts of knowledge need to be quickly

entered. Note that discoverability has been deliberately sacrificed here, in the hopes that users will quickly master and memorize the small number of operations.

2. Returning to previously recorded knowledge should be quick and painless. A simple key press brings up the search-as-you-type feature, allowing users to rapidly switch between contexts. If the user forgets the name or contents of a desired kernel, searching for anything related to it will allow them to "get close" with a search and then navigate locally within the desired realm.

3. Reorganizing knowledge should be easy. Users can drag kernels and notes freely in any kernel view, and can even drag them inside or outside of expanded kernels and parents. Kernels, notes, and relationships can be deleted simply by hovering over them and pressing the popup delete button. The intent is that reshuffling objects into different configurations will be easy with direct manipulation.

4. Users should be able to express knowledge both formally and informally. The user may choose to encode knowledge as groups of interrelated kernels, as natural language sentences in notes, or a combination of the two. Relationships may be typed or untyped, and kernels may be named or unnamed. And spatial positioning (e.g., clustering) is available when users want to express similarities or grouping informally.

5. Public content should be easy to assimilate. Dragging a snippet of text from another application automatically creates a note for it in Popcorn. And when such a snippet comes from the Web (through Firefox), not only the URL but also the position within the source page is captured so it can easily be referenced later in context.

6. The tool should work naturally with human memory. Numerous features of the interface have been designed with this in mind, including:

Semantic networks. Popcorn's basic data model is a semantic network, which the user constructs from kernels and relationships one view at a time.

Categories. The membership of elements in a user-perceived category can be expressed by placing kernels inside a kernel view. And Popcorn's transclusion permits an idea to be present in any number of categories. This does not address the more subtle notions of "fuzzy membership" (ie., some items are more perfect examples of a category than others), but the user has other options at their disposal to express such nuances, such as spatial positioning or annotation through notes.

Unlimited total memory, limited working memory. The entire interface has been designed for compatibility with the STM/LTM dichotomy of the human mind. The database (analogous to the LTM) is comprised of hundreds or thousands of kernels, only a select few of which are visible at any one time (just as the STM only works with small subsets at a time.)

Traversal between contexts. The multiple contexts of the active kernel are immediately visible and the position of the active kernel within them can be clearly seen. These contexts can be navigated to with a simple double-click, which mimics the "spreading activation" concept of closely related information.

Multiple retrieval paths. The autocomplete kernel-naming feature steers users towards reusing kernels in different contexts, which encourages the encoding of multiple retrieval paths.

Extension of prior knowledge. Similarly, when creating a new view, users can easily incorporate elements from previous views, which allows them to tie together new knowledge and old.

Visio-spatial support. The two-dimensional spatial metaphor is central in Popcorn; all activity takes place by manipulating items visually. And the spatial positioning is recorded along with the content, so that when the user returns to a previous view, the layout is consistent.

This evaluation is merely theoretical, however; the only way of knowing how well such a system would work in practice is to deploy it to real users. In the next chapter I present the results of a user study designed to give insight into the issues surrounding both this particular design, and the notion of PKBs in general.

CHAPTER 5

USER TESTING RESULTS

An effective personal knowledge base should be an integral part of a person's life. It is not so much used for a particular task in order to produce a particular artifact, but rather intermittently, in conjunction with the daily tasks that already confront the user. The user will employ the system much more on some days than others, and very rarely for its own sake: the goal is not "to sit down and use the personal knowledge base for a while," but "to work with knowledge for some external purpose, which will require, among other things, consulting the personal knowledge base."

The true utility of such an application cannot be determined instantaneously, but only over time, as the user retrieves and augments the knowledge they have previously stored. Simply having users experiment with the interface to generate knowledge diagrams does not tell us if the PKB is doing its job. The true test comes later, when that knowledge is to be recalled, reorganized, and exploited. Only then will it become apparent whether the tool is useful in maintaining a user's knowledge over time.

To evaluate the Popcorn design, a prototype version was deployed to twenty volunteer users for use in their real-life settings. Each tester was given a one-month trial period in which to use Popcorn, though several users requested to continue to use

it after the trial expired.[11] This group included students from a variety of engineering disciplines, computing professionals in various roles (developers, testers, administrators, marketers), a schoolteacher, a business consultant, a newsletter editor, a product development manager, and a Presbyterian minister.

The goal was to determine how naturally the tool would integrate with a user's daily life, how much it would be used and for what tasks, and so forth. Hence after the software was installed, testers received a brief tutorial and were then instructed to use the system in any way they wished during the testing period. No specific usage requirements (total amount of data to be entered, number of times per week the application was to be used, etc.) were stated or implied. Users were not told to invent artificial tasks for which the system might be useful (unless they wanted to do that), but rather to try and apply it in natural scenarios, where it would be genuinely valuable.

As might be expected, the results were widely variable. Some testers used the system heavily, nearly every day; others were more sporadic, only entering data on a few days each month. Some adopted Popcorn as a full-fledged personal knowledge base, recording knowledge about numerous diverse aspects of their lives; others concentrated only on a few domains or even a single domain. The types of information recorded were many and various, including such domains as culinary recipes, event planning and scheduling, personal health concerns, source code snippets, procedures for system administrators, guitar chords, contact information, lesson preparation, and notes on the Russian Revolution, just to name a few.

---

[11] For this reason, the statistics presented in this chapter represent a longer period of usage in some cases.

To give a concrete example, one computer science student used Popcorn to store a variety of programming-related knowledge. This included the relationship between components in the C++ standard template library, which ones he had used in certain modules of a class project, and snippets of Web pages that contained relevant tips and techniques. He also archived the syntax for certain Linux commands that he had trouble remembering, and useful ways he had seen them combined in scripts. This user stored more than just technical knowledge, however. Planning a trip to a dinner theatre involved the times, dates, and prices of shows, directions to the playhouse, and a record of a credit card transaction. Meeting minutes from his part-time job tracked the status of decisions on key projects, and the influences on them. A schedule of deadlines and relevant procedures helped him keep track of his graduation requirements. Notes from all of his class lectures captured and cross-referenced key points and highlighted pertinent sections of reading. And a multiply-categorized "to do" list, crossing several kernels, helped him manage his list of monthly, daily, and even hourly tasks. Note that Popcorn was used to manage knowledge in diverse domains, with varying degrees of structure, complexity, and malleability over time.

Quantitative analysis

Six of the testers were unable to use the system effectively enough to provide meaningful quantitative data, for various reasons described in the section on "Qualitative impressions," below. The others agreed to run a simple program which extracted certain statistical information from their knowledge base. This was

intended to get an idea of the general structure of Popcorn knowledge bases, and the degree to which they varied among users. A summary of these results is presented in Table 2. I will address each statistic in turn, and explain what light it sheds on Popcorn usage.

	Mean	*Min*	*Median*	*Max*	*Coefficient of Variance*
General:					
Total size of knowledge base	768.4	86	208	3661	137.5%
Notes with Web source (%)	17.7%	0.0%	13.3%	62.5%	104.2%
Measures of relationship usage:					
Typed relationships (%)	48.6%	12.5%	45.4%	100%	55.3%
Number of relationships per kernel	0.25	0.02	0.21	0.63	69.9%
Number of relationships per type	1.23	0.67	1.04	2.21	38.2%
Measures of kernel complexity:					
Note-to-kernel ratio	1.34	0.12	1.19	2.84	67.4%
Empty kernels (%)	52.9%	35.8%	50.8%	75.5%	24.7%
Measures of kernel containment:					
Avg kernels contained per view (and contained by other views)	1.03	0.76	1.00	1.45	18.8%
Avg objects contained by non-empty views	4.98	2.52	5.17	7.41	27.6%
Kernels that have one or more parents (%)	82.2%	66.0%	83.3%	96.2%	9.9%
Kernels that have one or more children (%)	28.0%	13.5%	28.1%	44.0%	27.2%
Kernels with both parents and children (%)	18.5%	2.0%	18.3%	37.5%	47.2%
Kernels with two or more parents (%)	15.1%	2.0%	14.8%	33.1%	65.2%
Island analysis:					
Number of islands	26.9	4	13	135	135.7%
Avg kernels per island	9.8	4.46	6.88	26.5	66.8%
Kernels in largest island	176.7	18	50	1045	172.3%
Total size of largest island (%)	61.1%	31.9%	61.8%	87.2%	31.5%

Table 2. Summary of knowledge base characteristics across the test group. "Mean" is the arithmetic mean of each statistical measure. "Coefficient of variance" is defined as the standard deviation divided by the mean, which gives a mean-adjusted idea of the stability of each statistic.

Total size of knowledge base

This total was computed as the sum of the number of kernels, notes, and relationships in the knowledge base at the time the data was collected. (Note that any objects created and then deleted before the analysis program was run would not be included in this total.)

As can be seen, the quantity of knowledge stored varies widely among individuals. Paradoxically, this measure did not always correlate with users'

qualitative assessments of the tool: several users with relatively small knowledge bases said the tool was very effective for them. The safest conclusion seems to be that different users have very different knowledge needs: some either deal with much more knowledge than others, or choose to materialize it much more often. Thus the absolute size of the knowledge base is not a very accurate measure of the tool's utility for a user.

Notes with Web source (%)

Also widely varying was the percentage of a user's notes that had a Web source attached; ie., that were dragged in from Firefox. Some testers did not use this feature at all, while others reported that they completely forsook the traditional "bookmark" functionality of their browser in favor of Popcorn notes. Overall, this measure is doubtless most dependent on where a user receives their information: if most of their information sources are on the internet, Popcorn's auto-URL capture facility can be extremely convenient. Most users made use of this feature to some significant extent, but also had numerous "plain" notes that they composed themselves (see Figure 18.)

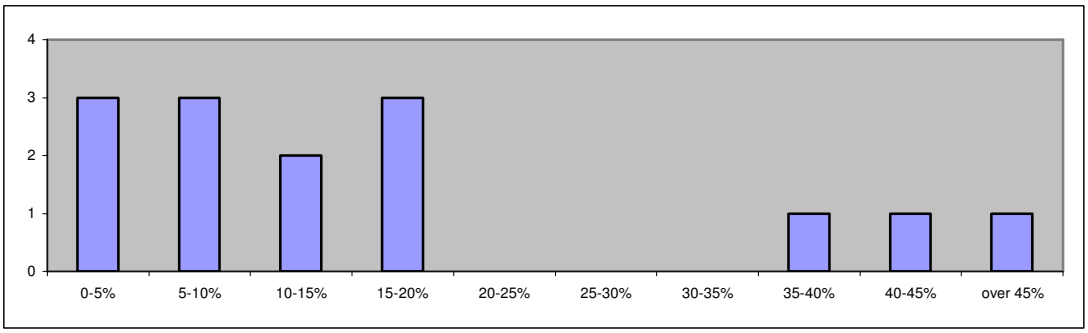

Figure 18. Histogram of the percentage of a user's notes that were assimilated from Firefox.

Typed relationships (%), Number of relationships per kernel

Relationships were used in very different ways, as can be seen by the variability in the overall percentage of relationships that were assigned types (ie., given names), and in the number of relationships per kernel. This underscores how users can differ significantly along the formality/informality continuum. Some users clearly prefer to precisely specify the nature of relationships between kernels, while others are content to simply draw informal connections. This likely depends on the type of knowledge being recorded, and how fully it is understood by the user. Many users had significant numbers of relationships with and without types; most often, around half of each (see Figure 19.)

The fact that some users had many more relationships per kernel than others tells us that users express the associations between ideas in different ways. Some most often create an explicit relationship between items, while others presumably indicate relationships less formally, perhaps through spatial positioning or containment. (See Figure 20.)

The correlation coefficient between these two measures is 0.43, which indicates that they are somewhat related: users that frequently assign names to their relationships also have many relationships per kernel, which further confirms the formality/informality hypothesis. Interestingly, however, even the user with the densest number of relationships had only .63 per kernel, or about two relationships for every three kernels. This tells us that even users who prefer formality also take advantage of informality, though the converse may not be true.

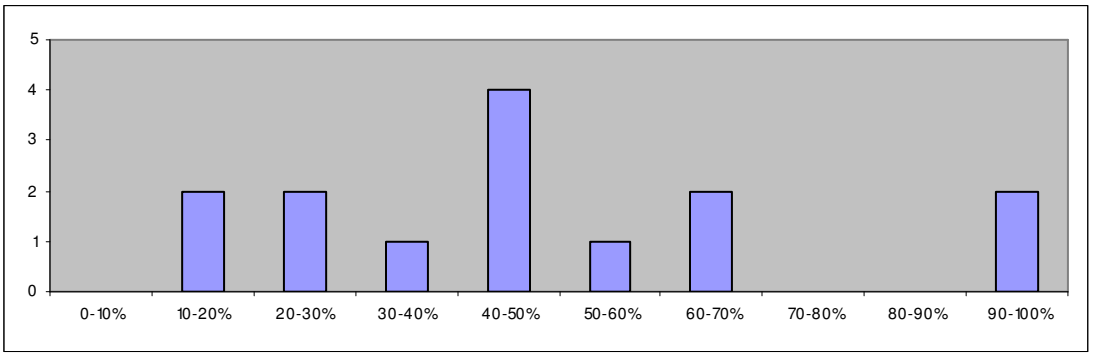

Figure 19. Histogram of the percentage of a user's relationships that were given types (names.)

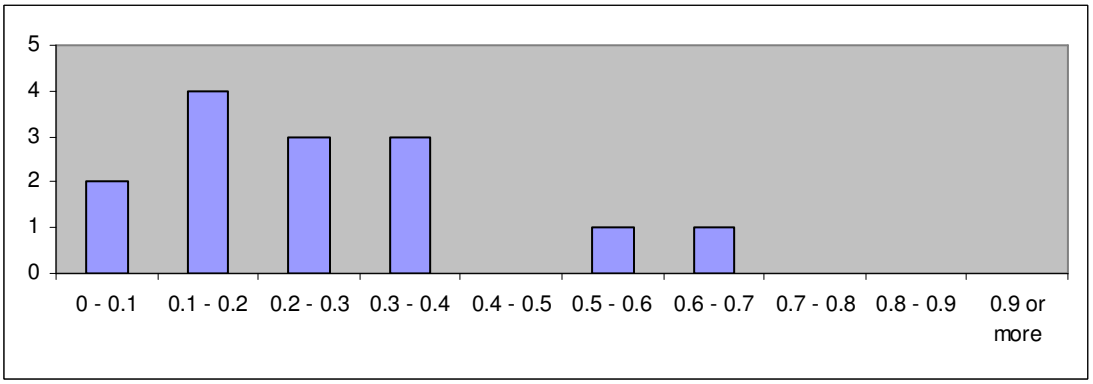

Figure 20. Histogram of the average number of relationships per kernel.

Number of relationships per type

This statistic relays how often a user gave the *same* type (or name) to multiple relationships. It is computed as the number of relationships that were given names divided by the number of unique names given. For example, if a knowledge base contained the three relationships "Caesar ruled Rome," "Cleopatra loved Caesar," and "Cleopatra ruled Egypt," this statistic would be 1.5, since there are three relationships, and two types ("ruled" and "loved.") Hence it helps us determine whether the verb phrases between kernels are often re-used, or whether they are normally unique. The analysis reports yield the latter conclusion. (See Figure 21.) It

appears that for most users, relationships are rarely given the same names. This tells us that at least in this regard, users tend to model knowledge informally. Empirically, relationship types are often multi-word, descriptive phrases, intended to convey precise and nuanced meaning, rather than being members of an oft-used set of common link types (as is the case with most current research efforts into knowledge ontologies.)

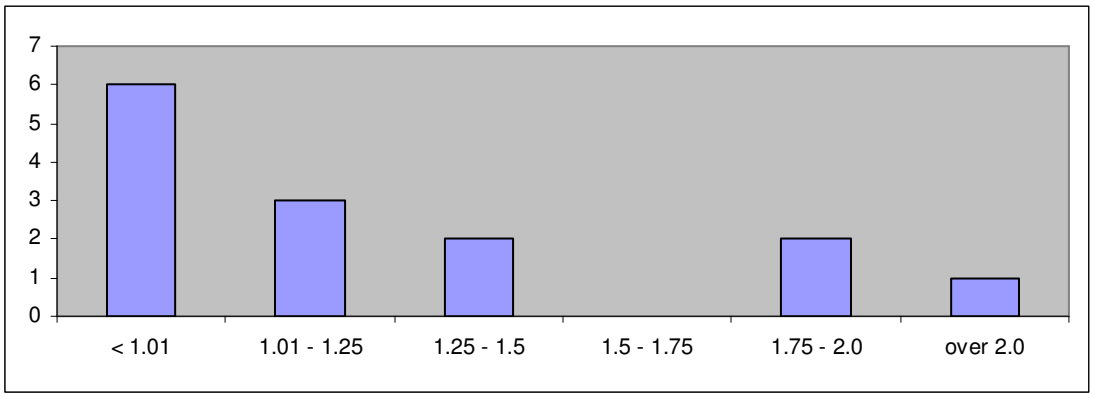

Figure 21. Histogram of the average number of relationships per type.

