

Information Scraps: Understanding and Design

by

Michael Scott Bernstein

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2008

© Massachusetts Institute of Technology 2008. All rights reserved.

Author.
Department of Electrical Engineering and Computer Science
May 23, 2008

Certified by.
David R. Karger
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by.
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Information Scraps: Understanding and Design

by

Michael Scott Bernstein

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

In this thesis I investigate *information scraps* – personal information whose content has been scribbled on Post-it notes, scrawled on the corners of sheets of paper, stuck in our pockets, sent in e-mail messages to ourselves, and stashed into miscellaneous digital text files. Information scraps encode information ranging from ideas and sketches to notes, reminders, shipment tracking numbers, driving directions, and even poetry.

I proceed by performing an in-depth ethnographic investigation of the nature and use of information scraps, and by designing and building two research systems designed for information scrap management. The first system, Jourknow, lowers the capture barrier for unstructured notes and structured information such as calendar items and to-dos, captures contextual information surrounding note creation such as location, documents viewed, and people corresponded with, and manages uncommon user-generated personal information such as restaurant reviews or this week's shopping list. The follow-up system, Pinky, further explores the lightweight capture space by providing a command line interface that is tolerant to re-ordering and GUI affordances for quick and accurate entry. Reflecting on these tools' successes and failures, I characterize the design process challenges inherent in designing and building information scrap tools.

Thesis Supervisor: David R. Karger

Title: Professor of Electrical Engineering and Computer Science

Acknowledgements

When I came to MIT I was, frankly, quite a bit lost. Many thanks to my advisors, David Karger, Rob Miller, and mc schraefel, for helping me find my place, define my interests, and develop as an academic. I am also indebted to electronic Max for being a constant travel companion in this journey. Thanks to members of the Haystack and User Interface Design groups, for listening to my crazy ideas and giving me many new ones: David Huynh, Greg Little, Max Goldman, Vineet Sinha, Adam Marcus, Robin Stewart, Chen-Hsiang Yu, Mihir Kedia, and Harr Chen. I would like to thank colleagues at Nokia Research Cambridge (Deepali Khushraj, Ora Lasila) and the University of Southampton (Paul André, Daniel Alexander Smith, Max Wilson). And finally, thanks to the extended research community who have listened, understood, critiqued, and contributed.

Few things in this line of work are as rewarding, or as challenging, as wrestling with tough questions. Unfortunately, social circles hear mostly about the challenges. Thanks first to Adi, who has stood by me and supported me through more than she should ever need to endure. I love you. Also love to my parents and sister, who single-handedly, or in fact sexta-handedly, got me to where I am today. (Sitting in front of a computer — they'd be so proud.) And finally, to my friends who have supported me through the past two years: thank you.

Boston: your weather's not as good as California, but I suppose you're alright.

Contents

1.	INTRODUCTION	13
1.1	What is an Information Scrap?	15
1.2	Information Scraps and Personal Information Management	15
1.3	Contributions	16
1.4	Thesis Outline	18
2.	RELATED WORK.	21
2.1	Psychological Foundations	21
2.2	Information Scraps in Studies of Specific Data Types	23
2.3	Organizational Practice.	25
2.4	Information Scrap Solutions	27
3.	ETHNOGRAPHY	29
3.1	Goals	30
3.2	Method	30
	3.2.1 Information Scrap Operationalization	31
	3.2.2 Triangulation Method	32
3.3	Results	35
	3.3.1 What do Information Scraps Contain?	35
	3.3.2 Scrap Encoding, Composition and Layout	40
	3.3.3 Use of Language in Scrap Text	42
	3.3.4 Tools and Locations	42
	3.3.5 The Information Scrap Lifecycle	47
	3.3.6 The Psychopathology of Information Scraps	50
3.4	Analysis	52
	3.4.1 Common Information Scrap Roles	52
	3.4.2 Organization and Fragmentation	54
	3.4.3 Constraints	55
	3.4.4 Caveats in Our Findings	56
3.5	Implications for Design	57
	3.5.1 Lightweight Capture	59
	3.5.2 Flexible contents and representation	61
	3.5.3 Flexible usage and organization	62
	3.5.4 Visibility and Reminding	63
	3.5.5 Mobility and Availability	64
3.6	Future Work	65

3.7	Conclusion	65
4.	JOURKNOW: INFORMATION SCRAP CAPTURE, MANAGEMENT AND RE-FINDING	67
4.1	Jourknow's Design and Research Contributions	70
	4.1.1 Notebook Interface	70
	4.1.2 Capture: Lightweight unstructured and structured entry	73
	4.1.3 Manipulation: structure exploitation and inspection	76
	4.1.4 Re-finding: Faceted Browsing and Context-based re-finding	78
	4.1.5 Mobility: Unique design needs in mobile scenarios	80
	4.1.6 Visibility and Reminding: Desktop Dashboard, Importance Indicator and Alarms	82
4.2	Implementation	82
	4.2.1 Data Model: Three Representations	85
	4.2.2 Episodes and saliency heuristics	87
	4.2.3 Structure extraction from text	87
	4.2.4 Jourmini: Simplified Data Model	89
	4.2.5 Synchronization	89
	4.2.6 Caching to Obtain Interactive Speeds	90
	4.2.7 Object-Oriented RDF Programming	91
	4.2.8 Saving and Transactions	92
4.3	User Study	92
	4.3.1 Study Results	93
4.4	Discussion	96
4.5	Conclusion	96
5.	PINKY: PERSONAL INFORMATION KEYWORDS	99
5.1	Motivation	99
5.2	A Command Line for PIM	100
5.3	Design of a GUI Keyword Command Line	102
	5.3.1 Commands	102
	5.3.2 Feedback	103
	5.3.3 Running a Command	104
5.4	Personal Information Keyword Commands	106
	5.4.1 Organizing Multiple Interpretations	106
	5.4.2 GUI Widgets for Command Arguments	107
5.5	Web Clips	108
5.6	Implementation	111
	5.6.1 Keyword Command Interpreter	111
	5.6.2 Web Clippings	113
5.7	Evaluation	113

5.8	Future Work	115
5.9	Pidgin and Keyword Commands: Lightweight Data Capture Mechanisms	116
5.9.1	Language Axes	116
5.9.2	Design Axes	117
5.9.3	Pidgin Languages and Natural Language Processing (NLP)	117
5.10	Conclusion	118
6.	DESIGN PROCESSES FOR INFORMATION SCRAPS.	119
6.1	The Design Process	120
6.1.1	Early Ideation and Design Space Exploration	120
6.1.2	Involving Related Work, Functional Prototyping	121
6.1.3	Expert Feedback	122
6.1.4	Needfinding and Ethnography	122
6.1.5	Scoping and Research Specification	123
6.1.6	Jourknow Client Redesign	123
6.1.7	Development	125
6.2	Study Design and Execution	126
6.2.1	Method	126
6.2.2	Study Results	127
6.2.3	Symptoms of a Wicked Design Process Failure	128
6.3	Reflection on Practice: What Went Wrong?	129
6.3.1	Considering the Obvious Solutions	129
6.3.2	Breakpoints: Process Inspection Points	130
6.4	Outcomes for Design Methodology	134
6.5	Conclusion	135
7.	CONCLUSION AND FUTURE WORK	137
8.	BIBLIOGRAPHY	139

A Note on Collaboration

This research has been a highly collaborative enterprise. In the year and a half since I identified information scraps as a topic of interest for my research, this work has involved three graduate students, a terminal masters student, five undergraduates, and three faculty members. The core team has featured myself and Max Van Kleek as the primary student researchers, and David Karger and mc schraefel as faculty advisors.

It is difficult if not impossible to tease out exact individual contributions on projects like Jourknow, so I have not endeavored to do so in this thesis. I designed and coded much of the user interface for Jourknow; Max helped. Max designed and coded the bulk of the context-capture mechanisms and RDF data model; I helped. Neither would be of much use without the other.

So, rather than confuse this thesis with contribution language throughout, I have simply chosen to tell the story that I wish to tell. My story is that of information scraps: understanding why they exist and designing new systems to help us manage them. Naturally, this story places weight on the problems I find interesting and solutions I am particularly proud of. Thus, this thesis focuses on the ideas that I take particular stake in or ownership over. It glosses over many issues that Max will take up with great zeal in his Ph.D. thesis.

But if you found a bug in Jourknow, it's definitely Max's fault.

1. INTRODUCTION

Despite the number of personal information management tools available today, a significant amount of our information remains out of their reach: the content is instead scribbled on Post-it notes, scrawled on the corners of sheets of paper, stuck in our pockets, sent in e-mail messages to ourselves, and stashed into miscellaneous digital text files. This scattered information ranges from ideas and sketches to notes, reminders, shipment tracking numbers, driving directions, and even poetry. It may never make its way into our usual applications – yet we carry it around with us, decorate our desks with it, and often even make sure to archive it. For a category of personal information with so little traditional support, it is all but ubiquitous in our lives.

I refer to these pieces of personal information as *information scraps* (Figure 1.1). The term suggests several images: notes that are written on a scrap of paper, that are incomplete, or that have been separated from our primary personal information tools.

This thesis focuses on information scraps as a topic of understanding and design. Why are information scraps so often held outside of our traditional locations and instead on scraps of paper or in text files? What kind of tools can support the lightweight capture and freeform expression common to information scraps? My goal in this research is to open an investigation of information scraps, so that we might begin answering these questions.

I. INTRODUCTION

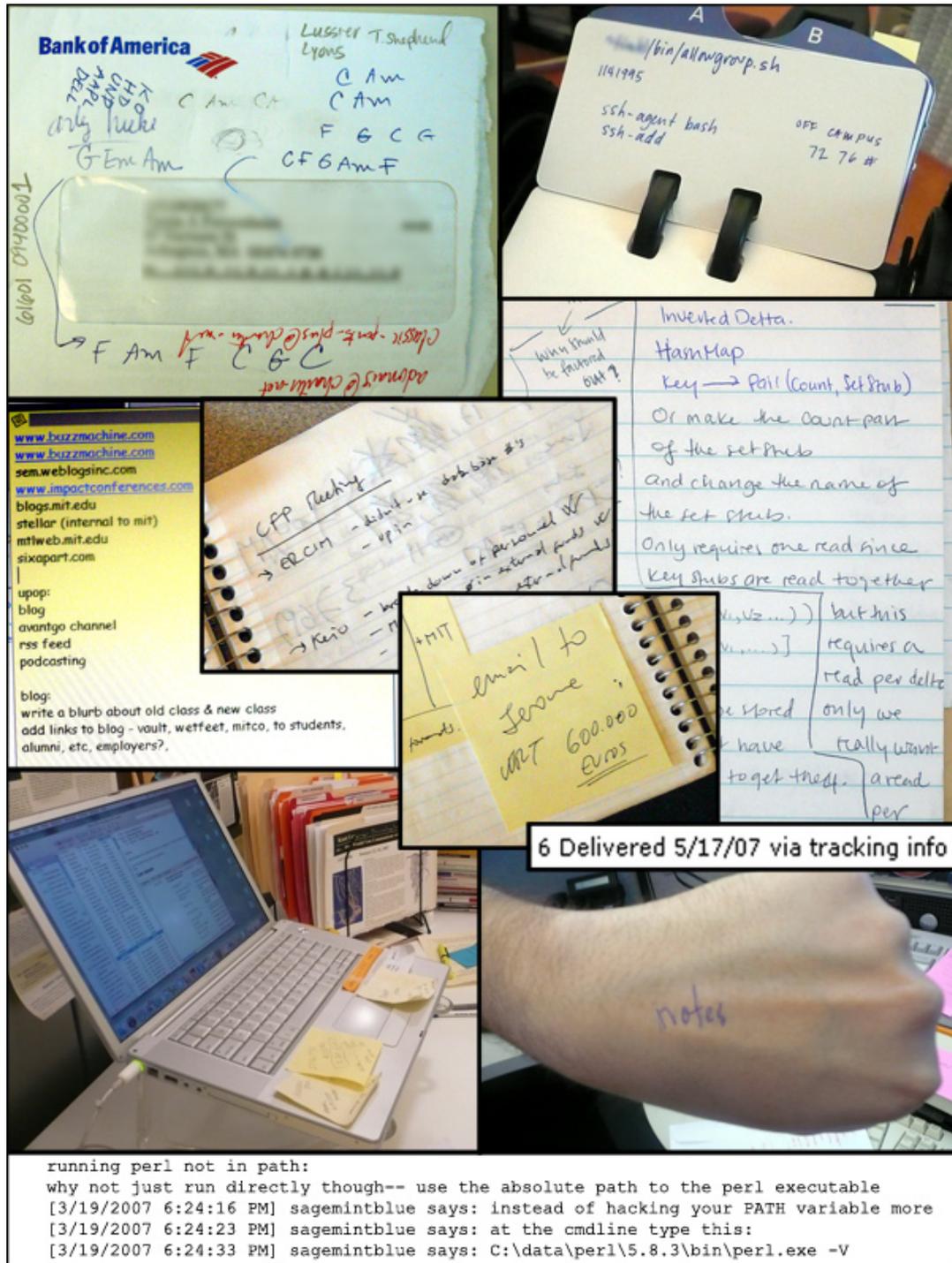


Figure 1.1. Information scraps live in our digital and physical worlds; they are recorded on envelopes, in text files, and in a myriad of other locations.

1.1 What is an Information Scrap?

An information scrap is an information item that falls outside all PIM tools designed to manage it. This definition suggests that canonical information scraps include items such as address information not in the address book, electronic communication not in the e-mail client, and to-dos not in a to-do manager. It intentionally includes information items for which no PIM tools currently exist, as well as information items stored and managed in general-purpose (e.g., non-PIM) information tools. For the purposes of our work, we choose to include in the set of general purpose tools artifacts such as notebooks, spreadsheets, and text editors/word processors because they tend to be catch-alls for PIM data. Our definition also intentionally makes no distinction between paper and digital PIM tools. To illustrate, here are some examples of information scraps from our research:

- Note of how to make a call abroad saved as a text file in a “Miscellaneous” folder
- To-do on a Post-it note
- Photo of a whiteboard from a discussion kept on the computer desktop
- Meeting notes in a general-purpose notebook
- Serial number for an application saved in an e-mail to yourself
- A friend’s phone number written on a piece of scratch paper
- Cooking recipe kept in a personal wiki
- Song lyrics and guitar tabs taped on the wall
- Copy of academic transcript saved in a text file

By this definition, information scraps are the personal information items that have fallen between the cracks of our PIM tools. An information scrap is evidence that there is no appropriate tool at a time of need; the user deliberately chooses a tool with affordances designed for other forms of information. In analysis, I treat the existence of such items as evidence of PIM design failures and thus suggestive of unfulfilled design opportunities.

1.2 Information Scraps and Personal Information Management

Personal information management (PIM) as a research field is concerned with the processes of capture, organization, re-finding and use of the information we deal with in our daily lives [79]. PIM’s research challenge is to understand existing information practice and leverage that under-

standing to deliver a set of tools that give us the affordances we need and desire.

Information scraps are largely a new domain of focus for PIM. I have targeted a type of personal information with many unanswered questions — one that highlights the ad-hoc, unorganized underbelly of personal information. Information scraps are highly personal, since they are often barely understandable to third parties. They are also inextricably tied to our personal spaces of information, living side-by-side with our files and often depending on them for meaning. Yet computers have thus far failed to provide compelling support for information scraps, forcing us to adopt coping strategies such as e-mailing ourselves or using free text files.

1.3 Contributions

In this thesis, I contribute knowledge in the domains of information scrap practice and information scrap manager design. These contributions are:

Characterization of existing information scrap practice and needs. I undertake an in-depth ethnographic study of information workers' use of information scraps. I describe the types of information contained in scrap form, tools commonly used, and common roles for information scraps. I derive design needs for information scraps: lightweight entry, unconstrained content, flexible use and adaptability, visibility, and mobility.

Design of an information scrap manager. I introduce Jourknow (Figure 1.2), an information scrap manager with design elements catering to context-based re-finding of notes, fast capture of structured information, and mobile capture and synchronization.

Design of a sloppy command line-style interface to reduce the need to create information scraps. A surprising amount of common personal information types such as to-dos and calendar items wind up as information scraps because applications require prohibitive time and effort to enter the data. To ameliorate this problem and prevent information scraps from being produced, I introduce Pinky (Figure 1.3), a command line-style interface for quick entry of personal information.

Challenges with Design Processes for Information Scrap Managers. Through the process of designing and implementing Jourknow, I have identified several design process issues particularly challenging when creating an information scrap manager: scope, prototyping techniques,

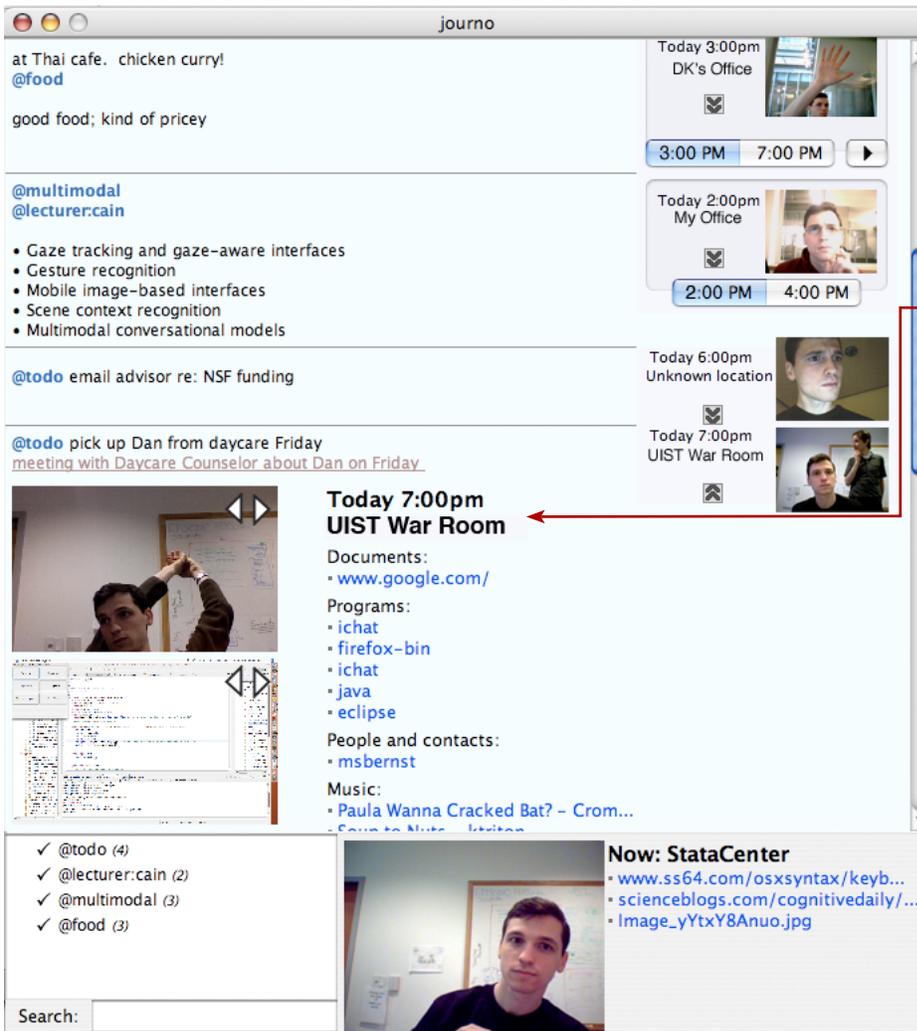


Figure 1.2. Jourknow is our prototype information scrap manager. It provides support for information scraps through lightweight capture, context-based re-finding, pushing information to relevant applications such as the calendar or to-do manager, and a mobile client.

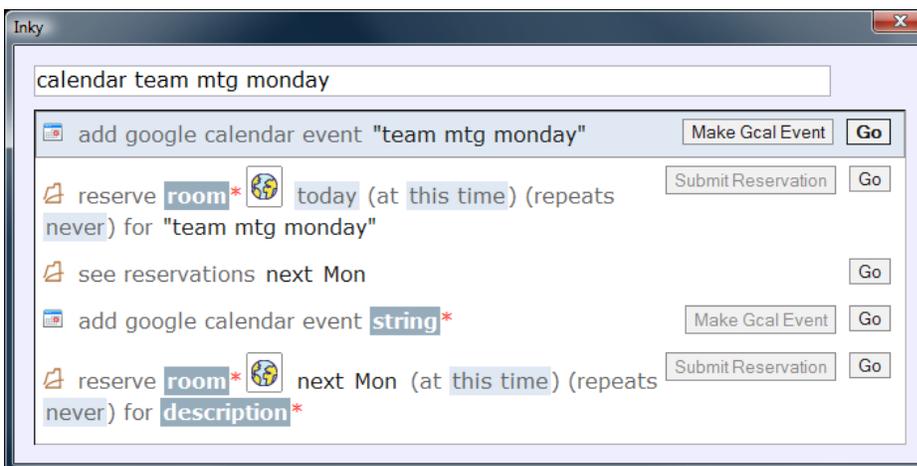


Figure 1.3. Pinky is a keyword command line for personal information. By speeding up the information capture process, Pinky may reduce the need to create information scraps in the first place.

evaluation techniques, and continuous population contact. In this section I discuss these issues, as well as possible solutions.

1.4 Thesis Outline

Following this introduction, CHAPTER 2 surveys related work, beginning with known psychological principles informing our use of information scraps (§2.1). It then covers studies bearing on specific tools or information scrap types (§2.2) and research on cross-tool organizational practices (§2.3). The related work section closes by canvassing existing tools and research systems which informed my work on Jourknow and Pinky (§2.4).

CHAPTER 3 is an in-depth discussion of existing information scrap practice – the most comprehensive one to date. It begins by describing a novel methodology for locating information scraps in practice (§3.2) and outlines the goals for our cross-tool study. I detail the information types [79] stored in scraps, the constitution of scraps, tools used in support of information scrap work, and the information scrap lifecycle (§3.3). I derive a characterization of the typical roles that information scraps serve in personal information practice (§3.4) and the needs that information scraps serve in support of these roles (§3.5).

The thesis then shifts to the design of novel information scrap management systems. CHAPTER 4 introduces Jourknow, an information scrap management system following a notebook metaphor. Jourknow contains support for context-aided re-finding, lightweight structured input, and mobile capture and retrieval. The chapter begins with a motivational description of Jourknow’s design decisions, details the Jourknow interface (§4.1), and reports on a weeklong evaluation of the prototype (§4.3 – §4.4). Implications of the (failed) rollout are discussed in CHAPTER 6.

CHAPTER 5 refocuses from general information scrap management to specific techniques for lightweight capture and reducing the need to create information scraps. I introduce Pinky, a keyword command engine for entering and querying personal information. Pinky includes flexible syntax (§5.3.1), GUI affordances (§5.3.2, §5.4.2), and contextually relevant information (§5.5) in support of fast, lightweight capture of common personal information types.

By referencing two major iterations on the Jourknow prototype and one on Pinky, CHAPTER 6 investigates the considerable design process challenges faced by designers of information scrap managers. These challenges include scoping the design, selecting a user study population and

maintaining contact with that population, and effective prototyping procedures.

CHAPTER 7 concludes the thesis with a discussion of future directions and a reprisal of my contributions through this endeavor.

2. RELATED WORK

Information scraps cross many boundaries in PIM – they are a cross-tool, cross-information type phenomenon that serve many purposes in our wide range of daily information-related activities. In this chapter I situate the study of information scraps among the rich body of research already surrounding PIM-related activities. I begin by reviewing related research on the psychological underpinnings behind information scrap practice. Then, I canvas general personal information management practices as they bear on information scraps, and specific practices related to individual information scrap types such as to-dos.

2.1 Psychological Foundations

There exists extensive psychological literature surrounding our motivations for creating and manipulating information scraps. Perhaps the simplest framing of the problem was done by Ross and Nisbett in what they termed *channel factors*, the “small but critical facilitators or barriers” to an action [118]. Ross and Nisbett demonstrated the amplified effects that small difficulties or facilitators will have on human action, just as a pebble placed at the fork of a stream can dramatically divert the course of the water. Seemingly small time and effort requirements such as booting up a laptop might thus be perceived as enough of a burden to cause us to use other means of capture such as writing on our hands.

2. RELATED WORK

There are many such channel factors that encourage us to create information scraps. Lansdale was the first to relate psychology to the study of personal information management [96]; in his work, he noted classification, or filing, as a cognitively difficult activity of special note. This result suggests that information scraps may be created when the cost of filing a piece of information is perceived to be too high – whether choosing a point in a folder hierarchy or deciding which of several related applications to use. Csikszentmihalyi identified humans’ desire to maintain a state of *flow* where uninterrupted concentration is highest [53]; Bederson translated this concept into interaction design principles in support of the flow state [29]. When the user is in a flow state, Bederson and Csikszentmihalyi argue that unrelated thoughts and ideas may be undesirable, explaining why we may attempt to write them down as quickly as possible before our original thought was lost.

Information scraps often serve as a memory prosthesis [95] or exosomatic memory, later used to remind us of the original thought. Scraps can help us index into our memory via a variety of cues. Location is a very powerful memory primer [57, 81, 116, 138]; a combination of knowing *what* and *when* can also effectively aid recall of the rest of a memory [134]. We are also able to recall a variety of contextual information about our documents to potentially aid in re-finding, such as textual content, visual elements, file type, or implicit narratives around file creation [28, 40, 65]. However, many information scraps do not include such metadata; it is unknown whether the highly abbreviated contents of many information scraps (e.g., “Joe the attorney” [30]) are more powerful memory cues than those above.

Often, we fail to create memory prostheses such as information scraps even when they might later be useful. We are habitually overconfident in our own knowledge and memory [98], potentially leading to conscious choices not to commit something to memory that later becomes valuable. These results imply that we may choose not to record critical information in an information scrap which later becomes necessary for re-finding or comprehension. Further, even if we chose to write down or make an effort to remember, we do not always utilize this information when recall is needed. Our memory’s faulty yet quick access is often preferred over accurate but slower external memory aids [66, 67, 86]. Such a preference suggests that many information scraps may only be deliberately re-accessed when our own memory has failed.

2.2 Information Scraps in Studies of Specific Data Types

Information scraps take many forms, and researchers have noted their existence across a number of type-specific studies. Here I review the relevant work by information type.

Perhaps the most canonical information scrap is the self-note. Through a series of semi-structured interviews, Lin et al. arrived at a model of such notes' lifecycles: trigger, record, transfer or maintain and refer, complete, discard or archive [99]. Campbell and Maglio identified salient characteristics for what they termed *notable information*, including transience, visibility, mobility, ability to post, transferability, short length, and ease of both creation and destruction [50]. The authors further observed a trend for personal notes: a strong preference for paper-based media over digital media. Dai investigated this preference by interviewing expert users of PDA memo applications to suggest future design directions; users were typically most hindered by a lack of organizational support for their digital notes [56]. Hayes et al. studied the phenomenon of *short important thoughts*, uncovering a strong need for ubiquity and mixed-initiative systems in the support of such information [70]. Strikingly, 73% of participants reported regularly transcribing such notes onto another medium, suggesting transfer as an especially potent pain point for personal notes and thus some information scraps.

Professional fields often encourage similar practices via engineering logbooks and meeting notes. Paper engineering logbooks, long a common practice for professional engineers to use for recording notes and ideas, were found to most commonly serve as reminders of work in progress and as a personal work record for future reference [104]. Meeting notes created by professional information workers contain a large number of facts (e.g. names, phone numbers, technical details, and procedures) and action items [90]. The degree to which these practices translate across other logbook-intense professions such as in the sciences [120] is not yet clear.

E-mail is now not just a tool but an entire habitat, an ecology in which we embed much of our personal information [59]. As a result of this embedding, we see e-mail used for a variety of information scrap purposes. E-mails are deliberately marked as unread or left unorganized in the inbox to serve as reminders or to-dos, and half-completed messages are saved along with notes for what to include [32]. Venolia et al. suggest that such coping strategies are due to the sheer volume of incoming messages [133]; other research suggests a complementary hypothesis that

2. RELATED WORK

they may also be the result of information scraps being archived in the inbox. For instance, Whittaker and Sidner's early study found that 35% of folders contained only one or two e-mails [136], suggesting that many of these e-mails may have had no natural application and required small, artificial homes to be created. More recently, we have learned that nearly a third of all archived e-mail is actually sent by the owner to herself [63, 83] – another common information scrap pattern.

The ubiquity of to-dos, scattered in unorganized locations across the physical and virtual workspace, suggests that they might constitute a particularly common form of information scrap. Bellotti et al. undertook the most rigorous investigation of to-do practice to date [30]. Their findings suggest that to-dos (and thus information scraps) will be created by expending as little effort as necessary, and “only elaborated enough to provide a salient clue” to the original author (e.g., a to-do with only, “Joe the attorney”). To-dos are often integrated as resources into ongoing work, incorporating state or links to other artifacts. Bellotti's investigation uncovered a large number of separate tools (average 11.25 per person) being used to manage to-dos, noting that often they are intentionally placed not in a typical organization but instead in the way of a typical routine to promote visibility. These fragments or notes are very much in keeping with our definition of attributes of information scraps: they are deliberately not kept in an application like a to-do list; they are in a specific “elsewhere.” These locations might include the backs of hands, scraps of paper, unstructured text files, and post-its [41].