Note to kernel ratio

Another highly variable figure was the ratio of the number of kernels to the number of notes in the knowledge base. This may be another indicator of formality/informality preference, since it shows the extent to which users represent knowledge as concept maps, rather than as raw phrases or sentences. It is clearly a matter of personal choice: one tester had eight times as many kernels as notes; another had nearly three notes for every kernel. The histogram (Figure 22) shows how variable this preference is.

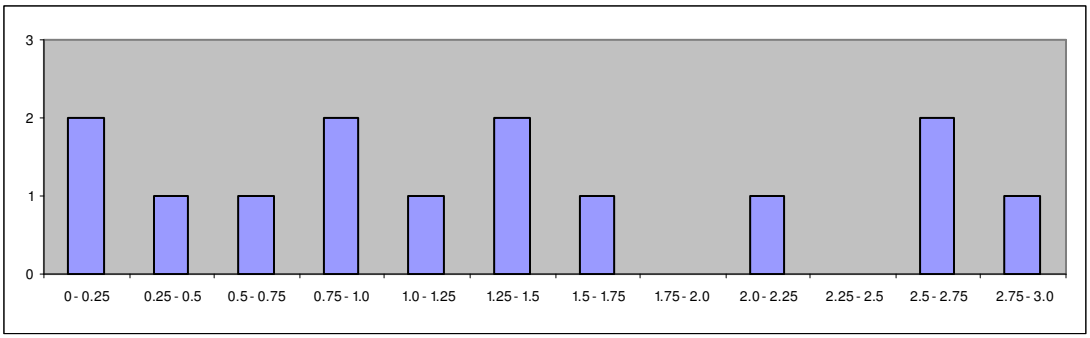

Figure 22. Histogram of the note-to-kernel ratio.

Empty kernels (%)

More constant was the percentage of kernels that were empty (ie., those with no notes or child kernels.) Most testers had very close to half their kernels containing inner detail, and half representing simple unelaborated concepts (see Figure 23.)

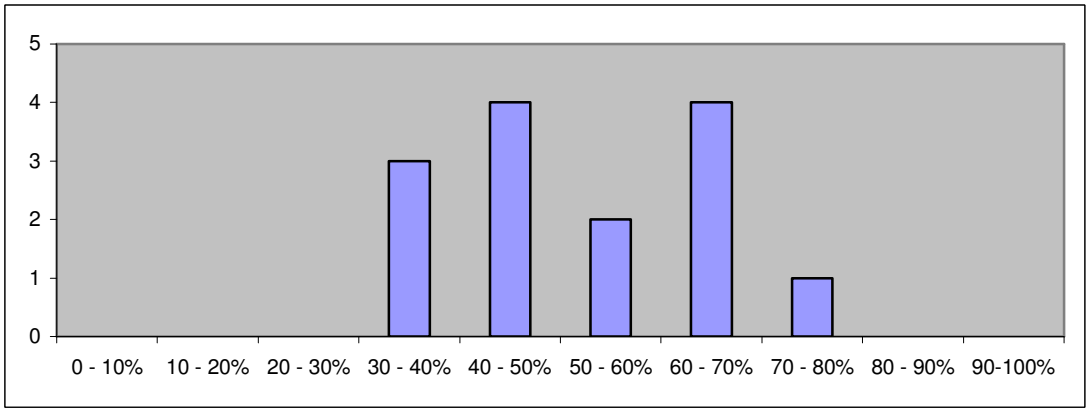

Figure 23. Histogram of the percentage of empty kernels.

Avg kernels contained per view (and contained by other views), Avg objects contained by non-empty views

A surprisingly stable statistic was the average number of kernels that were contained on a kernel view, or put another way, the average number of child kernels per view. (Since every parent relationship is also a child relationship, this is also equivalent to the average number of parents per kernel.) This was very close to 1 for most users. The average number of objects (kernels and notes) on a *non-empty* view

was also relatively stable (see Figure 24), and gives insight into how "full" a user tends to keep their views. This is especially important in light of the difficulty in reorganizing knowledge that plagued many users: the problem is more likely to develop for densely packed views, as explained below. On average, most users tend to keep between three and seven objects on a view.

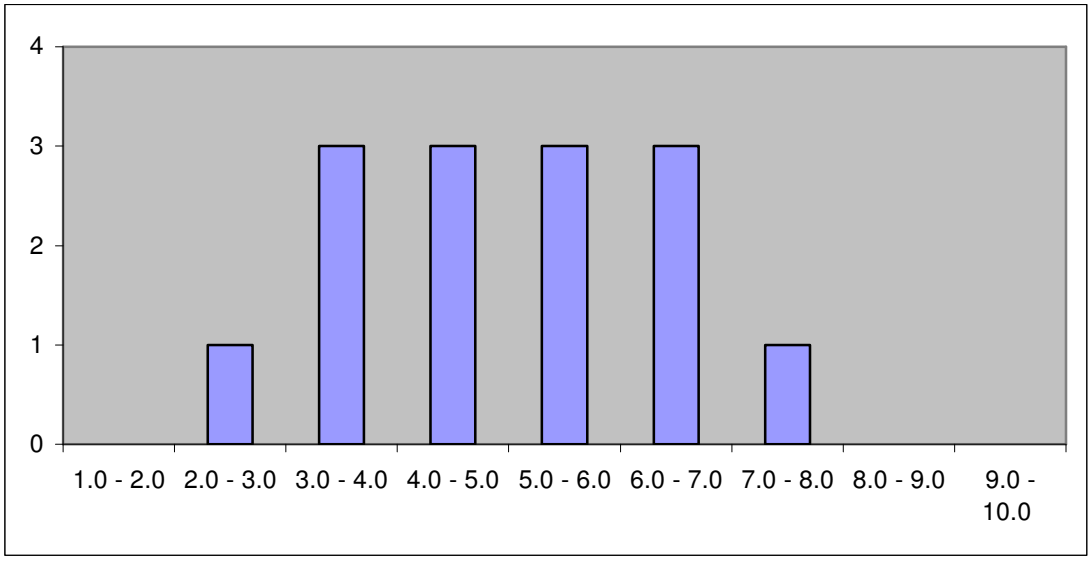

Figure 24. Histogram of the average number of objects per view.

Kernels that have one or more parents (%), Kernels that have one or more children (%), Kernels with both parents and children (%)

These three statistics were nearly constant across the user group. The percentage of kernels that had parents was normally about 83%, with very little variation, and the percentage with children was about 28%. The percentage of kernels that had both parents *and* children was somewhat more variable, but was still between 15% and 25% for over half the group. (See Figure 25.)

Overall, it seems that most users create relatively few "top-level kernels" (17% or so.) By combining these figures with the "empty kernel" data (above), we

can conclude that most people have about half their kernels empty, 1/4th with child kernels (and possibly notes), and 1/4th with notes only.

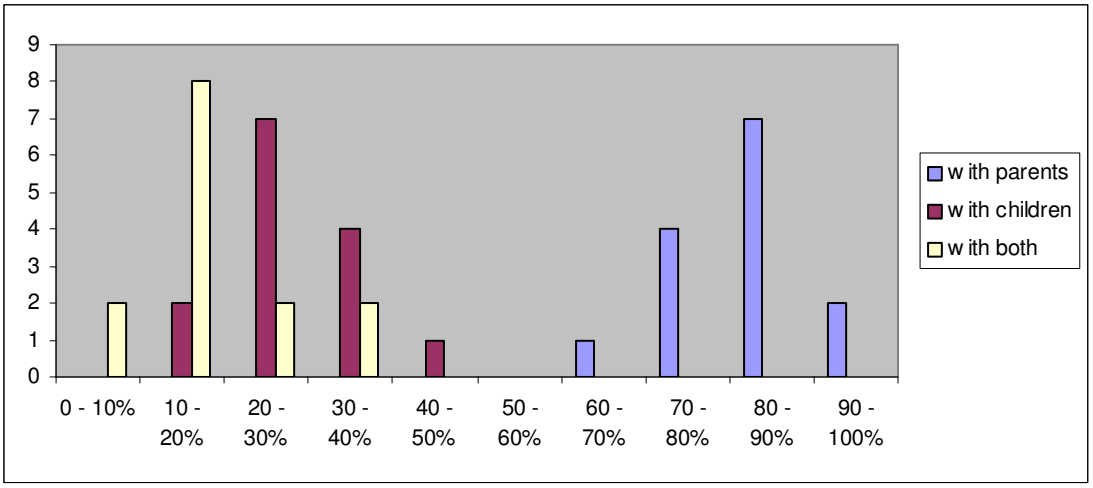

Figure 25. Histograms of the percentage of kernels that had parents, children, and both parents and children.

Kernels with two or more parents (%)

More than any other statistic here presented, the percentage of kernels with *two or more parents* was a predictor of the user's qualitative assessment of the tool. The correlation was extraordinary. Every user with more than 10% of their kernels having multiple parents was enthusiastic about the tool's effectiveness, without exception. Every user with a lower percentage found the tool difficult to use effectively, also without exception. (See Figure 26.)

From this data we can conclude that transclusion is perhaps the most critical ingredient in Popcorn's success. After all, "kernels with two or more parents" is just another way of saying "kernels that appear in multiple contexts," and it is this feature that all thriving Popcorn users seem to master. Placing the same item on multiple views is a technique that some users find immediately intuitive, but that others struggle to find uses for. Probably the best way to interpret these results is that when

a user does not make use of transclusion, Popcorn doesn't have enough other compensating advantages to make it worth their while. There are enough impediments (unfamiliar interface, change in workflow, etc.) that the cost of using Popcorn outweighs the perceived benefits. In the hands of a user who has mastered transclusion, however, the gains can be tremendous, and well worth the trouble of adapting to the quirks and even changing one's way of working.

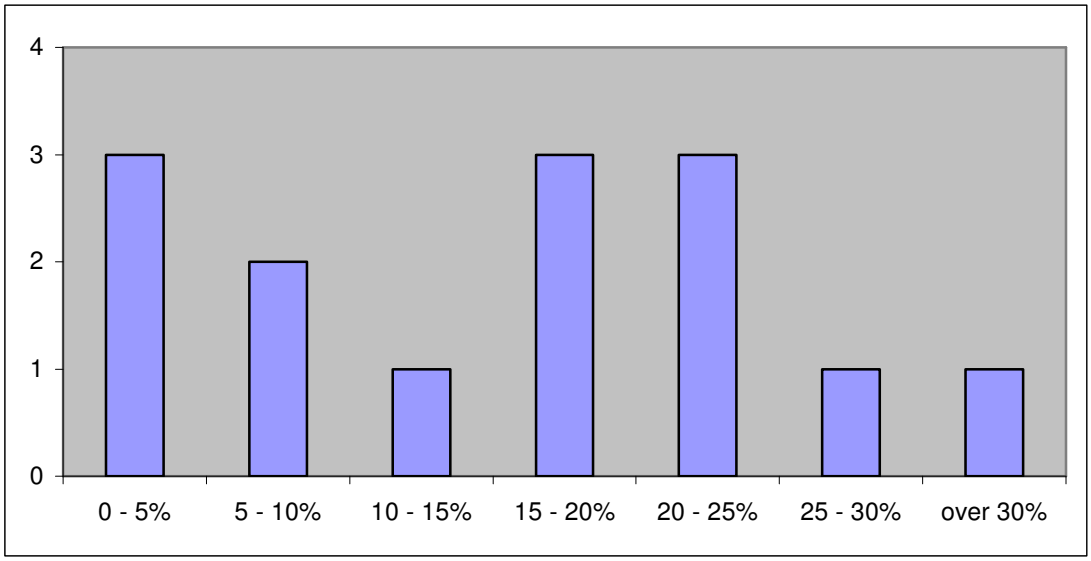

Figure 26. Histogram of the percentage of kernels that had two or more parents (ie., that appeared in multiple contexts.) This statistic, by far more than any other collected, correlated strongly with user satisfaction. Users with a percentage of greater than 10% always found Popcorn to be a very useful tool, while those with a lower value were invariably dissatisfied.

Number of islands, Avg kernels per island, Kernels in largest island, Total size of largest island (%)

The last set of general statistics relate to the *islands* in the user's knowledge base. An island is a set of kernels that are connected to each other, either directly or indirectly, and so reachable by navigation alone (as opposed to by quicksearch.) The kernels in an island may be related through relationships, containment, or both. Each island is thus a cluster of interrelated information, which is not connected to any

kernels outside of it. The purpose of island analysis is to discover how many isolated clusters of knowledge a user's PKB consists of, and how large those islands are. This should give a good picture of how diverse a user's domains are, and how often the user makes use of connecting kernels across contexts.

The results here were surprising, and remarkably consistent across users. Nearly every user's knowledge base followed the same pattern: many islands of knowledge, each with less than ten kernels, plus one gigantic island that contained the majority of the knowledge base. No user had less than 32% of their knowledge base consolidated into a single island, and several users had 80% or more in one cluster (see Figure 27.) This is all the more remarkable given the average number of kernels per island, which was nearly always quite low (see histogram in Figure 28.) These results hold even for the many users who stated (in face-to-face interviews) that they used Popcorn for many separate tasks and in diverse domains.

The pattern seems to be that users create small chunks of knowledge as they use Popcorn, but that over time the chances increase that each small chunk will be "absorbed" into the large island because of some association the user perceives. And once connected to the whole, it normally stays there, since relationship deletion and container removal are both relatively rare.

We can draw an important conclusion from this pattern that actually validates Popcorn's overall data model. Consider two graph-based approaches presented in chapter 4: the file-based architecture of tools like CMap[60] and Inspiration[162], and the "fully-connected" graph of PersonalBrain[311] and Compendium[77]. In the former, users create separate files for each knowledge diagram, and the elements on

these diagrams cannot refer to one other. This approach can be viewed as the forceful division of a user's knowledge into small islands. In the latter, the opposite is true: the user is prohibited from partitioning their knowledge into islands, because every element must be connected to the root or else it is lost. In contrast, Popcorn stakes claim to a flexible middle ground: every item of knowledge can exist on its own, or be freely connected to any other. Hence the islands can develop naturally, as the user sees fit. Users are neither required to partition, nor prohibited from doing so.

The island analysis data seem to attest that this approach, and *only* this approach, is ultimately viable. I stated earlier that forcing the user to divide up their knowledge into isolated pieces was unacceptable, and here is the proof. The fact is that, when given the opportunity, users naturally create numerous relationships between all kinds of elements in their knowledge base, even between those from domains that seemingly would be separate from each other. Therefore, requiring the user to isolate their knowledge into bite-sized pieces prohibits a great deal of expression. On the other hand, a complementary pattern is that users tend to create many small islands in addition to the single large one. They drop data into Popcorn without concern for fitting it into the larger picture, and these connections only emerge later, if at all. Hence the "fully-connected graph" approach is also problematic, because the user will be forced to create such associations at data entry time, when they are not yet naturally perceived. Popcorn's method of facilitating the creation of any relationship yet not mandating artificial ones seems to be the best solution here.

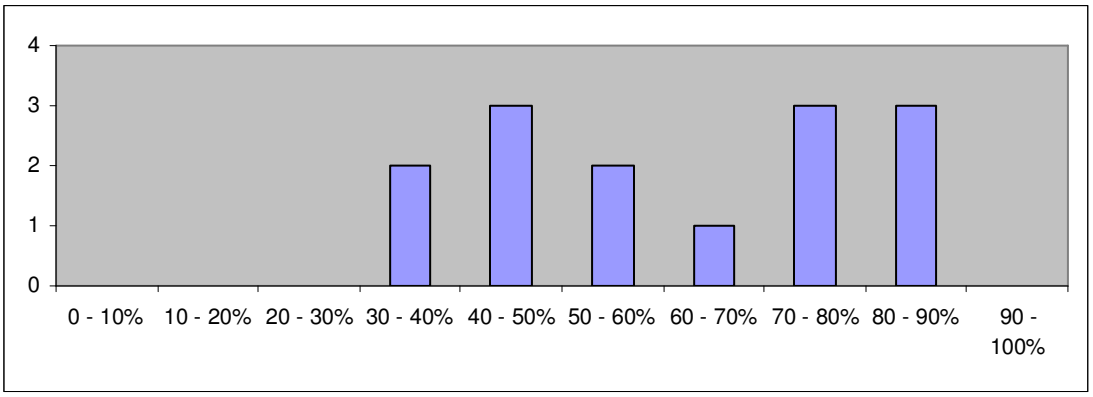

Figure 27. Histogram of the percentage of the knowledge base occupied by the largest island.