Calendaring tools also display many of the characteristics of information scrap work. Users keep a plethora of non-appointment (but still time-based) information scraps in their calendaring tools: notes of which week of the semester it is, pointers from a diary entry to supporting materials, reminders, reports of how time was actually spent, and notes of prospective but not finalized events, among others [41]. In their studies of e-mail and task management, Bellotti et al. also noted that participants would create calendar events as reminders [31].

The consideration of web and Internet material raises further issues. The Keeping Found Things Found (KFTF) project has investigated the means by which users keep web information [46, 82, 83]. Use of bookmarks and browser history are relatively anemic; participants instead e-mailed themselves URLs with comments, saved web pages to disk, or printed out information.

Camera phones have opened up new opportunities for information scrap collection and recording. Ito [77] describes the cameraphone's ability to elevate the mundane – allowing us to take photos of the seashell we

find on the beach, the street sign that will allow us to relocate a new restaurant, or other objects that are simply “interesting” in some way. Such one-off photos may fall into the domain of information scraps, especially as the mundane objects can be difficult to categorize or be of predictable utility. These scrap pictures may be numerous, as well: images captured for “personal reflection or reminiscence” were the most numerous of those indexed by Kindberg et al. [91].

2.3 Organizational Practice

Physical office organizations reveal many qualities that are common to information scraps, including an aversion to filing and an affinity for paper media. Malone’s seminal paper on office organization [102] directly examined the existence of unorganized *piles* in office work. He noted that while most workers’ desks were piled rather than filed, computers of the time required users to file rather than pile. I note this trend here, as the often uncategorizable nature of information scraps suggests that information scraps are particularly prone to piling. Whittaker and Hirschberg [135] discovered that *working notes* for current projects constituted 17% of the paper archives maintained across an office move – many of these were handwritten and irreplaceable, a population with likely overlap with information scraps (e.g., meeting notes and brainstorming). In *The Myth of the Paperless Office* [123], Sellen and Harper detail numerous reasons for the continued prevalence of paper in the workplace, including its ease of annotation, flexible navigation, spatial reorientability and support for collaboration. Many of these same forces are at play when we choose to use paper to record our information scraps.

There is a similar antipathy to filing scrap-like information in the digital realm. In parallel to Whittaker and Hirschberg’s description of working papers, Barreau and Nardi detail what they term *ephemeral information* – that which has a short shelf life. Many information scraps exhibit the characteristics and difficulties of ephemeral information: they are “loosely” filed or not filed at all, and difficult to manage in large quantities [27]. Boardman and Sasse noted that their participants tended to combine filing and piling strategies based on item priority, regularly filing items of high perceived value but otherwise leaving their collections to spring cleaning or no organization at all [42]. Boardman and Sasse further reported that 3% of files, 41.6% of e-mail, and 38.8% of bookmarks remained unfiled over their longitudinal study – again, the forces driving these artifacts to remain unfiled will likely also exist for information scraps.

2. RELATED WORK

Folder structures for scraps, when they do exist, may likely remain ad-hoc and relatively flat. Jones investigated folders in the service of ongoing projects [85], and found that folders' semantics were continually adapted to reflect each participant's "evolving understanding of a project and its components." This result will likely hold for information scraps as well, whose boundaries are less clearly delineated than project folders. Similarly, Barreau and Nardi [27] investigated digital file hierarchies and discovered that most participants' hierarchies were surprisingly flat due to low perceived future usefulness of complex archives. Rather, digital hierarchies were structured in the service of what they refer to as *location-based finding*: navigating to a directory of interest and proceeding to browse. This result is remarkable in that users preferred to depend on their own cognitive capacities for recognition of documents rather than ensure that everything is elaborately filed, suggesting that information scraps may be recalled in the same manner.

Generating accurate and personally relevant filing schemas for information scraps can be a cognitively difficult process. Malone's participants complained of the difficulty of accurately filing paper information [102]; Bowker and Star [45] expand this point to the digital realm: "A quick scan of one of the author's desktops reveals eight residual categories represented in the various folders of email and papers: 'fun,' 'take back to office,' 'remember to look up,' 'misc.,' 'misc. correspondence,' 'general web information,' 'teaching stuff to do,' and 'to do.' We doubt if this is an unusual degree of disarray or an overly prolific use of the 'none of the above' category so common to standardized tests and surveys." [45 p. 2]

Information fragmentation [78] of information scraps occurs across devices, applications, and media. In a cross-tool study, Boardman [43] reported on three consequences of fragmentation: 1) file compartmentalization across tools, 2) lack of ability to coordinate work activity between tools, and 3) inconsistent design vocabularies. Fragmentation leads to undesirable effects such as an inability to gather all information about a single person or topic, or to effectively link such data [88]. Information scraps can be particularly susceptible to fragmentation, as similar information may be scattered across multiple tools. While fragmentation mainly occurs between applications, mobile situations have fragmentation instead across devices such as cell phones, laptops, notebooks and other mobile devices [108]. Information scraps are susceptible to mobile fragmentation, as freeform, unsync'ed data inevitably becomes isolated from related data on other devices. Data unification is viewed as an eventual goal to remedy this situation [34, 43, 84, 88, 87].

2.4 Information Scrap Solutions

There are a growing number of commercial products and research systems aimed at information scrap management, allowing the user to capture unstructured notes. Here I discuss a few particularly illustrative products — a more complete list is maintained at <http://people.csail.mit.edu/msbernst/papers/pim-tools.htm>.

The most straightforwardly relevant class of products are those which adopt a notebook or free-text metaphor in their user interface, selling themselves as digital equivalents of the pocket notebook. These tools may be designed as such, such as Microsoft OneNote [17], Post-It Digital Notes [18] or Yojimbo [24], or they may be co-opted general purpose programs such as Notepad, emacs [6], or the e-mail client. There are a number of useful axes on which to compare the basic designs of these products:

- Does the tool discretize notes in the interface, or does it choose coarser units of granularity such as the page — or even no discretization at all?
- Are notes organized in any kind of hierarchy or tagging system?
- Is the tool desktop, web, or mobile-based? What kind of usage scenarios is the tool thus assuming will be particularly relevant?

Many tools aspire to more than simple note saving. Automatic recognition of structure embedded in the note text is common; this structure can be text used in searches [5, 17], people and places for organization [22], or (outside the notebook metaphor) dates and times for calendaring and reminding [7, 12]. Digital notebooking often involves linking or copy-paste; thus, tools such as Google Notebook [9] automatically embed source information whenever possible. This source data allows the user to re-find the information in its original context. By drawing on the ubiquitous Post-It note, specifically its constant reminding presence and its small form factor, designers have introduced tools with small desktop fingerprints such as Post-It Digital Notes [18]. Finally, we see tools experimenting with different input modalities. Jott [13] is a popular voice-to-text service; pen input is often cited as ‘killer apps’ of digital pens (e.g., Anoto [1]) and tablet computers (e.g., OneNote [17]).

Research programs have begun developing tools as well. Bederson instantiated his ideas of interface *flow* in NoteLens [29], providing a large number of keyboard shortcuts to keep the user’s experience as noninvasive as possible. The Cepher tool [71] addresses synchroniza-

2. RELATED WORK

tion and ubiquity issues associated with notetaking by synchronizing notes across multiple devices using a personal server. Lifelogging research attempts to capture context the user may not have otherwise digitized, such as pictures, sound, viewed web pages and GPS location. This information may be refound later for interest (e.g., Stuff I've Seen [60], SenseCam [73], MyLifeBits [64]), or embedded into notes to aid memory (e.g., ChittyChatty [86]).

3. ETHNOGRAPHY

As a class of personal information, we have much still to learn about information scraps. What similarities exist among scrap features and management practices? Why are information scraps so often held outside of our traditional PIM locations and instead on scraps of paper or in text files? Why do we manage other scraps by co-opting our traditional PIM applications against their intended modes of use, such as by composing e-mails addressed to ourselves? If these unorganized bits truly indicate the limits of our PIM tools, how might we begin to build better tools?

In this chapter, I investigate the nature and use of information scraps. I contribute a cross-tool methodology for studying existing information scraps, and apply this methodology to an investigation of information scrap practice. In the study, I investigate the information types [79] stored in scraps, the constitution of scraps, tools used in support of information scrap work, and the information scrap lifecycle. The artifact investigation reveals a large diversity of information types encoded in information scraps, contributing an appreciable percentage of the total scraps investigated. Through analysis of these results, I derive a characterization of the typical roles that information scraps serve in personal information practice: temporary storage, cognitive support, reminding, information archiving, and recording of unusual information types. These roles suggest a set of unmet design needs in current PIM tools: lightweight entry, unconstrained content, flexible use and adaptability, visibility, and mobil-

Work presented in this chapter is a collaboration with Max Van Kleek, David Karger and mc schraefel. It is in press, ACM Transactions on Information Systems [35].

ity. Finally, I describe approaches that I believe will be the most successful in the information scrap management of tomorrow.

3.1 Goals

I have targeted a type of personal information with many unanswered questions – one that highlights the ad-hoc, unorganized underbelly of personal information. My focus is on understanding why information scraps exist, what kinds of information they hold, why they end up in the tool or medium they do, and how they evolve through their lifetime. I entered this study hoping to understand to following:

- **Characterization of the phenomenon.** What is, and is not, an information scrap? Can we improve our intuitive understanding into a more precise characterization of the phenomenon?
- **Type variety.** What kind of data will be encoded in information scraps? How much variety will there be, and which information types will be the most popular?
- **Structure and expression.** How does a calendar item as an information scrap compare to a similar item in a digital calendar such as Outlook? Will the information scrap carry less information, or express it in a different way?
- **Tools.** Information scraps are by definition held in inappropriate or general-purpose tools. What tools are these, and why do we use them? How do we adapt the tools to hold information they may not have been designed to carry? To what extent does fragmentation take place across tools, and does this fragmentation inhibit later re-finding or re-use?
- **User needs.** What needs do information scraps serve? Why are they used in preference to other PIM tools?

3.2 Method

We conducted a study consisting of 27 semi-structured interviews and artifact examinations of participants' physical and digital information scraps. From our initial work [39, 131] it was clear that information scraps are a cross-tool phenomenon, so we chose a cross-tool study inspired by the cross-tool work of Boardman and Sasse [42]. This study design allowed us to examine information scraps in many locations rather than a single (possibly sparse) one. As our questions primarily surrounded information scrap content, organization and location, and lifecycle, we chose to focus on examining the scraps themselves rather than the capture or retrieval

process. Diary studies and experience sampling studies would have also allowed us to record information scraps as they were generated, but we were concerned that the additional time and energy burden on participants would have conflicted with the overriding importance participants place on ease and speed when capturing scraps – participants may have simply chosen not to record the scrap to avoid the effort associated with writing in their diaries.

We carried out the interviews with participants from five different organizations, following a five-person pilot study in our lab. Three of the organizations we visited were information technology firms, focusing on mobile communication, interactive information retrieval, and wireless communication. One was small (start-up), another medium sized, and the third a large multinational corporation. The fourth organization was an Internet consortium, working internationally in a highly distributed fashion. The fifth was an academic research lab. We interviewed 7 managers (MAN), 7 engineers (ENG), 6 administrative or executive assistants (ADMN), 2 finance workers (FIN), 2 usability professionals (UI), 1 technical writer (WR), 1 campus recruiting officer (REC) and 1 industrial researcher (RES). There were 13 males and 14 females; the median and mode age range was 30-35. Educational level ranged between some college (4), college degree (11) and graduate degree (12). This population was a diverse group of professional knowledge workers with a skew toward those vested in information technology.

Interviews were performed at each participant's main computer, located at his or her typical place of work. For privacy reasons, participants were free to refrain from sharing any particular piece of personal information. During the interviews, we did not use the term *information scrap*; rather, we asked participants to tell us about information that they had that was not formally recorded in a proper place, like a calendar or project folder. Participants then provided us with a stream of examples which we noted as they discussed these exemplar artifacts. Our questions focused on revealing the purpose of the item, the reason it was recorded the way it was, as well as where it fit in any context or process of use.

3.2.1 Information Scrap Operationalization

As discussed in the introduction, the term *information scrap* can be difficult to define. However, for purposes of internal validity, and lacking a rigorous definition at the time of our study, we required an operationalization of the term that would allow us to identify which artifacts

to record. We used this operationalization to decide which artifacts were in scope; we did not generally attempt to convey this definition to the subjects of the study. Throughout the discussion of the study, we refer to an information scrap as a piece of personal information that:

• Is in a tool with no explicit support for that information's schema, or	<i>e.g., a phone number on a Post-it note</i>
• Has no tool specifically designed to handle that kind of information, or	<i>e.g., an application serial number</i>
• Is in a tool that does not seem particularly well suited to the information type.	<i>e.g., a to-do with a "where to remind me" field shoehorned in to the details field</i>

The following are examples of artifacts that we excluded from our capture: e-mail serving a communicative purpose, word processor documents with papers or full essays, and contact information in the computer address book. This definition carries a connotation that what is and is not an information scrap depends on each participant's tools, needs, and practices. Based on the results of this study, we later refined our definition of information scraps to the one found in CHAPTER 1.

3.2.2 Triangulation Method

Information scraps are distributed among tools and locations, and strategies vary from person to person. We faced a challenge in our artifact examination – specifically, that we might not uncover some classes of each participant's information scraps. We could canvas the space by asking about all known tools – but what if the participant used a tool we didn't know about, or used a common application in a way we did not think to investigate? We could instead query by location, such as Desktop or Miscellaneous folders – but what if the data lived in an application rather than a folder? We could ask how participants dealt with common scrap types such as how-to guides and URLs – but certainly there would be types we might leave out.

Our solution was to fashion a methodology by which we would look for information scraps along all three axes: tool, location, and type. By exploring along each axis with participants, we would be able to zero in on, or triangulate, the location of an appropriate artifact. We first inquired after tools (typically the most prolific approach), continued with location and finally type. To our knowledge, this methodology is novel. We began by running a pilot study (5 participants) to generate a broad set of tools, locations, and types that we used as a seed list for our final study. Participants were free to generalize to other tools as they desired. We recorded information scraps that were digital, physical, and mobile. Table 3.1 lists a sample of the script we followed.

Table 3.1. The three categories of Tools, Locations and Types characterized the main starting points for our artifact study.

Triangulation Perspective	Examples	What was targeted
Tools	E-mail	Messages that do not serve communication purposes: e-mails sent to oneself, in the Drafts folder, or archived in the inbox.
	Calendar	Calendar entries that did not correspond to actual events; use of the “details” field.
	Bookmarks	Bookmarks carrying information beyond just a pointer to a web page – for example, “todo” or “toread” bookmark folders.
	Physical Notebooks	All available data (this location commonly holds information scraps).
	Physical Post-it Notes	All available data (this location commonly holds information scraps).
	Notetaking Applications	All available data (this location commonly holds information scraps).
	Freeform text files	“todo.txt” or “todo.doc” files containing personal notes, to-dos, and other data.
Locations	Computer Desktop	Documents of short-term interest and notes to self.
	Physical Desktop	Freeform notes and documents of short-term interest.
	“Miscellaneous” Folder	Data that was difficult to categorize.
	Office wall and whiteboard	Participant-authored decorations or annotations.
Types	Reminders and To-dos	To-dos not in the to-do manager or that did not fit the to-do manager’s schema.
	How-to guides	All examples (no known application to organize this information).
	URLs of interest or quotes from web sites	Examples not held in a bookmarking utility.
	Contact information	Examples not held in a contact utility.
	Notes	All examples (common information scrap).
	Short pieces of data (e.g., phone numbers, passwords, serial numbers, thank-you note lists)	All examples (common information scrap).

As the participant or interviewers pointed out information scraps in each tool, location or information type, the interviewers performed an artifact analysis and a semi-structured interview focused on techniques

3. ETHNOGRAPHY

surrounding the items of interest. One of the interviewers, performing the artifact analysis, probed for as many specific instances of the class of information scrap as feasible given the time constraints. For each artifact, the following were recorded:

- Information type: the information type as described by the participant (e.g., to-do, URL, shopping list).¹
- Tool: the tool used to author and edit the information scrap.
- Presence of material not directly created by the author: a binary variable, true if the information scrap contained any content that was copied into the scrap rather than authored directly.
- Content encoding: four binary variables, each true if the information scrap contained the particular kind of content:
 - Text: written or typed text or words
 - Photographs
 - Pictorial Drawings: representative drawings; drawings intended to look like a particular object, as in a design sketch
 - Abstract Drawings: non-representative drawings; graphs, arrows, abstract visual layouts, diagrams

Multiple items were often coded from a single tool, as distinguished from each other by the participant. The interviewers guided the interview so as to try and get a representative sample of information scrap from a variety of tools, skewing for breadth rather than depth, though we did spend extra time investigating tools with large numbers of scraps. At the conclusion of the study, we consolidated similar information type categories. To consolidate, we began with the specific types recorded by one of the interviewers and verified by the other. The two researchers then acted as coder/aggregators, consolidating types as aggressively as possible without sacrificing the participant's original intent with the scrap.

At the conclusion of the artifact examination the interviewers continued the semi-structured interview, following up on topics of interest. Interviews typically lasted sixty minutes.

¹ The data type was recorded as per the participant's primary classification of the artifact, even if it contained multiple data types. Thus, even though many to-dos included names of people, places to be, times of events, and so on, they were nonetheless coded as "to-do" if the participant viewed the overall artifact as a to-do.

3.3 Results

3.3.1 What do Information Scraps Contain?

In our artifact analysis, we coded each of the 533 information scraps for its information type, and then consolidated similar categories. The results can be seen in Table 3.3 and Figure 3.1.

3.3.1.1 Common information scrap types

The four most common information types we found in scraps were to-dos (92 instances), meeting notes (44 instances), name and contact information (38 instances), and how-to guides (25 instances):

- **To-dos.** Information scraps containing lists of items participants wanted to accomplish. Action items, traditional to-do lists, and other information interpreted as a to-do fell into this category.
- **Meeting Notes.** Notes taken down while the participant was in a meeting or discussion. These ranged from notes taken at formal meetings to hallway conversations.
- **Name and contact information.** Typical contact information: name, address, phone number or e-mail address.
- **How-to Guides.** Recipes noting how to perform certain tasks, kept for future reference. Examples included UNIX shell commands, login procedures for remote servers, and international calling/shipping/faxing instructions.

It is notable that two of these top four categories (to-dos and contact information) are easily managed by many PIM applications. §3.3.5.1, discussing the capture stage, will give possible reasons why this information still may end up in scrap form.

3.3.1.2 Diversity of Data Types and the Long Tail

Another compelling view of information scrap forms appears when one focuses on the least frequently occurring items. Figure 3.1 illustrates the frequency of each information type we found within all participants' scraps, ordered from most to least frequent. Immediately noticeable is the fast drop in the histogram after the most common types described above, and the large mass of types with few occurrences. Furthermore, as can be seen in Table 3.2, the least frequently occurring types (the tail of the distribution) comprised a significant percentage of the information stored in all the scraps; in particular, forms that occurred only once comprised 13%

3. ETHNOGRAPHY

Table II. Table 3.3. We noted large numbers of scraps containing as to-dos, meeting notes, contact information, and how-to guides. However, there is was also a group of scrap types that only occurred once or twice across our entire investigation, highlighting the wide variety of personal information encoded in information scraps.

Occurrences	Types with the Given Number of Occurrences
92	To-Do
44	Meeting Notes
38	Name And Contact Information
25	How-Tos
16	Work-In-Progress
14	Directory, File Path Or URL
13	Desired Items
12	Login/Password
9	Brainstorm, Calendar Or Event Details, Event Notes
8	Progress Report, Receipts And Confirmations
7	Computer Repair Status, Conversation Artifact, Correspondence (Chat), Financial Data, Products Of Interest, Reminder
6	Calendar Or Event List, Correspondence, Debugging Notes
5	Archived E-Mail, Computer Address, Ideas, Pre-Emptive Calendar Scheduling
4	Account Number, Airplane Flight Information, Annotations, Design Layout, People Of Interest, Plans And Goals, Timeline, Airplane Flight Information, Archived Document
3	Agenda, Configuration Settings, Jobs And Classifieds, Math Scratchwork, Project Notes, Shipping Information, Template E-Mail Response, To Read, Whiteboard Capture
2	(Mixed type), Academic Record, Bug List, Changelog, Company Organization Chart, Debugging Program Output, Favorite Quote, File Backup, Frequent Flier Information, Hotel Information, Performance Tracking, Room Setup Diagram, Tax Information, Template Text, Time Log, To Share
1	(No Memory of Meaning), Announcement, Application Instructions, Architecture, Archived Document, ASCII Art, Baseball Schedule, Blue Chip Stocks, Book Margin Comments, Book Outline, Calculation Chart, Car Supply Shops, Citation, Class Assignments, Client ID Number, Concert Tickets, Correspondence (E-Mail), Credit Card Information, Deadlines, Decorative Drawing, Definition, Demographic Breakdown, Documentation, E-Mail Lists Of Interest, Employee Desires And Goals, Event Planning, Expense Report, Fantasy Football Lineup, File Transfer, Flow Diagram, Funding Options, Gift Certificate, Guitar Chords, Gym To Join, Insurance Claim, Kayaking Resources, Library Number, Moving Plans, Network Diagram, Newsletter Outline, Notes From Old Job, Parking Location, Part Number, Patent Information, Phone Payment Statistics, Picture Of Car, Picture Of Poster, Pictures Of Team Members, Planned Trip (Map), Presentation, Price List, Project Overview, Public Notice, Puzzle Answers, README File, Rebate UPCs, Recipe, Resume, Room Location, Salary Calculation, Serial Number, Sign Out Sheet, Song Lyrics, Talks Given, Travel Agent, Word To Spellcheck

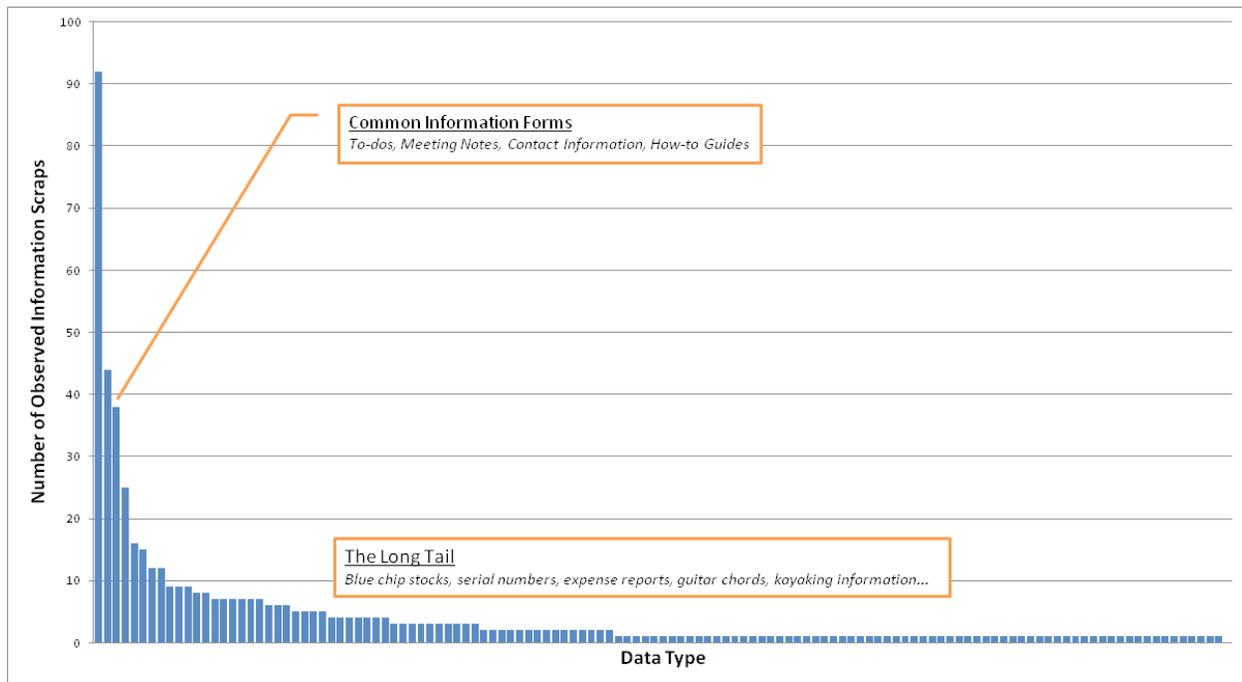


Figure 3.1. This visualization of Table 3.3 demonstrates the span of the long tail of personal information in information scraps. The ordering is identical to that in Table 3.3.

of all scraps; twice or less, 18% of all scraps. Information scraps encode a wide variety of personal information.

This result suggests that the distribution of information types contained with scraps exhibits the *long-tail* property of statistical distributions, where the scraps containing rarer forms cumulatively rival the number of occurrences of commonly occurring forms. Our result is consistent with a variety of other natural occurrences distributed as power laws, patterns known variously as The Pareto Principle and Bradford's Law. These principles suggest that a large percentage of our information will fall into a small number of categories, and that a large diversity of unpopular categories will make up the remainder.

Table 3.2. Information types that appeared only once make up approximately one eighth of all information scraps. The 50% threshold is crossed at nine occurrences out of 533 recorded information scraps.

Upper Bound on Number of Occurrences of an Information Type	% of all Information Scraps
1	12.8%

3. ETHNOGRAPHY

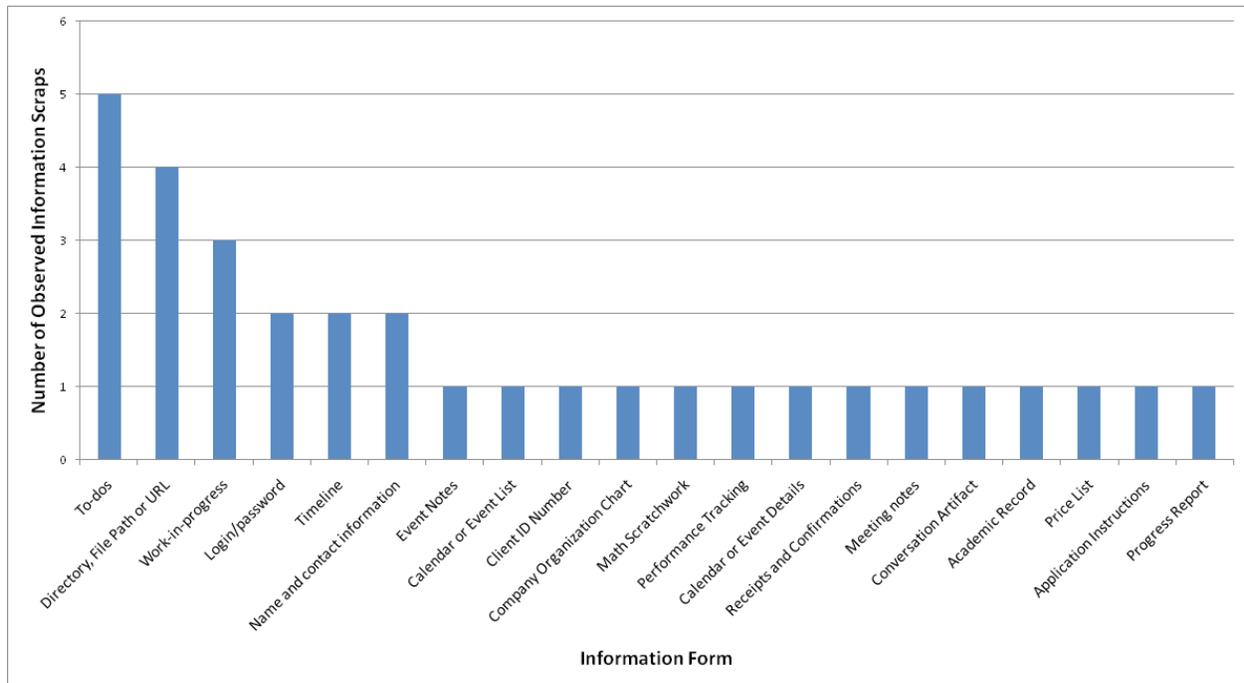
Upper Bound on Number of Occurrences of an Information Type	% of all Information Scraps
2	18.4%
3	24.0%
4	29.3%
5	33.0%
6	36.4%
7	44.3%
8	47.3%
9	52.4%

We uncovered a large number of rarely-occurring information types, including book wish lists, application serial numbers, expense reports, resumes, guitar chords, and information about kayaking. All of these types could benefit from an application tailored to their semantics (e.g., comparing kayaking in Cambridge and Palo Alto, sorting resumes by years of experience, or finding the most similar expense reports); however, unlike information types like to-dos, for the majority of these rarely occurring types, such applications are either nonexistent or unpopular.

In addition to looking at the distribution of types for all scraps as a single group, we also looked at each individual's scrap distributions, to gain a sense for comparison. In doing so, we observe a similar distribution for individuals' scraps, which implies that individuals use scraps to keep a large number of infrequently occurring information types themselves; see Figure 3.2a and Figure 3.2b. This result is again consistent with The Pareto Principle and Bradford's law, which state that subsets of the general population also follow similar power laws.

Figure 3.2b provides an interesting link between the global (i.e. inter-participant) and personal distributions: ENG3's most popular item is notes on the repair status of the computers he managed; however, he was the only participant to record such information. Thus, even though computer status notes are in the head of ENG3's personal distribution, the form still exists in what is globally the tail of the distribution. Several other participants exhibited such behavior, as well: ADMN1 maintained a set of artifacts she needed to share with her superior, ENG2 keeps an extensive set of personal progress reports, MAN6 has a sizable folder of documents to read in his free time, and so on. Yet each of these patterns was minor when averaged over the entire sample.

a)



b)

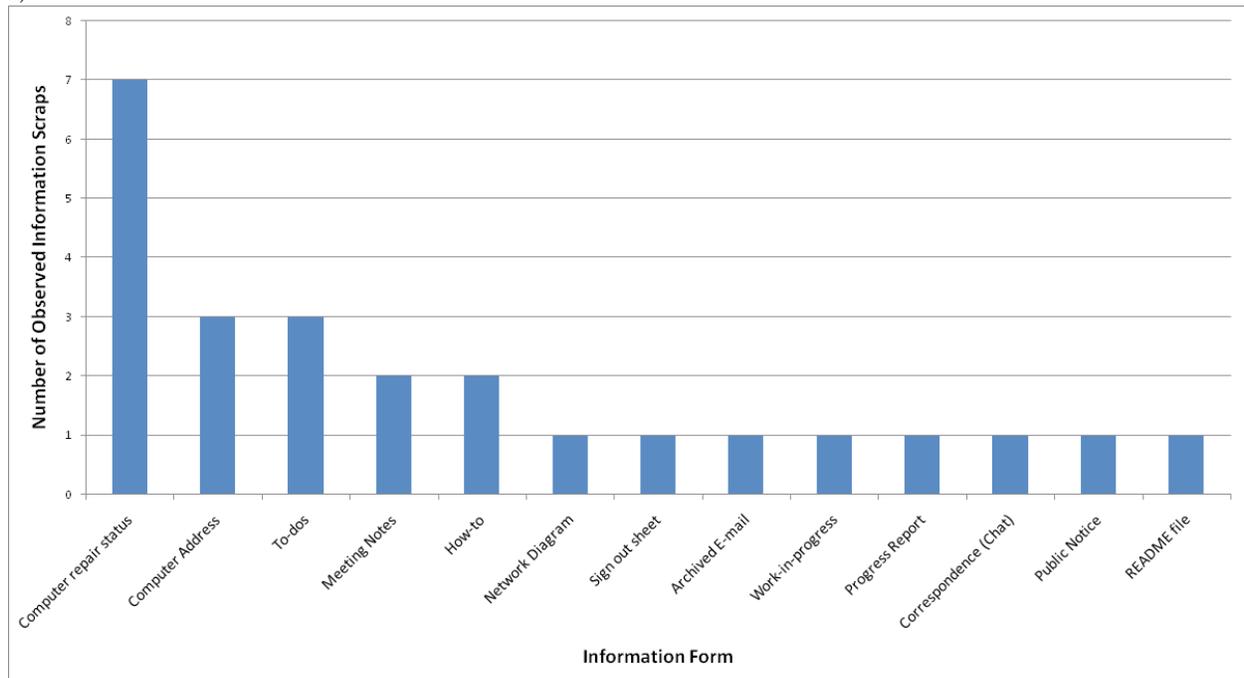


Figure 3.2. a) FINI's information form distribution, which also follows a long tail. Here the histogram closely mirrors the accumulated distribution across all participants. b) ENG3's information form distribution, evidencing a large number (7) of computer status note scraps. Though these notes are in the head of his distribution, he was the only participant to collect such data, so when accumulated the computer status notes fall into the long tail.

3.3.2 Scrap Encoding, Composition and Layout

The predominant method of encoding information in scraps was text, typed or handwritten. However, it was not uncommon for information scraps to also contain abstract drawings (Figure 3.3). Such drawings included arrows, graphs or timelines, stars, organizational lines and boxes, and markings indicating emphasis. We coded our data to record the number of information scraps that contained text, abstract drawings, pictorial drawings, or photographs. We found 96% of our information scraps to contain some sort of text (95% of the digital scraps, 96% of the physical scraps), 5% to contain abstract drawings (1% digital, 10% physical), 2% to contain pictorial drawings (no digital examples, 4% physical), and 2% to contain actual pictures (4% digital, < 1% physical). In two-proportion z-tests, the differences between digital and physical media for abstract drawings, pictorial drawings, and actual pictures are significant ($p < 0.01$); the difference for text was not significant. A small number of information scraps contained other kinds of media, digital or physical attachments such as laundry receipts.

Due to the varied nature of the types of information scraps, there was a corresponding variation in scraps' elements, such as phone num-

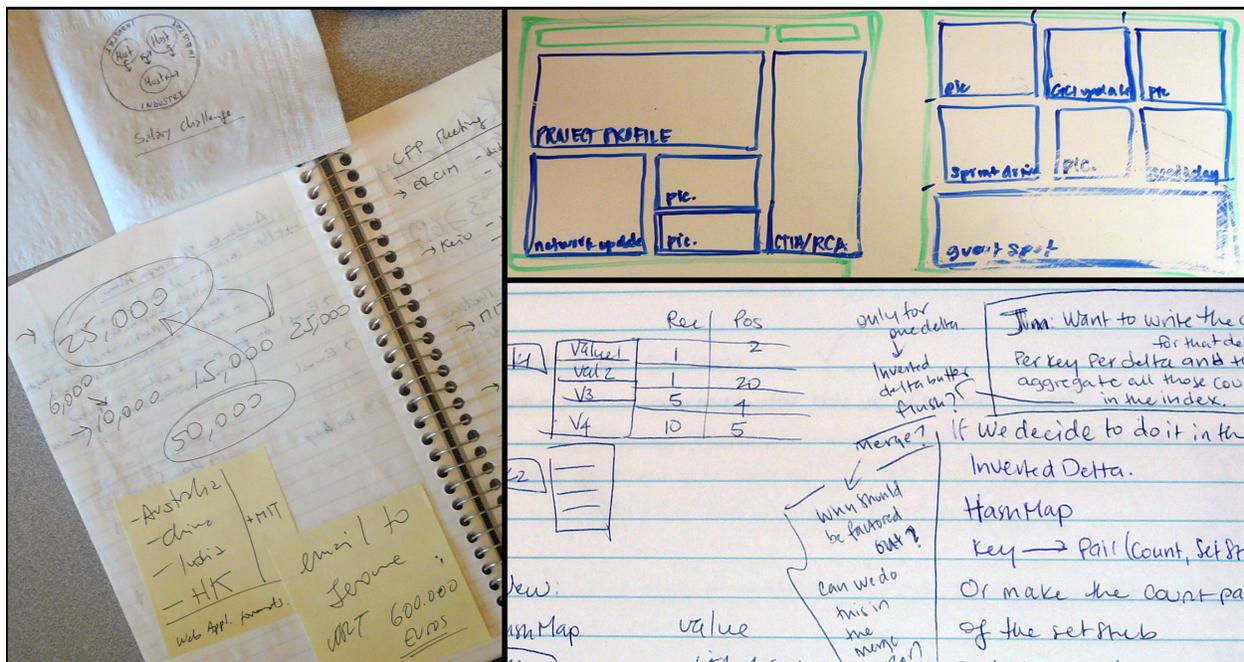


Figure 3.3. All three of these information scraps contain abstract drawings, including arrows, boxes, and so on.

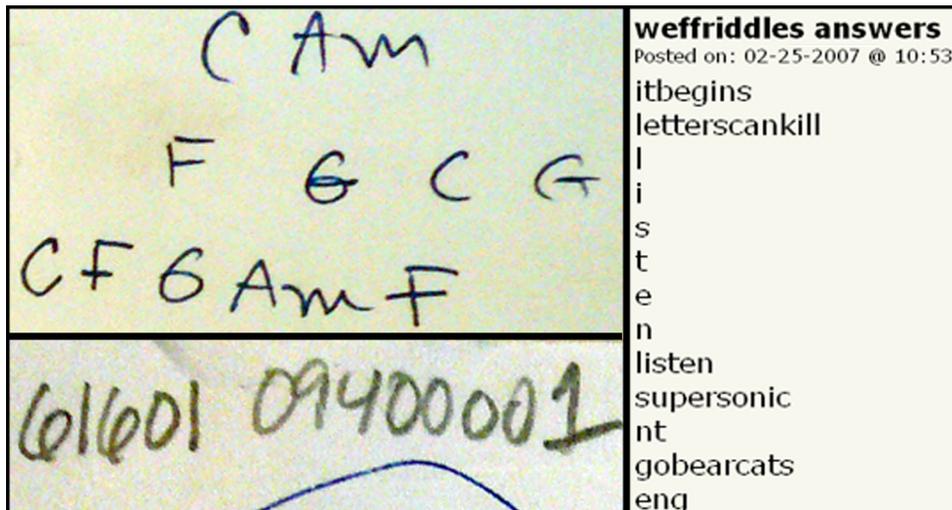


Figure 3.4. Information scraps containing unusual data. Counterclockwise from upper left: guitar chords, an unknown string of numbers, and answers to an online riddle.

bers, e-mail addresses, or URLs. It was not uncommon to see several elements intermixed in a single information scrap or several unlike information scraps together in one location – Figure 3.5b contains a URL, a PIN number, two UNIX commands, and a phone number sequence, all mainly unrelated. The information types were rarely labeled, which occasionally made it difficult for the interviewers to understand the content of some notes (Figure 3.4).

Information scraps were a common way to capture data that did not match existing schemas. Underspecifying data was an important affordance: “I don’t know exactly when [my visitor] will come today,” MAN3 explained about the calendar event in his to-do manager, “If we’ll agree on the details later, I prefer to use a to-do.” Information scraps could also capture data with more fields than applications knew how to handle; for example, MAN maintained his own contact list where he could record previous deals and other personal notes on each person he worked with.

Flexible content traded off the ability to manipulate the data in a structured manner later. Thus, many participants were reduced to keyword search or page flipping to re-find particular scraps. Groups of scraps that had grown large enough sometimes established their own lightweight structure to support manipulation, such as by using spreadsheet software with headings.

Several participants who kept free text files on their computer utilized this ability to mix types or lay out thoughts as they desired – even creating ASCII art in the case of ADMN6. Paper and physical tools were particularly preferred for their encoding flexibility, allowing participants freedom over visual structure and sketching (fully 10% of physical scraps involved some sort of drawing annotation).

While most information scraps were very short (a few words or lines long), we observed several instances of scraps of approximately a hand-written page in length, particularly meeting notes. This result indicates more variation in length than we were originally expecting.

3.3.3 Use of Language in Scrap Text

We found that text written in information scraps used extremely terse language; many scraps consisted exclusively of key words, such as lists of names of people, places or objects, and raw bits of data, such as phone numbers, addresses, passwords, and other strings. Figure 3.5 gives examples of text used in scraps. Scraps used as temporary storage locations in particular exhibited short language, listing single words or pairs of noun-object or noun-data value, often omitting the verb or relevant predicate, as well as articles and particles. For scraps comprising notes, such as from a meeting, class or brainstorming, phrase structure was more common. We also noticed a tendency to omit the subject title or description of what the data actually represented. Several participants, when sending e-mails to themselves, intentionally left the subject field blank or wrote something general such as “note to self.”

3.3.4 Tools and Locations

We noted 51 different tools in use across our investigation, 33 digital and 18 physical. Figure 3.6 shows the distribution of the number of information scraps we located in each tool or location. Again there is a power law (Pareto Principle) pattern, beginning with a set of extremely popular tools and trailing off to a large number of less popular ones. Participants maintained a small set of main tools for capturing information scraps, supplemented with a large number of less-used auxiliary tools.

The digital and physical world each held popular tools. E-mail was by far the most dominant digital tool for recording information scraps (74 instances, 26.4% of digital scraps), followed by a text editor (47 instances, 16.8%; e.g., TextEdit or Notepad) and word processor (27 instances, 6.4%; e.g., Microsoft Word). Demonstrating the use of text files in particular, a few participants generated large, separate files of contact information be-

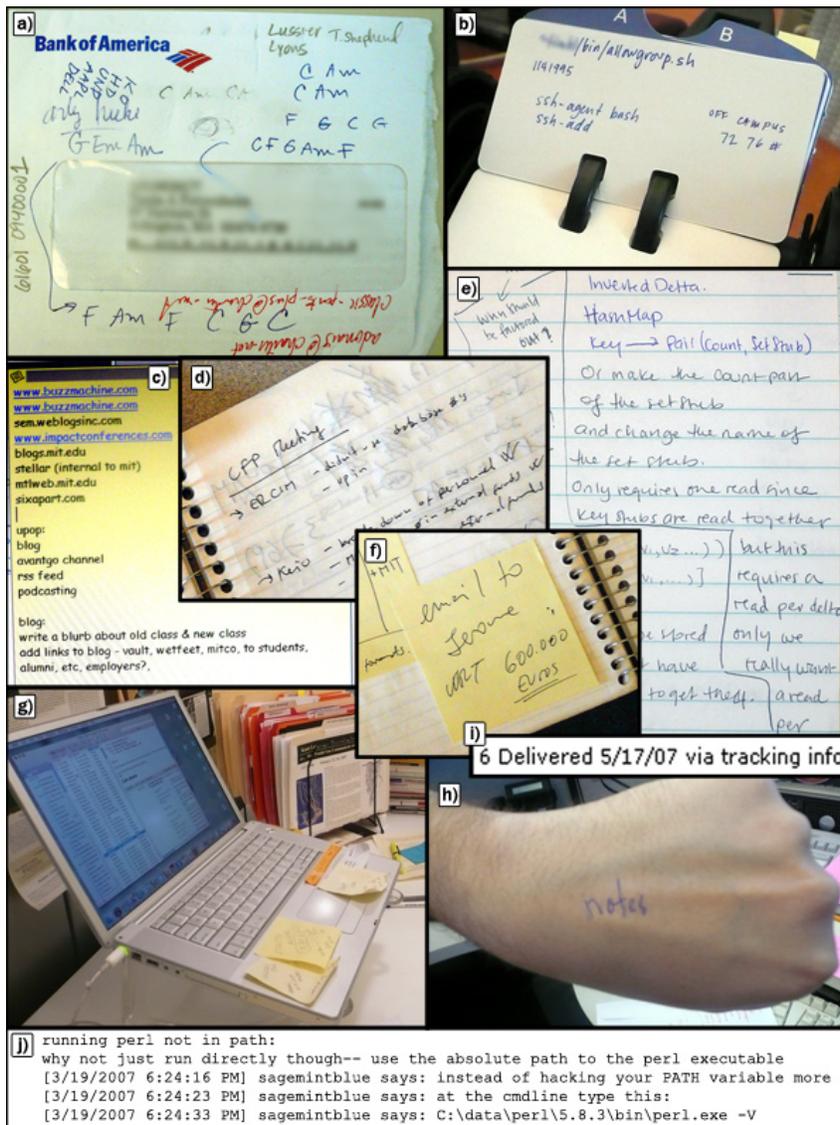


Figure 3.5. Information scraps we noted, focusing on typical examples of minimal use of language in scraps: a) an envelope with several scraps: guitar chords, stock ticker symbols, e-mail addresses, and an unknown number, b) a web site address, password and helpful SSH commands in a rolodex, c) use of the Outlook notes facility to maintain links of interest and the outline of a blog entry, d) “CFP Meeting, ERCIM, didn’t use database #’s, up in [incomplete],” meeting notes, e) a brainstorm on a programming decision, f) “email to Leone w.r.t. 600.000 euros,” a reminder, g) several post-its on the laptop palm rest, reading “XIA HUA,” “ANDREW talk,” and “Gopal’s cell #,” h) a reminder written on the participant’s hand, a single word, i) text at the top of an e-mail the participant received, condensing it into a few memorable words, j) an annotated, copy/pasted chat transcript detailing a slightly arcane UNIX command.

3. ETHNOGRAPHY

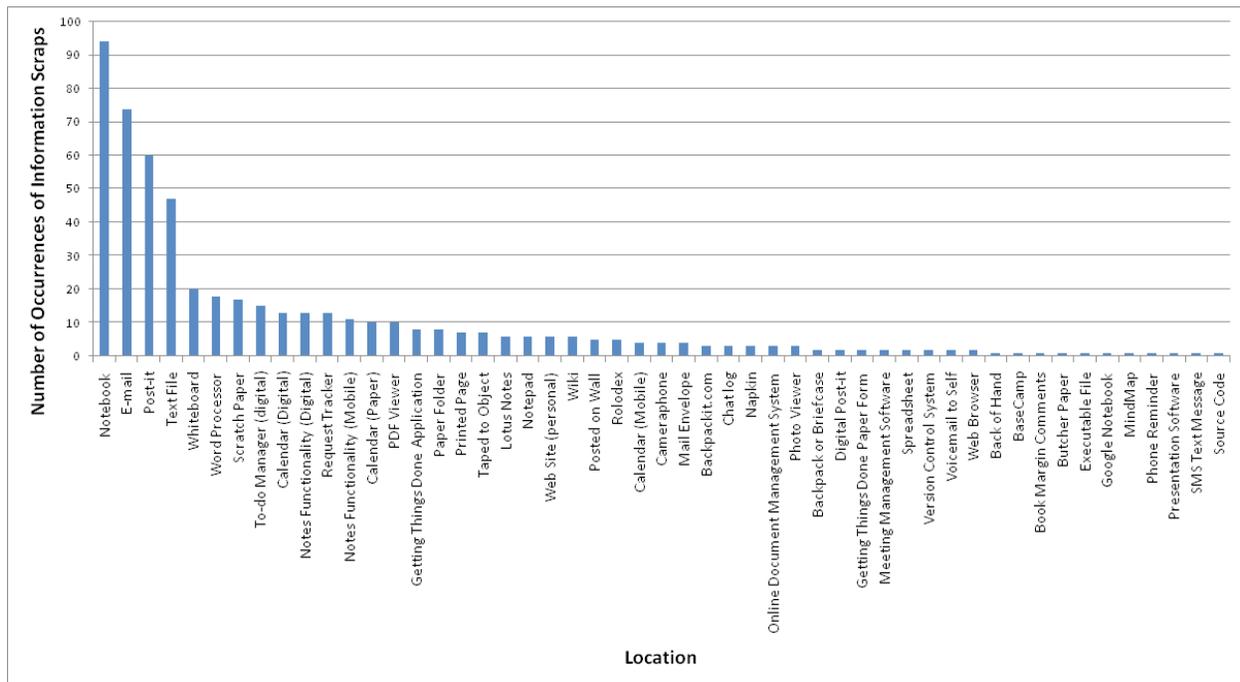


Figure 3.6. There are a small number of tools and locations used frequently, such as notebooks, e-mail, and post-its, and a large number of locations used a small number of times.

cause they wanted to keep the information conceptually separate from their contacts or needed data fields not available in the typical contact manager.

In the physical world, paper notebooks (94 instances, 37.2% of physical scraps) and Post-its (60 instances, 23.7%) were the most popular choices. Participants reported that paper notebooks were often an appropriate choice because they were portable and more socially acceptable in face-to-face meeting settings. Thus, paper notebooks were the most popular tool for meeting note-keeping, and physical meeting notes were three times as common as digital meeting notes.

3.3.4.1 Physical/Digital Divide

Overall, there was an approximate parity in the number of physical (253, 47.47%) and digital (280, 52.53%) information scraps we gathered. However, this statistic is slightly misleading, as participants adopted widely different strategies. A chi-squared test comparing the number of digital and physical artifacts between each participant rejects the null hypothesis ($p < 0.01$), indicating that there is some dependency between the partici-

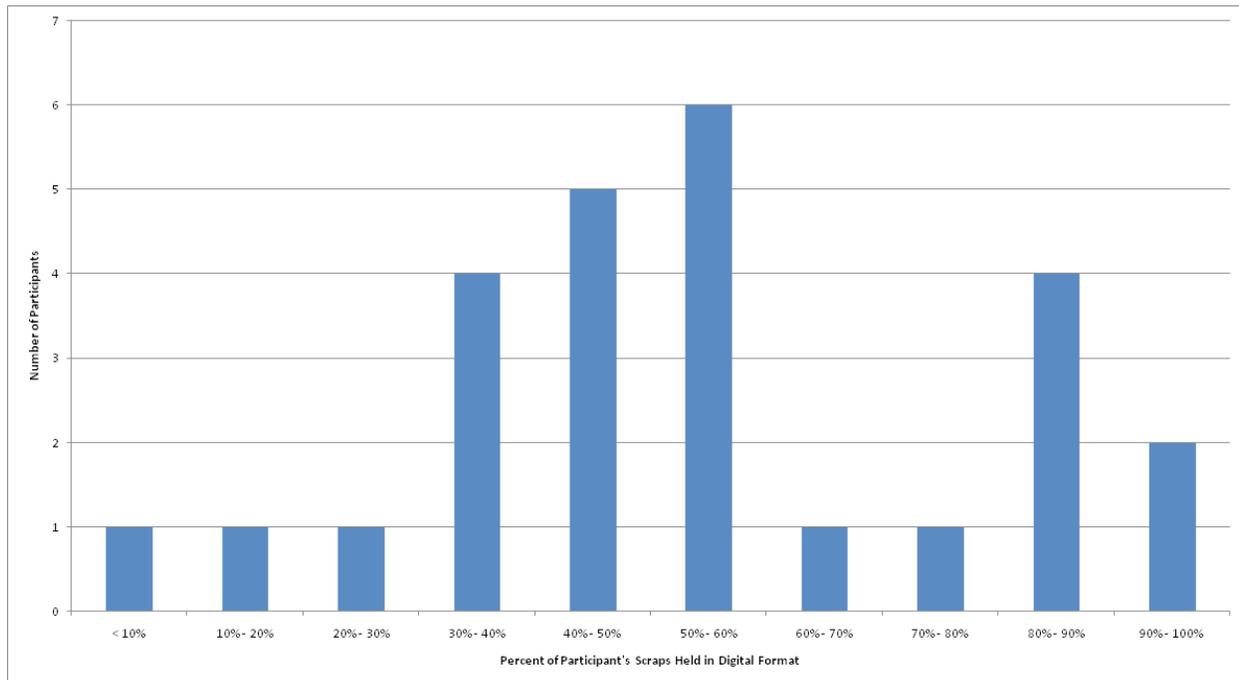


Figure 3.7. Examining the number of participants at each level of digital data, we see two groups: one centered around half digital, half physical, and the other almost completely digital.

pant and the distribution of their information scraps between digital and physical tools. Figure 3.7 indicates what this connection might be: a bimodal distribution with most participants centered at the 50% mark and a smaller group being almost entirely digital. These digital participants tended to be technophiles or mobile workers, including two managers, an engineer, an administrative assistant and a research scientist. Their existence is a somewhat surprising conclusion given previous research's claims that paper is still an overriding favorite for many information scraps [50, 99, 123].

3.3.4.2 Mobility

When the scenario called for information workers to go mobile, participants often generated information scraps to carry important data around or to capture information as events occurred. A small number of information scraps (22 instances) were in mobile digital form: primarily smartphones, but also PDAs, SMS messages and cameraphone pictures. ENG7 and MAN7 in particular used smartphones heavily for capture, and relied on synchronizing functionality with their desktops. Though we were unable to note which physical information scraps were used in mobile

scenarios and which were not, our interviews suggested that paper information scraps were particularly useful in mobile scenarios. UI2 was often mobile: she described how she would reference a note file on her smartphone with relevant conference call numbers, and was likely to send herself a voicemail or add a smartphone note file if the situation required. In addition to mobile scenarios, social constraints came into play: when laptops were not socially appropriate at meetings, paper notebooks were used instead.

3.3.4.3 Tool Adaptation

Information scrap tools were characterized by an adaptivity to novel uses. Post-Its are perhaps most illustrative of this fact: we observed Post-its stuck on walls, on monitors, and inside notebooks; we saw them being folded up and carried into other rooms, crossed out and annotated, and stuck on to cell phones. The tools' generality enabled a wide variety of adaptation and innovation.

This adaptability enabled re-appropriation of the tools. For example, ENG4 used a popular web-based software bug tracking tool as his personal to-do list because it afforded an organizational principle that he liked (individual issues as commitments with deadlines), he could access it from anywhere, and because he could easily update his list of commitments by emailing the system. Similarly, MAN3, who stuck Post-It notes to the back of his cell phone, took advantage both the availability of an unused area on the back of his cell phone and Post-It notes of the right size to devise a solution that worked around the physical impossibility of taking down notes on the phone while using it.

Annotation and revision were also quite common – documents were annotated with comments on what the recipient should do, calendar events contained explanatory notes, and last-minute amendments were appended to agendas. For example, ADMN5 printed out the day's schedule for her supervisor (produced using a calendaring tool) and marked it up with physical notes.

We found that several participants attached commonly referenced artifacts to highly visible objects, such as the wall or computer monitor. Participants often depended on scraps' visibility and in-the-way qualities [30, 102]: for example, DOC left Post-Its with to-dos in an intrusive location right next to her monitor until she finished them. Sometimes the information was even attached to the item of interest; in Figure 3.8, ENG3 wrote on tape affixed directly to the computers he was attempting to annotate.



Figure 3.8. ENG3 wrote notes on masking tape, then affixed the information directly to the computers of interest.

3.3.5 The Information Scrap Lifecycle

In this section we build on Lin et al.'s micronote lifecycle [99], revisiting our results using the lens of the information scrap lifecycle. Here, we discuss Capture, Transfer, Organization, and Re-use:

- **Capture:** the process of translating a thought into a physical or digital information scrap.
- **Transfer:** optionally translating an information scrap from one form into another, either to put it in a more permanent form or enable mobility.
- **Organization:** the addition of structures and metadata to aid re-finding of scraps later.
- **Re-use — Reference, Retrieval and Recall:** the need to re-find scraps (reference), the process of re-finding those scraps (retrieval), and memory for scrap contents (recall).

3.3.5.1 Capture

One of the most commonly cited situations prompting information scrap creation was the need to write something down quickly. Participants reported that quick capture was often prompted by situations such as unintended distractions or being away from their desks. Thus they were a common means to archive important thoughts before the ideas were forgotten

3. ETHNOGRAPHY

[70]. “Mind like water,” MAN6 explained, zenlike; getting the thought out of the head and written somewhere as quickly as possible was the first priority. This offloading of data to the scrap freed our participants to focus on their primary tasks, such as holding a conversation, paying attention to a meeting, or even driving the car (as with MAN7).

When time and effort were at such a premium, the fastest tool would often win out. Even seemingly minor difficulties or annoyances with tools could have striking effects on behavior. “If it takes three clicks to get it down, it’s easier to e-mail,” FIN1 explained. MAN3 would write notes on Post-its and stick them to his cellular phone to transfer into Outlook later rather than enter the data directly into his smartphone, even though the phone supported note syncing. When asked why not enter the note digitally in the first pass, he responded, “Starting in Outlook forces me to make a type assignment, assign a category, set a deadline, and more; that takes too much work!” Similarly, paper notebooks were often chosen instead of laptops because they require no time to boot up. The effect is similar to the one described with mobile applications by Oulasvirta and Sumari [108].

Even when data was implicitly structureable, such as with potential calendar events, participants chose the faster, structureless route of recording a scrap: for example, plain text such as `mtg @ 5pm in cafe`. In the semi-structured interviews, participants explained that structuring the data could often double or triple the time it would take to simply type the information. Thus, there was a tension between the desire to capture the information quickly and the desire for rich representation and structure, often later achieved via the Transfer process.

We observed three major sources of information for information scraps: directly authored material, automatically archived material, and copy/pasted material. Directly authored material, the most common, was intentionally written by the user in a direct effort to record information. Indirectly authored material consisted of scraps that were created as the result of some external action not initiated by the participant, such as receiving an e-mail or paper correspondence from someone else, and then explicitly kept by the participant. Thus, e-mails received and then saved in a “Miscellaneous” directory constituted indirectly authored material. In our interviews, copy/pasted information included examples such as photocopies of a credit card in a notebook, pieces of an online FAQ pasted into a text file, and internet chat transcripts manually saved. We coded our data to examine how often participants included any material they did not directly author in their information scraps. We found that 113 of the 533 scraps (21.20%) overall contained portions copied from other sources, breaking down as 28.93% of the digital scraps and 14.48% of the

physical scraps. In a two-proportion z-test, this difference between physical and digital was significant ($p < 0.01$).