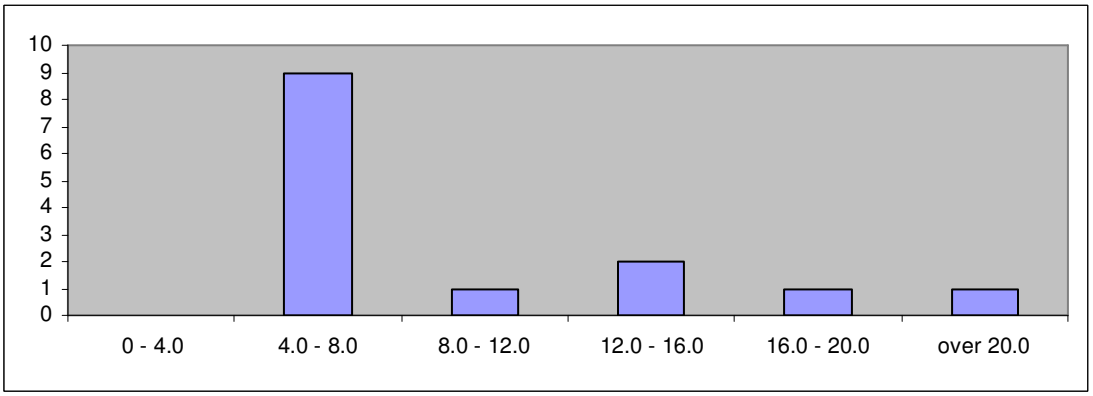

Figure 28. Histogram of users' average number of kernels per island.

Temporal analysis

The creation and most recent modification date of each kernel is stored in the database, so it was possible for the analyzer program to track each tester's usage over time. Figure 29 gives time series plots of this data, as a percentage of the user's overall activity during the trial. The message seems to be that users fall into two categories: those who use Popcorn in bursts, and those who use it more steadily. Several users had large spikes on particular days, indicating that they used the tool very heavily on certain occasions, while others demonstrated more constant usage

throughout the trial. (Note that data *retrieval* statistics were not captured, which means that these usage loads only reflect the creation of data.)

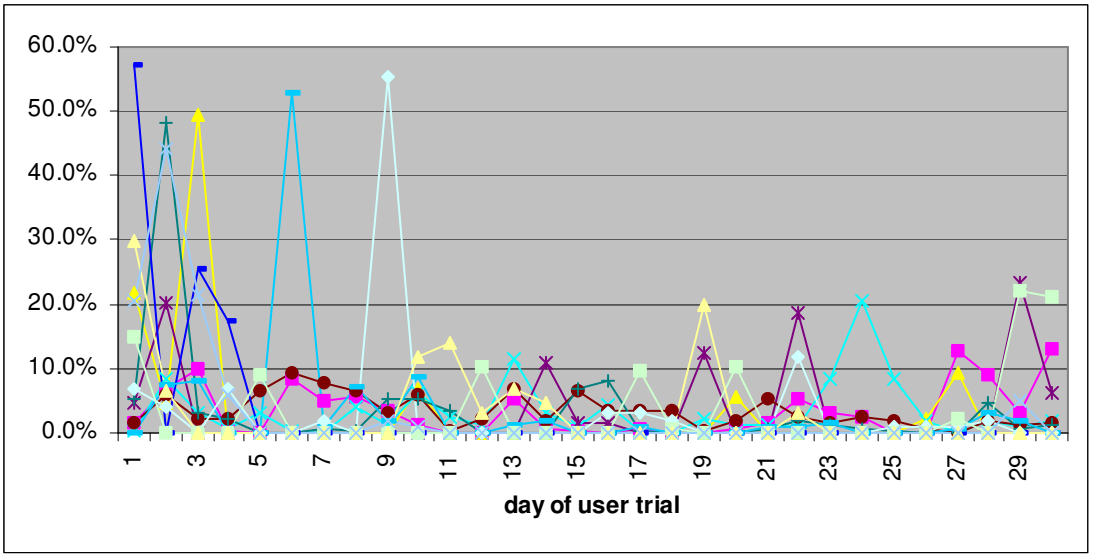

Figure 29. Usage over the trial period, as a percentage of total usage. (Note: some user trials lasted longer than one month, and in those cases the usage statistics on this graph have been truncated to show only the first 30 days.)

Day of week analysis

Finally, the analysis program extracted the day of the week that knowledge elements were created, in order to compile weekly patterns of usage. As expected, this varied considerably from user to user, and the results are shown in Figure 30. The only constant seems to be that the system is rarely used on weekends or (bizarrely) on Tuesdays. Perhaps the most important fact that we can glean from this data is that Popcorn usage is highly dependent on a user's schedule. Nearly every tester had a large spike on at least one day, suggesting that users settle into patterns of data entry that conform well to their work week. This probably indicates that Popcorn is often used for deliberate knowledge modeling sessions, in addition to simply capturing bits of data that the user encounters randomly through the day. (As

above, these statistics only reflect the creation of data. How the frequency of knowledge retrieval correlates with the day of the week is therefore not known.)

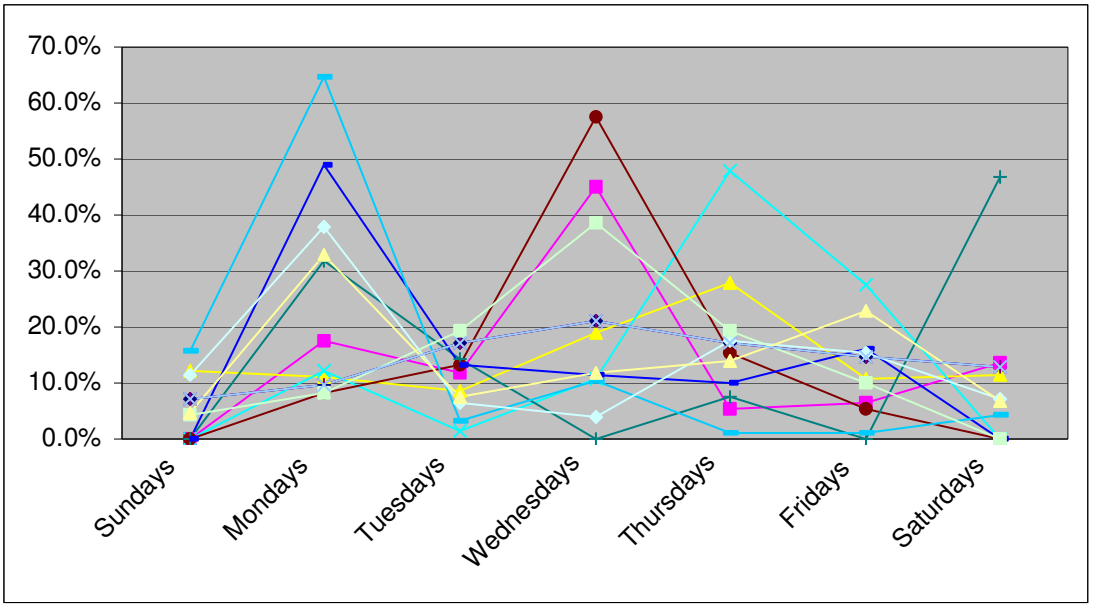

Figure 30. Testers' usage load by day of week, as a percentage of overall usage.

Qualitative impressions

After the one-month trial period, each of the twenty testers engaged in a face-to-face interview lasting about two hours.[12] Interviewees were asked a broad range of questions designed to elicit how the tool functioned for them practically, and what obstacles they faced. In many cases, a tester would provide an unexpected insight into their experience, which would trigger a more lengthy follow-on discussion to further flesh out its implications. The results presented in this section are largely my attempt to ferret out the common themes that emerged from these interviews, even

[12] In two cases, the interview was conducted by means of a confidential e-mail exchange.

though a particular idea was often stated in quite different ways by different testers answering different questions.

General impressions

Overall, the qualitative impressions were mixed and strikingly bimodal: users tended to either embrace the system enthusiastically, often using it as a replacement for all other forms of knowledge management (8 out of 20 testers), or else have great difficulty in adapting to the paradigm (7). A few (5) occupied a middle ground where they could use the system effectively enough to see its advantages, but ultimately rejected the tool as it stands because of one or more inhibiting factors. As discussed in detail below, the ability (or willingness) of a tester to understand, accept, and effectively use Popcorn's data model (as opposed to its interface, or *raison d' etre* in general) seemed to be the largest determining factor in the overall impression.

Incentives and obstacles to usage

A personal knowledge base is entirely subject to the user's initiative. It never proactively prompts its owner to complete a particular task, but sits passively, waiting to be expanded or consulted at the user's whim. The question arises: when a knowledge-based task presents itself, what ultimately causes a user to decide to reach for a tool like Popcorn? What benefits do they gain, or believe they will gain? And conversely, what perceived hindrances cause them to *not* use the tool for such a task, in the key moment of decision?

To try and answer these questions, interviewees were presented with a series of statements about the reasons they might have used or *not* used the tool, and asked to rate how strongly they agreed with them. Responses were given on a five-point

scale: 1 for "strongly disagree," 2 for "disagree," 3 for "neutral," 4 for "agree," and 5 for "strongly agree."

Incentives to usage

The questions regarding incentives are given in Table 3, and the average responses on the 1 to 5 scale in Figure 31.

"Think about the times you chose to use Popcorn. On a scale of 1 to 5, where 1 means 'strongly disagree' and 5 means 'strongly agree,' how would you rate each of the following statements: "1. I often choose to use Popcorn because using it helps me **understand** my **knowledge** better." "2. I often choose to use Popcorn because it helps me **manage** my **ongoing**, changing **knowledge**." "3. I often choose to use Popcorn because it helps me **archive and return to** the **knowledge** I once knew." "4. I often choose to use Popcorn because it helps me **return to Web pages** of interest again." "5. I often choose to use Popcorn because it helps me **combine/synthesize/**integrate multiple sources of information."

Table 3. Interview questions designed to determine why people use Popcorn.

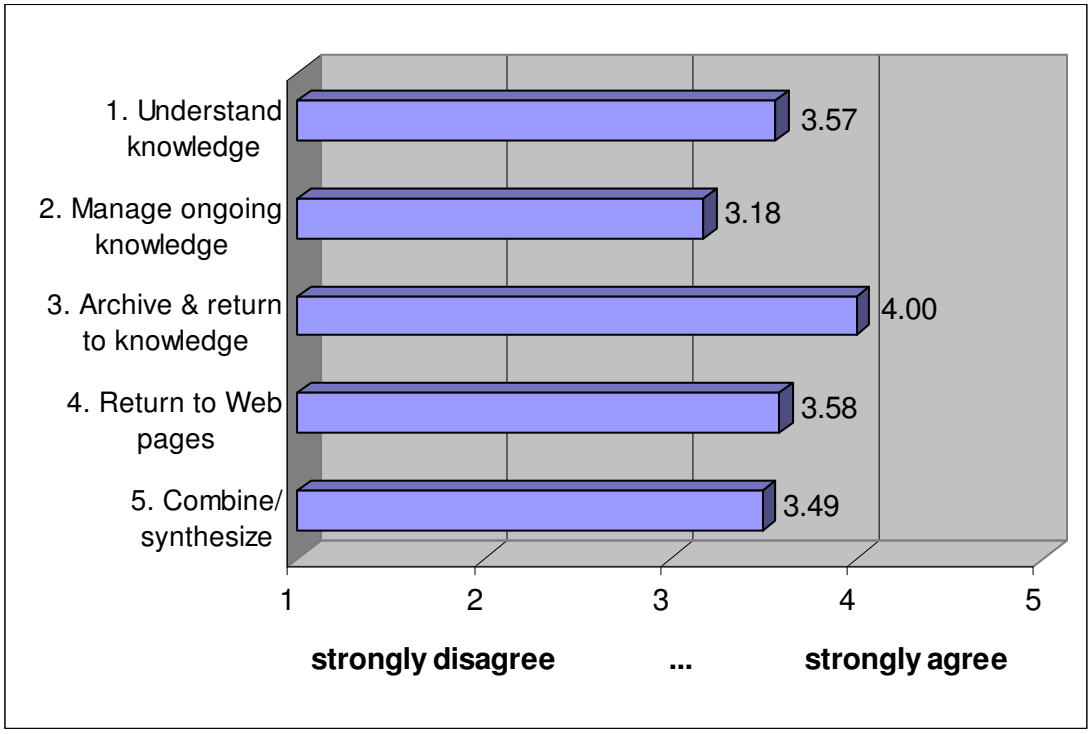

Figure 31. Average responses to the questions given in Table 3.

No one factor sticks out above the others in Figure 31. Popcorn's stated purpose of "a place to store and retrieve your knowledge" is clearly a motivating factor, but others are almost equally so, including helping one to understand a difficult domain, and facilitating the integration of sources. The ability to return to Web pages via auto-URL capture met with mixed results, with six users (30%) rating the statement a 5 ("strongly agree") and another six never having even tried the feature. Perhaps the most significant finding here is that users had no consensus on *which* task(s) Popcorn was most helpful with. Only one user agreed (or strongly agreed) with every statement, and only one user failed to agree with any. The others were mixed. Clearly Popcorn is used for a variety of purposes, and there is considerable disagreement about what kind of tool it really is.

Obstacles to usage

The questions regarding disincentives are given in Table 4, and the average responses on the 1 to 5 scale in Figure 32.

Hardware availability

The responses about inhibiting factors (see Figure 32) are more revealing. Clearly the most important barrier to using Popcorn is simply that the application is tied to the platform on which it runs (statement #2.) Hence it is easily accessible only at one's desk, when one is normally working alone on a specifically computer-supported task. Users said they wanted Popcorn to be available to them in the kitchen, during meetings, on the road, even in the shower, and in such settings they were forced to rely on other means. Even laptop users reported that their machines were not always available or convenient when needed.

"Think about the times you chose *not* to use Popcorn. On a scale of 1 to 5, where 1 means 'strongly disagree' and 5 means 'strongly agree,' how would you rate each of the following statements: "1. I often choose not to use Popcorn because I just don't have that much knowledge worth managing." "2. I often choose not to use Popcorn because the hardware is not convenient: I'm not around the computer when I need to store knowledge." "3. I often choose not to use Popcorn because the software is not convenient: it's too hard to start up and use the application." "4. I often choose not to use Popcorn because the possible benefit of storing my knowledge just isn't worth the effort." "5. I often choose not to use Popcorn because I don't have confidence I'll be able to find my knowledge again." "6. I often choose not to use Popcorn because there's nothing pushing me to use the tool – it relies on my own intiative." "7. I often choose not to use Popcorn because Popcorn's paradigm of concept maps isn't intuitive." "8. I often choose not to use Popcorn because Popcorn's user interface is foreign and non-intuitive." "9. I often choose not to use Popcorn because there are other tools (electronic or not) that work better for me than Popcorn." "10. I often choose not to use Popcorn because I just can't seem to develop the habit of using it." "11. I often choose not to use Popcorn because I have trouble integrating Popcorn into my daily routine."

Table 4. Interview questions designed to determine why people *don't* use Popcorn.

This was expected to be a problem, but a surprising discovery was that knowledge *recovery* was not the feature most sorely missed, but in fact knowledge *entry.* Evidently, users do not need to access their previously entered data in remote settings so much as they need to record the new knowledge that they generate there. The reason for this, as the interviews made clear, is that knowledge is elusive: it disappears quickly. One user, for example, recounted that he would often discuss a key idea in a meeting and have some critical insights, but that by the time he got back to his desk to record the knowledge into Popcorn, many of the important nuances had already faded from his memory. For maximum effectiveness, it seems, Popcorn must be made available at the point of knowledge creation. And considering how fleeting newly discovered knowledge appears to be, it could be an extremely valuable tool in this setting.

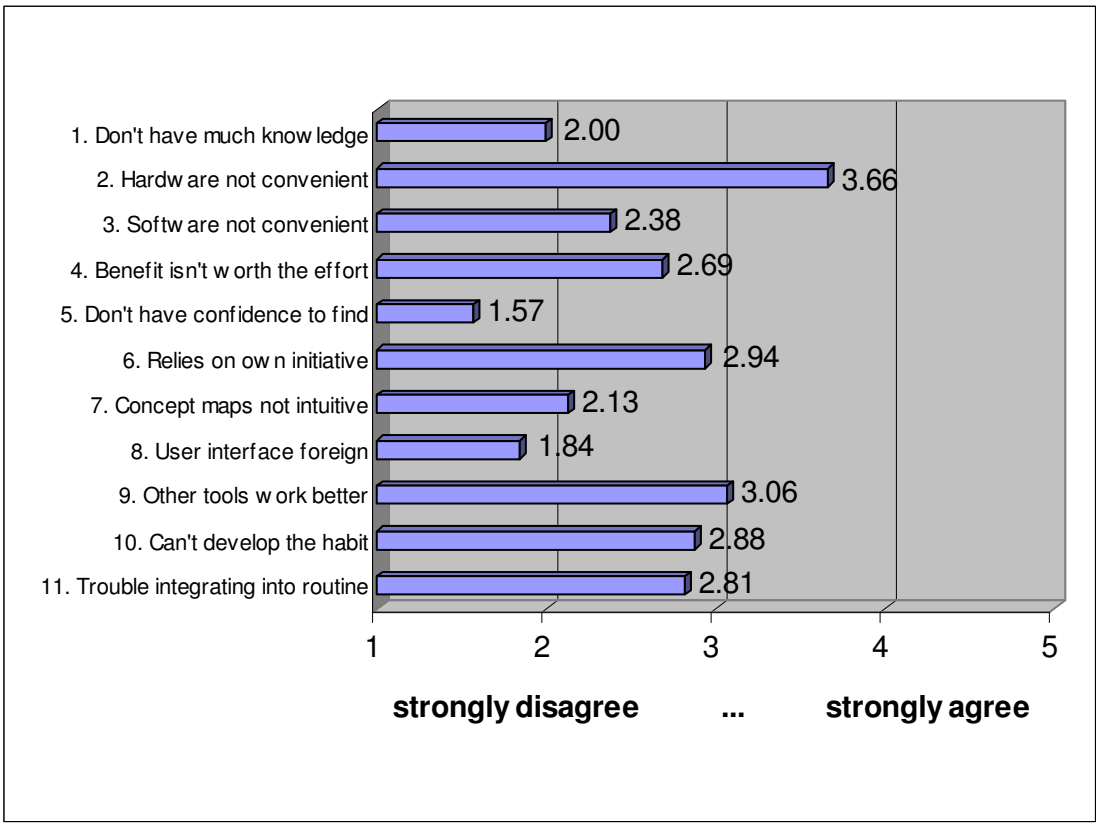

Figure 32. Average responses to the questions given in Table 4.

An obvious solution is to provide a distributed architecture, where one's knowledge base is hosted on a centralized server and accessed by client machines. These clients could include palmtop devices and cell phones as well as workstations or laptops. This would also solve the problem faced by some users who needed Popcorn to be available in two locations, typically work and home. There are many issues to be addressed here: security and privacy, meaningfully displaying knowledge on small screens, and keeping multiple clients "in sync" if the same knowledge base is accessed from more than one location. But a networked solution is probably necessary for a personal knowledge base to truly fulfill the vision outlined in this thesis.

Integration with toolset

Statement #9 ("other tools work better for me") was also a popular answer, but for two different reasons. A small number of users already had in place another information-keeping system (most often paper-based), and because they had ironed out the difficulties with those systems and formed habits around them, Popcorn was found lacking in comparison. But more often the reason was that the user already had a considerable amount of information entered into a different tool, or was required to use a different tool to generate a particular kind of knowledge artifact. And Popcorn, as it stands, does not interface well with most of the other software a typical user employs.

The level of integration desired varied among users. Some suggested that if the data they currently managed in other applications could be easily imported into Popcorn, they would be content to abandon those applications entirely and use Popcorn as the sole repository. These users were invested only in their data, not in the other tools themselves. Others were looking for more task-specific support so that their Popcorn knowledge could be more conveniently leveraged. (Ideas included exporting a network of kernels to create a Microsoft Word outline or a set of PowerPoint slides; including shortcuts to files in a kernel view; and directing e-mail messages from a particular sender into a kernel representing that person.)

The basic message here is that an individual uses a variety of applications to manage their knowledge effectively. And although a personal knowledge base should perhaps be the centerpiece of this toolset, it should work seamlessly with the other applications to carry out tasks. At the very minimum, Popcorn needs to be able to

import and reference data stored in other tools, and it might also be advantageous to more tightly integrate with certain tools so that the knowledge it contains can be used to maximum effect.

Habits and initiative

As users "talked out loud" about their answers to statements 6, 10, and 11, it was apparent that these ideas were related. Many users had simply never thought in terms of intentionally managing their knowledge before, and so while a promising concept, there were challenges in putting it into practice. Some of this had to do with getting used to identifying items of knowledge as they were encountered throughout the day, and considering them as candidates for entry into the tool. Some had to do with developing more diligence in acquiring information, and being less content to make decisions based on assumptions.

These problems had less to do with Popcorn *per se,* and more to do with the whole idea of a personal knowledge base. In general, a PKB (like Popcorn) is not an easy tool to properly use. It is easy to use "in the small" – creating and retrieving a knowledge diagram is trivial – but much harder to user "in the large"; that is, maintaining enough consistent, disciplined usage over a long period of time for its benefits to show themselves. One user commented that his problem with Popcorn was that its supposed added value was too distant; it was hard for him to convince himself that it was worth investing the time to record knowledge, when the benefits of doing so, even if they materialized, were remote.

The cost/benefit tradeoff

Related to this issue were the responses to statement #4 ("the benefit isn't worth the effort.") Several users stated that they often struggled with this decision at the time knowledge was encountered. Recording knowledge in Popcorn, while fairly streamlined, still requires effort, and it is hard for users to know whether this is worthwhile in the instant that it must be recorded or lost (see [168].) Making an effective "to record or not to record" decision requires a good deal of prescience, both about whether the knowledge is likely to be retrieved later, and if so, how best to encode it (and associate it with other items) so as to facilitate this retrieval. It is expected that if a user makes a commitment to a PKB, over time they will recognize their own habits and patterns, and make wiser decisions here.

Non-factors

This verbal questionnaire also helped identify some red herrings. Two of the areas expected to be problems at the outset of this project – the unfamiliar UI paradigm, and the anticipated difficulty in remembering kernel names – were plainly insignificant. Nearly every user (17 out of 20) reported that although the interface took some getting used to initially, they were quickly able to adjust to it, and soon preferred it for speed's sake. This is probably because Popcorn supports such a small number of operations, all of which are used commonly, and so are easily remembered.

Too, Popcorn users were surprisingly successful at retrieving old knowledge from their knowledge base. This was expected this to be a problem since recall ought to be more difficult than recognition, and Popcorn requires the recollection of

verbatim text (or the verbatim text of a connected kernel) in order to retrieve an item. However, nearly every user (also 17 of 20) stated flatly that remembering the names of (or retrieval paths to) desired kernels was never an issue for them.

There are several possible reasons for this. For one, the trial period was brief, and it is possible that after a year or more of using Popcorn, users would find recovery substantially more difficult. It is also possible that users' retrieval patterns are usually short-term in nature: one normally needs to return only to kernels that have been created recently, thereby lessening the burden on biological memory. It may well be, however, that Popcorn's recipe of "free associate to a related kernel, then navigate from there" is in fact an effective retrieval technique for the long-term. Longer-term testing should reveal whether this is the case.

The response to statement #1 ("I don't have that much knowledge worth managing") demonstrates that nearly everyone agrees with Popcorn's main goal. The reason to reject Popcorn, whatever else it might be, is not because it is a solution for a non-existent problem. On the contrary, the problem of drowning in knowledge that one cannot easily manage is well-appreciated by all – the only question is how best to solve it.

Finally, remarkably few users expressed any difficulties with concept maps themselves (see statement #7), even though no formal training was provided. In most cases, just a few examples were sufficient to demonstrate the paradigm of association-based knowledge representation, and users were comfortable working with it. This at the very least affirms that semantic networks come naturally to people

with very different backgrounds, and may in fact further suggest that our minds actually store knowledge in a representation quite like them.

Specific themes

As testers talked about their experiences using Popcorn, several other themes emerged that form some of the most important "lessons learned" from this project.

Common patterns of use

Somewhat surprisingly, the typical Popcorn user tends to use the system in two very different, non-overlapping ways. One is for information capture. Popcorn users stored various bits of important data in the tool, loosely grouping them and naming them, and taking great comfort in knowing that they were "safe." A few examples would be phone numbers, driving directions, passwords, and quotes from Web pages, but the variety here was quite large. This kind of information is typically relevant for a short to medium duration, has very little structure, and its primary purpose is to be later recalled.

The second common area is knowledge formulation. Users brainstorm difficult domains with Popcorn, and their intent is to emerge with a better understanding of the topic. They most often do not begin with a thorough grasp of the knowledge, but rather discover it as they go, using Popcorn as a sort of multidimensional scratch pad. Interestingly, many users report that once they have generated a concept map, even if it is very detailed and crosses over numerous kernel views, they have little interest in being able to return to it again. Hence Popcorn's primary function as a storage repository is not even used in this case. The knowledge gained theoretically has very long-term value (especially as compared with a phone

number or password), and yet apparently its chief purpose is to be formed in the mind, not recalled via the tool.

The interesting fact here is that nearly all users (18 out of 20 in the trial) employ *both* of these methods to some degree, but with little to no overlap. Over time, their knowledge base becomes a growing collection of two spheres: loosely connected bits of information, entered hurriedly and recalled frequently; and richly structured knowledge maps, created painstakingly and reviewed rarely. Both have value, but it is surprising that one tool seems to serve both purposes.

A third, less common use for Popcorn (recall Figure 31) is to manage ongoing, changing knowledge. Users keep a set of kernels up-to-date so that it always represents the "current state of affairs" in some area. Examples here are personal "to do" lists, project management resource allocations, and the state of ongoing dialogues with customers. Users report that this can be very helpful in staying on top of tasks that involve multiple interdependencies, but that it is impractical for more than a small number of areas simply due to the time required to keep them up to date.

Domains

As a knowledge management tool, Popcorn is best-suited to ill-understood, unexplored domains with many arbitrary relationships. This is clearly something that most existing software applications have trouble with, and is what can make Popcorn a very valuable tool. It was apparent, however, that the knowledge most users generate is a combination of these kinds of domains plus hierarchical (e.g., outlines, org charts), tabular (accounts, contact info), and unstructured (memos, essays)

information. Users liked Popcorn for the first kind of knowledge, but for these other, more traditionally organized areas, many of them (9 of 20) became frustrated with it. The unrestrained flexibility, which is Popcorn's hallmark, makes it ill-suited to domains that implicitly have a regular, predictable structure: for these areas, users prefer tools that more rigidly guide them. And using Popcorn for completely unstructured information (free text) is equally frustrating because its text editing features do not compare with those of word processors. Hence it appears that the original vision for Popcorn as a "store everything" knowledge base needs to be adjusted somewhat. As discussed previously, it needs to be better integrated into a user's overall toolset so that each application can be used to process the type of information for which it is intended.

Visual representation

Popcorn's method of displaying knowledge visually through concept maps was mostly well-received, though not without a few criticisms. Users reported that the spatial nature of kernel views helped them more quickly zero in on important information, more easily get their bearings when returning to a previously created view, and more firmly ingrain the knowledge in their biological memories (nine testers mentioned one or more of these benefits, even without being asked.) One user said that after constructing a kernel view, a certain concept was thereafter "always in the bottom-right corner" of their mind, even when not using the tool, and this sort of feedback was not uncommon. This suggests that the act of using Popcorn helps create and solidify dependable cognitive structures.

Spatial placement on a canvas involves more degrees of freedom than writing on a lined notebook page, and users varied in their ability to adjust to that. For some, it was extremely positive: one stated that after using Popcorn, it was painful to return to linear text because it felt arbitrarily confining. Another commented that working spatially let them "think like a human, rather than the computer forcing me to think like a machine." A few users (4 of 20), however, found this extra dimension unnecessary and even inhibiting. The knowledge itself was sometimes harder to enter, because in addition to specifying the concepts the user is forced to also specify their spatial positions, which can be distracting. Another frequent comment (from 7 users) was that certain patterns of information – most commonly, lists – did not lend themselves well to a spatial representation. One can implement a list as a kernel view, of course, with the items arranged as kernels or notes inside of it, but some users were unsatisfied with this, preferring a strictly one-dimensional listing of bullet items.

Difficulty in reorganization

Despite attempts to facilitate the reshuffling of knowledge, Popcorn did not live up to user's expectations in this regard. Many users (about 15) related that they found it frustrating to restructure kernels to conform to new understandings. This is partially due to the specifics of the tool's interface. There is currently no "multi-select" operation, which could be used to select and then move a number of related kernels in concert. Also, the screen features just one kernel view at a time (the active kernel in the viewport), with all others relegated to a smaller, thumbnail status. This makes it difficult to break one kernel view into two, since one has to create a new,

expanded kernel on the viewport, move some of the contents into it, name it and delete it from the current view, and then reorganize the remnants, an unnatural and tedious process. Also, since knowledge acquisition is nearly always additive, even sparsely populated kernels tend to fill up over time as more facts are learned about a topic. An eventual reorganization of the kernel is actually the most fortunate outcome here; more problematic is that the user will become reluctant to add newly acquired knowledge for fear of the view becoming overcrowded. Clearly, better methods for restructuring are needed. I expect that modest enhancements to the user's repertoire of operations (such as adding multi-select and multiple top-level windows) would help a good deal.

A replacement for bookmarks

At least six users stated that Popcorn was a complete replacement for their Web browsers' "bookmarks" or "favorites" facility. The ability to drag text into Popcorn from a Web page and auto-capture the URL has several advantages. First, the granularity of "bookmarking" is smaller: instead of pointing to an entire page, snippets of important text *within* a page can be captured. One user observed that this was particularly useful when gathering information from "blogs" or "forum" websites. These sites often feature lengthy exchanges between individuals on some topic, and important morsels of information can be buried inside a response many screenfuls down within the page. The ability to instantly return to the surrounding context of the excerpt is powerful.

Another advantage is that the URL of the webpage is captured in the context in which the user is naturally thinking about it. A Popcorn view contains a user's

understanding of a subject, and if some Web content bears upon that understanding it is quite natural for the annotated URLs to appear directly on that view. Compare this with using (say) a Word processor to express one's thoughts, and having to wade through a hierarchical arrangement of bookmarks every time one needs to view a related web page.

Finally, this feature helps solve the volume problem. More than one user remarked that their browser had become so overloaded with (often outdated) bookmarked pages that it was now impossible to navigate them. The basic problem here is that the browser is forcing all bookmarks for all topics to be contained within a single hierarchy, and this proves to be unwieldy over time. With Popcorn, however, the field of view is never uncluttered with irrelevant bookmarks. Each one appears on the view for which it is relevant, and never obscures the user's field of vision otherwise. A Popcorn knowledge base can accommodate thousands of bookmarks over a period of years, since none will ever appear outside the right context.

The effect of allowing duplicate names

The original Popcorn design mandated that the name of every kernel in a knowledge base be unique. This was for the sake of simplicity and to make the search process more straightforward – nothing but the kernel name itself would need to be shown in the results list. Early testing, however, revealed that this was problematic in some situations. It is probably true that most of our ideas could be given a unique name that would unambiguously denote them: for instance, a businessperson might be thinking of "barriers to entry for the Z1000 product in the medical sector." However, people normally operate in the presence of so much

implied context that it is unnatural to specify this level of detail in a name.[13] If a colleague were to ask what this businessperson was discussing at lunch, they would be much more likely to simply hear "barriers to entry," with the other details being left out because they were understood.

In Popcorn terms, this manifests itself in the naming of kernels. A user wants to quickly assign the most natural name for a kernel, based on what they are thinking about at the moment. In the above example, the obvious name for the kernel is "barriers to entry." But if names are required to be unique, this is problematic. Surely over the lifetime of a knowledge base, the tag "barriers to entry" is likely to have more than one meaning depending on context. (Different products, different sectors, or even different conversations about the same product in the same sector.) If the user does not fully appreciate the ramifications of global uniqueness, they are likely to want to create a different kernel with a duplicate name later on, which the system wouldn't allow. (Renaming the original kernel is possible, of course, but requires the user to lay aside their current task and temporarily return to a former one, which is very disruptive.) And if the user *does* appreciate the ramifications, the problem can be even worse. Some early testers reported that they sometimes felt paralyzed when naming kernels, because they recognized they needed to be absolutely certain to fully specify the concept with a globally unique name. At the

---

[13] Note that this is one advantage of the hierarchical data model. Every node is implicitly "in" exactly one other (ie., it has one parent) and so it can be considered a proper subset of that parent. Thus, for example, in a filesystem it is common to see a folder named "meeting minutes" underneath a folder named "XYZ project." The very structure implies that the former contains the meeting minutes *for the XYZ project.* There is no need to actually name it "XYZ project meeting minutes." When we move from the hierarchy to a directed-graph-based representation, however, the notion of "in" changes, since any node can have any number of parents, which is what leads to this naming dilemma.

very least, this slows down the process of knowledge entry, which we want to avoid at all costs. And it can also lead to uncertainty and doubt.

In response to this issue, Popcorn's data model and interface were modified slightly to allow duplicate names. As described in chapter 4, two kernels with the same name can (sometimes) be distinguished in the search results list by including the names of their parents. The effect of this change was positive, but in an unexpected way. After using the modified application, users reported that in actual fact, they did not encounter name collisions very often at all. The fears that led to naming paralysis were largely unjustified (at least during the relatively brief trial period.) The impact of the change, however, was substantial: it simply relieved users' anxiety at kernel naming time. Since users can assign names freely, knowing that if necessary they could create a different kernel with the same name later on, they can proceed with confidence.

"Fuzzy" vs. "crisp" kernels

Popcorn models an abstract "thought" as a concrete, named entity. This works better for some thoughts than others, especially those that appear in multiple contexts. A kernel named "John Wilkes Booth," for example, works well because of its tangibility: there is very little ambiguity about what real-world object is meant. But consider a kernel like "rebellion." Suppose it first appears in a kernel view about the Civil War. Now the user is creating a knowledge representation of the Warsaw Uprising, and wants to include a kernel called "rebellion." They double-click, type the first few letters of the name, and see the previous kernel pop up in the results pane. Should the user place the previous kernel on this view, or create a new one

with the same name? And how would this change if the user were describing a mutiny on a ship, or problems in dealing with teenagers?