3.3.5.2 *Transfer*

Transfer is the process of moving an information scrap from one medium to another after it has been captured. Information is moved from tool to tool, from physical to digital media, or from digital to physical media. Relatively few scraps were transferred; participants explained that those that were transferred often held additional importance.

We discovered three major reasons for initiating transfer. The first was to transform and re-interpret the information to fill in incomplete details, making the notes appropriate for consumption by others or for permanent archiving. For example, MAN4 religiously transferred all of his handwritten meeting notes into e-mails to “fill in the gaps” and make the notes “sixty-day proof” (ensuring they would be understandable sixty days later). Second, transfer occurred when information was in a work-in-progress state and needed to be available in a tool that offered additional affordances or the ability to accumulate several information scraps into one location. Third was mobility: scraps were sometimes transferred onto other media to carry to another room, or sent in e-mail so that it would be retrievable from home.

3.3.5.3 *Organization*

Among scraps that were archived, techniques varied; some consolidated scraps of similar types/purposes, while others situated scraps with other scraps that were created at the same time, creating a chronological ordering. ENG1 in particular maintained three text files corresponding to three different types of how-tos accumulated over several years.

Several participants expressed difficulty filing information scraps accurately. This effect was especially powerful for single, unique thoughts: as ENG3 jokingly complained, “where *would* you put the last two octets of a MAC address?” REC concurred: “It’s too much work to decide which section it should go in – because sometimes things don’t fit in just one, or fit in multiple places. It’s hard to decide what to do.” When such difficulties occurred, participants reported dedicating areas to unorganized information scrap collection, ranging from a special e-mail folder, “Miscellaneous” file folders, misc.txt text files, or a catch-all notebook.

We also noted that participants used ad-hoc solutions to circumvent organizational barriers, replicating information scraps across multiple

digital tools. For example, several participants copied contents from e-mails, wikis, bug tracking tools and groupware into their to-do list management tool or calendar. Participants reported this behavior was an effective coping mechanism for linking information from one tool into another which better fit their workflow. For example, ENG4 pasted emails into job tickets and summarized them in one line at the top (Figure 3.5i), because he wanted to keep all of the information relevant to an outstanding ticket in one location. Similarly, ADMN5 copied relevant email threads into calendar note fields when reminding her superior of a meeting so that the superior would be able to reconstruct the context and purpose of the meeting.

3.3.5.4 *Reference, Retrieval, and Recall*

Participants reported that few of their scraps were actually referenced regularly. One group of information scraps, typified by the to-do list on a Post-it, was referenced actively until its usefulness was exhausted, and then was either archived or thrown away. A second group, including meeting notes, was archived immediately without a period of active reference. After archiving the scraps, participants reported not needing them except for special occasions.

Because our study methodology directly located the information scraps, we did not rigorously examine re-finding techniques; however, participants often spontaneously recalled the existence of a particular scrap and we could observe as they located it for us. When re-finding, participants used a technique similar to the *orienteering* behavior described by O'Day and Jeffries [107] and Teevan et al. [128]. That is, they would directly navigate to a folder location thought to be relatively close to the desired information scrap or sort the items in a useful manner, then take small local steps to explore the results and their neighbors.

We found that participants exhibited good memory for the meaning of almost all of scraps we uncovered. Out of the 533 information scraps indexed, only one was ultimately left with the participant unable to identify its meaning. We did not investigate memory for the meaning of specific details in each information scrap, only focusing on the gist; specific details would likely have fared more poorly, due to human memory for meaning outperforming memory for details.

3.3.6 **The Psychopathology of Information Scraps**

During our study, we encountered a series of affective and psychological dimensions of information scrap management more powerful than we

had expected. The most prominent factors will need to be accounted for in any information scrap management system, so we report them here.

There was often perceived social pressure to be an organized individual; admitting to the existence of information scraps ran directly counter to this perception. Similarly to as recounted by Boardman and Sasse [42], participants often began the interview proud to demonstrate their complex personal information solutions. However, when the interviewers began to inquire after those pieces of the workspace that were unorganized, the same participants would often become uncomfortable and embarrassed, much like as recounted by Bellotti and Smith [33]. Our participants reported:

- “I would like to have time to organize what I’ve captured, but this never happens.” DOC adds that she wishes she were an adherent to the Getting Things Done [25] methodology.
- “By Friday, no stickies and no papers on the desk,” UI2 preemptively excused the existence of information scraps on her desk.
- “Ideally, I wouldn’t need this anymore,” ADMN2 says of the disorganized notebook in which she keeps all of her important information. (Clearly, it’s not going anywhere.)

Several participants apologized for the state of their office or computer desktop.

Exceptions to the embarrassment trend were found both in participants who put extra time into organizing their lives, colloquially known as *lifehackers* [16], and those who simply embraced the mess. MAN6, as a Getting Things Done devotee [25], was quite proud to demonstrate his array of tools. Several participants expressed pride in keeping a tight reign over their information scraps. In contrast were those who had simply accepted that their lives would be messy; ENG1 repeated to the interviewers what had become an affirmation of her love for a messy notebook: “It’s OK to have a notebook!”

Often participants were forced into a cognitive dissonance between their perceptions of themselves as organized individuals and the messy reality of their lives at the time of the interview. UI2 described her regimen of “always” transferring all her Post-it notes into Outlook tasks, but when we noted that there were several such notes that had remained untransferred for some time, the response was defensive: “I’ve been too busy lately.” Believing oneself to be an expert re-finder of information scraps also seemed *de rigueur*. When asked about problems participants might have re-finding information scraps later, responses were curt:

- “I just remember.” (ADMN3)
- “Generally, I remember where things are.” (RES)
- “I remember things.” (MAN5)

In contrast to these reports, participants usually spent considerable time while we observed them trying to re-find scraps they wanted to share with us. Many participants thus overestimated their memory for scrap locations.

3.4 Analysis

3.4.1 Common Information Scrap Roles

We have consolidated a list of common information scrap roles (Figure 3.9):

Temporary Storage. Information scraps' small, discardable presence enabled their common use as temporary storage or exosomatic short-term memory. ADMN2 kept Post-it notes on her laptop palm rest for just this purpose, recording visitors' names and contact information later to be disposed of. Mobility came into play here, as information scraps could be used to bring information along, for example driving directions written on a Post-it. These scraps were self-regulating in number, as they tended to be thrown away or to disappear in piles quickly after creation. Participants often expected this graceful degradation from temporary storage scraps.

Cognitive Support. Our participants shared with us many works-in-progress such as half-written emails, ideas for business plans, brainstorm, and interface designs – scraps used to aid the process of thought. “Before I put anything in the computer, I like to put it on the whiteboard first,” ADMN4 explained of her newsletter layout design process. Information scraps were commonly in cognitive support roles because scrap creation tools supported, and even encouraged, messy information work.

Two such support functions were epistemic action and external cognition. Epistemic action is thinking-by-doing, similar to the benefits expert Tetris players enjoy by rotating falling puzzle pieces to directly visualize and interpret the result of the rotation action [92]. Information scraps thus allowed authors to reflect on what they were doing, change text, cross out ideas, and reposition elements simply to see the result. Information scraps also served as external cognition, enabling our participants to off-load difficult thought processes onto the scrap. We observed a wide variety of information scraps being utilized in external cognition roles – for example, ENG2 and ENG5's in-progress notes taken down while debug-

ging, UI2's sketches of interface designs, and a large number of work-in-progress documents.

Archiving. In contrast to temporary storage scraps, which were intended to have short lifetimes, many information scraps were intended to hold on to important information reliably for long periods of time. For example, many participants used information scraps to archive notes from meetings – as well as information they could not rely on themselves to remember, such as passwords. Our participants expressed comfort knowing the information had been safely saved.

Reminding. Participants took advantage of information scraps' visibility and mobility by placing them in the way of their future movements to create reminders for themselves. Participants used techniques such as colored Post-its, unread or unfiled e-mails, and files left on the desktop, reminding them to take action later or to return to a piece of information

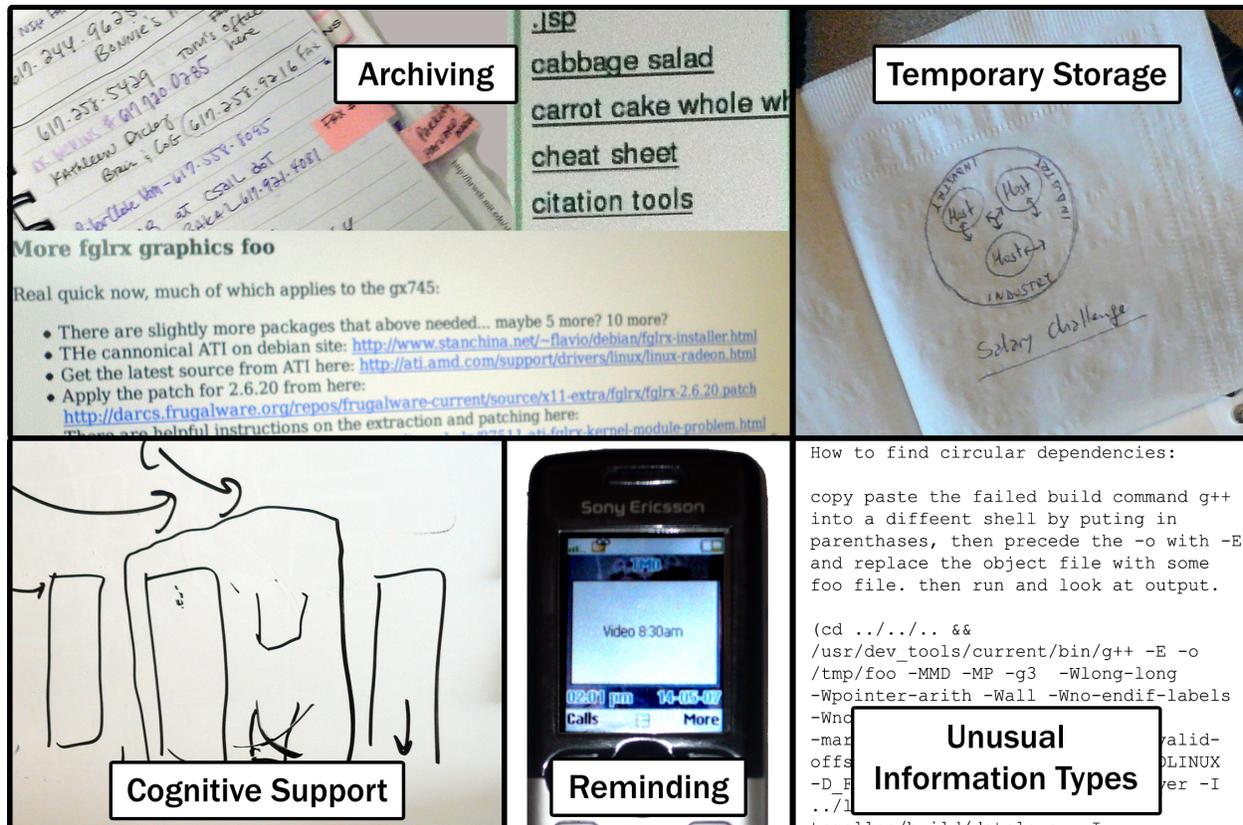


Figure 3.9. The five main roles information scraps played were archiving, temporary storage, work-in-progress, reminding, and a place to put information that wouldn't fit elsewhere.

at a later date. To-dos were an extremely common type of information scrap filling this role.

Unusual Information Types. This role was a catch-all for personal information that did not quite fit existing tools’ expectations. Taking advantage of information scraps’ freeform nature, participants corralled unique information types that might have otherwise remained unmanaged. For example, ENG3 created an information scrap system to manage a library-style checkout for his privately owned construction tools, and MAN4 maintained a complex document of contact information annotated with private notes on clients. This role was particularly prominent in situations where the information did not quite fit existing tools’ expectations, such as calendar items with a date but no start time chosen.

3.4.2 Organization and Fragmentation

We may characterize information scraps of each type by the amount of effort that participants have invested in organizing them. We use web site passwords as illustrative examples:

<ul style="list-style-type: none"> • Low Invested Effort: scraps that are fragmented, unique, or separate from similar data. This includes information of a type that has not recurred often enough to warrant collection in a specialized repository, or temporary information such as might be encoded on a Post-it. 	<p><i>Example: Web site passwords are archived using whatever is handy: e-mails, Post-its, or text files.</i></p>
<ul style="list-style-type: none"> • Medium Invested Effort: information types with many instances archived together, but that remain unorganized. 	<p><i>Example: The user has made sure that all of his or her passwords are archived somewhere in the e-mail inbox, though they are not tagged or filed in any consistent way.</i></p>
<ul style="list-style-type: none"> • High Invested Effort: information types with many instances archived, and that the user has organized. 	<p><i>Example: Passwords are all kept in the e-mail inbox in a special folder called “passwords.”</i></p>

Of the scraps we collected, a large proportion exhibited low effort – once captured, they were allowed to remain where they were placed. This placement was usually dictated by convenience of capture. In notebooks or text files, this pattern resulted in a chronological stream of scraps as new scraps were simply added to the beginning or end. For e-mail, participants seemed to leave scrap e-mails (e.g., e-mails to self) in their inbox

rather than filing them away in a sub-folder. Our results support earlier observations regarding engineers' lack of organization in their logbooks [104].

A fragmentation-like problem arose from participants' voluntary placement of scrap information in different places. The primary reason participants cited for writing information of the same type at different locations was convenience at time of capture. For instance, ADMN2 kept contact information (names, phone numbers, addresses) in her main notebook, a paper desk calendar, and a mini address book; and reported that where any piece of contact information ended up was determined by the location of the closest notepad. When asked about retrieval, she reported having to "rummage around" when she didn't remember where something was placed, that this often took time, and that she re-copied contact information between locations so that it would later be more easily found.

3.4.3 Constraints

Information scrap practice exhibits particular physical, temporal, social or structural attributes of a situation that may necessitate tool use. We suggest that these conditions play a particular role in information scrap generation that may not be as evident in tools that have a stronger correlation between task and tool. One such strong correlation is checkbook balancing: we typically open a personal finance application and start a file; we do not create an information scrap. While paying bills, we are fairly indifferent to the physical environment, time constraints, or social conditions – the tasks of personal financial management predicate the use of the recognized tool for that genre of information. We may each use different tools for that task, but generally, we would agree there is a tool that exists to support that task, and that's the one we use when performing that task. There are also certain agreed social conventions around this task. Writing a paper usually means that paper-writing is a primary task in terms of one's attention. It would be unusual to meet with a colleague while trying to pay bills. As such, with such strong conventions and tool support around a particular practice, we do not see information scrap challenges in capture, storage and retrieval.

If we were to instead jot down thoughts on the paper while meeting with a colleague about that paper, the temporal, physical and social constraints would have more of an effect on our choice of tool. For instance, it may be socially taboo to be seen either using a computer in this context, or to take lengthy notes during the meeting. Either condition may predicate quick gestures on the back of a note card. The social conditions of an

exchange may be such that it would break the flow of the conversation to reach for a more formal mechanism than a scrap of paper. Likewise, one may be on the move, so that not only is it awkward to use a more formal mechanism for recording data, it may be that there is simply not the time to engage in a detailed recording of an observation. So, currently, physical, temporal, social and structural factors have particular bearing on the devices selected and the kind of input generated with information scraps.

3.4.4 Caveats in Our Findings

The boundary between information scraps and the rest of our personal information remains fuzzy, especially in the case of high organizational effort. Once an amateur chef decides to collect all of her favorite recipes on a private blog and tag them by cuisine type, are the individual recipes still information scraps? In one sense, no – they have a place to reside and are reliably indexed. However, we cannot help but think that a blog might be a suboptimal method to collect and organize recipes, because it cannot take natively take advantage of recipe-specific needs such as ingredient filters or shopping list generation.

With regard to the variety of information types we catalogued, one difficulty with analyzing the frequency distribution of information types in scraps lies in drawing distinctions between similar information types. It is inevitable to question whether the categories we list could have been combined further, and thus the diversity lessened or erased. Our approach has been to group as aggressively as possible without losing the essence of the scrap's composition, attempting to find a rough lower bound on the strength of this long tail effect. A less aggressive grouping strategy produced results where ~25% of all scraps were unique types. Though individual pairs of categories might be further merged, we believe that the long tail effect is quite strong and worth noting.

Reflecting on our methodology, a clear limitation of our study stemmed from our use of interview and artifact analysis instead of live observation via shadowing. We were thus unable to study, *in situ*, how information scraps were used and created, but instead only how people reported they used their scraps, along with evidence from their workspace and the physical and digital artifacts we collected. The kind of statistics that our procedure was unable to capture included the number of information scraps participants generated each day and significant contextual associations. We observed that participants often exhibited a kind of confirmation bias toward their own organizational skills by mainly acknowledging well-organized work, so it is further possible that this also affected our observations; for example, our participants may have ignored particularly

embarrassing examples of disorganization. The methodology's strength was that it allowed us to observe a broad number of information scraps, more so than would have been possible *in situ*. We believe that an ethnographic shadowing methodology would complement ours well; it could objectively investigate many of the questions we could not.

Our triangulation method for locating information scraps was also not a perfect lens. We found that it was successful in unearthing a large number of information scraps in a variety of locations. On several occasions, only the last of the three dimensions we attempted (tool, location, and then type) successfully located a particular scrap. The triangulation method's strength lies in unearthing a wide variety of information; its weakness is that the number of information scraps found can be too numerous to examine thoroughly within the allotted time. In the future, we suggest the triangulation methodology might be modified to serve as a fast "tour" through a participant's information scrap landscape, allowing the investigators to then choose a small number of tools or locations to focus on.

3.5 Implications for Design

In this section, we ask: what design affordances would enable digital personal information tools to better serve these important and underserved roles of personal information practice? Do there exist any designs that support these affordances, and if not, how realistically can we expect such affordances to be supported by current technology? To ground this investigation, we discuss aspects of the design of Jourknow, our own note-taking research tool designed to better support information scrap activity, to be fully discussed in CHAPTER 4.

We derive these design opportunities to identify the design deficiencies in our current tools that most explain users' preference for information scraps in fulfilling the roles in §3.4.1. While we cannot predict whether addressing these needs would cause the demise of unmanaged scraps, we have evidence that addressing these needs would positively impact daily practice.

Table 3.4 outlines the set of affordances we have identified, including lightweight capture, flexible content, flexible organization, visibility and reminding, and mobility and availability. In Table 3.4 we have contextualized each affordance in terms of the activities and constraints that influence scrap generation.

3. ETHNOGRAPHY

Table 3.4. The major design affordances necessary for management of information scraps derive from access, scraps' contents, and organization.

Category	Observed Behaviors and Constraints	Derived Design Needs
Access	Finding the easiest and fastest tool to use. Constrained by effort required, limited time and attentional resources. “If it takes three clicks to get it down, it’s easier to e-mail.” - FIN2	<i>Lightweight Capture</i> : Record information with minimal effort or distraction. §3.5.1
	Adapting tool use to physical locations and social situations. Constrained by available tools. “At off-site meetings, I don’t have my infrastructure there, and keeping notes on paper is less rude.” - MAN1	<i>Mobility and Availability</i> : different capture methods may be necessary in different situations. §3.5.5
	Information scraps kept always in peripheral view, or in a location to be tripped over the next time the information would be relevant. Constrained by tools’ ability to be placed in-the-way. “If it’s not in my face, I’ll forget it. Like if it’s on the wiki – I have no idea it’s there.” - REC	<i>Visibility and Reminding</i> : information appears in the right place, at the right time. §3.5.4
Content	Scribbling, sketching, and annotation. In-progress, vague and underspecified information. Constrained by tools’ expressiveness. “Drawing is the way you really see it!” - ADMN4	<i>Flexible Content</i> : record any kind of data, at any level of completeness. §3.5.2
	Coping strategies when information does not fit other applications’ models, and collections of unusual information types. “There’s this problem: I wanted to assign dates to notes, but Outlook would only allow dates on tasks.” - MAN3	<i>Flexible Schema</i> : information may not fit existing molds. §3.5.2
Organization	Organizational strategies varying in degrees from completely disorganized to carefully filed, and avoidance of cognitively difficult filing decisions. “It’s too much work to decide which section it should go in – because sometimes things don’t fit in just one, or fit in multiple places. It’s hard to decide what to do.” - REC	<i>Flexible Categories</i> : Support for a variety of organizational strategies, as well as for transforming unfiled items into more structured, organized forms. §3.5.3
	Information scraps attached to or placed near related items. “I can never remember which [computer] is which. So I grabbed the gaffer’s tape and marked them!” - ENG3	<i>Flexible Linkage</i> : enable information to be linked to and in view with arbitrary other information §3.5.3

3.5.1 Lightweight Capture

As described in §3.3.5.1, we found information scraps often to be generated in response to a need to capture data quickly. This need occurred most commonly while the individual was performing some other attentionally, socially, or physically engaging primary task. For this reason, lowering both the actual and perceived cost of cognitive, social and physical effort may improve our tools. We see the following opportunities for reducing effort required during capture:

Avoiding upfront decisions and postponing disambiguation. Since information scraps are often captured at a moment when time, attention and cognitive resources are scarce, requiring individuals to make significant upfront decisions might incur sufficient cost to impede capture. Such decisions may include forcing the categorization of a new piece of information, choosing a reminder time, or setting parameters ultimately unimportant to the captured information. Thus, tools might aim to immediately handle information in whatever form provided to them by the user and postpone forcing user choices until a more appropriate time.

Avoiding task-switching, cognitive and navigational burdens. Navigational and cognitive costs associated with launching or switching applications contribute significantly to the time elapsed between the moment an individual forms an intention of writing something down and the moment they can actually start doing so. Since perceived time and effort during this critical interval have been found to dictate which tool will be chosen [66, 67, 86], we believe that minimizing navigational effort will improve a tool's capture rate and therefore overall usefulness to the user.

Supporting abbreviated expression of information. As described in §3.3.3, we found idiosyncrasies in participants' language – notes often represented very little explicitly and instead served as memory primers. Our finding contrasts considerably with most PIM applications' requirements that users complete forms with formal expressions for properties such as who, when and where. Our hypothesis is that tools can lessen the time and effort associated with entering information by supporting incomplete, informal capture methods.

Supporting diversity. If a tool is restrictive about the information forms it will accept, individuals will inevitably resort to a coping strategy – either imperfectly fitting the information into the tool, or fragmenting information by encoding it in another tool. Since coping strategies incur non-zero costs to devise and implement, and further lead to decreased effectiveness of future retrieval, we believe it worthwhile to accommodate

whatever information the user wishes to express. We discuss further issues with supporting diverse information forms in §3.5.2.

Capture from other tools. Given that a significant portion of the artifacts we examined originated from other applications and devices, (e.g., mobile phone, emails, web pages, IM conversations), we may reduce the need to create scraps by making it easy to select and pipe the relevant information from any application into an appropriate place. This desire also pertains to the need for ubiquitous availability of capture tools, discussed in §3.5.5.

Tablet-based notetaking tools such as Microsoft OneNote [17] have granted digital note-taking some of the expressive freedom of paper and pen. OneNote in particular also allows a users to categorize their notes post-hoc, such as by tagging to-do and contact items, thereby reducing the upfront time to capture. However, the OneNote user interface is monolithic and large, and therefore difficult to have “on the side” or to switch to while in the middle of another task.

Natural language expression interfaces for intuitive expressions of PIM information are another promising approach. These interface let users easily add information such as events, contact info and to-do reminders to their calendars and online PIM apps. Specifically, Google’s Quick Add [7] and I Want Sandy [12] have demonstrated potential for advantages in terms of navigational burden and efficiency over form-based GUI equivalents.

A number of cross-application launchers have recently gained popularity by turning common application launch and navigation activities into reflexive keyboard actions. The Quicksilver interface for MacOS X [20], for example, can be summoned from any application using a hotkey trigger, employs a small, unobtrusive interface to avoid obscuring the display of one’s main task display, and adaptively re-ranks to ensure that the most frequently used commands are easily accessible. However, these tools are generally application launch tools rather than information capture tools; they quickly land the user in the desired application and delegate information capture to the application interface.

Finally, with respect to tool integration, “snippet-keeper” application Yojimbo [24] facilitates cross-application content grabbing by letting the user simply select the content they want grabbed and pressing a hotkey. Users can immediately tag their grabbed items or choose to defer organization; the items are then added to collections in the person’s own tag-based Yojimbo repository.

While general-purpose drawing tools and word processors are potential candidates for information scrap management because they afford fast, unconstrained input of text or drawings, they are not ideal for several reasons. First, the design needs for creating published documents and free drawings differ significantly from those of scraps, especially in creation, use and semantic/structural characteristics. As discussed in §3.3.2, information scraps are often implicitly structured but sketchy and rough, whereas word processors and drawing tools are designed to create published or shared documents and illustrations. These tools thus foreground design affordances that are important to publishing but relatively useless for scrap creation. Also, these tools are not optimized for handling a large number of small data items: we observed a number of participants compiling large collections of scraps into a single text or word processor document in order to circumvent the overhead of creating and managing many documents (§3.3.5.1).

3.5.2 Flexible contents and representation

Our artifact analysis in §3.3.1.2 revealed that information scraps were considerably more diverse and irregular than the commonly considered set of PIM information types. Information often did not match expected schemas: some properties were missing, and others were introduced. Participants also commonly combined information types inside of a single scrap. These behaviors resulted in scraps such as contact items consisting of a name and a phone number, a time indicating when to contact the individual, and driving directions to the person's house – but no last name. Furthermore, as the distribution of types discussed in §3.3.1.2 suggests, there is a very large potential set of truly personal data types that collectively make up a significant portion of all information scraps but that do not fit at all in PIM applications today.

These observations indicate a potential need for more flexible representations of PIM data. Some PIM tools such as Microsoft Outlook have started to blur distinctions in PIM types, e.g., to-do items with calendar entries. Research tools such as Haystack [87] have taken a more radical approach, in which a general relational model called RDF [21] is used as a basis of representation. In such a representation, rather than having disparate collections of data records of particular fixed schemas, instances are defined in terms of how they link their atomic data components (e.g., dates, times or names), and linking is possible among arbitrary data components. Thus, under such a model it becomes possible to create and represent the (often implicit) meanings implied by the freeform scraps we found in the study.

A remaining challenge surrounds developing a means by which the user can utilize this expressiveness. Current interfaces for directly specifying instances using similarly rich vocabularies (e.g., [19]) carry high comprehension and execution overheads. Another option would be to automatically extract semantics; however, in practice this is a challenging problem because the language used in scraps (§3.3.3) is often incomplete, ungrammatical, and highly personalized. The difficult nature of the problem is reflected in the fact that personal notes are often ambiguous and unintelligible to people other than the author.

Controlled naturalistic languages such as those first proposed for databases [112] provide a possible solution. In a simplified natural language, user are informed that the system can only interpret a restricted set of simple, common phrasings for information (or using some fixed syntactic convention) but that they are free to express anything they wish to using this language. This technique trades off expressiveness for perceived naturalness of expression.

3.5.3 Flexible usage and organization

The tool adaptations described in §3.3.4.3 are particularly interesting because they reveal individuals' needs: participants devised new custom organizational systems out of existing tools to better fit their needs, for example ENG4's re-appropriation of a bug-tracking tool as a personal to-do list and MAN3's use of Post-it notes on the back of his cell phone as a capture solution. Our study also revealed several instances where tools had to be adapted in order to accommodate information that didn't fit, such as when an application failed to provide free-text annotation capabilities.

Re-appropriation requires tools to be sufficiently flexible to accommodate novel use of their affordances or data. Many high-functionality information tools (e.g., emacs [6] and Eclipse [4]), have long allowed scripting and extension facilities for customization. The practical difficulty associated with developing application-specific plug-ins was such a high barrier to entry that few users attempted it. However, the past few years have seen a new trend in the popularization of open web-based APIs to data services. These APIs and reusable widgets have allowed web developers to combine functionality from multiple online applications into their own custom applications, spawning the phenomenon of mash-ups. Tools such as Intel's MashMaker [61] and Dontcheva et al.'s card metaphor [58] have attempted to make this process even more accessible to end users.

Participants also devised a rich set of organizational techniques and strategies, as summarized in §3.3.4. Some of these behaviors are already

well supported in digital information tools, while others are poorly supported. For example, unlike notes written on physical media, digital PIM tools are good at automatically organizing collections of information records. However, these tools are generally less capable of adapting to novel organizational strategies, so our participants tended to use spreadsheet software if they needed to manage or sort novel fields. We propose three potential solutions: letting users manually specify organizational rules, learn organizational rules by example [69], or dynamically construct faceted views by combining of a simple set of operators (e.g., [75, 139]) in a user interface familiar from online commerce.