The answer is not as simple as it first appears. If the user chooses to create a new kernel, then there will be no "link" between the Civil War and the Warsaw Uprising. Later on the user could not, for example, browse the Civil War view, observe the kernel on "rebellion," and then double-click on it to see "Warsaw Uprising" among the contexts in which it appears. To the system, these would be two completely unrelated kernels, which prevents the user from asking the question, "where are all the places in my knowledge base that deal with rebellion?" Choosing to use the same kernel, however, is also not without difficulties. If two distinct topics are really in view (rebellion in a war versus teenage "rebellion") then it is probably not a good idea to use the same kernel to represent them. But even when they are essentially the same topic, the "rebellion" kernel may have information *inside* it (other kernels and notes) that pertains specifically to the Civil War (or the Warsaw Uprising.) In this case, it is awkward to have a kernel appear on one view that contains information obviously meant for another.

The problem seems to be that knowledge is often less well-defined than we realize. We pretend that it can be broken down into crisp, unambiguous concepts, but using Popcorn for any length of time exposes that sometimes it can't be. The same basic idea may be proliferated throughout a knowledge base, but its semantics change slightly depending on its context, which makes rendering it in a Popcorn-like data model difficult. I know of no great solution to this problem, other than gaining experience with the tool over time as such situations are discovered.

Communication through views

On the whole, Popcorn users were quite optimistic that if they were to share their kernel views with another person, they would be easily and quickly understood, with little to no accompanying explanation. Some reported that they had actually done this with friends or colleagues and had success. One tester commented that she used Popcorn to "show her thinking to others" and that the diagrams were actually more for their benefit than her own. Another planned to use it as a tutor in order to encourage students to lay out their notes in a certain way.

While perhaps a bit naïve (the implicit expression through spatial positioning would not be obvious to a newcomer, for instance) these findings do bode well for extending Popcorn into a collaborative tool. Users could perhaps work together remotely on views, share portions of their knowledge base for another's perusal, etc. This is outside the scope of personal knowledge bases *per se*, but since knowledge in some sense is contained within organizations as well as individuals, this may be a fertile area of future research.

Oft-requested features

The most commonly requested feature was the aforementioned distributed architecture, so that a user's knowledge base would be accessible from multiple locations. Another desirable enhancement was a "global browser" feature through which one could see the entire contents at a glance. This would fill a gap in Popcorn's search paradigm. Unlike, say, hierarchical tools, where one can start at the root and explore down the tree, there is no way to access "everything" in Popcorn. If a user knows something is in the knowledge base, it is easy to find, but the user has to

start by knowing (or suspecting) that it is there. No mechanism exists to *discover* what is there, other than by typing random letters and seeing what appears in the search results!

Three reasons were cited for wanting this feature. One was to periodically "clean up" the knowledge base by purging it of dated information. This was admittedly for aesthetic reasons rather than performance, since Popcorn suffers virtually no performance degradation as more information is added (due to its database indexing mechanisms.) The second was to serve as a directory of information into the knowledge base – a user could browse a list of "starting kernels" which would direct them into the main areas of knowledge. The final reason was simple curiosity: after working with Popcorn for a long period of time and building up a large knowledge base, some users were simply interested in going back and surveying what they had created, which is difficult to do with the product as it stands.

One other feature users mentioned was a "chronological search" similar to that of LifeStreams[121, 126] or Circus Ponies[70]. They expressed that it would sometimes be convenient to go back to "yesterday's kernels" without having to name them, or to be able to look for information that they knew had been recorded last summer in case the existing search mechanism failed.

Lack of structure

If freedom and fluidity are Popcorn's greatest assets, it appears that they can also be its greatest liabilities. Several users (8) commented that they sometimes felt paralyzed by too many options: since Popcorn imposes no particular structure, the onus is completely on the user to generate their own, and this can be an intimidating

proposition. It is often not clear at the outset which of several modeling choices will turn out to be the best. So the user must simply choose one, and proceed half-convinced, always wondering if they made a suboptimal choice. Later on, new knowledge may arise that shares features with something they previously recorded, but the user will inadvertently encode it differently than they did before. Such inconsistencies can be maddening when they are later exposed, and can actually lead to fear and uncertainty.

Part of the difficulty is the fact that much creativity is involved in encoding knowledge at all. Popcorn is obviously not a structured tool that presents the user with, for instance, a template of fields to fill out. One user commented that in many ways he found such "idiot-proof" tools more comforting, because they left no doubt about exactly what kind of input they expected. Another echoed the thoughts of many when he said, "I'm very impressed with the way I see Popcorn working for others. But when I sit down to input something myself, I often just stare at that blank screen and don't know how to proceed." The flexibility itself, it appears, sometimes stifles thought.

There are several ways this could be addressed. One is simply through training. Popcorn is a tool quite unlike most applications users have ever worked with, and it may be too much to ask to simply teach them the basic GUI operations and then leave them to their own devices. A brief "introduction to knowledge mapping through Popcorn" course could explain the principles of concept mapping, plus common techniques and tricks to materialize one's knowledge within the constraints of the tool. Another idea would be to introduce sample skeletal structures

for particular domains. When modeling a new area, users could perhaps browse a central location (or even the Web, with a service discovery approach) to find "Popcorn starter structures" for download. These would contain basic patterns that have been proven successful by other Popcorn users in a particular field, which the user could fill in, imitate, modify, and extend. This would help bootstrap a beginning user by giving them some guidance as to which way to proceed initially.

The difficulty of knowledge formulation

The most significant barrier to using Popcorn, however, and also the most enlightening one, was simply this: knowledge formulation is a more difficult endeavor than the testers (or I) imagined. The hope was that since the mind can be said to store knowledge as a semantic network, allowing users to transcribe that structure directly into electronic form would be easy and painless. What was discovered, however, was that this can be a vexing task, not because the interface is suboptimal, nor because the data model is insufficient, but because *the knowledge users wish to record is often not fully understood*.

It was abundantly clear from user experiences that Popcorn exposes ill-understood knowledge. Its strength is that it allows knowledge to be expressed as a cognitive structure; but the catch is that it essentially *demands* the user enter it in that form. Time and again, users would be frustrated in trying to properly model their knowledge in the tool, and they often reported that their lack of understanding is what really stopped them. They couldn't figure out how to express the knowledge in terms of concepts and relationships because they didn't truly *understand* the concepts and

relationships. The experience was frustrating because Popcorn confronted them with their lack of understanding and forced them to deal with it.

A related user complaint was that it was often troublesome to try and record knowledge "in real time." Many users attempted to create concept maps for newly acquired knowledge even as they were learning it. The idea was that as new concepts and their relationships are encountered, it would be optimal to simply record them as one went along, compiling information into knowledge as it was consumed. Most quickly discovered that this was impossible. It takes time to mull over a difficult domain and ferret out the essential insights, and rushing this process is a certain road to chaos. For this reason, most users spent significant time thinking and doodling before recording a "mostly correct" concept map for a complex domain. Some formed the habit of recording fragments of text in Popcorn notes when a new domain was initially encountered, returning later to process the information more deeply and "compile" it into a concept map of kernels. With this technique, the ratio of notes to kernels on a particular view could serve as a rough indicator of how deeply the information had been processed.

When confronted with difficulty in knowledge formulation, users had two basic responses. They either gave up using the tool for that task out of frustration, or else they took much longer than expected to record their knowledge. Those who took the first course were conscious that they did not understand the topic as well as they thought, and this frank realization became Popcorn's primary contribution for them. The others engaged in more time-consuming investigations and longer brainstorming sessions, after which they invariably emerged with a more thorough grasp of the

domain. Popcorn's value for them was perhaps not that it stored the knowledge itself, but that it dispelled the illusion that they really understood it, and encouraged them to explore it further. A very common remark from users was that although it was sometimes time-consuming to construct a set of kernel views, in the end the process was well worth it.

Summary

In all, Popcorn seems to be a reasonably effective tool in the hands of someone who understands and commits to the data model (including transclusion.) Interestingly, there are a number of different *ways* it can be effective, and this varies greatly depending on the particular user's information needs. In some ways, the tool becomes whatever the user wants it to be, which is both a strength and a weakness. Its strength is that it can serve many purposes and adapt to many different working environments; its weakness is that it does not direct the user towards a course of action, which can be confusing and in some cases deprive the tool of its utility.

A personal knowledge base makes great claims on a user's life – attempting to manage all of their knowledge in every conceivable domain. And yet it can only be effective if it does this without interfering much in the user's daily activities. We are rarely conscious of "using our brain" as we carry out tasks; if a PKB is to be a "surrogate brain" we should ideally be unaware of it, too. What this means practically is that it must smoothly integrate into a user's workflow. It must be available on demand, and in exactly the way the user needs it, but it must never intrude when not wanted or require any explicit upkeep. This is a delicate balance.

One fact brought out by the user testing is that people are sensitive to a number of parameters, including the time taken to process knowledge, the physical location in which knowledge is acquired, the specific tasks that the knowledge is needed for, and simply their existing habits around knowledge management. Attempting to inject an added value into this equation without disturbing the established processes is a difficult operation for any tool to achieve.

As for knowledge itself, the user trial demonstrates two important realities. The first is that non-linear knowledge representations like concept maps are promising. Most users, even without training, seem to understand them, and are even confident that others could look at their diagrams and discern the meaning easily. The second reality, however, is that knowledge itself is often difficult to form, and trying to express it as a concept map exposes that. Users discovered that expressing knowledge to Popcorn can be vexing at times because it is often elusive, fuzzy, and frankly, not fully understood.

Lastly, a tool like Popcorn seems to require a great deal of commitment to be effective. Most users are either "Popcorn geeks" who embrace the tool and mold their whole workflow around it, or else fail to derive any lasting benefit. Apparently, it can only function well as a major component in a person's life, which means the stakes are high indeed.

CHAPTER 6

CONCLUSION: KNOWLEDGE AS A COMMODITY

Personal knowledge bases can be very effective tools in the hands of dedicated users, as the testing phase of this thesis made clear. They require significant discipline, even to the point of being willing to change habits, in order to fulfill the vision outlined in the introduction. But in these cases they can be a tremendous aid to knowledge management and personal effectiveness. Users who fail to make this commitment can still reap some shorter-term benefits, especially in the area of focused knowledge formulation, though the gains are much smaller.

But perhaps the most interesting finding from these experiments has less to do with PKBs themselves, than with what it revealed about the ways humans work with knowledge. A personal knowledge base is intended to be a reflection of a user's mind, and this is a double-edged sword. It enhances our fleeting thoughts with persistence and tangibility, which can be valuable indeed, but these very attributes also *expose* what it is in the mind, often in a surprising and uncomfortable way. I conclude this thesis with some observations about human behavior that this project revealed for me.

We live in a world that moves at breakneck speed. Many of us spend most of our time just trying to keep pace with a swarm of short-term obligations. And we are evaluated – by our managers, our peers, and even ourselves – largely in terms of the concrete tasks we complete. Our overriding concern tends to be how many papers we publish, or deliverables we meet, or clients we see, or sales deals we close. This rate-

of-task-completion may or may not be the best gauge of success, but it is certainly measurable, and our rewards systems tend to be heavily based on it.

Now when we turn to electronic tools, what do we usually want them to do for us? Naturally, to speed up our rate of task completion. The word processor helps us to write documents more quickly. E-mail helps us communicate more quickly. The Internet helps us access information and make purchases more quickly. Databases help us access customer records more quickly. Upon reflection, the underlying goal of a huge majority of software applications is simply to *automate* something for us. We're too busy, it seems, to spend the time on a new, potentially enlightening endeavor. All we want the computer to do for us is to speed up our rate of processing, so that we can continue to execute the tasks we already understand faster and faster.

The above description is a caricature, of course. But it does seem to me that our society is largely predicated upon doing, not knowing. We often fail to value knowledge itself as a commodity, instead valuing only the concrete applications of that knowledge. This sounds innocent enough, and even pragmatic, but it is all too easy for us to start focusing only the completion of the task, rather than the understanding that made the accomplishment possible. Soon, we begin tackling tasks *without* having the knowledge necessary, which leads to faulty decisions and suboptimal solutions. Worse, we may not even realize that these solutions *are* suboptimal, because the only way to evaluate them is by means of the knowledge that we carelessly discarded. We soon find that we are in a race to complete a certain task

in a certain way faster than our competitors, when it may not be the right way, or even the right task.

This phenomenon struck me in a new way during this thesis. When I was explaining Popcorn to the volunteer testers, I found myself describing it in these very terms. It was intended to "let you quickly record your knowledge," I claimed, and to "speed up your access to all your information." The central fact in my sales pitch was automation: Popcorn would help you complete your existing tasks more quickly by putting your knowledge at your fingertips.

What the user testing revealed was that this is not Popcorn's main function at all. For the most part, Popcorn is not a tool that makes simple things quicker and easier. Rather, it's a tool that gives support to tackle *hard* problems, and to encourage the user to work on them. Users were frustrated by trying to quickly enter knowledge, and were brought face-to-face with their lack of understanding. "No," the tool would say if it had a voice, "my job is not to let you hurry up and enter this so you can return to the whirlwind of deadlines. My job is to make you step back for a moment and consider whether you really understand what you think you do."

It is actually a frightening thought that many of us evidently live out our lives unaware that we understand as little as we probably do. It is even more frightening that we regularly make decisions based on this weak foundation. Perhaps the world would be better served if we valued not so much the speed of task completion, but the depth to which we understood the task and pondered the best possible outcome, and even the degree to which we were certain it was the correct task in the first place.

I believe we live in a world that is mostly very free-form and ill-understood, but that has little pockets of well-defined, structured material that we understand well. Much of our lives are spent dealing with these little pockets. We intuitively flock to them because we can understand them and find them comfortable. It's a relief from the chaos, and we take refuge in it. Most of our electronic tools are built around this structured data: account balances in Quicken, contact information in Outlook, perfectly hierarchical directory structures. We operate in these areas because we *can* operate in them, without much risk or confusion.

Popcorn's perplexity is that it ventures into the unknown realm – it challenges the user to make sense of the unfamiliar parts of their world, and to do it in ways that are unfamiliar. It dares them to forge order out of chaos. This requires innovation, courage, determination, and the willingness to recognize mistakes and start over. But the potential reward is considerable: an ever-expanding realm of understanding that provokes new questions and challenges the status quo.

A personal knowledge base is not a complete solution by itself, of course. No mere tool could be. The change would have to be cultural, a fundamental shift in values. But a PKB could support it better than any other application. Because it is *designed* to handle the fluidity of thoughts in the human mind.

I close with a quote from Albert Einstein: "I lived in solitude in the country and noticed how the monotony of a quiet life stimulates the creative mind." Clearly, here was a man who made major discoveries, furthered our understanding, and changed the world for the better. Perhaps it was because he removed himself from the hustle and bustle of shortsighted deadlines and took the time to think things

through. Perhaps it was because he valued knowledge itself as a commodity worth seeking. Perhaps taking a step back from the fray and truly trying to *understand* would be the best way to improve our world.

BIBLIOGRAPHY

1. The Mozilla Foundation. The Firefox Browser. Available at: www.mozilla.org. 2005
2. Grokker 2.1. Available at: www.grokker.com. 2004
3. Sunburst Technology, Inc. HyperStudio 4. Available at: www.hyperstudio.com. 2005
4. Knowledge Interchange Format (KIF) draft proposed American National Standard NCITS.T2/98-004. Available at: http://logic.stanford.edu/kif/kif.html. 1991
5. Solutions Etcetera. SuperCard 4.5. Available at: www.supercard.us. 2005
6. XML TopicMaps (XTM). Available at: http://www.topicmaps.org/xtm/index.html. 2001
7. XPertRule Software Ltd. xPertRule eLearning. Available at: www.xpertrule.com. 2005
8. Aberer, K., Cudre-Mauroux, P. and Hauswirth, M., The Chatty Web: Emergent semantics through gossiping. in *Proceedings of the 12th International World Wide Web Conference*, (Budapest, Hungary, 2002).
9. 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.
10. Agrawal, R., Imielinski, T. and Swami, A., Mining association rules between sets of items in large databases. in *Proceedings of the ACM SIGMOD Conference on Management of Data*, (Washington, D.C., 1993), 207-216.
11. Ahlberg, C., Williamson, C. and Shneiderman, B., Dynamic queries for information exploration: An implementation and evaluation. in *Proceedings of the ACM Conference on Human Factors in Computer Systems*, (Monterey, California, 1992), 619-626.
12. 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).
13. AKS-Labs. Mind Pad v1.1. Available at: www.mind-pad.com. 2005
14. Institute of Computer Science, FORTH. The ICS-FORTH RDFSuite: Managing voluminous RDF description bases. Available at: http://www.ics.forth.gr/isl/publications/paperlink/semweb2001.pdf. 2001
15. Advanced Learning Technologies in Education Consortia (ALTEC). NoteStar. Available at: http://notestar.4teachers.org. 2005
16. Anderson, J.R. *Cognitive Psychology and Its Implications, 3rd Ed.* W.H. Freeman, New York, 1990.
17. Ankerst, M., Ester, M. and Kriegel, H.-P., Towards an effective cooperation of the user and the computer for classification. in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (Boston, Massachusetts, 2000), 178-188.
18. Ankolekar, A., Burstein, M., Hobbs, J.R., Lassila, O., Martin, D., McDermott, D., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T. and Sycara, K., DAML-S: Web Service description for the Semantic Web. in *First International Semantic Web Conference (ISWC 2002)*, (Sardinia, Italy, 2002), 348-363.
19. Antonietti, A. Why does mental visualization facilitate problem-solving? in Denis, M. ed. *Mental Images in Human Cognition*, Elsevier Science Publishing Company, Inc., New York, 1991, 211-227.

20. AquaMinds Software Corporation. NoteTaker 1.9. Available at: www.aquaminds.com. 2005

21. Arens, Y., Knoblock, C.A. and Shen, W.-M. Query Reformulation for Dynamic Information Integration. *Journal of Intelligent Information Systems*, 6 99-130.

22. Assini, P., NESSTAR: A Semantic Web application for statistical data and metadata. in *Eleventh International World Wide Web Conference: Workshop on Real World Applications of RDF and the Semantic Web*, (Honolulu, Hawaii, 2002).

23. Ausubel, D.P. *Educational Psychology: A Cognitive View*. Holt, Rinehart, and Winston, Inc., New York, 1968.

24. Ausubel, D.P. The use of advance organizers in the learning and retention of meaningful verbal material. *Journal of Educational Psychology*, 51 267-272.

25. Ayers, Danny. IdeaGraph. Available at: www.ideagraph.net. 2005

26. Bachler, M., Shum, S.B., De Roure, D., Michaelides, D. and Page, K., Ontological mediation of meeting structure: argumentation, annotation, and navigation. in *Proceedings of the 1st International Workshop on Hypermedia and the Semantic Web*, (Nottingham, UK, 2003).

27. The KartOO Visual Metasearch Engine. Available at: www.kartoo.com. 2004

28. Banxia Software Ltd. Decision Explorer. Available at: http://www.banxia.com/demain.html. 2005

29. Bartlett, F. *Remembering: A Study in Experimental and Social Psychology*. The Syndics of the Cambridge University Press, Cambridge, 1964.

30. Bayardo Jr., R.J., Bohrer, W., Brice, R., Cichocki, A., Fowler, G., Helal, S., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A. and Woelk, D., Infosleuth: Semantic integration of information in open and dynamic environments. in *Proceedings of the 1997 ACM-SIGMOD International Conference on Management of Data*, (Tuscon, Arizona, 1997), Morgan Kaufman, 195-206.

31. World Wide Web Consortium. OWL Web Ontology Language Reference. Available at: http://www.w3.org/TR/2004/REC-owl-ref-20040210/. 2004

32. World Wide Web Consortium. RDF/XML Syntax Specification (Revised). Available at: http://www.w3c.org/RDF/. 2004

33. University of Bristol, UK. Redland RDF Application Framework. Available at: http://www.redland.opensource.ac.uk. 2000

34. Benjamins, V.R., Wielinga, B., Wielemaker, J. and Fensel, D., Towards Brokering Problem-Solving Knowledge at the Internet. in *Eleventh European Knowledge Acquisition Workshop (EKAW-99)*, (Dagstuhl Castle, Germany, 1999).

35. Berendt, B., Hotho, A. and Stumme, G., Towards semantic web mining. in *Proceedings of the First International Semantic Web Conference*, (Sardinia, Italy, 2002), 264-278.

36. Network Working Group: Request for Comment 2396. Uniform Resource Identifiers (URI): Generic Syntax. Available at: http://www.ietf.org/rfc/rfc2396.txt. 1998

37. Berners-Lee, T., Hendler, J. and Lassila, O. The Semantic Web. *Scientific American*, 2001.

38. Bernstein, M. The bookmark and the compass: orientation tools for hypertext users. *ACM SIGOIS Bulletin*, 9, 4 34-45.

39. Bernstein, M., Collages, Composites, Construction. in *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, (Nottingham, UK, 2003).

40. Bertino, E., Catania, B. and Zarri, G.P. *Intelligent Database Systems*. Addison-Wesley, Edinburgh, England, 2001.

41. Bignell, Elliott. HeadCASE: Mind Mapping for Windows. Available at: www.bignell.de. 2005

42. Bitsmith Software. Personal Knowbase. Available at: http://www.bitsmithsoft.com/product.htm. 2005

43. 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.

44. World Wide Web Consortium. XQuery 1.0: An XML Query Language. Available at: http://www.w3.org/TR/xquery/. 2003

45. Axon Research. Axon Idea Processor. Available at: web.singnet.com.sg/~axon2000. 2005

46. Boley, D., Gini, M., Gross, R., Han, E.-H., Hastings, K., Karypis, G., Kumar, V., Mobasher, B. and Moore, J. Document categorization and query generation on the World Wide Web using WebACE. *AI Review*, 13, 5-6 365-391.

47. World Wide Web Consortium. Web Services Architecture. Available at: http://www.w3c.org/TR/ws-arch/. 2004

48. The Bosley Group. MindMapper Professional v4.2. Available at: www.mindmapperusa.com. 2005

49. Bower, G.H. and Clark, M.C. Narrative stories as mediators for serial learning. *Psychonomic Science*, 14 181-182.

50. World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Second Edition). Available at: http://www.w3.org/TR/2000/REC-xml-20001006.html. 2000

51. World Wide Web Consortium. Resource Description Framework (RDF) Schema Specification 1.0. Available at: http://www.w3.org/TR/2000/CR-rdf-schema-20000327/. 2000

52. Broekstra, J., Kampman, A. and van Harmelen, F., Sesame: A generic architecture for storing and querying RDF and RDF schema. in *First International Semantic Web Conference*, (Sardinia Italy, 2002), Springer-Verlag Heidelberg, 54-68.

53. Bruce, H., Jones, W.P. and Dumais, S.T. Information behaviour that keeps found things found. *Information Research*, 10, 1.

54. 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.

55. Buckingham Shum, S., Motta, E. and Domingue, J., Representing scholarly claims in internet digital libraries: a knowledge modelling approach. in *Proceedings of ECDL'99: Third European Conference on Research and Advanced Technology for Digital libraries*, (Paris, France, 1999).

56. 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.

57. Bush, V. As we may think. *The Atlantic Monthly*, 1945, 101-108.

58. Buzan, T. and Buzan, B. *The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential*. Plume Books, 1996.

59. Canas, A.J., Carvalho, M., Arguedas, M., Leake, D.B., Maguitman, A. and Reichherzer, T., Mining the Web to suggest concepts during concept map construction. in *Proceedings of the 1st International Conference on Concept Mapping*, (Pamplona, Spain, 2004).

60. 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.

61. Card, S.K., Robertson, G.G. and York, W., The WebBook and the WebForager: an information workspace for the World Wide Web. in *Proceedings of the 1996 CHI Conference on Human Factors in Computing Systems*, (1996).

62. Carenini, G., An analysis of the influence of need for cognition on dynamic queries usage. in *Proceedings of the Conference on Human Factors in Computing Systems*, (Seattle, Washington, 2001).

63. Carlson, D.A. and Ram, S. HyperIntelligence: the next frontier. *Communications of the ACM*, 33, 3 311-321.

64. Chakrabarti, S. and Batterywala, Y., Mining themes from bookmarks. in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (Boston, Massachusetts, 2000).

65. Chapura, Inc. KeySuite v3.3.2. Available at: http://www.chapura.com/keysuite.php. 2005

66. Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. and Widom, J., The TSIMMIS Project: Integration of Heterogeneous Information Sources. in *16th Meeting of the Information Processing Society of Japan*, (Tokyo, Japan, 1994), 7-18.

67. Chen, Q., Wu, X. and Zhu, X., OIDM: Online interactive data mining. in *Proceedings of the 17th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*, (Ottowa, Canada, 2004).

68. Chronos. StickyBrain 3. Available at: www.chronosnet.com. 2005

69. Texas Tech University. A Conceptual Model of an Adaptive Decision Support System (ADSS). Available at: http://hsb.baylor.edu/ramsower/ais.ac.96/papers/chuang.htm. 1996

70. Circus Ponies. NoteBook 2.0. Available at: www.circusponies.com. 2005

71. Claro Software. MindFull V2. Available at: http://clarosoftware.com. 2005

72. CoCo Systems Ltd. VisiMap Professional 4.0. Available at: www.visimap.com. 2005

73. Cognitive-Tools. Easy-Mapping-Tool. Available at: www.cognitive-tools.com. 2005

74. Cohn, D. Learning the structure of unstructured document bases. Microsoft Research, Seattle, Washington, 2001.

75. Collins, A.M. and Loftus, E.F. A spreading-activation theory of semantic processing. *Psychological Review*, 82 407-428.

76. Collins, A.M. and Quillian, M.R. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Memory*, 8 240-247.

77. The Compendium Institute. Compendium Version 1.3.04. Available at: www.compendiuminstitute.org. 2005

78. Conklin, J. Hypertext: an introduction and survey. *Computer*, 20, 9.(Sept 1987), 17-41.

79. 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.

80. 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).

81. Conway, M.A., Kahney, H., Bruce, K. and Duce, H. Imaging objects, routines, and locations. in Denis, M. ed. *Mental Images in Human Cognition*, Elsevier Science Publishing Company, Inc., New York, 1991, 171-182.

82. Cooley, R., Mobasher, B. and Srivastava, J., Web mining: Information and pattern discovery on the World Wide Web. in *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence*, (1997).

83. 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.

84. Craven, M., Freitag, D., McCallum, A., Mitchell, T., Nigam, K. and Quek, C.Y., Learning to extract symbolic knowledge from the World Wide Web. in *Proceedings of the 15th National Conference on Artificial Intelligence*, (Madison, Wisconsin, 1998), 509-516.

85. Daconta, M.C., Orbrst, L.J. and Smith, K.T. *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. John Wiley & Sons, 2003.

86. Davies, J., Duke, A. and Stonkus, A., OntoShare: Using ontologies for knowledge sharing. in *WWW2002 Semantic Web workshop, Eleventh International World Wide Web Conference*, (Honolulu, Hawaii, 2002).

87. Davies, J., Fensel, D. and van Harmelen, F. (eds.). *Towards the Semantic Web: Ontology-Driven Knowledge Management*. John Wiley & Sons, Ltd., 2003.

88. Davies, J., Krohn, U. and Weeks, R., QuizRDF: search technology for the semantic web. in *WWW2002 workshop on real world RDF & Semantic Web Applications, 11th International WWW Conference*, (Hawaii, USA, 2002).

89. Davies, S., Leveraging metadata inductively and subjectively. in *Proceedings of the Fourth International Conference on Dublin Core and Metadata Applications*, (Shanghai, China, 2004), 163-167.

90. Davies, S. and King, R., Crossing the objective-subjective divide in information space organization. in *Proceedings of the Eighth Joint Conference on Information Sciences*, (Salt Lake City, Utah, 2005).

91. Davies, S., Velez-Morales, J. and King, R., Building the Memex sixty years later: trends and directions in personal knowledge bases. University of Colorado Technical Report CU-CS-997-05. Available at: http://www.cs.colorado.edu/department/publications/reports/docs/CU-CS-997-05.pdf. 2005

92. 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.

93. Davis, R.C., Brotherton, J.A., Landay, J.A., Price, M.N. and Schilit, B.N., NotePals: lightweight note taking by the group, for the group. Available at. 1998

94. Dede, C.J. and Jayaram, G., Designing a training tool for imaging mental models. Available at. 1990

95. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T. and Harshman, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41, 6 391-407.

96. 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.

97. DEVONtechnologies. DEVONthink Personal Edition. Available at: http://www.devon-technologies.com/. 2005

98. Di Giacomo, M., Mahoney, D., Bollen, J., Monroy-Hernandez, A. and Meraz, C.M.R. MyLibrary, a personalization service for digital library environments.

99. Diaz, G. Decision Explorer. *Social Science Computer Review*, 18, 3 344-350.

100. diSessa, A.A. and Abelson, H. Boxer: a reconstructible computational medium. *Communications of the ACM*, 29, 9 859-868.

101. 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.

102. The Dublin Core Metadata Initiative. Available at: www.dublincore.org. 2005

103. 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.

104. Dvorak, Martin. MindRaider. Available at: mindraider.sourceforge.net. 2005

105. Electric Pocket, Inc. BugMe! Notepad. Available at: http://www.electricpocket.com/bugme-palm/. 2005

106. Enfish Software, Campbell, California. Enfish Find. Available at: http://www.enfish.com/Products_Find.asp. 2005

107. 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.

108. Engels, R.H.P. and Lech, T.C. Generating ontologies for the Semantic Web: OntoBuilder. in van Harmelen, F. ed. *Towards the Semantic Web: Ontology-Driven Knowledge Management*, John Wiley & Sons, Ltd., 2003.

109. Erdmann, M. and Studer, R. How to structure and access XML documents with ontologies. *Data Knowledge Engineering*, 36 317-335.

110. Etzel, B. and Thomas, P.J. *Personal Information Management: Tools and Techniques for Achieving Professional Effectiveness*. New York University Press, New York, 1996.

111. EvolutionCode Pty Ltd. RecallPlus V3. Available at: www.recallplus.com. 2005

112. Fails, J.A. and Olsen, D.R., Interactive machine learning. in *Proceedings of the Eighth International Conference on Intelligent User Interfaces*, (Miami, Florida, 2003), 39-45.

113. Farquhar, A., Fikes, R. and Rice, J. The Ontolingua Server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46, 6 707-727.

114. 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.

115. Fensel, D., Benjamins, V.R., Motta, E. and Wielinga, B., UPML: A framework for knowledge system reuse. in *Proceedings of the 16th International Joint Conference on AI*, (Sweden, 1999), 16-23.

116. Vrije Universiteit Amsterdam. The Web Service Modeling Framework WSMF Extended abstract. Available at: http://informatik.uibk.ac.at/~c70385/wese/wsmf.bis2002.pdf. 2002

117. Fensel, D., Hendler, J., Lieberman, H. and Wahlster, W. (eds.). *Spinning the Semantic Web*. The MIT Press, Cambridge, Massachusetts, 2003.

118. Fensel, D., Horrocks, I., van Harmelen, F., McGuinness, D. and Patel-Schneider, P.F. OIL: Ontology infrastructure to enable the Semantic Web. *IEEE Intelligent Systems*, 16, 2.(March/April 2001), 38-45.

119. Fensel, D., Staab, S., Studer, R., van Harmelen, F. and Davies, J. A future perspective: Exploiting peer-to-peer and the Semantic Web for knowledge management. in van Harmelen, F. ed. *Towards the Semantic Web: Ontology-Driven Knowledge Management*, John Wiley & Sons, Ltd., 2003.

120. World Wide Web Consortium. Scalable Vector Graphics (SVG) 1.1 Specification. Available at: http://www.w3.org/TR/SVG11/. 2003

121. 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.

122. Fluit, C., ter Horst, H., van der Meer, J., Sabou, M. and Mika, P. Spectacle. in van Harmelen, F. ed. *Towards the Semantic Web: Ontology-Driven Knowledge Management*, John Wiley & Sons, Ltd., 2003.

123. The Friend of a Friend (FOAF) Project. Available at: www.foaf-project.org. 2005

124. FranklinCovey. PlanPlus 4.0 for Windows XP. Available at: www.franklincovey.com. 2005

125. Freebyte. TreePad Business Edition 7.1.7. Available at: www.treepad.com. 2005

126. Freeman, E. and Gelernter, D. Lifestreams: a storage model for personal data. *ACM SIGMOD Record*, 25, 1.(March 1996), 80-86.

127. FreeMind. Available at: http://freemind.sourceforge.net. 2005

128. Frye, C., Plusch, M. and Lieberman, H. Static and Dynamic Semantics of the Web. in Wahlster, W. ed. *Spinning the Semantic Web*, The MIT Press, Cambridge, Massachusetts, 2003.

129. 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.

130. Gael Ltd. MindGenius Home v2.22. Available at: www.mindgenius.com. 2005

131. Gaines, B.R. and Shaw, M.L.G. Concept maps as hypermedia components. *International Journal of Human Computer Studies*, 43, 3 323-361.

132. Gaines, B.R. and Shaw, M.L.G., WebMap: concept mapping on the web. in *Proceedings of the Fourth International WWW Conference*, (1995).

133. 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.

134. 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.

135. Gentner, D. and Stevens, A.L. (eds.). *Mental Models*. Lawrence Erlbaum Associates, Inc., New Jersey, 1983.

136. Getoor, L., Friedman, N., Koller, D. and Pfeffer, A. Learning probabilistic relational models. in Lavrac, N. ed. *Relational Data Mining*, Springer-Verlag, 2001.