Capturing physical organizational strategies in the digital realm has been more challenging. Visual layout systems involving stacks and piles [103, 116] enable 2- and 3-dimensional layouts and implicit Gestalt grouping. Other possibilities include visual wear and tear on heavily-used items [72] and time-based metaphors [62].

3.5.4 Visibility and Reminding

Nearly all participants employed a strategy of physically situating scraps in places where they could serve as references or reminders. As described in §3.3.4.3, the desire to be able to reference useful information frequently easily inspired participants to place items in prominent, always-visible locations, for example sticking Post-Its to their workstation monitor or writing information in the corner of whiteboards. For prospective reminding, several participants reported strategically placing notes in locations where they knew they would later serendipitously “trip over” them at the right time.

It is difficult to support these behaviors in digital tools because we cannot easily situate pieces of information in particular locations of easy access or strategic significance. The problem of physically situating digital information is still solved most straightforwardly by first converting the information to physical form (*i.e.*, printing it) and then sticking the physical version in the appropriate place. Display technologies could contribute to making it easier to physically situate digital information, including low-cost electronic displays such as e-ink [3], multisensory ambient displays [48, 55], and pervasive displays such as the EveryWhere Display [110]. An alternative approach is to build location sensors into portable displays and to display information grounded at the user’s location; efforts in this vein include the Remembrance Agent [114] and augmented reality research (*e.g.*, [2]). Still other work [74] has sought to reproduce the kind of passive reinforcement that occurs when we shuffle through our

notes or flip through the pages of our physical notebooks while looking for something else.

A separate reminding problem surrounds reminding individuals of what their notes mean. As we discussed in §3.3.2, the brief, incomplete nature of scraps meant that they were not necessarily future-proof, although memory for the general gist was strong. One participant, MAN4, strongly expressed a desire for helping him remember the significance of notes: “The thing I want the most in a note-taking tool would be to be able to ask it – Who? What? ‘Why?’ – ‘Why’ is absolutely crucial. If my sticky-note could answer this for me I’d be golden.” Research surrounding ways to support long-term recall of events includes *Stuff I’ve Seen* [60], which attempts to contextualize events in terms of other more memorable events, or features by long-term video transcripts [54], or images taken at random from a wearable camera [122]. These projects have demonstrated substantial gains in duration and fidelity of recall of routine workplace situations and events. We believe that similar recall effect would occur for the meaning of notes, as well as for helping in the re-finding of lost notes.

3.5.5 Mobility and Availability

Information scraps are often closely tied to mobile scenarios (§3.3.4.2). Social constraints may dictate the availability or appropriateness of tools in certain settings (§3.4.3). In response, many of our participants resorted to carrying legal pads, day planners or pocket sketchbooks whenever they were away from their desks.

To support capture in a mobile context, nearly every mobile smartphone and personal digital assistant (PDA) provides some basic PIM and freeform notetaking functionality. However, adoption varies widely among users [56]. Our study elicited two major impediments to the use of such devices for notetaking and personal information management: 1) a choice between fragmenting information across devices and synchronizing the mobile device, and 2) the difficulty, time and attention costs associated with mobile information entry.

Fragmentation can be ameliorated by synchronization, and the synchronization problem is mainly an engineering one. One option is for web- or desktop-based tools to offer mobile-accessible capture and access interfaces, as is the case with Google Calendar [7] and Microsoft OneNote Mobile [17]. Increasingly high-bandwidth wireless networks have opened the door for automatic synchronization [Nokia], though automatic synchronization is not standard practice as yet. Another choice is for interme-

diary clients to pass information between the mobile client and the desktop: for example, Jott [13] translates phone voice commands into calendar appointments, to-dos, etc. via an API agreement with web services.

With respect to the barrier of data entry, two options are innovation in text entry methods and innovation in mobile capture modalities. A variety of keypad form factors are exploring the former option, for example placing full QWERTY keyboards on phones or implementing text prediction as with the iPhone [11]. However, alternative approaches to notetaking incorporating paper-and-pen are also gaining traction in products [10] and with additional functionality in research circles [126, 140].

3.6 Future Work

An important next step for this work is to extend our artifact and interview study by observing scrap creation and re-finding in situ. We have focused thus far on understanding scraps' contents, tools and organization, examining artifacts after they have been created and before re-finding was needed. Our research does not paint a complete picture of creation and re-finding as they occur. What exactly triggers the need to record or reference an information scrap? What kind of information is recalled about each scrap at intervals after its capture? What are the most typical re-finding procedures, and how might we support them? A shadowing study would likely elicit many interesting (and likely unexpected) answers to these questions.

3.7 Conclusion

In this chapter we examined the phenomenon of the information scrap: personal information that does not make its way into our current personal information management tools. Information scraps are pervasive – our participants shared with us an impressive number of scraps, both physical and digital, scattered over diverse parts of their information environments. We analyzed the wide variety of information types held in scrap form, underlining the criticality of supporting unusual or unique types in information scrap work. By examining the composition and life-cycle of information scraps, we have identified a set of user needs for information scrap management: lightweight capture mechanisms, freeform and potentially unstructured data representation, flexible organizational and usage capacities, visibility and the in-the-way property, and mobility. These strengths drive a set of typical information scrap roles, including

3. ETHNOGRAPHY

temporary storage, cognitive support, reminding, archiving, and capture of unusual information types.

Our goal with this work is dualistic and somewhat contradictory: we want to understand and support information scraps, so that we can reduce their number. Information scraps are an outcome – the result of a coping strategy – rather than an outright goal. The needs driving information scrap creation will remain constant, as we will always require lightweight capture, support for unusual data and flexibility. The scraps, however, can be reduced, eliminated or redirected by carefully considered redesigns of our personal information management tools.

The wide variety of information encoded in information scraps is galvanizing, as it suggests an opportunity for PIM to engage new types of information. The data indicates that a significant percentage of our personal information is beyond the reach of our current generation of tools, and furthermore will likely remain so without a significant recalibration of our goals. We identified over 125 information types from our sample of participants, and surely there are others. The diversity suggests that it may not be tractable for each of these information types to be managed by its own tool. Instead, we suggest that the future of personal information management may lie in finding a flexible, unified platform to corral as much data as the user cares to support [88]. First steps have been made in this direction (for example, [84, 87, 131]), but we have much distance yet to cover. Data flexibility will be in tension with a clear demand for rapid capture, driving a need for novel methods for capture and organization.

Taken in sum, these conclusions specify a set of problems that will be challenging at best. Yet the challenge is necessary, even revolutionary. For PIM to move beyond its current limitations, it must venture beyond its established boundaries – and into the world of the information scrap. This is the world we enter in the following chapters.

4. JOURKNOW: INFORMATION SCRAP CAPTURE, MANAGEMENT AND RE-FINDING

This chapter details a novel scrap keeping program called Jourknow (Figure 4.1, Figure 4.2). Jourknow attempts to re-envision the entire information scrap lifecycle: from capture to organization, transfer and retrieval.

Jourknow's goal is to be an information scrap catalyst of sorts, ensuring that scraps reach a more useful state without extra effort on the user's part. If the user is willing to use a quick-launch command to capture a thought, we can attach contextual information to it in order to aid re-finding. If the user is willing to add some lightweight structure to the text, we can then provide further support by pushing the information into the appropriate application or providing structured retrieval mechanisms. As a result, Jourknow attempts to manage an incredibly broad range of personal information, from calendar events and to-dos, to guitar tabs, comic book collections, and wholly freeform notes. It approaches this broad design space via designs catering to different subsets of this information.

At its core, Jourknow is a text-based notebooking tool for computer desktops and laptops. Text is an appropriate capture mechanism in such environments; when we enter text, each stroke is being used to record the actual information we care about – none is wasted on application navigation or configuration. The linear structure of text means that there is

Work presented in this chapter is a collaboration with Max Van Kleek, David Karger and mc schraefel. It is published in UIST '07 [131] and a CSAIL Technical Report [37], as well as workshop papers at PIM 2008 [38], SWUI 2008 [130], and HCIR 2007 [36].

4. JOURKNOW

always an obvious place to put anything – at the end. And the freeform nature of text means we can record anything we want to about anything, without worrying whether it fits some application schema or should be split over multiple applications.

However wonderful its capture affordances, however, text’s weakness lies in retrieval. Text’s fixed linear form reduces us to scanning through it

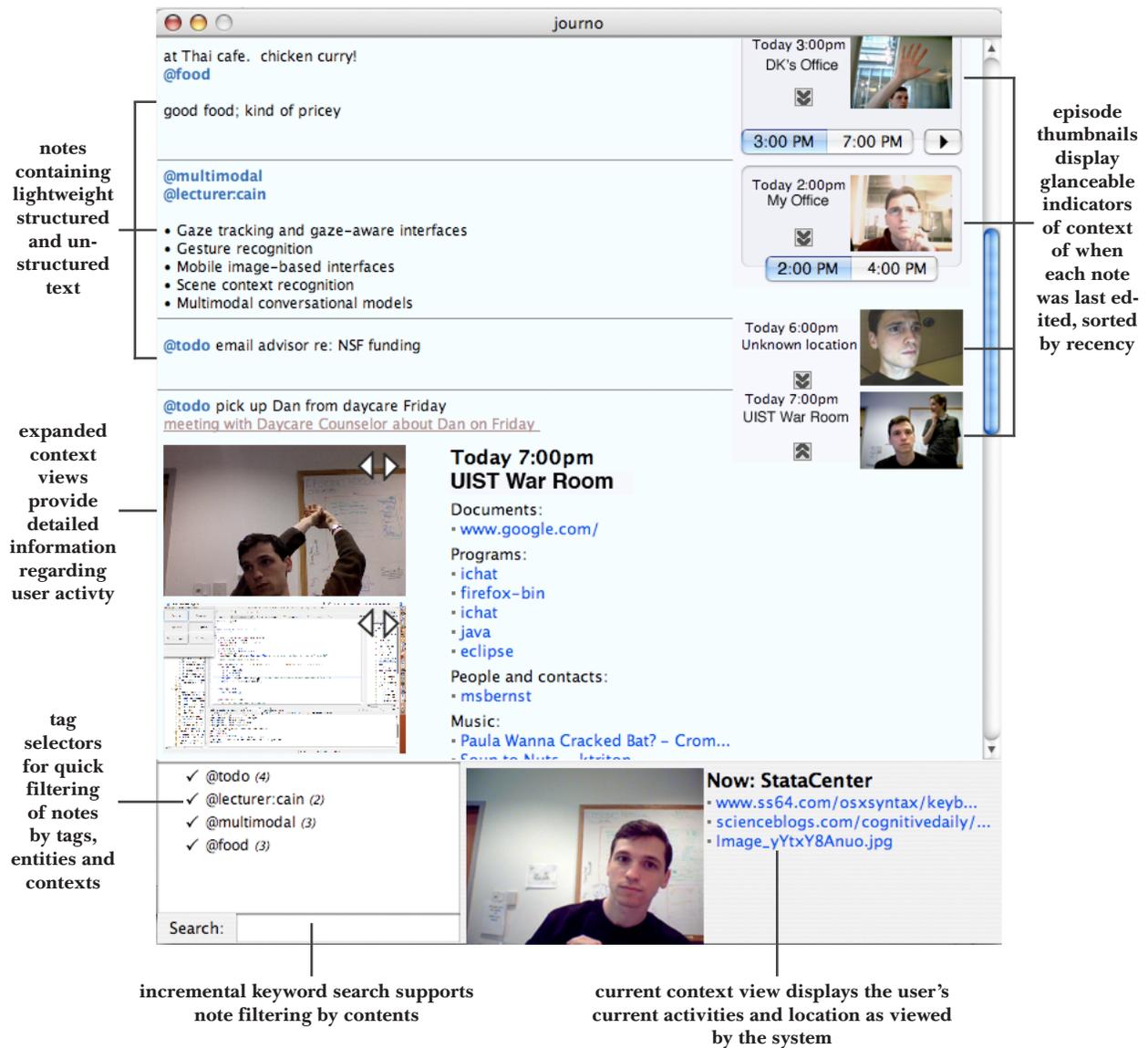


Figure 4.1. The first-generation Jourknow user interface. The annotations point out context thumbnails and tags associated with notes.

for information we need. Even with electronic text, the lack of structure means we cannot filter or sort by various properties of the information. When we aren't sure what we want, a blank text search box offers few cues to help us construct an appropriate query [34]. The shorthand we use to record information in a given context can make it both hard to find and incomprehensible when we return to it later without that context [19]. Furthermore, only text we explicitly enter is recorded, without any of the related contextual information (such as a timestamp) that might be known to a sophisticated application.

Jourknow's goal is to combine the easy input affordances of text with the powerful retrieval and visualization capabilities of graphical applications and personal information management tools.

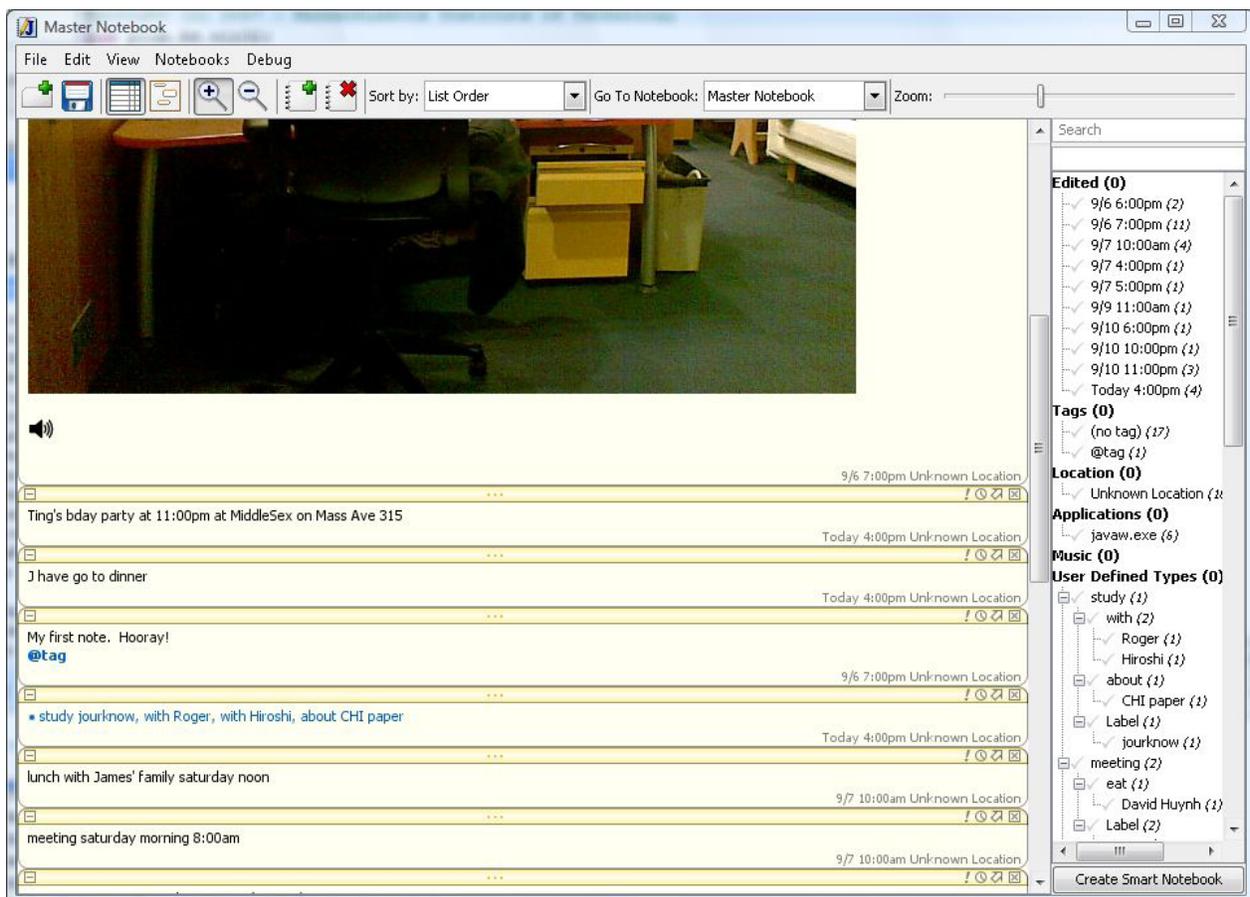


Figure 4.2. The second-generation Jourknow user interface. The second iteration includes the faceted browsing panel on the right, as well as images and sound files in notes.

4.1 Jourknow's Design and Research Contributions

Jourknow (Figure 4.1, Figure 4.2) is designed after a notebook metaphor, much like commercial tools such as OneNote [17], Yojimbo [24] and Evernote [5]. Its research goals can be viewed as falling into several major categories:

- Capture: Lightweight unstructured and structured entry
- Manipulation: structure inspection and exploitation
- Re-finding: Faceted Browsing and Context-based re-finding
- Mobility: Unique design needs in mobile scenarios

In the sections that follow, I detail design innovations Jourknow makes in each of these categories.

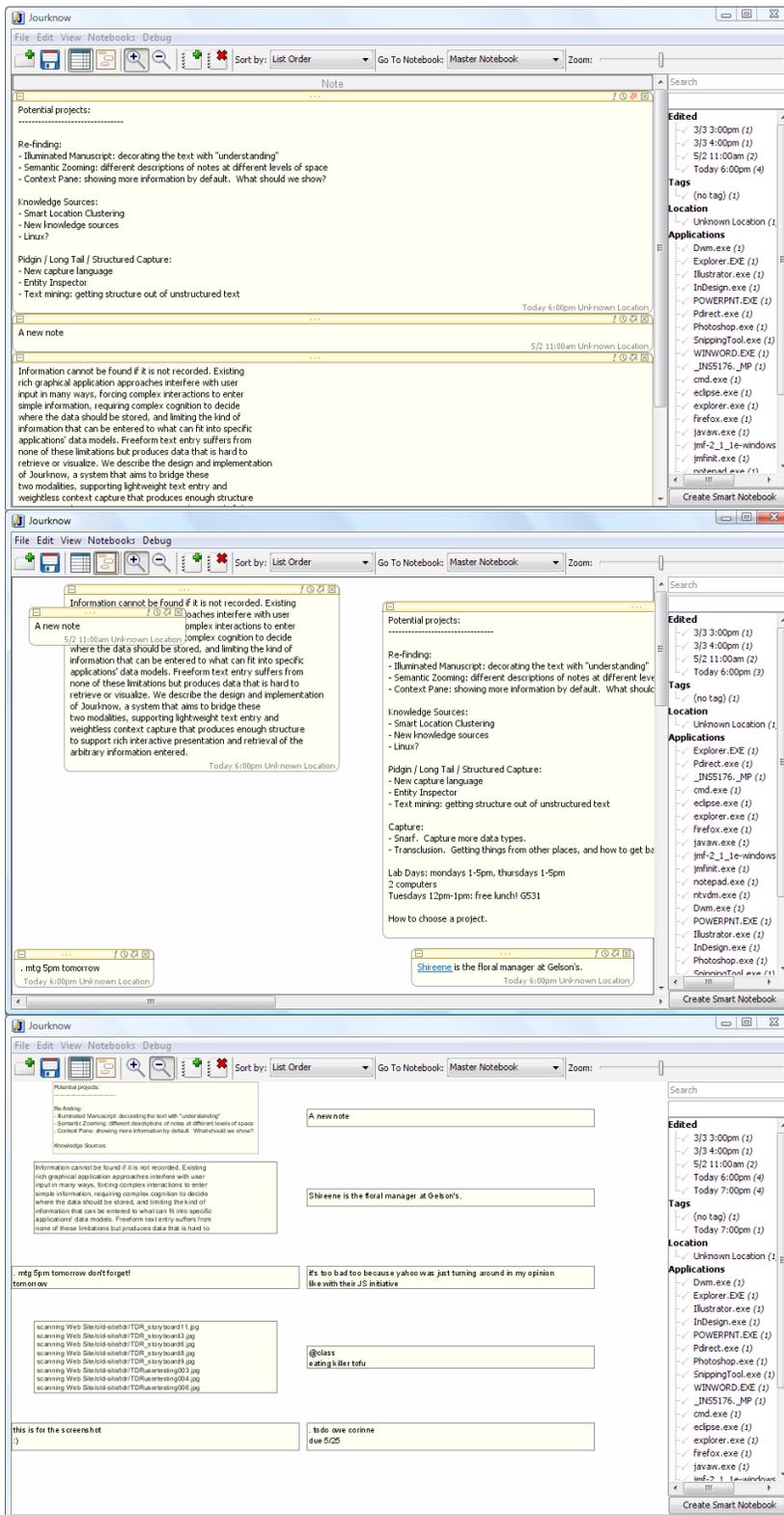
4.1.1 Notebook Interface

Jourknow's organizing principle is that of the notebook. The default view (called the Master Notebook) contains all notes. The user may create additional notebooks and add notes to them, or instead create smart notebooks. Smart notebooks are analogous to smart searches or smart playlists in Windows Vista and iTunes, respectively – the user creates a persistent query such as “all notes tagged as @jourknow or edited at Starbucks,” and the smart notebook will continually update to contain all notes matching the query. Interacting with notebooks is straightforward – a drop-down menu allows the user to switch between notebooks in view.

Jourknow provides three main views of notes: a list view (the default), a freeform 2-dimensional canvas, and a zoomed-out view that utilizes semantic zooming.

The list view (Figure 4.3) is a tabular view common in WIMP interfaces. Notes are added to the bottom of the list as they are captured, and are rearrangeable via drag-and-drop. Additional columns are available: creation time, last modified time, user location, programs open, files edited, music listened to while recording the note, desktop photo, and user photo. Notes may be resorted by the columns that are well-ordered.

In the canvas view, notes are laid out like cards on a large 2-dimensional canvas (Figure 4.3). The notes may be arbitrarily resized and moved around on the canvas. The canvas view was designed to support the physical grouping and repositioning our study users employed so successfully. Each new note is added to the bottom of the canvas initially. The canvas view can be zoomed out to an arbitrary viewpoint, employing the same



List View
notes in a vertical scrolled list

Canvas View
notes repositionable in a 2D plane

Flow View
zoomed-out view for browsing and search results

Figure 4.3. The three main views in Jourknow.

4. JOURKNOW

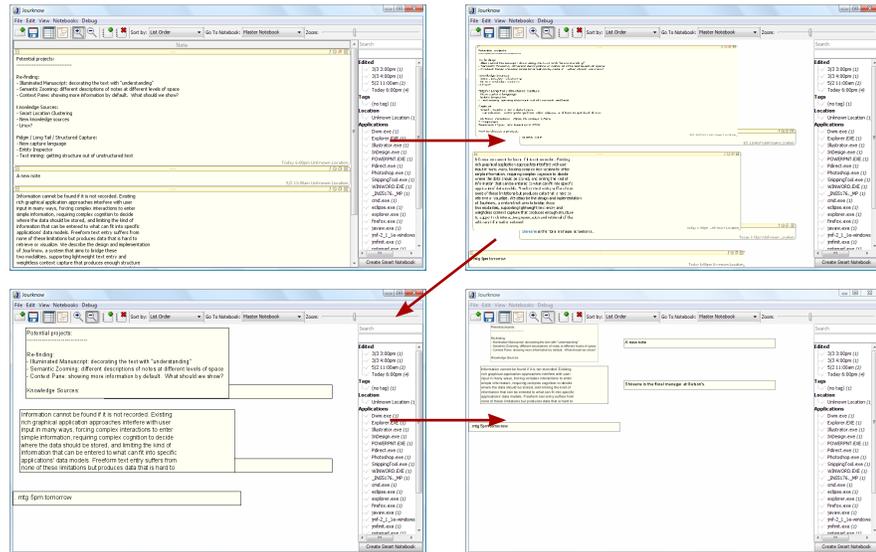


Figure 4.4. Animating from list view to flow view.

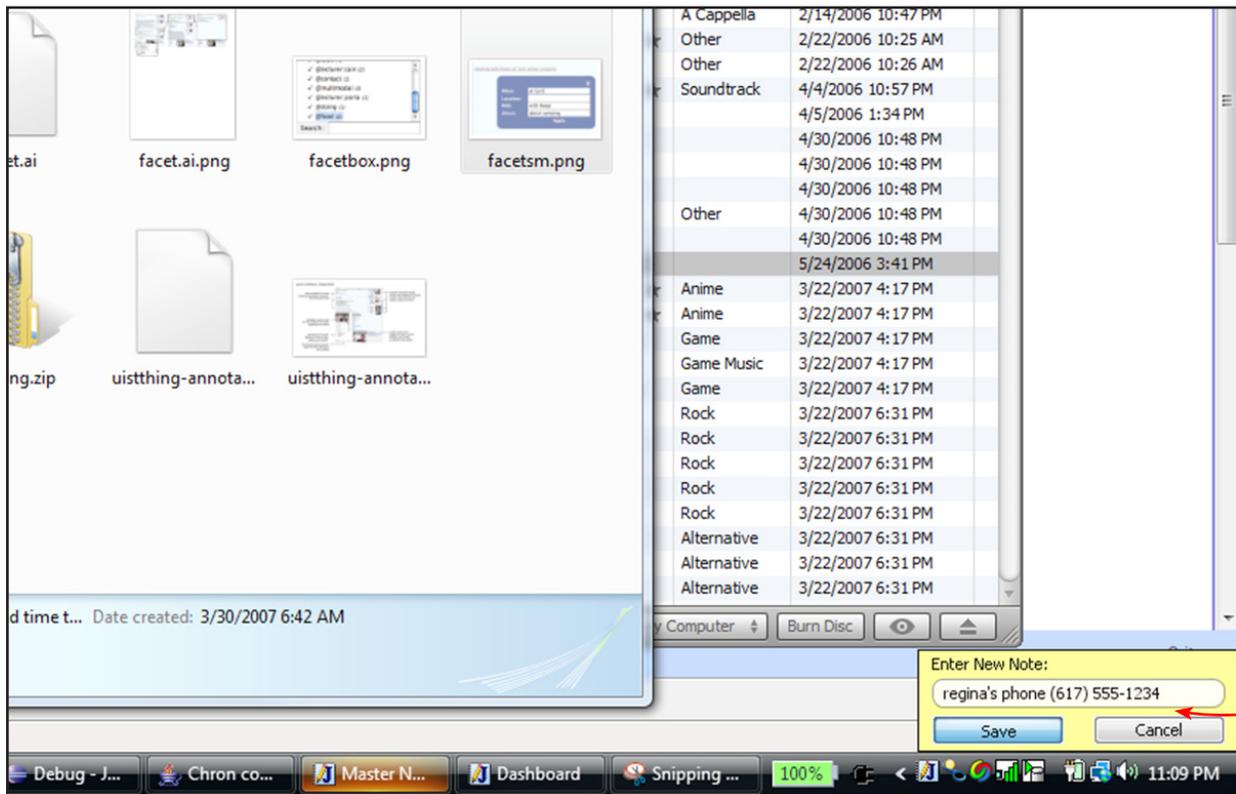


Figure 4.5. The quick capture window can be called up via a hotkey. It overlays the contents of the screen and optionally pastes in the selected contents of the previous application.

semantic zooming techniques described below. Canvases can be created containing subsets of notes by creating notebooks with only those notes inside.

Upon text search, the list view and canvas view animate to a zoomed-out flow view of note previews (Figure 4.3, Figure 4.4) to support visual scanning. As search terms eliminate notes from consideration, they fade from the flow layout and new results take their place. Search results are sortable by the same criteria described above. When the user clicks on a desired note, the program zooms into the note in the previous view mode.

Semantic zooming [109] is a user interface technique wherein the program continues to display an object's most salient identifiers at normal size when zoomed out, and zooming in may expand previously singular items into constituent parts. Jourknow employs semantic zooming when the canvas view or list view are zoomed out or in search view. The zoomed-out notes continue to display images and the first few lines of text, so that the note is still readable.

4.1.2 Capture: Lightweight unstructured and structured entry

Information scraps cannot be managed if they are never captured into our system, and our ethnographic work (CHAPTER 3) suggests that users are extremely unforgiving of time and effort wastes when they wish to capture a thought.

Jourknow provides two major facilities for capture: quick launch, for all notes, and Pidgin languages, for notes which carry implicit structure such as meeting times and locations.

4.1.2.1 Quick Launch

Jourknow's main method of note capture is via a hotkey launcher: a key combination mapped to Ctrl-Alt-Spacebar that launches a small capture window (Figure 4.5). Mac OS X-specific versions of Jourknow allow for a second combination that automatically copies whatever is selected in the current application into the window, allowing the user to seed notes with web pages, blog posts, documents, or whatever else he or she has encountered. The user may then type into the window and record a note or annotate the text that was copied in. This technique is not a terribly innovative interface design; however, it may save precious seconds at capture time.