137. Ghani, R., Jones, R., Mladenic, D., Nigam, K. and Slattery, S., Data mining on symbolic knowledge extracted from the Web. in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (Boston, Massachusetts, 2000).

138. Gil, Y. and Ratnakar, V., TRELLIS: An interactive tool for capturing information analysis and decision making. in *Proceedings of Ontologies and the Semantic Web: 13th International Conference*, (Siguenza, Spain, 2002).

139. Godwin-Jones, B. Blogs and wikis: environments for on-line collaboration. *Language Learning and Technology*, 7, 2.(May 2003), 12-16.

140. The Mozilla Foundation. XML User Interface Language (XUL) version 1.0. Available at: http://www.mozilla.org/projects/xul/xul.html. 2001

141. Goodman, D. *The Complete Hypercard Handbook*. Bantam Books, New York, 1988.

142. Grosky, W.I., Sreenath, D.V. and Fotouhi, F. Emergent semantics and the multimedia semantic web. *SIGMOD Record*, 31, 4 54-58.

143. Gruber, T., Ontolingua: A mechanism to support portable ontologies. Available at. 1992

144. World Wide Web Consortium. SOAP Version 1.2. Available at: http://www.w3.org/TR/2003/REC-soap12-part1-20030624/. 2003

145. Halasz, F.G. Reflections on NoteCards: seven issues for the next generation of hypermedia systems. *Communications of the ACM*, 31, 7 836-852.

146. Halasz, F.G., Moran, T.P. and Trigg, R.H. NoteCards in a Nutshell. *ACM SIGCHI Bulletin*, 17 45-52.

147. Halasz, F.G. and Schwartz, M. The Dexter hypertext reference model. *Communications of the ACM*, 37, 2.(February 1994), 30-39.

148. Harman, G. *Thought*. Princeton University Press, Princeton, New Jersey, 1973.

149. Hastie, T., Tibshirani, R. and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, New York, 2001.

150. 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.

151. 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.

152. Hearst, M.A., Levy, A.Y., Knoblock, C.A., Minton, S. and Cohen, W. Information integration. *IEEE Intelligent Systems*, 13, 5.(Sept./Oct. 1998), 12-24.

153. Heflin, J. and Hendler, J., Searching the Web with SHOE. in *Artificial Intelligence for Web Search. Papers from the AAAI Workshop*, (Menlo Park, CA, 2000), AAAI/MIT Press, 35-40.

154. Hendler, J. and McGuinness, D. The DARPA Agent Markup Language (DAML). *IEEE Intelligent Systems*, 15, 6.(November/December 2000), 67-73.

155. Hendry, D.G. and Harper, D.J. An informal information-seeking environment. *Journal of the American Society for Information Science*, 48, 11 1036-1048.

156. Hicks, D.L. and Tochtermann, K., Personalizing information spaces: a metadata-based approach. in *Proceedings of the International Conference on Dublin Core and Metadata Applications*, (Tokyo, Japan, 2001), 213-220.

157. Hog Bay Software. Hog Bay Notebook v3.5. Available at: www.hogbaysoftware.com. 2005

158. Horrocks, I. and Patel-Schneider, P.F. Optimizing Description Logic Subsumption. *Journal of Logic and Computation*, 9, 3 267-293.

159. 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).

160. Hypersoft-net. Knowledge Manager v8.4. Available at: www.knowledgemanager.us. 2005

161. IBM Lotus Software. IBM Lotus Notes 6.5. Available at: http://www.lotus.com/products/product4.nsf/wdocs/noteshomepage. 2005

162. Inspiration Software, Inc. Inspiration 7.6. Available at: www.inspiration.com. 2005

163. Intelligents, LLC. NoteWorthy. Available at: http://www.intelli-gents.com/noteworthy.htm. 2001

164. Iosif, V., Mika, P., Larsson, R. and Akkermans, H. Field experimenting with Semantic Web tools in a virtual organization. in van Harmelen, F. ed. *Towards the Semantic Web: Ontology-Driven Knowledge Management*, John Wiley & Sons, Ltd., 2003.

165. Jain, R., Emergent semantics and experiential computing. Available at: http://lsdis.cs.uga.edu/SemNSF/Jain-Position.doc. 2000

166. James, W. *The Principles of Psychology*. Harvard University Press, Cambridge, 1890.

167. Jasper, R. and Uschold, M. Enabling Task-Centered Knowledge Support through Semantic Markup. in Wahlster, W. ed. *Spinning the Semantic Web*, The MIT Press, Cambridge, Massachusetts, 2003.

168. First Monday. Finders, keepers? The present and future perfect in support of personal information management. Available at: http://www.firstmonday.dk/issues/issue9_3/jones/index.html. 2004

169. 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.

170. Jones, W.P. and Dumais, S.T. The Spatial Metaphor for User Interfaces: Experimental Tests of Reference by Location versus Name. *ACM Transactions on Office Information Systems*, 4, 1 42-63.

171. 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.

172. Karvounarakis, G., Christophides, V. and Plexousakis, D., Querying Semistructured (Meta) Data and Schemas on the Web: The case of RDF & RDFS. Available at: http://www.ics.forth.gr/isl/publications/paperlink/querying-semistructured-metadata-and.pdf. 2000

173. Kifer, M., Lausen, G. and Wu, J. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42, 4.(July 1995), 741-843.

174. Kintsch, W. *Learning, Memory, and Conceptual Processes*. John Wiley & Sons Inc., 1970.

175. Kiryakov, A. and Ognyanov, D., Tracking changes in RDF(S) repositories. in *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management*, (Siguenza, Spain, 2002).

176. Klas, W., Aberer, K. and Neuhold, E. Object-oriented modeling for hypermedia systems using the VODAK modeling language (VML). in Sellis, T. ed. *Advances in Object-Oriented Database Systems*, Springer-Verlag, Berlin, 1993.

177. Kosslyn, S. *Ghosts in the Mind's Machine*. W.W.Norton & Co., New York, 1983.

178. 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.

179. Koy, A.K., Computer Aided Thinking. in *Proceedings of the 7th International Conference on Thinking*, (Singapore, 1997).

180. Kushmerick, N. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118, 1-2 15-68.

181. Landauer, T. and Dumais, S.T. A solution to Plato's problem: The Latent Semantic Analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104, 2 211-240.

182. 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

183. Le Grande, B., Soto, M. and Dodds, D., XML Topic maps and semantic web mining. in *XML Conference and Exposition 2001*, (Orlando, Florida, 2001), IdeAlliance.

184. Leger, A., Michel, G., Barrett, P., Gitton, S., Gomez-Perez, A., Lehtola, A., Mokkila, K., Rodrigez, S., Sallantin, J., Varvarigou, T. and Vinesse, J., Ontology domain modeling support for multilingual services in e-commerce: MKBEEM. in *Fourteenth European Conference on Artificial Intelligence (ECAI-'00): Workshop on Applications of Ontologies and Problem-Solving Methods*, (Berlin, Germany, 2000).

185. Lenat, D.B. and Guha, R.V. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, Boston, 1990.

186. Levine, J., Tate, A. and Dalton, J. O-P^3: Supporting the planning process using open planning process panels. *IEEE Intelligent Systems*, 15, 5.(Sept/Oct 2000), 56-62.

187. Levy, A.Y., Rajaraman, A. and Ordille, J.J., Querying heterogeneous information sources using source descriptions. in *Proceedings of 22th International Conference on Very Large Databases*, (Mumbai, India, 1996), Morgan Kaufmann, 251-262.

188. Micro Logic. Info Select 8. Available at: http://www.miclog.com/is/index.shtml. 2005

189. Lewis, P.M., Bernstein, A. and Kifer, M. *Databases and Transaction Processing: An Application-oriented Approach*. Addison-Wesley, 2002.

190. Lieberman, H. Personal assistants for the web: an MIT perspective. in Klusch, M. ed. *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet*, Springer-Verlag, Berlin, 1999, 279-292.

191. Lieberman, H. and Selker, T. Out of context: computer systems that adapt to, and learn from context. *IBM Systems Journal*, 39, 3 617-631.

192. IEEE Distributed Systems Online. Amazon.com recommendations: item-to-item collaborative filtering. Available at: http://dsonline.computer.org/0301/d/w1lind.htm. 2003

193. Liu, B., Chin, C.W. and Ng, H.T., Mining topic-specific concepts and definitions on the web. in *Proceedings of the 12th International World Wide Web Conference*, (Budapest, Hungary, 2003), ACM, 251-260.

194. Logie, R.H. and Marchetti, C. Visio-spatial working memory: visual, spatial, or central executive? in Denis, M. ed. *Mental Images in Human Cognition*, Elsevier Science Publishing Company, Inc., New York, 1991, 105-115.

195. Lorayne, H. and Lucas, J. *The Memory Book*. Stein and Day, New York, 1974.

196. SHOE 1.01 (proposed specification). Available at: http://www.cs.umd.edu/projects/plus/SHOE/spec.html. 2000

197. University of California at Berkeley School of Information Management and Systems. How Much Information. Available at: http://www.sims.berkeley.edu/research/projects/how-much-info-2003/. 2003

198. MacGregor, R.M. Inside the LOOM Description Classifier. *SIGART Bulletin*, 2, 3 88-92.

199. Retrospective on Loom. Available at: http://www.isi.edu/isd/LOOM/papers/macgregor/Loom_Retrospective.html. 1999

200. MacNamara, J. *Names for Things: A Study of Human Learning*. The MIT Press, Cambridge, 1982.

201. Maedche, A. and Staab, S., Discovering conceptual relations from text. in *Proceedings of the 14th European Conference on Artificial Intelligence*, (Berlin, 2000), IOS Press.

202. Maedche, A. and Staab, S., Measuring similarity between ontologies. in *Proceedings of the 13th European Conference on Knowledge Engineering and Knowledge Management*, (Madrid, Spain, 2002).

203. Maedche, A. and Staab, S., Mining Ontologies from Text. in *Knowledge Acquisition, Modeling, and Management: Proceedings of the European Knowledge Acquisition Conference*, (Berlin, 2000), Springer-Verlag.

204. Maedche, A., Staab, S., Stojanovic, Studer, R. and Sure, Y., SEAL - A framework for developing SEmantic portALs. in *Proceedings of the 18th British National Conference on Databases*, (Oxford, UK, 2001), Springer-Verlag, 1-22.

205. Mantei, M.M. Disorientation behavior in person-computer interaction. *Communications Department*, University of Southern California, 1982.

206. Marshall, C., The future of annotation in a digital (paper) world. Available at. 1998

207. Marshall, C. NoteCards in the age of the web: practice meets perfect. *ACM Journal of Computer Documentation*, 25, 3.(August 2001), 96-103.

208. 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.

209. Marshall, C. and Shipman, F. Spatial hypertext: designing for change. *Communications of the ACM*, 38, 8 88-97.

210. Masuda, Y., Ishitobi, Y. and Ueda, M., Frame-axis model for automatic information organizing and spatial navigation. in *Proceedings of the 1994 ACM European Conference on Hypermedia Technology*, (Edinburgh, Scotland, 1994), 146-157.

211. Matheus, C.J., Piatetsky-Shapiro, G. and McNeill, D. Selecting and reporting what is interesting. in Uthurusamy, R. ed. *Advances in Knowledge Discovery and Data Mining*, AAAI Press/The MIT Press, Menlo Park, California, 1996, 495-516.

212. McGuinness, D., Fikes, R., Rice, J. and Wilder, S., An Environment for Merging and Testing Large Ontologies. in *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*, (Breckenridge, Colorado, 2000), Morgan Kaufmann, 483-493.

213. FindUR: Knowledge-enhanced online search. Available at: http://www.research.att.com/~dlm/papers/findur-chi98.ps. 1998

214. Mena, E., Illarramendi, A., Kashyap, V. and Sheth, A., OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. in *Conference on Cooperative Information Systems*, (Eliat, Israel, 2000).

215. Micro Logic. Info Select for Windows, Version 8. Available at: http://www.miclog.com. 2005

216. Microsoft Corporation. Microsoft Outlook 2003. Available at: www.microsoft.com/outlook. 2003

217. Microsoft Corporation. OneNote 2003. Available at: http://www.microsoft.com/office/onenote/prodinfo/default.mspx. 2003

218. Miller, G.A. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63 81-97.

219. Mind Technologies. Visual Mind 7, Business Edition. Available at: www.visual-mind.com. 2005

220. Mindjet. MindManager X5 Pro. Available at: www.mindjet.com. 2005

221. MindManager(R) by MindJET LLC. Test version V5.2.344. Available at. 2004

222. Missiontrek. Cartagio Home. Available at: http://www.missiontrek.com/cartagio/chome.asp. 2005

223. Montes-y-Gomez, M., Gelbukh, A. and Lopez-Lopez, A., Comparison of Conceptual Graphs. in *Proceedings of the First Mexican International Conference on Artificial Intelligence*, (Acapulco, Mexico, 2000), Springer-Verlag, 548-556.

224. Flying Meat, Inc. VoodooPad v2.0.2. Available at: http://www.flyingmeat.com/voodoopad/. 2005

225. Musen, M.A., Fergerson, R.W., Grosso, W.E., Noy, N.F., Crubezy, M. and Gennari, J.H., Component-Based Support for Building Knowledge-Acquisition Systems. in *Conference on Intelligent Information Processing of the International Federation for Information Processing World Computer Congress*, (Beijing, 2000).

226. 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.

227. Neches, R., Abhinkar, S., Hu, F., Eleish, R., Ko, I.-Y., Yao, K.-T., Zhu, Q. and Will, P. Collaborative information space analysis tools. *D-Lib Magazine*, 1998.

228. Nelson, T.H. The heart of connection: hypermedia unified by transclusion. *Communications of the ACM*, 38, 8 31-33.

229. 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.

230. 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.

231. NetManage, Inc. Ecco Pro. Available at: www.netmanage.com. 1997

232. Netscape Communications Corporation. The Open Directory Project. Available at: dmoz.org. 2004

233. 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.

234. Nosleep Software. Idea Pad 3.1. Available at: www.nosleep.net. 2005

235. Novak, J.D. The promise of new ideas and new technology for improved teaching and learning. *Cell Biology Education*, 2 122-132.

236. Novak, J.D., The theory underlying concept maps and how to construct them. Available at: Available at http://cmap.coginst.uwf.edu/info. 2003

237. NovaMind Software Pty Ltd. NovaMind v2.4.5. Available at: www.nova-mind.com. 2005

238. Novell, Inc. Novell Evolution 2. Available at: http://www.novell.com/products/desktop/features/evolution.html. 2005

239. Nyce, J.M. and Kahn, P. *From Memex to hypertext*. Academic Press, Boston, 1991.

240. Oberon, askSam Systems. Citation. Available at: http://www.citationonline.net/brochure.asp. 2005

241. Computer Systems Odessa. ConceptDraw MINDMAP v3.5. Available at: conceptdraw.com. 2005

242. O'Hara, K. and Sellen, A., A comparison of reading paper and on-line documents. in *Proceedings of CHI-97, Special Interest Group on Computer and Human Interaction*, (1997), 335-342.

243. Omni Development, Inc. OmniGraffle 3. Available at: http://www.omnigroup.com/applications/omnigraffle/. 2005

244. Omni Development, Inc. OmniOutliner 3. Available at: http://www.omnigroup.com/applications/omnioutliner/. 2005

245. Oostendorp, K.A., Punch, W.F. and Wiggins, R.W., A Tool for individualizing the web. in *Proceedings of the 2nd International Conference on the World-Wide Web*, (Chicago, IL, 1994), 49-57.

246. Open Source Applications Foundation (OSAF). Chandler. Available at: www.osafoundation.org. 2005

247. Ostertag, E., Hendler, J., Ruben, P.-D. and Braun, C. Computing similarity in a reuse library system: an AI-based approach. *ACM Transactions on Software Engineering and Methodology*, 1, 3.(July 1992), 205-228.

248. Owsinski, J. Software review: "Decision Explorer" and "Frontier Analyst" from Banxia Software. *Control and Cybernetics*, 30, 4 479-485.

249. 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.

250. KRSS Group of the ARPA Knowledge Sharing Effort. Description-Logic Knowledge Representation System Specification. Available at: http://www-db.research.bell-labs.com/user/pfps/papers/krss-spec.ps. 1993

251. 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.

252. 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.

253. Peters, R., Exploring the design space for personal information management tools. in *Proceedings of the 18th Conference on Human Factors in Computing Systems*, (Seattle, Washington, 2001), 413-414.

254. Pirolli, P. and Card, S.K. Information foraging. *Psychological Review*, 106, 4.(1999), 643-675.

255. Pirolli, P. and Card, S.K., Information foraging models of browsers for very large document spaces. in *Proceedings of AVI'98, the Working Conference on Advanced Visual Interfaces*, (L'Aquila, Italy, 1998), ACM Press, 83-93.