4.1.2.2 *Pidgin: Lightweight Structured Input*

In linguistics, a Pidgin is a simplified language used to communicate with another person whose language is not your own. The foreign tongues at odds here are the abbreviated and incoherent text common in information scraps on one hand:

```
mtg w/ karger @ 5pm
```

and highly structured representations used by computers on the other:

```
type: meeting;  
title: [none];  
attendees: Michael Bernstein and David Karger;  
time: 5:00pm;  
date: May 23, 2008.
```

We know that users are often unwilling to enter the structured version of information, creating information scraps instead. Can we instead bring users and computers to a middle ground – quick and simple for the user to enter, but providing enough structure for the computer to interpret and, for example, place in the user’s calendar or add to the shopping list?

This bridging between information scrap language and computer-interpretable language is the goal of Pidgin. When the user enters a note with the text:

```
mtg w/ karger @ 5pm
```

Jourknow recognizes the text as a meeting and attaches the relevant structure. Jourknow makes this structure useful by pushing the information into the user’s calendar application and by making it browseable within the interface (described more fully in §4.1.4.1).

The meeting Pidgin above is restricted to a predefined grammar – here, one which captures various means of expressing meeting events. Predefined grammars allow for considerable flexibility in the user’s expression of data:

```
mtg w/ karger @ 5pm  
meeting 5pm karger  
calendar tomorrow 5
```

However, these grammars can often be inscrutable (“why did it parse that way?”), and rely on a specialist authoring each grammar. We can author Pidgin grammars across a variety of common PIM types found in information scraps such as to-dos and contact information.

Our studies also revealed that users maintain large numbers of uncommon data types such as shopping lists, UNIX commands, and recipes. Such items can be captured via a second type of Pidgin language that is

designed to be extensible by the end user. The following are examples of such a Pidgin, called TurtleDove, available in Jourknow:

```
meeting with karger, time 5pm, date tomorrow
```

```
thesis Information Scraps: Understanding and Design, due 5/23/08, author Michael Bernstein
```

```
shoppingList this weekend, dairy milk, dairy butter, meat ground beef, vegetables onions
```

```
restaurant In 'n Out, review 4, cuisine American
```

```
restaurant Kaze Shabu Shabu, review 5, cuisine Japanese
```

TurtleDove is syntactically unambiguous, so it can be used to capture any information the user cares to enter. The grammar adheres to the following general form:

```
TypeName ObjectName, [PredicateName ValueName,]*
```

TypeName and PredicateName are constrained to be only a single word. So, a TurtleDove entry above would be structured as the following:

restaurant	In 'n Out	,	review	4	,	cuisine	American
type	name		predicate	value		predicate	value

The parse would translate into the following object:

```
type: Restaurant;
name: In 'n Out;
review: 4;
cuisine: American.
```

Jourknow makes this structure available to the user for exploration and browsing purposes, as described in §4.1.3.2.

TurtleDove requires the user to signal the beginning of a Pidgin phrase with a period. This syntactic clutch incurs an extra cost on the user – however, Jourknow can provide extra input support when it knows the user

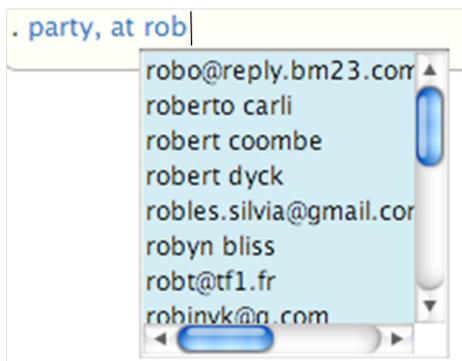


Figure 4.6. The TurtleDove autocomplete drop-down assists with completion of Pidgin statements. Here it is suggesting all known values for the predicate ‘at’ that begin with ‘rob.’ The list has been initialized with names from the user’s e-mail.

is entering a TurtleDove expression. Autocomplete is one such technique – Jourknow automatically suggests previously used values to speed entry (Figure 4.6). Specifically, the autocomplete dropdown will suggest all previously used types at the beginning of a phrase, all previously used names for the given type, all previously used predicates at the beginning of each predicate clause, and all previously used values for the chosen predicate. Thus, the autocomplete dropdown will suggest ‘review’ and ‘cuisine’ when the user is typing a new predicate for a restaurant; once the user chooses ‘cuisine,’ then the autocomplete dropdown displays ‘Chinese,’ ‘American’ and other previously-entered cuisine types. The dropdown also auto-populates with entities from the user’s computer, such as those mined from e-mail (Figure 4.6).

A more full discussion of the space of possible Pidgin designs is postponed until CHAPTER 5).

4.1.2.3 *Implicit Structure Capture: Context*

Contextual information is a final source of lightweight structure – one that requires no explicit user intervention. Jourknow utilizes the Personal Lifetime User Modeling (PLUM) system [132] to capture user activity at all times. This activity information includes:

- Application in Focus
- File System Access
- Active/Idle State
- Web Browsing (Firefox, Safari, Internet Explorer)
- E-Mail (Microsoft Outlook, Apple Mail)
- Document Viewing (Microsoft Office, Adobe Acrobat, Apple Preview)
- Location (GPS coordinates based on WiFi triangulation)
- Chat (iChat, Adium)
- Music Listening (iTunes)
- User Photo (Webcam embedded in laptop)
- User Desktop Screenshot

The preceding activity information is automatically logged and associated with timestamps on each character in the note, as described in §4.2.1.1.

4.1.3 **Manipulation: structure exploitation and inspection**

Capturing structured data is only worthwhile if that data can be put to work for the user. The most straightforward application of structure is for re-finding: note structure becomes immediately searchable and browsable within a faceted browsing interface [139]. Visual context summaries

assist users in reconstructing situations surrounding note capture. Recognized Pidgin expressions are pushed into external applications such as calendars and to-do managers, and benefit from specialized browsing interfaces.

4.1.3.1 *Application Integration*

Many Pidgin expression can be “brought to life” and manipulated like objects in traditional PIM applications, used to set reminders, or sorted and filtered by property or value. To maximize their availability and utility, Jourknow exports a view of Pidgin expressions that represent PIM data types such as events, contacts, and to-do items to the user’s standard PIM applications. Edits via these external representations are reflected in the text, and are made visible in the through a revision indicator.

For example, the Pidgin `mtg at Luna Cafe @ 5pm w/ Akemi cell 617-851-1294 re:camping this weekend` encodes the fact that there is a meeting that is happening at a location known as the “Luna Cafe,” at 5pm today, with a person named “Akemi,” whose cell phone number is “617-851-1234,” on the topic of “camping this weekend.” Instantiation of this Pidgin element causes an event to appear in the user’s calendaring application with the appropriate date, time and subject, as well as contact information to appear (if one didn’t already exist) for a person named “Akemi” with the appropriate phone number, in the user’s address book.

4.1.3.2 *Structure Inspection*

Jourknow allows visual exploration of Pidgin expressions. By right-clicking on the name of the object in the text, the user can see a tabular view of all information known about this entity (Figure 4.7). This information is gathered across all notes, so if the user were to assign Mike’s Pastries a rating of “5” in one note and leave a comment in another note, the inspection window will show both elements. This technique is useful, for example, when the user wishes to jot down individual shopping list elements throughout the week and have them collated into a list right before a shopping trip.

This inspection panel is sensitive to the content being selected. If the user were to right-click “restaurant” instead of “Mike’s Pastries,” the entity panel would list all known restaurants (Figure 4.7); if he or she clicked on the “5” in “rating 5,” the panel would display all entities which the user had given 5 stars.

4. JOURKNOW



Figure 4.7. The structure inspection interface. Top: three TurtleDove statements. Middle: the user has right-clicked on “Mike’s Pastries” and is viewing a tabular version of all information known about Mike’s Pastries. Bottom: the user has right-clicked “restaurant” and is viewing a list of all restaurants in Jourknow.

All Pidgin expressions, not just those pushed to external applications, support browsing and exploration in the Jourknow interface. Pidgin objects are added to the facet panel (Figure 4.8), so that the user may quickly find all notes with Pidgin meeting agendas in them, or all meetings that occurred with Victor Zue present.

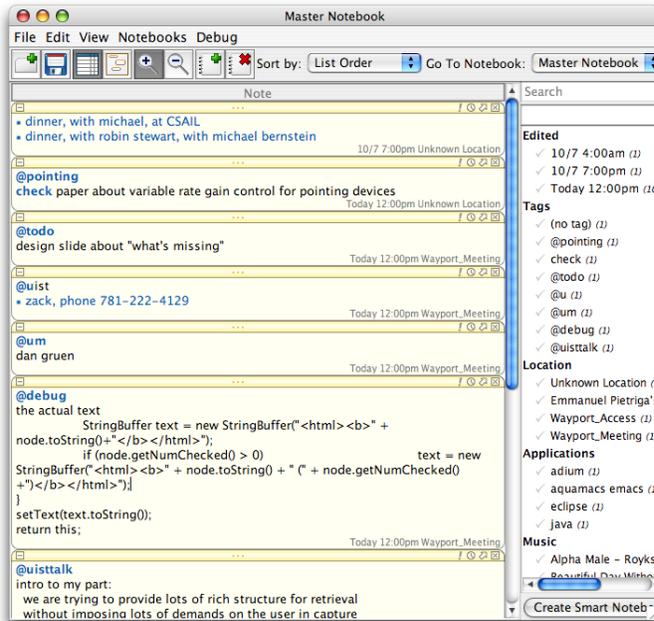
4.1.4 Re-finding: Faceted Browsing and Context-based re-finding

Given the speed at which we typically fill our notebooks, Jourknow’s interface needs to scale appropriately to hundreds of thousands of notes. This section details two techniques Jourknow includes for just such a purpose: faceted browsing and context-based re-finding.

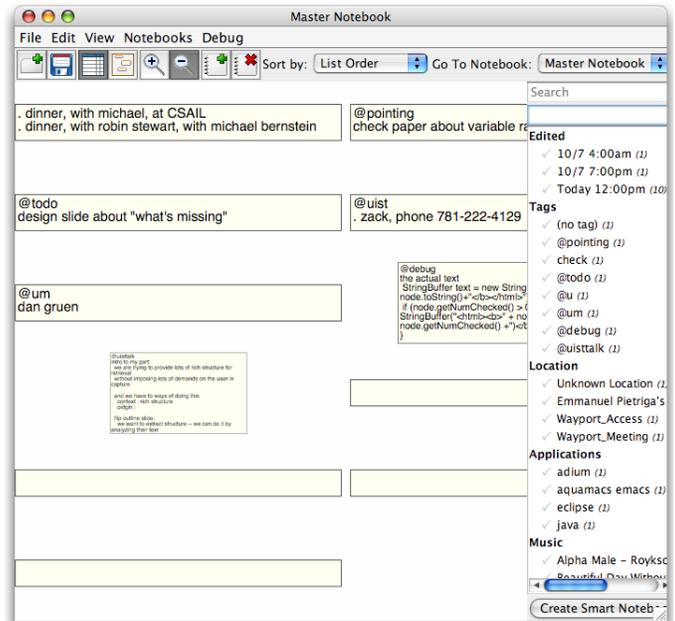
4.1.4.1 Faceted Browsing

Jourknow makes all of the information it captures available to the user in a faceted browsing interface (Figure 4.8). This information includes mined context associated with notes, Pidgin information, and basic note information such as creation time. Faceted browsing [139] allows the user

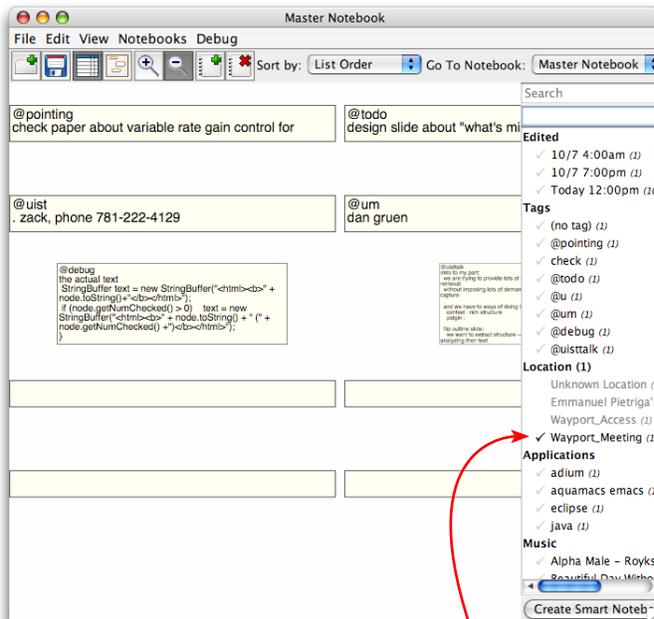
4. JOURKNOW



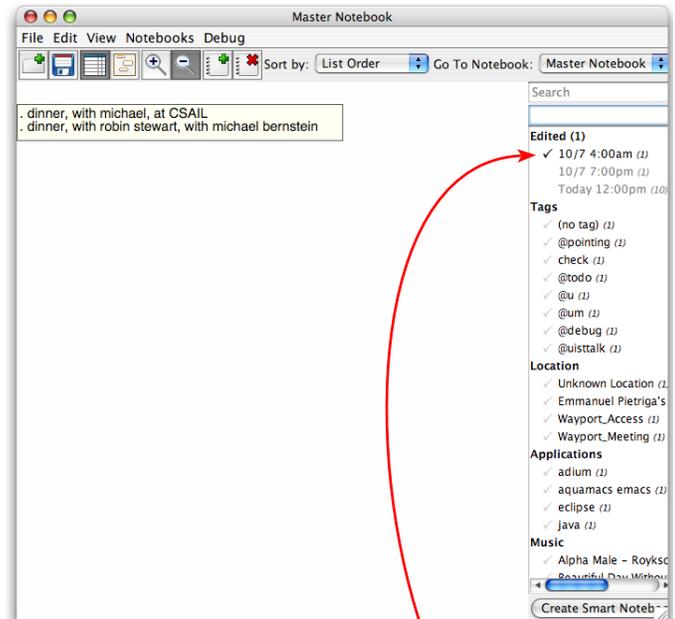
Default List View



Zoomed out to Flow View



Wayport Meeting selected in Location facet



4am selected in Edited Time facet

Figure 4.8. The faceted browsing panel is a means for browsing notes. Upper left, the typical list view. Upper right, the user has entered the Flow view on the same notes, preparing to search. Lower left, the user has selected Wayport Meeting in the Location facet, restricting note display to only those notes taken while on the Wayport Meeting wireless. Lower right, the user has selected only notes taken in the 4:00 hour on October 7th. (This user is a night owl.)

to explore his or her notebook by a combination of any of these features, visually constructing queries such as “all notes created today when I was in Starbucks and listening to Metallica.”

4.1.4.2 Context-based Re-finding

Consider `john 617-555-6835`: “Which John? How do I know this person? Was I supposed to call him? *What was I thinking?*”

Many notes deliberately leave out information, leaving only salient clues to aid reconstruction of meaning. Often this tactic works over the short term, but memory degrades within a month [86].

Jourknow harnesses the contextual information it has mined to aid this memory process. Each note contains an expandable context pane that displays the most salient contextual information surrounding the note capture (Figure 4.9). In the case of John’s phone number, it might remind our user of where he was when he took down the note, who he had been interacting with via e-mail and possibly a picture of the individual if he happened to be standing over the user’s shoulder when the webcam triggered.

The contextual information is available for reverse lookups as well: “I met a business contact at Starbucks last time I was in Chicago, and he gave me his phone number. I know I wrote his name and number down, but I can’t remember either of them!” This contextual information can be used to create a search in the facet panel, narrowing down the set of visible notes to just those taken in a Chicago Starbucks.

4.1.5 Mobility: Unique design needs in mobile scenarios

Our ethnography participants reported that their tools were often rendered useless when not available, for instance when away from their desks, driving to work, or at home. Scraps already entered into the tool could not be retrieved, and new scraps could not be added unless a temporary medium such as e-mail or cameraphone were employed. Our hypothesis was that the Jourknow system would become much more viable when note creation and note referencing is possible in a mobile scenario, allowing for the unique affordances of mobile phones such as picture taking and audio recording.

To address this issue, we created Jourmini, a mobile phone Jourknow client (Figure 4.10). Jourmini implements a subset of the functionality in

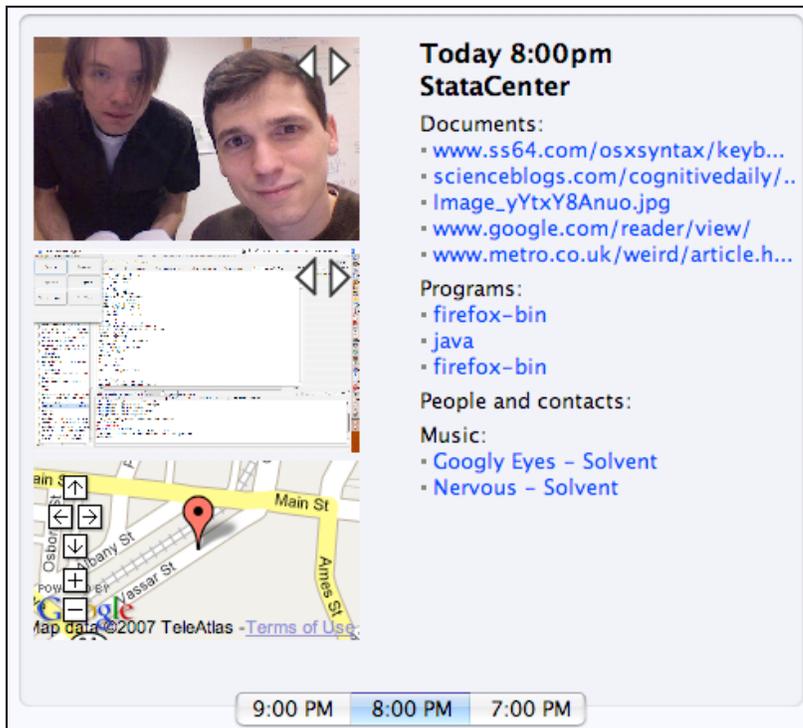


Figure 4.9. The expanded context view associated with a note.

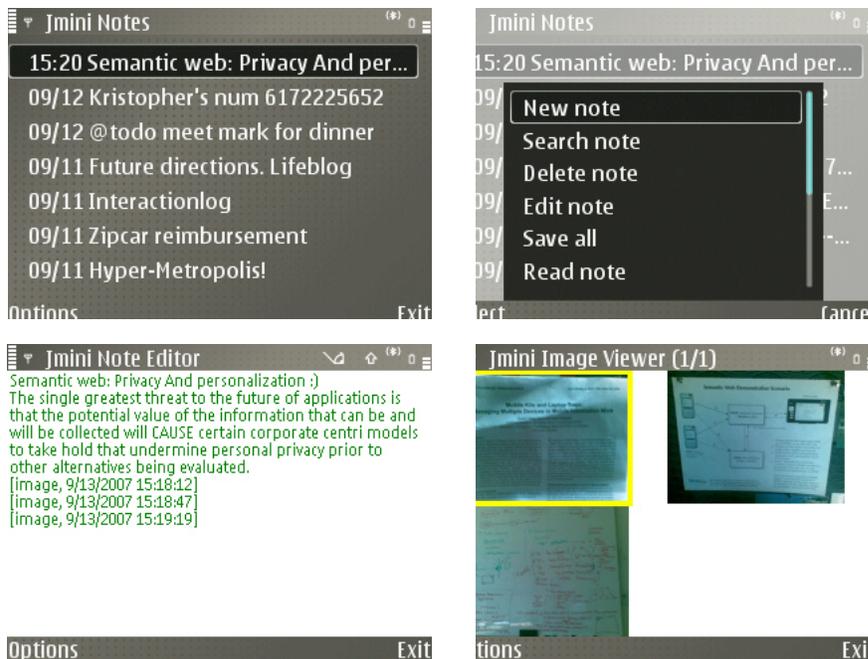


Figure 4.10. Jourmini, the mobile phone Jourknow client. Clockwise from the upper left: a list of notes, commands available, single note editor interface, and a view of all images attached to a note.

Jourknow, as well as a few phone-only features. Jourmini synchronizes its notes with Jourknow, so any notes created on the desktop are available on the mobile device and visa versa. The phone interface allows for note creation via keypad text entry, full-text searching and viewing. In addition, it allows users to attach photos from the phone's camera and audio recordings directly to the note, and later review them.

4.1.6 Visibility and Reminding: Desktop Dashboard, Importance Indicator and Alarms

Visibility and reminding are important in any information scrap manager (§3.5.4). Jourknow addresses these needs through three major design points: a note dashboard on the desktop, a visual importance indicator on notes, and scheduled alarms.

The note dashboard is a thin display visible on the user's desktop even when Jourknow is hidden or minimized (Figure 4.11). Notes may be promoted to or demoted from the dashboard by clicking on the Dashboard icon  in the note. By dragging a note out of the dashboard, the user may position it in an arbitrary location on the desktop. A pin icon  may be used to anchor the note on top of all visible windows. We designed the dashboard and note pinning facility in support of visibility – notes may be deliberately placed in the way of future activity to serve as reminders.

The importance indicator is another method of making notes highly visible. The importance icon  toggles note importance, which highlights the note contents in a bright red (Figure 4.12). The note may be returned to normal color by toggling note importance off.

It was common for participants in our ethnography to set alarms on arbitrary items such as e-mails, thus signaling a to-do or reminder – we enable this pattern by including a note alarm mechanism. By clicking the alarm icon , users may set an alarm for a later date and time – when the alarm goes off, users are prompted with a dialog box containing the contents of the note and the note turns red as described above.

4.2 Implementation

The key technical challenges were in supporting the following functionality:

- Unconstrained textual input,

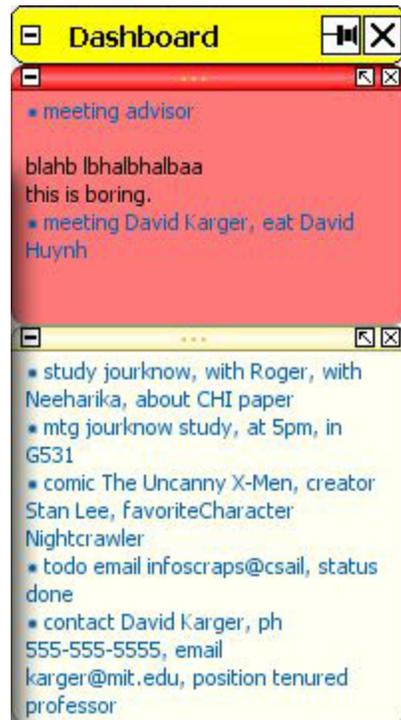


Figure 4.11. The note dashboard would appear on the user's desktop even when the Jourknow client was minimized.

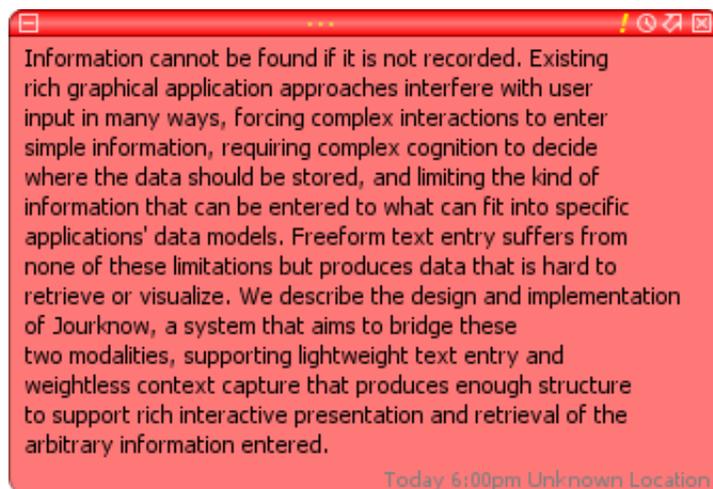


Figure 4.12. A note that has been marked as important turns red to promote visibility.

4. JOURKNOW

- Flexibility in how information can be structured without requiring people to predefine (or adhere to predefined) ontologies,
- Interfacing with desktop applications, specifically, alignment of subtext with applications' data ontologies,
- Extraction of subtext from unconstrained text, particularly supporting incomplete grammatical input, partial phrases, and informal language,
- Capture of context, and subsequent selection and presentation of relevant contextual events for supporting effective re-finding and memory priming,
- Maintaining appropriate correspondences among the user's text, extracted subtext, and captured context,
- Synchronizing information across desktop and mobile clients,
- Obtaining interactive speeds while feeding a GUI from an RDF database.

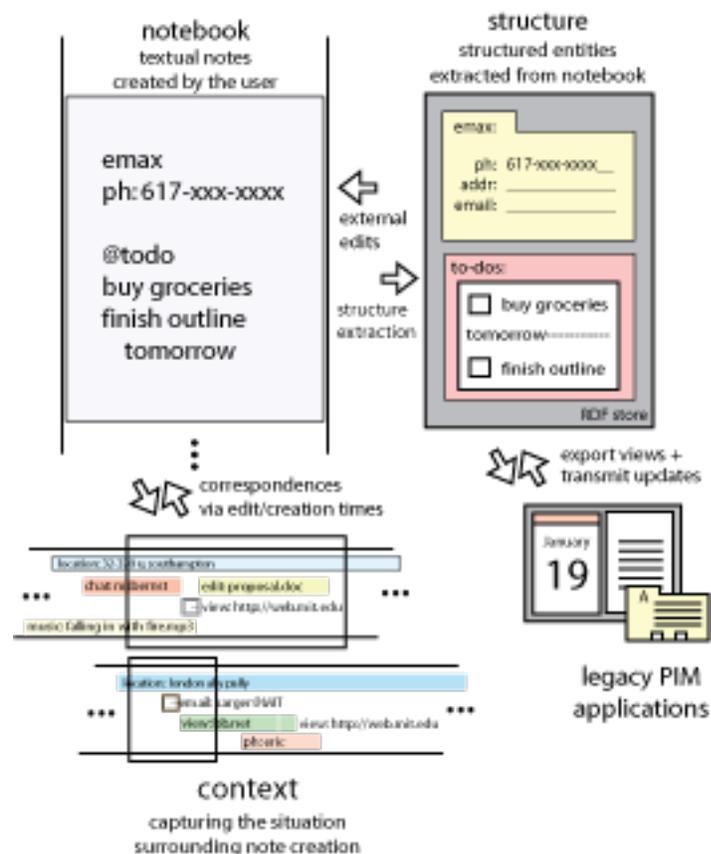


Figure 4.13. The Jourknow architecture. One knowledge base (KB) holds the contents of notes as well as all structure, and second KB holds context information. The program makes calls to external APIs to export items to legacy PIM applications.

In this section, I describe Jourknow’s current solutions to these challenges; Figure 4.13 illustrates the general architecture of Jourknow.

4.2.1 Data Model: Three Representations

4.2.1.1 *Text*

Jourknow maintains two different knowledge bases (KBs) to hold the structures that become the notebook (text and subtext, or the “meaning behind the text”: structure embedded in the text via Pidgin), and the context. Jourknow employs an expressive approach to storage: each character in every note is stored as a unique object in the text KB. (‘Character’ is liberally interpreted to include all entries in the note, such as audio and picture elements.) These characters have creation timestamps associated with them, as well as a link to any subtext object associated with the character. Maintaining a creation timestamp for every character enables Jourknow to identify exactly when each character was created, edited or automatically generated in response to an external edit to underlying subtext. Being able to identify the time of creation efficiently for each character in the codex is critical to Jourknow’s functionality, because it is the key by which Jourknow establishes a correspondence among text, subtext, and the context chronology. This creation time is kept with the character for the entire duration of its existence, and follows each character as it moves due to edits to surrounding text. New characters assume the current time, and characters that are cut or copied from one location in Jourknow and pasted elsewhere gain additional timestamps corresponding to each paste. This makes it possible to “manually re-plant” context associated with a note simply by moving appropriate text between notes.

4.2.1.2 *Subtext and application integration*

Subtext entities are represented as graphs in RDF [21]. Subtext that originates from pattern extractors such as the Pidgin parser or the Notation3 processor are grounded in the PLUM and Jourknow ontologies which are mapped to standard RDF ontologies, such as iCalendar [52] for events and vCard [76] for contacts. Use of these standard ontologies makes it possible to use existing tools [94] to help with ontology alignment to schemas used by external sources, simplifying the process of importing subtext such as events from atom/RSS feeds, contacts from the user’s LDAP server or IMAP e-mail account, and bookmarks from the user’s web browser. Importing subtext items can assist the name ambiguity problems mentioned earlier by providing additional information with which to identify people, places and things mentioned in the codex.

Similarly, when manifestations of the user’s subtext (“shadows”) are exported to the user’s applications, ontologies must first be aligned with the target application’s schema. Unfortunately, establishing a good mapping from instances grounded in the rich, expressive ontologies of the semantic web, to rigid schemas of PIM applications often requires somewhat arbitrary decisions regarding determining which fields best align (e.g., should “about” correspond to “comment” or “description?”). As more applications adopt flexible data representations in the future, we hope better mappings will become possible.