256. Pirolli, P., Pitkow, J. and Rao, R., Silk from a sow's ear: extracting usable structures from the Web. in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, (Vancouver, British Columbia, 1996), 118-125.

257. Pompi, K.F. and Lachman, R. Surrogate processes in the short-term retention of connected discourse. *Journal of Experimental Psychology*, 75 143-150.

258. Popsecul, A., Ungar, L.H., Pennock, D.M. and Lawrence, S., Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. in *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, (San Francisco, 2001), Morgan Kaufmann.

259. Posner, M.I. Abstraction and the process of recognition. in Bower, G.H. ed. *The Psychology of Learning and Motivation, III*, Academic Press, New York, 1969.

260. Powers, S. *Practical RDF*. O'Reilly & Associates, Inc., Sebastopol, California, 2003.

261. Quan, D., Bakshi, K., Huynh, D. and Karger, D., User interfaces for supporting multiple categorization. in *Proceedings of INTERACT 2003*, (2003).

262. Quillian, M.R. Semantic memory. in *Semantic Information Processing*, The MIT Press, Cambridge, Massachusetts, 1968.

263. Rada, R., Mili, H., Bicknell, E. and Blettner, M. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19, 1.(Jan/Feb 1989), 17-30.

264. Ralston Technology Group. Clarity 2.0. Available at: www.ralstontech.com. 2005

265. Raymond, D.R., Canas, A.J., Tompa, F.W. and Safayeni, F. Measuring the Effectiveness of Personal Database Structures. *International Journal of Human Computer Studies*, 31.(Sept. 1989), 237-256.

266. 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.

267. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J., GroupLens: an open architecture for collaborative filtering of netnews. in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, (Chapel Hill, North Carolina, 1994), 175-186.

268. 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.

269. Robin, F. and Denis, M. Description of perceived or imagined spatial networks. in Denis, M. ed. *Mental Images in Human Cognition*, Elsevier Science Publishing Company, Inc., New York, 1991, 141-152.

270. Rocha, L.M. and Joslyn, C., Simulations of embodied evolving semiosis: emergent semantics in artificial environments. in *Proceedings of the 1998 Conference on VirtualWorlds and Simulation*, (1998), The Society for Computer Simulation International, 233-238.

271. Rowley, J. and Farrow, J. *Organizing Knowledge: An Introduction to Managing Access to Information, 3rd Edition*. Gower Publishing Limited, Hampshire, England, 2000.

272. Russell, S.J. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

273. Sagonas, K., Swift, T. and Warren, D.S., XSB as an efficient deductive database engine. in *Proceedings of the 1994 ACM SIGMOD International Conference on Very Large Databases*, (Athens, Greece, 1994), Morgan Kaufmann, 266-275.

274. Sahuguet, A. and Azavant, F. Building intelligent Web applications using lightweight wrappers. *Data Knowledge Engineering*, 36, 3 283-316.

275. Santini, S., Gupta, A. and Jain, R. Emergent semantics through interaction in image databases. *IEEE Transactions on Knowledge and Data Engineering*, 13, 3.(May/June 2001), 337-351.

276. Santini, S. and Jain, R. Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21, 9.(Sept 1999), 871-883.

277. Sayer, P. *Understanding Hypertext: Concepts and Applications*. Windcrest Books, Blue Ridge Summit, Pennsylvania, 1991.

278. 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.

279. Schohn, G. and Cohn, D., Less is more: active learning with support vector machines. in *Proceedings of the 17th International Conference on Machine Learning*, (San Francisco, California, 2000), Morgan Kaufmann, 839-846.

280. 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.

281. Schultz, J., Cantrill, S. and Morgan, K., An initial operational problem oriented medical record system. in *AFIPS Proceedings*, (1971), 239-264.

282. Selvin, A.M. Supporting collaborative analysis and design with hypertext functionality. *Journal of Digital Information*, 1, 4.

283. 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).

284. Seyer, P. *Understanding Hypertext: Concepts and Applications*. Windcrest Books, Blue Ridge Summit, Pennsylvania, 1991.

285. Shahabi, C. and Chen, Y.-S., Web information personalization: challenges and approaches. in *Proceedings of the Third International Workshop on Databases in Networked Information Systems*, (Aizu-Wakamatsu, Japan, 2003).

286. Shardanand, U. and Maes, P., Social information filtering: algorithms for automating "word of mouth". in *Proceedings of the CHI-95 Conference*, (Denver, Colorado, 1995), ACM Press.

287. Shepard, R. and Cooper, L. *Mental Images and Their Transformations*. The MIT Press, Cambridge, 1982.

288. Sheth, A., Data Semantics: what, where and how? in *Database applications semantics: proceedings of the 6th IFIP Working Conference on Database Applications Semantics*, (Atlanta, Georgia, 1995), Chapman and Hall, 601-610.

289. Shipman, F., Hsieh, H. and Airhart, R., Analytic workspaces: supporting the emergence of interpretation in the Visual Knowledge Builder. Available at: Available at http://www.csdl.tamu.edu/~shipman/vkb/vkb.html. 2000

290. shirky.com. The Semantic Web, syllogism, and worldview. Available at: http://www.shirky.com/writings/semantic_syllogism.html. 2003

291. Shneiderman, B. Creativity support tools. *Communications of the ACM*, 45, 10 116-120.

292. Shneiderman, B. Dynamic queries for visual information seeking. *IEEE Software*, 11, 6 70-77.

293. University of Maryland. Information visualization: dynamic queries, starfield displays, and LifeLines. Available at: http://www.cs.umd.edu/hcil/members/bshneiderman/ivwp.html. 2000

294. Shortliffe, E.H., Perreault, L.E., Wiederhold, G. and Fagan, L.M. (eds.). *Medical Informatics: Computer Applications in Health Care and Biomedicine*. Springer-Verlag, New York, 2001.

295. Sintek, M. and Decker, S., TRIPLE - A query, inference, and transformation language for the semantic web. in *Proceedings of the First International Semantic Web Conference*, (Sardinia, Italy, 2002), 364-378.

296. SMART Technologies, Inc. SMART Ideas Concept-mapping Software. Available at: www2.smarttech.com. 2005

297. 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.

298. Sowa, J.F. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley Publishing Company, 1984.

299. Srikant, R. and Agrawal, R. Mining generalized association rules. *Future Generation Computer Systems*, 13, 2 161-180.

300. Staab, S., Angele, J., Decker, S., Erdmann, M., Hotho, A., Maedche, A., Schnurr, H.-P., Studer, R. and Sure, Y. Semantic community Web portals. *Computer Networks*, 33, 1-6.(June 2000), 473-491.

301. Staab, S. and Maedche, A. Knowledge Portals - Ontologies at work. *AI Magazine*, 2001.

302. Staab, S., Maedche, A. and Handschuh, S., An annotation framework for the semantic web. in *Proceedings of the First Workshop on Multimedia Annotation*, (Tokyo, Japan, 2001).

303. Staab, S., Santini, S., Nack, F., Steels, L. and Maedche, A. Emergent Semantics. *IEEE Intelligent Systems, Trends and Controversies*, 2002, 78-86.

304. Staab, S. and Schnurr, H.-P. Smart task support through proactive access to organizational memory. *Knowledge Based Systems*, 13, 5 251-260.

305. The StayAtPlay Co. Idea Knot. Available at: www.stayatplay.com. 2005

306. Stoffel, K., Taylor, M. and Hendler, J., Efficient Management of Very Large Ontologies. in *Proceedings of the American Association for Artificial Intelligence Conference*, (Providence, Rhode Island, 1997), AAAI/MIT Press, 442-447.

307. Strehl, A., Ghosh, J. and Mooney, R., Impact of similarity measures on web-page clustering. in *Proceedings of the 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search*, (Austin, Texas, 2000), 58-64.

308. Takkinen, J. and Shahmehri, N. Are you busy, cool, or just curious? -- CAFE: A model with three different states of mind for a user to manage information in electronic mail. *Human IT*, 2, 1.(March 1998).

309. Tate, A. Roots of SPAR - Shared Planning and Activity Representation. *Knowledge Engineering Review*, 13, 1 121-128.

310. Thacker, S., Sheth, A. and Patel, S. Complex Relationships for the Semantic Web. in Wahlster, W. ed. *Spinning the Semantic Web*, The MIT Press, Cambridge, Massachusetts, 2003.

311. TheBrain Technologies Corporation. PersonalBrain v3.02. Available at: www.thebrain.com. 2005

312. Thuraisingham, B. *XML Databases and the Semantic Web*. CRC Press, 2002.

313. Tilley, S.R., Whitney, M.J., Muller, H.A. and Storey, M.-A.D., Personalized information structures. in *Proceedings of the 11th Annual International Conference on Systems Documentation*, (Waterloo, Ontario, 1993), 325-337.

314. Tong, S. and Koller, D., Support vector machine active learning with applications to text classification. in *Proceedings of the 17th International Conference on Machine Learning*, (San Francisco, California, 2000), Morgan Kaufmann, 999-1006.

315. Travers, M., A visual representation for knowledge structures. in *Proceedings of the Second Annual ACM Conference on Hypertext*, (Pittsburgh, Pennsylvania, 1989), 147-158.

316. Trigg, R.H. and Weiser, M. TEXTNET: a network-based approach to text handling. *ACM Transactions on Information Systems*, 4, 1 1-23.

317. TruTamil, LLC. ndxCards. Available at: ndxcards.com. 2005

318. Tulving, E. Subjective organization and effects of repetition in multitrial free-recall learning. *Journal of Verbal Learning and Verbal Behavior*, 5 193-197.

319. Tversky, A. Features of similarity. *Psychological Review*, 84, 4.(July 1977), 327-352.

320. Uren, V., Buckingham Shum, S., Li, G. and Bachler, M., Sensemaking tools for understanding research literatures: design, implementation, and user evaluation. Available at: Available at http://kmi.open.ac.uk/projects/scholonto/docs/ScholOnto-Evaluation-2004.pdf. 2004

321. Vdovjak, R. and Houben, G.-J., RDF-Based Architecture for Semantic Integration of Heterogeneous Information Sources. in *Workshop on Information Integration on the Web*, (2001), 51-57.

322. Vega, J., Gomez-Perez, A., Tello, A. and Pinto, H., How to find suitable ontologies using an ontology-based WWW broker. in *Proceedings of the International Work-Conference on Artificial and Natural Neural Networks*, (Alicante, Spain, 1999), Springer, 725-739.

323. The World Wide Web Consortium. Available at: www.w3c.org. 2004

324. The Wikimedia Foundation. Wikipedia. Available at: en.wikipedia.org. 2005

325. Wanner, H.E. On remembering, forgetting, and understanding sentences: A study of the deep structure hypothesis. Harvard University, 1968.

326. Want, R., Pering, T., Danneels, G., Kumar, M., Sundar, M. and Light, J., The Personal Server: changing the way we think about ubiquitous computing. in *Proceedings of Ubicomp 2002: 4th International Conference on Ubiquitous Computing*, (Goteborg, Sweden, 2002), 194-209.

327. Ware, M., Frank, E., Holmes, G., Hall, M. and Witten, I.H. Interactive machine learning - letting users build classifiers. *International Journal of Human Computer Studies*, 55, 3 281-292.

328. Watson, I. *Applying Case-based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann Publishers, Inc., San Francisco, 1997.

329. MIT. Footprints: Visualizing histories for web browsing. Available at: http://web.media.mit.edu/~wex/Footprints/footprints1.html. 1997

330. Wiederhold, G. and Genesereth, M. The conceptual basis for mediation services. *IEEE Expert / Intelligent Systems*, 12, 5.(Sept/Oct 1997), 38-47.

331. Williams, M.D. What makes RABBIT run? *International Journal of Human-Computer Studies*, 21, 4.(Oct 1984), 333-352.

332. Williamson, C. and Shneiderman, B., The dynamic HomeFinder: evaluating dynamic queries in a real-estate information exploration system. in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, (Copenhagen, Denmark, 1992), 338-346.

333. UserLand Software Inc. ThinkTank 2.41NP and MORE1.1c. Available at: www.outliners.com. 2005

334. Winston, P.H. *Artificial Intelligence*. Addison-Wesley, 1992.

335. Witten, I.H. and Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, San Francisco, California, 2000.

336. Wittrock, M.C. Learning as a generative process. *Educational Psychologist*, 11 87-95.

337. Wjj Software. MyBase Desktop Edition. Available at: wjjsoft.com. 2005

338. 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.

339. Woods, W.A. What's in a Link: Foundations for Semantic Networks. in Levesque, J. ed. *Readings in Knowledge Representation*, Morgan Kaufmann, 1985.

340. Yao, K.-T., Neches, R., Ko, I.-Y., Eleish, R. and Abhinkar, S., Synchronous and asynchronous collaborative information space analysis tools. in *Proceedings of the International Workshop on Collaboration and Mobile Computing*, (Fukushima, Japan, 1999).

341. Yates, F.A. *The Art of Memory*. William Cloves and Sons, London, England, 1966.

342. YellowPen, Inc. YellowPen Professional Service. Available at: www.yellowpen.com. 2005

343. ECECS Department, University of Cincinnati. Fuzzy similarity measures for signal pattern classification. Available at: http://www.ececs.uc.edu/~fit/MAICS/PAPERS/Roshdy.pdf. 2003

344. Zoot Software, Delray Beach, Florida. The Zoot Information Processor. Available at: http://www.zootsoftware.com. 2005

APPENDIX A

DATABASE SCHEMA FOR POPCORN PROTOTYPE

The Popcorn prototype interfaces with a MySQL database in order to store the user's knowledge persistently. The schema for this database is very straightforward, consisting of only seven tables, and is presented here for completeness.

```
TABLE ids
      id : INTEGER (primary key)
      type : { 'kernel', 'katom', 'relationship' }
```

Each Popcorn "object" – whether kernel, note, or relationship – has a unique ID in a row in this table. This globally unique ID approach was intended to allow an as-yet-unimplemented feature of promoting a note to a kernel while still preserving the relationships to other objects. Also, it gives room for relationships to be reified as kernel-like objects to permit more sophisticated modeling.

```
TABLE kernels
      id : INTEGER (primary key; foreign key references ids[id])
      name : TEXT
      created : DATETIME
      lastModified : DATETIME
```

Every kernel has a unique ID and a (not necessarily unique) name. Timestamps are not surfaced to the user, but were used to collect usage data. One could imagine users being interested in such information, however, and even using it as the basis for a chronological search as with Lifestreams[121, 126] or PlanPlus[124].

```
TABLE notes
      containerId : INTEGER (foreign key references kernels[id])
      id : INTEGER (primary key; foreign key references ids[id])
      content : TEXT
      source : TEXT
      created : DATETIME
      lastModified : DATETIME
      x : INTEGER
      y : INTEGER
      w : INTEGER
      h : INTEGER
```

Each row of this table represents a note that appears on some kernel view. The containerId field contains the ID of the kernel. The source field is empty for hand-created notes, but contains the URL of any note created via a drag-and-drop operation from Mozilla. One could imagine other kinds of "sources" being used and manipulated by the user directly; book titles, for instance, or the names of individuals who relayed some bit of information. The x, y, w, and h fields store the position, width, and height of the note in relative coordinates (ie., as a percentage of the enclosing view's width and height.)

```
TABLE containedObjects
      id : INTEGER (primary key)
      containerId : INTEGER (foreign key references kernels[id])
      containedId : INTEGER (foreign key references ids[id])
      x : INTEGER
      y : INTEGER
      zoomlevel : INTEGER
      collapsed : BOOLEAN
```

For each kernel that appears on a kernel view, a row in this table maintains the spatial position within the view (x and y), the "zoomlevel," or how large it is when expanded (in relative coordinates identical to those in the notes table), and whether it is currently expanded or collapsed. Note that this scheme permits a kernel to have any number of other kernels as children, and also as parents, yielding full transclusion.

```
TABLE reltypes
      reltypeId : INTEGER (candidate key)
      reltype : TEXT (primary key)
```

Each unique *type* of relationship ("is married to", "influenced", "collaborated against") is held in a row of this table, and given a unique ID.

```
TABLE rels
      id : INTEGER
      participant1Id : INTEGER (together with participant2Id,
reltypeId
            forms primary key; foreign key references ids[id])
      participant2Id : INTEGER (foreign key references ids[id])
      nav : { 'fromleft', 'fromright', 'bi', 'non' }
      reltypeId : INTEGER (foreign key references
reltypes[reltypeId])
```

Each relationship between two objects – be they kernels or notes – is represented by a row of this table. The "nav" field indicates the directionality, and the last field the type of relationship. Note that this scheme permits relationships between *relationships*; ie., a kernel could have a relationship to a relationship, rather than to a note or another kernel. This advanced modeling technique was not made available through the prototype UI, but a few testers mentioned that this would have been a natural solution to certain knowledge modeling problems they faced.

```
TABLE preferences
      prefkey : TEXT (primary key)
      prefvalue : TEXT
```

Finally, various bits of information that needed to be stored between Popcorn sessions were stored in this table. Examples include the id of the active kernel when the application was last shutdown, the size and position of the window, etc.