In order to effectively maintain the illusion of a single unified data model across the user’s PIM applications, Jourknow stores explicit bidirectional pointers between each subtext item and all its exported shadows. Some external APIs such as GData [8] already generate unique IDs for identifying elements; in this case, these IDs are used as-is; in other cases, Jourknow generates an identifier and stores this both with the subtext item and in a miscellaneous field of the shadow. Once this correspondence is established, maintaining synchronization is simple; Jourknow periodically polls applications using its data transfer API, and examines all items that are tagged with a subtext identifier. For each such item, it casts it back to the Jourknow representation, and compares field values; if values have changed, this indicates that the user has edited the subtext externally. Likewise, when the user updates the subtext by editing the note, Jourknow first determines whether the subtext already has exported shadows; and if so, updates these shadows with new values. Jourknow currently exports shadows to Google Calendar via the GData API, and Apple’s iCal and Address Book via Applescript.

4.2.1.3 Context

The context KB consists of a chronology of observations made of the user’s desktop state and actions, and of their situation/environment, which is created and maintained by PLUM [132]. PLUM executes a sequence of observer knowledge sources at a regular frequency (usually 2-3Hz) that call various facilities in the underlying operating system to yield observations. Examples of activities currently observed by PLUM knowledge sources include the identity of the application that has focus at each moment, the names and locations of any documents or web pages being viewed or edited, the user’s activity in chats, writing emails, or music listening, as well as periodic desktop screen captures and the user’s activity/idle state. Examples of environment/situational state captured by observers include the user’s location (as perceived through Placelab [93]) and web cam snapshots of the user. Observations each have an associated “validity” time interval, which represents the largest contiguous time interval

for which, according to the observer, the observed phenomena remained unchanged. All observations made by knowledge sources are encoded as RDF graph structures (for representational flexibility), grounded in the PLUM ontology.

4.2.2 Episodes and saliency heuristics

Jourknow uses the concept of an *episode* to segment time into discrete units for context association. By default, Jourknow assumes an episode length of an hour, so a note written at 1:44pm would be associated with context from 1:00pm–1:59pm. In order to find the set of episodes associated with particular text in the codex, Jourknow finds all the episodes whose intervals intersect the creation times of the text. Once associations are made between text and episodes, Jourknow extracts observations relevant to each episode by finding all observations that overlap with each relevant episode’s interval.

However, there may be a great number of observations; in order to prevent inundating the user with a record of their activities, we have designed saliency heuristics to select observations that are likely to be memorable and relevant to the user. These heuristics include, for each type of context observation, most recently observed, most frequently observed, and longest total duration. We additionally defined an “outlier” saliency heuristic inspired by TF-IDF [119] which weights context proportionally to its total observed duration during a particular episode, and inversely proportionally to its total frequency observed across episodes. This latter heuristic has proven useful in filtering out routine visits to commonly revisited web sites, as well as intermediary pages that consumed little face time.

4.2.3 Structure extraction from text

To support the various modes of input described in the Interaction section, Jourknow provides three types of textual pattern analysis: simple syntactic forms (i.e., regular expressions), recursive-descent parsing, and Notation3 interpretation. As the recognition process can be computationally intensive (particularly for the recursive-descent parsing), Jourknow runs recognizers only on regions containing changed text, and executes recognizers asynchronously and independently on their own threads.

The syntactic recognizer uses a standard regular expression engine to find syntactically structured elements in the text, including tags, file paths, and URLs. The Pidgin parser for common types such as to-dos and calendar events can accept any context-free language. Jourknow features

Pidgin grammars designed to handle the most common found types of PIM data: events, contacts, and to-dos. NLTK_lite's `rdparser` [15] is used to implement these grammars — `rdparser` is a simple deterministic top-down parser for context-free grammars. Our modifications involved allowing the inclusion of regular expression as terminals, which we defined to match if and only if the regular expression matched the entire token. This modification greatly simplified the recognition of tokens by syntactic form (such as dates) without a separate lexical analysis phase, and enabled us to add “wildcards” to the grammars, to stand in for words or combinations of words that could not be known a priori. This occurred frequently in our Pidgin grammars, such as with names of people, locations, or the topics of a meeting.

While enabling wildcard terminals greatly enhanced the expressiveness of our grammar language, it also dramatically increased parse ambiguity. Modifying the grammar to not require phrase headwords (usually prepositions such as “at” or “with”) made the number of ambiguous parses combinatorially larger, but also dramatically improved usability of the grammars, as we observed in CHAPTER 3 that prepositions were often omitted in people's information scraps. Jourknow tackled the complexity of parse ambiguity resolution in three ways. First, interleaving wildcard token matching into parsing made it possible to include ambiguous or incomplete interpretations. Second, because the recursive descent automatically returned all possible parses for a particular sentence under the grammar, it reduced the problem of ambiguity resolution to choosing the correct parse tree from the returned set of possible parses. Finally, a simple heuristic worked well in most cases: to choose the parse tree that was broadest at its base. This corresponded to the parse tree that recognized the greatest number of separate clauses, and attached them closest to the root. This heuristic eliminated the most common source of incorrect parses, the consumption of clauses by wildcard terminals, as evidenced by errors such as interpreting the meeting Pidgin expression “meeting with Michael at Stata” as a meeting with a person named “Michael at Stata” rather than a meeting with a person named “Michael” at a location named “Stata”. Jourknow allows the user to override the heuristic's choice by browsing through potential parse trees.

The TurtleDove interpreter is implemented as a simple finite state machine. This machine transitions between four states: type, name, predicate and value. Transitions are governed by spaces and commas as described in the grammar design. TurtleDove must be implemented with a think-ahead technique to analyze the state the user will enter upon typing the next character in order to display the autocomplete drop-down at the correct moment.

4.2.3.1 *Associating text with subtext*

When a pattern extractor first identifies a previously unseen structure in the text, it generates a new subtext entity to represent the item. To allow future edits of the original text to properly update the correct subtext entities, it is necessary for Jourknow to be able to uniquely identify a text's corresponding subtext. To effectively and reliably support this lookup, Jourknow creates a pointer from each character in the subtext to the subtext object. When the source text changes, the corresponding subtext may be updated or removed to reflect the change.

4.2.4 **Jourmini: Simplified Data Model**

Jourmini is implemented on Symbian Series 3 Linux Nokia phones using Python and PyGTK. The data model is also RDF, stored using a Python implementation of Wilbur [97]. The phone's built-in camera and microphone are used to record picture and speech notes.

Due to limited memory constraints of phone technology, Jourmini uses a simplified data model. Specifically, rather than storing each character as an individual RDF object with a timeprint, Jourmini maintains a string representation of the entire note and a series of edit timeprints for the entire note. This is a lossy transformation; we no longer know when each character was created.

4.2.5 **Synchronization**

An arbitrary number of Jourknow and Jourmini clients may try to update a single collection of notes – creating a classic problem of offline-online synchronization. Jourknow relies on a synchronization server to manages updates and commits of note changes. When the user chooses to synchronize, a two-step process begins: update (to bring any other note changes to our local client and reconcile conflicts) and commit (to push changes to the central server).

Versioning is accomplished using timestamps: specifically, the most recent edit time of any character in the note serves as a de facto version number. Comparing timestamps between the server and the client will usually uncover which is the most recent. However, this approach fails if the client edits an out-of-date note: for example, if the client receives the note from the server and edits it, but another client pushes changes to the server before the first client can push its updates. The Jourknow server detects this situation by inspecting a special lastSync timestamp associated with each note, marking the date and time of the last synchronization with

the server. If the times do not match between client and server, we have a conflict, much like in commercial version control systems like CVS or Subversion. Our solution in conflict scenarios, since there is no way to know which note is the “right” one, is to defer to the user: the two versions are pasted together into a new note, so no information is lost.

The lossy data structures in Jourmini make synchronization a bigger problem. How does the system go about assigning timestamps to characters, given only a list of timestamps at the note granularity? Missing timestamps might lead to context being un-associated with a note. This problem is not generally solvable, so the synchronization server applies a heuristic: randomly assign each character in the note a timestamp from the set of known edit times. More formally, we sample without replacement from the note set of timestamps. This technique gives us as strong an assurance as possible without adding placeholder characters for missed timestamps. The heuristic works in practice because the typical note has many more characters than it does edit times (Jourmini collapses edit times to larger, minutelong intervals), and the timestamp information is only used at high levels of abstraction to generate context thumbnails. However, the note is left with nonsensical and incorrect timing data, and if Jourknow used timing information at higher granularity than hourly episodes this solution would need to be revised.

4.2.6 Caching to Obtain Interactive Speeds

Jourknow’s persistent data model is held in an RDF [21] database using the Jena framework [14]. Semantic Web relational databases (triple stores) are extremely difficult to optimize, and as a result it is difficult to extract interactive speeds from disk-based triple stores.

Our solution has been to maintain an in-memory copy of the database called the cache model to hold the text KB and the context KB. In practice the cache model performs writes and reads at interactive speeds, and is small enough to fit in RAM. The physical disk database (the base model) is overwritten with the contents of cache model upon save.

Many operations require further caching by using Java in-memory data structures. For instance, the code commonly needs to query the string representation of a note’s contents; rather than reconstruct the buffer by traversing a list of character objects in RDF, Jourknow caches this string in memory. Context summaries require more complex caching – asking PLUM to return a list of all observations during a given hour is a nontrivial operation, so these observations are likewise cached once completed.

This caching architecture will not scale; Jourknow already uses upwards of 100MB of memory with fewer than 50 notes. Caching is a convenient interim solution because the RDF representation conveniently overlaps with many of the data model needs of our system. To make a fully interactive system, however, a more complex in-memory data model would be required.

4.2.7 Object-Oriented RDF Programming

Programmer efficiency is a major issue when attempting to program using an RDF data model. Programming using an RDF data model has many dissimilarities to object-oriented programming paradigms. Most strikingly, there is no function abstraction: all changes must be at the statement level, because RDF is conceived as a series of <subject, predicate, object> statements. To create a new blank note and insert a sentence requires a carefully-crafted series of statements, paraphrased as:

- There is an object with the following URI: <http://projects.csail.mit.edu/#plum/a8bd6b78s...>
- The object is of type Note
- The note has a list of characters
- That list of characters is an object
- That list of characters is of type Sequence
- The first entry in the list of characters is an object
- The first entry in the list of characters is of type Character
- The first entry in the list of characters has a contents property with a value of the following URI: <http://projects.csail.mit.edu/#plum/quc876fd8...>
- The contents object is an object
- The contents object is of type xsd:string
- The contents object is the string “a”
- The second entry in the list of characters is an object
- ...

One option is simply to write the application’s model code directly in terms of these statements. This is the most efficient implementation for the machine, as the programmer can manipulate only the necessary parts of the data model, but it is unfamiliar and difficult to inspect or debug.

To address this issue, I have created Java object wrappers around RDF types in Jourknow. The wrapper classes serve as sets of functions around objects – each object corresponding to a particular URI. Getter methods make the appropriate calls into the RDF data model to retrieve data, for example asking for all statements with the given URI as subject and a predicate called character, and parsing the result as a Java char.

These object wrappers enable traditional OOP design patterns. For example, wrapper classes may be inherited: entries in notes can be characters, images, or audio files, sharing common manipulation code. Wrapper classes also are straightforward places for any caching to occur, and simplify both parameter passing and model-view separation. (These objects are a simple means of access control to the data model – only a note object can inspect its own contents.)

4.2.8 Saving and Transactions

Early users of Jourknow complained about saving – a lightweight application should not require explicit saving, and instead automatically back up data behind the scenes. Given the verbosity of Jourknow’s data representation, explicit pushes of the cached data model to the database could take minutes. Such a wait is unacceptable, especially when the application needs to block user input during that period to avoid writing incomplete data to disk.

Jourknow’s solution is to use transactional saving in a background thread. The implementation of this technique is fairly straightforward, as Jourknow’s persistent memory is kept in a RDF database. To save, a background thread begins a transaction, removes stale statements from the physical memory database, copies fresh ones from the cached version, and ends the transaction. If the application is closed or crashes during save, database ACID transactional semantics guarantee a consistent database.

4.3 User Study

The Jourknow study consisted of 14 participants from MIT (ages 18-41, median 26), external to our research group. 7 were students at the business school, 1 was visiting Computer Science faculty at our university, 2 were undergraduates and 3 were graduate students in computer science. There were 10 men and 4 women. The group was randomly divided into seven participants who received just the desktop version of Jourknow, and seven participants who received both the desktop and the mobile version of Jourknow (MiniJour). This division enabled us to perform a between-groups investigation on the mobile client, examining its effect on take-up of the tool. Both groups had the context capture and TurtleDove elements of Jourknow enabled; however, common PIM type recognition and application push was disabled.

Following standard practice (e.g. [60, 117]), Jourknow was installed on participants' computers, participants were instructed in the use of the interface. We described several of the shortcomings of the current version of the research prototype — slow loading and saving, occasional GUI bugs, and a remaining server bug that was patched near the beginning of the study. Participants were instructed to introduce Jourknow into their everyday note-taking practices, and to make extra effort to use the software to capture their thoughts and notes. They then used the Jourknow client for a period averaging eight days, including one weekend. Throughout the study, I used e-mail announcements to promote use of the tool, remind participants to integrate the tool into their lives, and keep in constant contact. This level of contact was fell short of other studies which made regular visits to participants (e.g., [31]), but was more direct than those with no reported communication during the study (e.g., [124]).

4.3.1 Study Results

4.3.1.1 General Feedback and Anemic Usage

Mid-study warning signs. Our team began to receive indications midway through the study that participants were not making regular use of Jourknow. On the 6th day of the rollout, we observed that only four of the seven participants with mobile phones had tried synchronizing their notes on the server. In response to an e-mail suggesting everyone synchronize, two participants e-mailed us admitting that they had not yet opened the tool, with a third participant experiencing trouble starting the tool on his computer. We helped the participant debug the problem, then sent an e-mail reminding all participants that we had asked them (as per the study agreement) to make daily use of the tool.

Usage analysis. At the conclusion of the study, three participants (P2, P3, P11) had never launched the client after their initial installs (Figure 4.14). A fourth participant only used the client once right before his exit interview (P1). Others' usage varied significantly. As can be seen in Figure 4.15, most participants created notes on the day that they received the client, and note creation tailed off sharply in time. Usage picked up with the release of a major software patch. Another short jump occurred on the 12th, most likely in response to an e-mail that reminding participants that the study was half over and that we expected them to “continue using the client.”

4. JOURKNOW

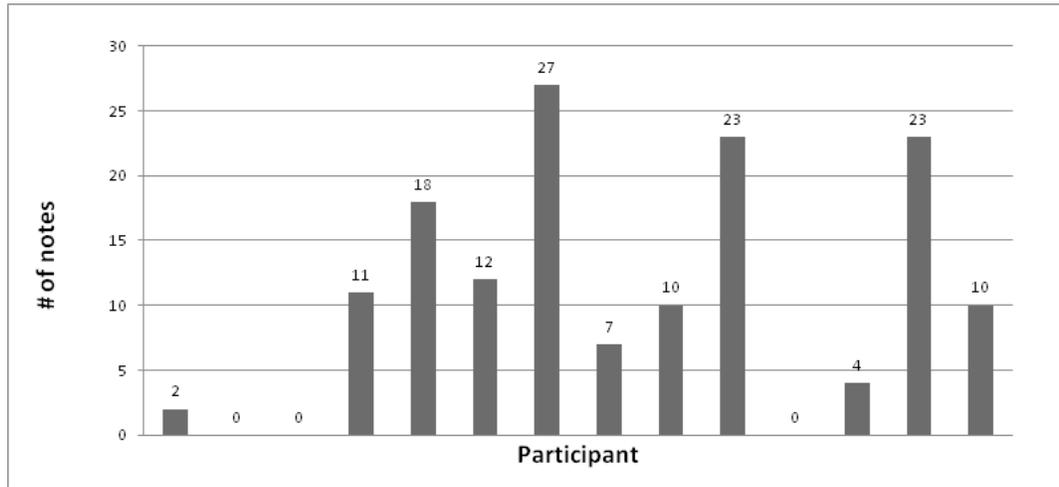


Figure 4.14. Number of notes captured by each participant during our weeklong trial. Several participants barely used the prototype at all, and barely over half captured more than one note per day.

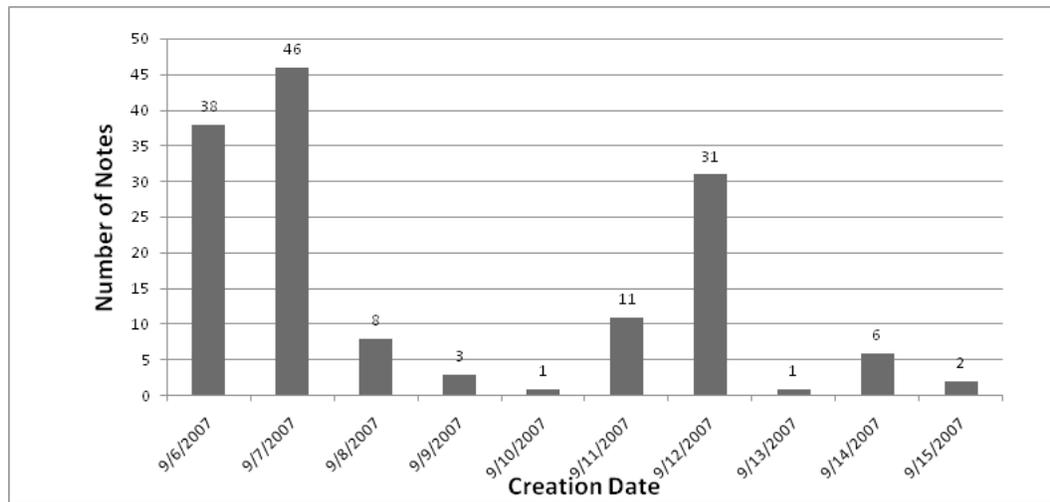


Figure 4.15. Number of notes captured each day during our study. The first two days were the installation and tutorial days for all participants. After the installation dates there is a precipitous drop. The spike in the middle is in response to a reminder e-mail from the researchers.

Closing interview feedback. The closing interview indicated that many participants were generally unsupportive of the prototype or exhibited mixed feelings. Feedback was often focused on non-research elements of the design such as load times and bugs. It was clear that the prototype needed additional design iterations in several respects; I return to this conversation in CHAPTER 6.

4.3.1.2 *Specific Feature Feedback*

Here I report the feedback we received on the three research aspects of Jourknow: context, TurtleDove, and JourMini.

Context-based Re-finding. We found that our participants made very little use of the contextual features, rarely viewing the context associated with a note or navigating via the associated context to re-find a note. Participants pointed out that much of the contextual information was incomprehensible, for instance wireless router ssid's – and since Jourknow was logging a great amount of context, such information overwhelmed the facet panel. Most participants generated a very small number of notes by the end of the experiment, and could thus linearly scan the list more quickly than might locate the correct facet values.

TurtleDove Language. Our participants generally found the comma-delimited syntax of TurtleDove too complicated to use. Engineers and programmers understood the format, but others found the comma-delimited predicate/value approach abstruse and gave up after a small number of attempts. We found few instances of uncommon PIM type capture beyond shopping and to-do lists – both examples given during the training period. Participants tended to forget syntax quickly after the training session, likely contributing to our null result. Several participants requested that we re-introduce the original Pidgin support for common PIM types such as meetings and calendar events, and pushing the information into applications such as the calendar.

Mobile Client. Mobility was our most successful venture – reactions to Jourmini were generally positive. Notes taken on the mobile client included many text scraps, as well as several cameraphone pictures. Participants using phones with full QWERTY keyboards were in general more positive about the mobile experience than those with traditional 12-key number pads, due to the ease of typing. Furthermore, participants who lived primarily digital lives and used little or no paper also found the system more useful than those who used paper regularly for such notes, as they already had a functioning mobile solution in scraps of paper.

4.4 Discussion

I believe that the somewhat anemic take-up of Jourknow reflects on the design and evaluation process for the artifact. This is a complex topic, and one I postpone for full discussion in CHAPTER 6.

The Pidgin languages were originally designed to serve as lightweight entry of two data classes: common PIM types (using simple language), and uncommon data types (using more complicated syntax). TurtleDove disabled the former by removing the functionality to push Pidgin expressions into common applications — due mainly to a lack of engineering time for integrating with the wide variety of tools our participants used. Our hypothesis was that the opportunity to record uncommon data types would provide enough of a benefit to users. The hypothesis was incorrect: participants indicated that these common PIM types would be at least as important as the uncommon item capture already enabled.

Testing with a population heavy on business students underscored the importance of integration with users' critical work pathways. Several of these participants had extremely well-defined information handling routines that Jourknow could not penetrate without integrating into specific tools. Whether this result is an issue of adoption for evaluation purposes (as Kelley and Teevan suggest [89]) or a more general design critique is an open question.

Data scale is of importance in studies of systems like Jourknow. The context information in particular proved unbeneficial to our participants over such a short period of time, due to both the small number of notes they accumulated and our participants' still-intact memory of notes' contents. To better stress-test the context features of Jourknow, our participants must gather notes over a long enough period of time that these mechanisms may become useful, or instead seed the application with existing notes.

4.5 Conclusion

In this chapter I have presented Jourknow, an information scrap management client for the desktop and mobile phone. Jourknow explores several dimensions of the design space, including lightweight capture, flexible organization and re-finding, context as a memory cue, and multimodal input. Through a weeklong evaluation of the tool on a population on predominantly nontechnical users, we were able to identify some of the

tool's most salient design and implementation weaknesses moving forward. I continue the discussion of the user study and design process in CHAPTER 6.

5. PINKY: PERSONAL INFORMATION KEYWORDS

If Jourknow is the Swiss Army Knife of information scrap management, attempting a variety of designs for capture, organization and re-finding, then Pinky is a surgeon's scalpel: it attacks a subset of the problem much more directly. Pinky (**P**ersonal **I**nformation **K**eywords) is a quick capture mechanism for common personal information types such as to-dos and calendar events. Its design goal is to prevent information scraps in the first place by reducing the time and energy associated with capturing structured information. This chapter describes Pinky, its design and innovations, and ongoing work.

5.1 Motivation

Some information scraps exist because the user has no specific tool for managing them (e.g. guitar tabs or how-to guides). But other scraps are precisely the sort of data that PIM tools are designed to store. Our study of the scraps on knowledge workers' computers and physical desktops in CHAPTER 3 found that over 25% of scraps were to-do items or contact information, the two largest categories overall.

This result begs the question, what goes wrong? Why doesn't this information make it into a PIM tool? The cost of starting, navigating, and

Work presented in this chapter is a collaboration with Max Van Kleek, Vikki Chou, Rob Miller, David Karger and mc schraefel. It is under review.

entering data in PIM applications is one reason why users turn to scraps, despite the difficulties of organization and refinding that information scraps will pose later [14]. Participants in our study in CHAPTER 3 reported the need for quick capture as a common reason for creating scraps. “If it takes three clicks to get it down, then it’s easier to email [a scrap to myself],” reported one participant. Another said, “Starting in Outlook forces me to make a type assignment, assign a category, set a deadline, and more; that takes too much work!”

Starner et al. [125] found a similar effect for users of mobile PDAs and paper day planners. When prompted to schedule an appointment, almost half of PDA users and over half of day planner users wrote down information scraps (generally on bits of paper) rather than open up and navigate their calendars.

The Jourknow system in CHAPTER 4 was designed to capture and manage information scraps, to catch these bits falling between the cracks of current tools and give them life. In particular, we presented Pidgin languages for capturing structured information, potentially pushing it into appropriate application. One finding from our studies of Jourknow, however, is that users are loath to abandon their current PIM tools, necessitating automatic synchronization between Jourknow and the universe of other PIM tools – a substantial engineering undertaking.

In this chapter I focus on the structured information capture process. Pinky, a close relative to Jourknow’s Pidgin languages, is a prototype capture tool for these common personal information types. Rather than being another PIM tool to manage, Pinky exists only to capture information into other tools.

5.2 A Command Line for PIM

Pinky offers a possible solution to the lightweight capture problem. Pinky is a hotkey-invoked popup command line extending the Inky web command system [51] that allows the user to enter information quickly as text, then pushes the information into the appropriate PIM tool (Figure 5.1). The text is parsed using keyword matching to extract PIM data (like todo items, calendar events, or contact information), which is then filed immediately in the appropriate PIM tool. As a web browser plug-in, Pinky currently automates web-based PIM tools including Google Calendar and Remember the Milk to accomplish this filing. However, this technique could be extended to many desktop applications as well.

The rest of this chapter describes Pinky's design, research contributions and implementation. Some of the new ideas embodied in Pinky include: (1) using GUI widgets for choosing and changing arguments on the command line; (2) displaying relevant clippings from the back-end web site while the user is entering a command; and (3) reorganizing the display of alternative interpretations to make them easier to scan and select.

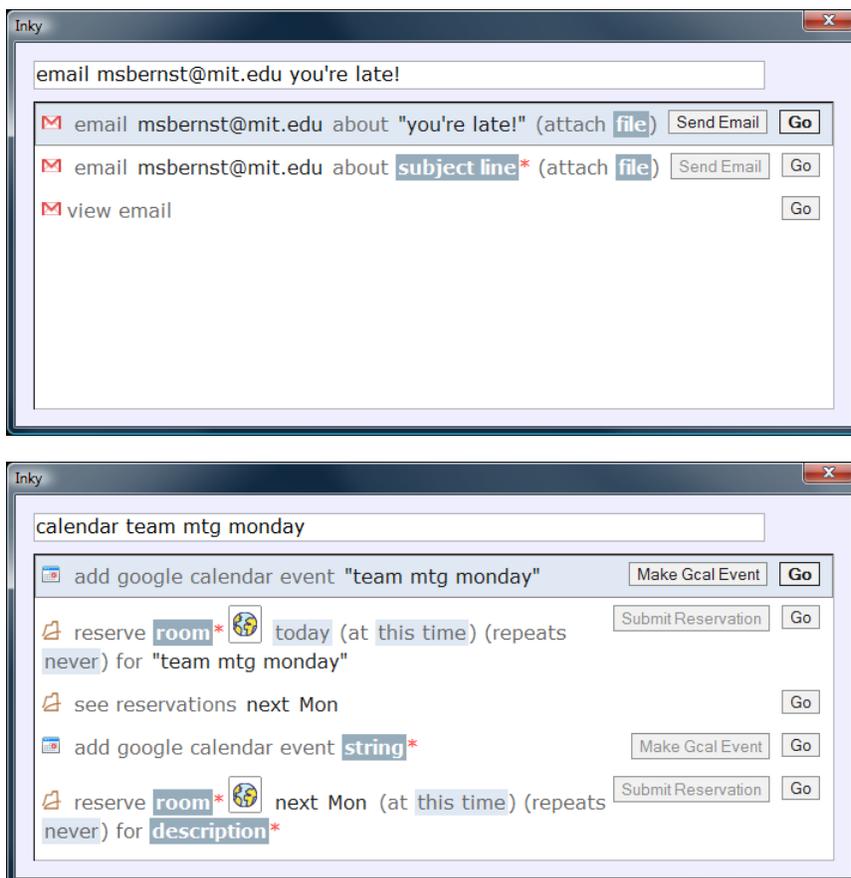


Figure 5.1. The Pinky command line for personal information. The text box at the top of each window has been typed by the user, and the list below the break represent Inky's interpretations of the command. The user selects a parse, and Inky will execute the command.

5.3 Design of a GUI Keyword Command Line

Pinky builds on techniques explored in the Inky internet keyword command system [51] — the interaction techniques in this section are derived from Inky. Pinky-specific innovations are detailed in §5.4.

Pressing Control-Spacebar in the web browser pops up the Inky window (Figure 5.1). This keyboard shortcut was chosen because it is generally under the user’s fingers, and because it is similar to the Quicksilver shortcut (Command-Space on the Mac).

The Inky window has two areas: a text field for the user to type a command, and a feedback area that displays the interpretations of that command. The Pinky window can be dismissed without invoking the command by pressing Escape or clicking elsewhere in the browser.

5.3.1 Commands

A command consists of keywords matching a web site function, along with keywords describing its parameters. For example, in the command `todo call mom 3pm`, the `todo` keyword indicates that the user wants to capture a to-do, and `call mom` and `3pm` are arguments to that function.

To reduce the burden of learning and remembering syntax, Inky insensitive to keyword ordering and synonyms. For example, `todo call mom 3pm` and `3pm call mom todo` will produce the same interpretation. Keywords that represent arguments to a function can be reordered and interspersed with keywords matching the function to be called. Commands can use synonyms for both function keywords and arguments. For example, to set up a meeting with david at 4pm in D463, the user could have typed `calendar` or `mtg` instead of `meeting`, used a full room number like `32-D463` or a nickname like `Star Room`, and used various ways to specify the time, such as `15:00` and `3:00`.

Function keywords may also be omitted entirely. Even without function keywords, the arguments alone may be sufficient to identify the correct function. For example, `D463 15:00` is a strong match for the room reservation function because few other PIM functions take both a room location and a time as arguments.

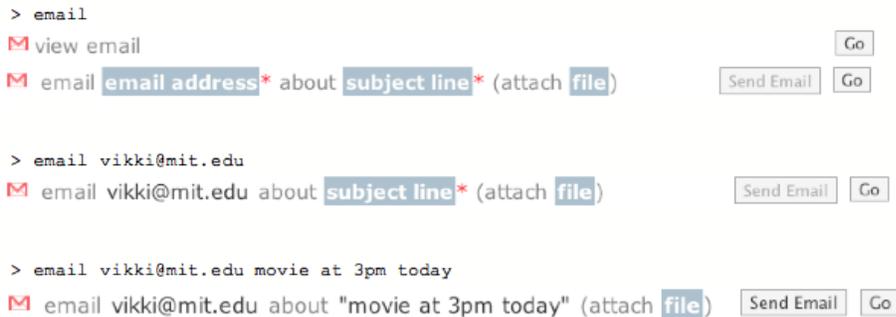


Figure 5.7. Feedback is offered continuously as the command is entered.



Figure 5.8. Feedback showing different kinds of argument feedback, including required (*room*), default values (*this month*), and rarely-used arguments (*repeats never*).

5.3.2 Feedback

As the user types a command, Inky continuously displays a ranked list of up to five possible interpretations of the command (Figure 5.7). Each interpretation is displayed as a concise, textual sentence, showing the function’s name, the arguments the user has already provided, and arguments that are left to be filled in. The interpretations are updated as the user types in order to give continuous feedback.

The visual cues of the interpretation were designed to make it easier to scan. A small icon indicates the website that the function automates, using the favicon image displayed in the browser address bar when that site is visited. Arguments already provided in the command are rendered in black text. These arguments are usually exact copies of what the user typed, but may also be a standardized version of the user’s entry in order to clarify how the system interpreted it. For example, when the user enters `mtg star room`, the interpretation displays “meeting D463” instead to show that the system translated `star room` into a room number.

Arguments that remain to be specified appear as `white text in a dark box`. Missing arguments are named by a word or short phrase that describes both the type and role of the missing argument. If a missing argument has a default value, a description of the default value is displayed, and the box is `less saturated`. In Figure 5.8, `name`,



Figure 5.9. Commands can be run in either view mode (*Go*) or submit mode (e.g., *Send Email*).

`e-mail office, etc.` is a missing argument with no default, while `this month` is an argument that defaults to the current month.

For functions with persistent side effects, as opposed to merely retrieving information, Inky makes a distinction between arguments that are required to invoke the side effect and those that are not. Missing required arguments are marked with a red asterisk*, following the convention used in many web site forms. In Figure 5.8, the room and description are required arguments. Required arguments exist for functions with side effects because those side effects cannot be automated without all necessary arguments. Note that the user can run partial commands, even if required arguments are omitted. The required arguments are only needed for running a command in the mode that invokes the side effect immediately.

The feedback also distinguishes optional or rarely-used arguments by surrounding them by parentheses, like the (repeats never) argument in Figure 5.8. It should be noted that this feedback does not dictate syntax. The user does not need to type parentheses around these arguments. If the user did type them, however, the command could still be interpreted, and the parentheses would simply be ignored.

5.3.3 Running a Command

Pressing Enter on a command runs the top-ranked interpretation by default. The arrow keys or the mouse can be used to select a different interpretation from the list, or the user can click the Go button next to the desired interpretation. When a command is run, the Inky window disappears, and Inky directs the browser to visit the target web site and fill in the form automatically.

Commands without side effects can be run with as few arguments as the user chooses to give. If the function is missing arguments, Inky relies on the fact that the website will either fill in appropriate defaults or prompt the user for required arguments when Inky tries to submit the form. By delegating these tasks to the website, Inky is able to use defaults

that are stored by the web site. For example, AccuWeather.com uses an HTTP cookie to remember the last city used for looking up a weather forecast. By letting AccuWeather handle the default, Inky users can look up the weather in their usual area just by running the command `weather`.

A website's prompt may also include useful UI feedback and constraints that are not available in our textual prototype. For example, the command `travelocity SFO LAX` would start searching Travelocity for flights from San Francisco to LA, but Travelocity would prompt for departure and return dates with a custom calendar widget.

Functions with side effects can also be run in submit mode. When a command is run in submit mode, Inky takes the final step of causing the side-effect to occur. Submit mode is selected by pressing Control-Enter, or by clicking the submit button in the desired interpretation. The submit button is labeled with the effect that it has, such as "Make Event," "Reserve Room," or "Send Email" (Figure 5.9). This button is disabled until all required arguments are provided to Inky, since Inky is taking responsibility for running the command. For example, typing `email vikki@mit.edu remember to buy milk today` and pressing Control-Enter will immediately send an email, with no further interaction. The command is run by automating the web site, however, so any confirmation pages or opportunities to cancel or undo would be visible.

Separating commands with side effects from those without side-effects helps discourage Inky users from making errors that would be difficult to reverse. Since the default run methods always execute the com-

The screenshot shows a command prompt with the text "meeting with max and rob 5:00 tomorrow". Below it is a multi-tabbed input area. The tabs on the left are "calendar", "todo", "CSAIL", "jogging", "e-mail", and "note". The "calendar" tab is active, showing a search input with "max and rob" and a list of results: "max and" (highlighted) and "max". Below the search input are three input fields: "date" with a calendar icon, "time" with a clock icon, and "location" with a globe icon. The "date" field contains "4/3/2008". The "time" field has a dropdown menu with "5:00am" and "5:00pm" (highlighted). The "location" field has a dropdown menu with "G531" and "D507" (highlighted).

Figure 5.3. Pinky has redesigned the command interface to speed disambiguation between multiple parses. Most notably, it splits top-level functions into tabs on the left, and enables disambiguation by argument via drop-down.

mand in a view mode, it is more difficult for a user to unknowingly cause a persistent side effect. However, by making it possible for users to run in a submit mode, Inky increases the efficiency of users who trust the system and want to commit to the side effect.

5.4 Personal Information Keyword Commands

In this section I detail the improvements and adaptations I have performed to redesign Inky for personal information capture. This new version of the software is called Pinky.

5.4.1 Organizing Multiple Interpretations

One observation from the Inky user study was that the list of alternative interpretations was rarely used. Several study participants expressed the concern that the alternatives on the list often looked very similar, which made them hard to compare. For example, the top few choices may all be the same function, differing only in how the user's keywords are assigned to arguments. As a result, it often felt easier to change the command until the top suggestion was right, rather than visually scan the list of suggestions.

The Pinky prototype has a new feedback interface aimed at addressing this problem (Figure 5.3). The suggestion list is categorized by function, indicated by the tabs on the left, so that each function that matches the command appears only once. Within each command, the alternative parses for each argument are shown in a drop-down list under the argument, which the user can select. However, this higher granularity adds complexity to the user interface: the user can no longer simply page through a list of options.

This interface allows the user to select the right interpretation by focusing on just one component at a time: first the function name (by picking a tab), then each argument (by picking from drop-down lists). Each choice may cascade to other choices, since the keyword interpreter does not permit two arguments to use the same keyword. For example, when the user in Figure 5.4 indicates that G531 is the location of the calendar event, G531 can be removed from the list of guesses for the title of the meeting. The new interface also incorporates the GUI widgets mentioned earlier (Figure 5.4 includes the calendar widget and the location widget).

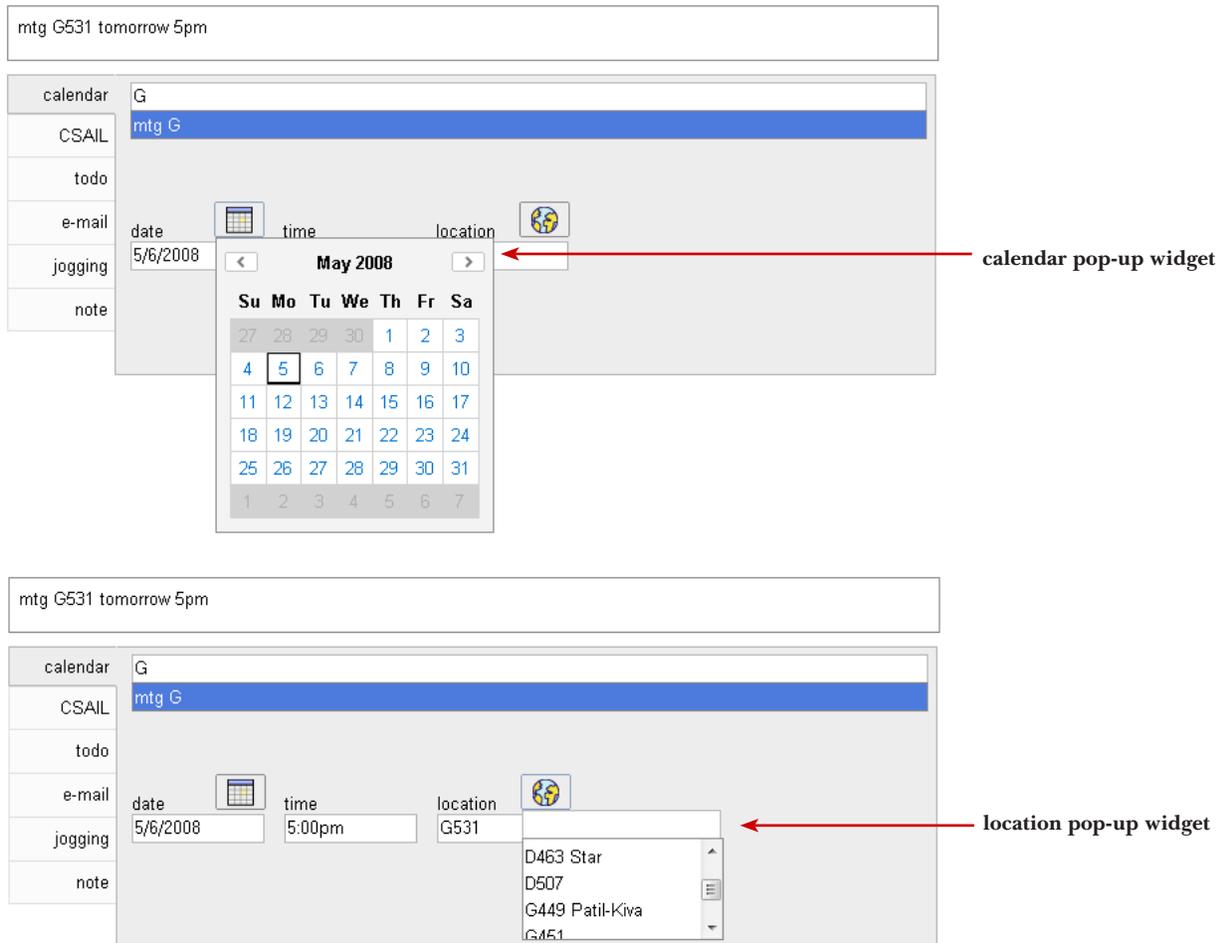


Figure 5.4. Pinky provides GUI widgets to aid completion for certain argument types, such as dates, locations, and people.

5.4.2 GUI Widgets for Command Arguments

Sometimes arguments may be easier or faster to select from a GUI widget, such as a calendar picker, than to type. GUI widgets also inherently offer additional feedback that can reduce errors. For example, a calendar widget makes it obvious that April 12 is a Saturday, so it may not be a good day to schedule a work meeting.

The Pinky prototype incorporates three kinds of GUI widgets into its interface: people, places, and dates. The people widget  pops up an auto-completing list of people's names and email addresses, drawn from the user's email contacts. The places widget  has a similar list of relevant places, which for our environment are the rooms in our building. The



Figure 5.5. Variables chosen explicitly from drop-down lists or widgets are visually indicated by the predicate they represent (left). When the user clicks the expand box , the entire expression is displayed (right).

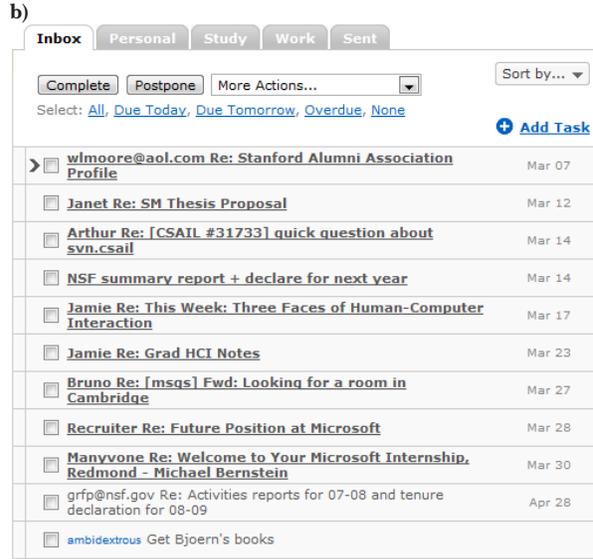
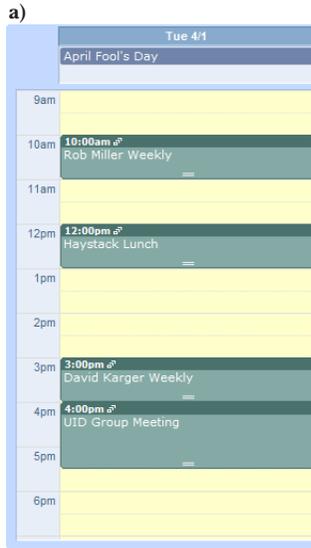
date widget  is a conventional calendar widget. These three widgets are implemented in HTML and Javascript using the Yahoo User Interface library.

GUI widget buttons are incorporated into Pinky's feedback window, so that argument slots of the appropriate type (people, places, and dates) are automatically followed by the relevant button. Clicking the button pops up the widget to fill in the missing argument (or change the value already assigned to it by command parsing).

When an argument's value is set with a GUI widget, the command in the textbox automatically reflects the change as well. The relevant text is annotated with the property name (Figure 5.5). This text represents a *mandated variable* and is prefixed by an argument name, as in `mtg 5pm with:emax at:G725`. This syntax forces the keyword interpreter to use those keywords only for the specified argument. To avoid changing the user's command too drastically, this extra syntax is normally hidden, and mandated variables are shown by underlining them as in `mtg 5pm emax G725` (Figure 5.5). Clicking on the mandated variable expands it into its full syntax. Expert users can also directly type the syntax for mandated variables, but this requires the user to learn and remember names of argument.

5.5 Web Clips

GUI widgets like the calendar widget provide generic support for entering arguments accurately, reminding the user for example that April 12 is actually a Saturday and that msbernst@mit.edu is the intended email address. For more personalized context, Pinky uses web clippings extracted from relevant web sites. These web clippings bring just-in-time information to the user, anticipating the user's information needs so that the user does not need to break off the entry to consult less efficient menu and form interfaces, or worse, choose not to record the information at all.



c)

	#	Date	From	Subject [Thread]	Size
	2032	09/22/2006	Rob Miller	Re: [uid] HCI Reading and Discussion Group	5 KB
	1934	09/20/2006	Rob Miller	Re: [uid] HCI Reading and Discussion Group	3 KB
	1870	09/19/2006	Rob Miller	[uid] [Fwd: [HCI Seminar] TALK:Wednesday 9-20-06 End-User Creation, C	9 KB
	1858	09/19/2006	Rob Miller	call for papers, WWW 2007	6 KB
	1818	09/19/2006	Rob Miller	Re: [uid] synergy - tool for sharing keyboard/mouse	4 KB
	1773	09/18/2006	Rob Miller	[uid] hacking session today, 4 pm	3 KB
	1736	09/18/2006	Rob Miller	[uid] papers for this week's meeting	3 KB
	1650	09/15/2006	Rob Miller	Re: Thanks	2 KB
	1189	09/07/2006	Rob Miller	Re: Dropping by	5 KB
	1141	09/07/2006	Rob Miller	Re: Dropping by	3 KB
	885	08/31/2006	Rob Miller	Re: Dropping by	2 KB

d)

Time:	Conf Room G531
07:00am	*
07:30am	*
08:00am	*
08:30am	W3C comm meeting
09:00am	
09:30am	
10:00am	
10:30am	
11:00am	
11:30am	
12:00pm	
12:30pm	
01:00pm	
01:30pm	
02:00pm	
02:30pm	DIG
03:00pm	
03:30pm	
04:00pm	userstudy
04:30pm	
05:00pm	
05:30pm	
06:00pm	*
06:30pm	*
07:00pm	*
07:30pm	*
08:00pm	*
08:30pm	*
09:00pm	*
09:30pm	*
10:00pm	*
10:30pm	*
11:00pm	*
11:30pm	*

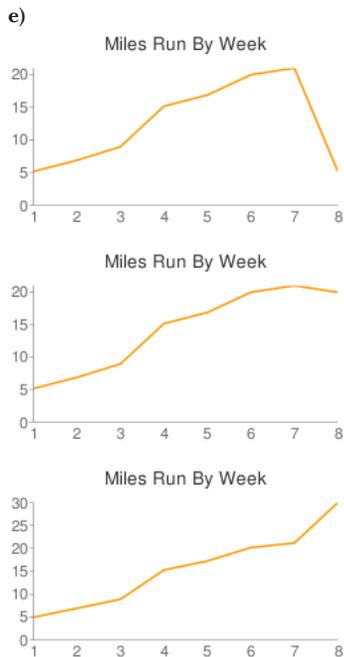


Figure 5.2. Web clips available in Pinky:
 a) Google Calendar;
 b) Remember the Milk to-do manager;
 c) Horde webmail client;
 d) CSAIL room reservation system;
 e) Google Charting API, fed from a Google Spreadsheet.

5. PINKY



Figure 5.6. Web clippings appear as satellites of the Pinky window. The clippings update to match the arguments — here, the date of the reservation.

For example, when the user is scheduling a calendar event or reserving a conference room, Pinky automatically pops up a clipping of the user’s calendar (Figure 5.2a) for that day, to help confirm the date and time of the meeting and avoid overbooking. When the user is sending an e-mail, Pinky shows a clipping of recent e-mails exchanged with the intended recipient (Figure 5.2c). Other clippings appear in Figure 5.2. Clippings appear as satellites around the main Pinky popup window, and update as the command is edited (Figure 5.6).

Pinky’s clippings are reminiscent of WinCuts [127] for desktop windows and Apple Web Clips [23] and Web Tracker [68] for web pages. Unlike these systems, however, Pinky extracts a clipping not by retrieving a single URL and extracting a snippet of HTML or a screenshot, but instead by automating a web application until it reaches the desired state. The resulting web page can thus be customized much more dramatically than with other tools; for example, by showing the calendar focused on the time under consideration, by displaying only e-mails exchanged with the person of interest, or by skinning the conference room schedule down to only the room under consideration rather than a large matrix showing all rooms. The automation and customization scripts are pre-authored by a domain expert.

Since the clipping is a live rendition of the underlying web application, it can update immediately when the user changes arguments in the command, like the day of a meeting. To interact with the web application directly, the user can click on the clipping, which expands the clipping to make the whole web page visible. Clicking away from the expanded page shrinks it again.

By automatically navigating web applications and showing appropriate clippings, Pinky helps bridge the gap between the user and the web application. For PIM data capture, clippings bring useful bits of the PIM tool out to the user, on demand, rather than requiring the user to find their own way into the tool.

Automatic clippings suggest another use for Pinky – not just a shortcut for data capture, but for queries as well. By typing a partial command, like `apr 17`, the user can immediately bring up a web clipping with useful information, in this case their calendar for that day.

5.6 Implementation

Pinky, like Inky, is a Firefox extension built on top of the Chickenfoot end-user automation system [44]. Pinky’s user interface is implemented in HTML, Javascript and CSS; the back-end keyword interpreter is implemented in Java; XML files are used to specify top-level functions.

5.6.1 Keyword Command Interpreter

The keyword command interpreter takes in a command from the user and returns an ordered list of possible interpretations of that command. The goal is to make entries in that list as mirror the text’s intention as closely as possible. Our approach proceeds in two stages for each possible function: first, identifying potential matches for each argument, and second, merging the argument matches into coherent interpretations. Pinky uses a different keyword command algorithm than Inky: one based on the Koala sloppy programming paradigm [100], called Koalalicious.

Each function definition consists of set of name keywords, (e.g., “calendar, addCalendar, add, cal, gcal, schedule, appointment, meeting, rekky, mtg”) and a list of arguments to that function. Each argument likewise consists of a set of keywords (e.g., “time, at, dueTime, due, by, before, on, @, the”), as well as the base type: string, regular expression, date, time, or an enumerated type. These type definition files represent the database against which the user’s command is matched.

In the first stage, each argument searches the string for potential matches and assigns numeric scores to substrings. This score reflects how likely the substring is to match the argument (for example, if it

looks a lot like a time and has the word “at” before it). Each argument may tag as many substrings as it likes, and these substrings may overlap.

In the second stage, the goal is to assign an interpretation based on these sets of argument scores and substrings. We want interpretations that explain the highest percentage of the command text, giving preference to those substrings which have received high scores as likely matches for particular arguments.

The general approach is a close relative of the A* search algorithm. In traditional A* on a graph, search proceeds by successively removing the first item from a priority queue as sorted by a distance heuristic. Each heuristic guess is guaranteed to equal or underestimate the actual distance to the goal: the defining characteristic of an *admissible* heuristic. When an item comes off the queue, all outgoing edges are explored and all new possibilities are placed in the queue with fresh heuristic guesses. If there are no more edges to explore, then either the search is at a dead end, in which case it simply discards the item, or it has arrived at the goal. If it has arrived at the goal, the current path is guaranteed to be the shortest path because all admissible heuristic guesses are underestimations, so any shorter path would have already been examined.

Koalalicious adapts this approach by using the argument scores as a distance heuristic. The heuristic guess is the sum of all argument scores in a single interpretation. As a result, rather than searching for the shortest path using underestimates, Koalalicious’s A* algorithm searches for the largest summed score, and requires equality or overestimation on all score estimates. The initial overestimate score sums the maximum possible scores for each argument, based on all possible substring matches each argument provided. Suppose the interpretation were the following:

```
function: todo
  arg name:
    5pt, [1, 2]
    10pt, [2, 6]
    6pt, [2, 3]
  arg date:
    15pt, [4, 10]
    2pt, [4, 6]
    8pt, [6, 10]
```

The initial score would be 10pt for max(name) + 15pt for max(date) = 25pt.

At each following step, the algorithm removes the highest-scoring (incomplete) interpretation from the queue. It then makes a single argument assignment to maximize its score: in the preceding example, date would be assigned the 15pt interpretation, since that choice adds the most to the overall score. The algorithm then removes any interpretations that

conflict with the now-reserved substring bounds. In our example, we would be left with:

```
function: todo
  arg name:
    5pt, [1, 2]
    10pt, [2, 6] (pruned due to substring overlap)
    6pt, [2, 3]
  date:
    15pt, [4, 10] (chosen by the algorithm)
    2pt, [4, 6]
    8pt, [6, 10]
```

The algorithm then sums the highest possible remaining scores, here 15pt + 6pt = 21pt, and places this new guess back on the queue. Removing all possible interpretations for an argument does not signify a dead end, as the user may have omitted an argument.

When the algorithm removes a fully-assigned interpretation from the queue, that interpretation is guaranteed to be the highest-scoring interpretation for that function. The algorithm may then continue to run and generate a ranked list of interpretations.

5.6.2 Web Clippings

To render a web page as a clipping, it is opened in an HTML iframe element set to reasonable browsing dimensions (800x600), which ensures that the clipping is rendered in a familiar and readable way. A pre-authored Chickenfoot script automates the web page using arguments supplied from the Koalicious parse, for example the date and time of the calendar event under consideration. Once the automation is complete, the desired region in the page is located (e.g., a single day in Google Calendar), and its bounding box is used to clip the iframe by positioning the frame appropriately inside a viewport element, a div of the appropriate width and height. The div element is absolutely positioned on the page to appear as a satellite, and the iframe is absolutely positioned inside the div to place the upper-left of the clipping bounds at the upper-left edge of the div.

5.7 Evaluation

As Inky had previously been evaluated with a small field study [51], we can extrapolate appropriate conclusions to Pinky. Inky's field study involved seven users, all members of MIT CSAIL who regularly use Firefox, who used Inky over a period of approximately a week. The data gathered from the study shed light on learnability, accuracy, the

importance of synonyms, the importance of order independence in commands, and the importance of suggestions. In brief:

- Almost all commands executed were presented at the top of the interpretation list
- Synonyms and re-ordering are critical to successfully recognizing a small but significant number of commands

Pinky has not yet been rigorously evaluated. The main aspects of Pinky which require evaluation are its novel layout, its inclusion of web clippings as just-in-time information, and GUI widgets.

To evaluate the layout and usability of the interface, a laboratory usability setting would be appropriate. A question for investigation: does the Inky's list interface or Pinky's tabular layout lead to faster, more error-free capture under time pressure? This question could be answered via a within-subjects experiment using a precompiled list of personal information items to capture.

Web clippings need to be evaluated in two areas: 1) do they provide useful information at appropriate moments?; 2) is the compressed interface to that information usable? A laboratory study may begin to shed some light on these issues. We can amend the previously-detailed study by asking participants to role-play as a fictional person whose schedule, contacts list, etc. already exist online. Some commands given to participants might be straightforward, with no conflicts or other information needed, while others might depend on the information from the web clippings such as a time conflict. In this way, we can encourage participants to interact with the web clippings and uncover both quantitative data answering the question "does it help?" and usability feedback. The broader question of whether the clippings are engaged in practice would need to be evaluated through a longitudinal study.

A longitudinal evaluation could be a mix of an experience sampling study and a traditional rollout. Participants would install Pinky on their computers and receive training on the software. Follow-up meetings would ensure that they understand the software and reinforce the importance of continued usage (see CHAPTER 6). Furthermore, participants would be text-messaged at semi-random intervals with information to record immediately using Pinky. This experimenter-fed information would be based on reference tasks [137] we wish to compare across participants, and also further reinforce Pinky usage.

5.8 Future Work

We suggest that Pinky may be useful as a query mechanism in addition to a capture mechanism. It is an intriguing idea to be able to call up a keyword command window, type `board meeting` and have the interface show you the time and location of your next board meeting. Pinky already supports a measure of this style of interaction: typing `mtg tomorrow` will display tomorrow's calendar in a web clipping. However, this notion could be extended by allowing the information to be acted upon, for example by changing the date or time, or by pasting the selected information into the current application. Such a search-and-paste approach might speed up common interrupting tasks like finding a contact's e-mail address to type into a web form.

Long tail personal information may also be an appropriate target for Pinky. For a user who maintains a spreadsheet of jogging activity, entering `jogged 10 miles` might capture today's activity; for a user who is applying to graduate schools, `MIT recommendations due 12/15` might capture an additional bit of information about the application process into a spreadsheet. Is long tail information the best candidate for a Pinky-style interface, however? The ethnographic data suggests that reliable archiving and extensible formats are much more important than lightweight capture for long tail information.

Command history might be integrated into the Pinky interface to speed entry and reduce ambiguity. For instance, when the user enters `board meeting`, Pinky can search its command history and display all previous board meeting commands in a satellite window. The user might select entries in the list and use them to populate arguments in the main command window, for instance carrying over the previous time, location and attendees. Command history can also be used to disambiguate `board meeting Blair 6`, if previous commands reveal that Blair has been used as a location and board meetings are typically at 6pm rather than 6am.

The context information mined by PLUM for Jourknow can be used to disambiguate commands as well. A command `jourknow mtg w/ david 5pm` is inherently ambiguous: do I mean David Karger, David Huynh, or David Bowie? PLUM's topic models of my e-mail might suggest that David Karger co-occurs with Jourknow much more often than David Huynh or David Bowie, and then implicitly store that information. Potentially more powerfully, PLUM might notice that I have been trading e-mails about Jourknow with David Huynh recently or that David Bowie (but not David Karger) is in the room when I make the command, and adjust interpretation accordingly.

5.9 Pidgin and Keyword Commands: Lightweight Data Capture Mechanisms

Pinky exhibits many similar characteristics to the Pidgin languages described in CHAPTER 4. They both expose lightweight textual capture mechanisms. What can we say about the meaningful axes of distinction between these tools?

5.9.1 Language Axes

We can characterize several important design features of the textual languages themselves:

- General/ontology-specific: whether the language contains domain specific representations (e.g., events, contacts) or generic entities and relations
- Resolution of entity and property names: the ability for the user to refer to entities using short/familiar names instead of a long but unambiguous identifier.
- Literal type deduction: the ability to automatically determine the types of literal values without extra work by the user, e.g., to parse relative dates and coerce numeric values
- End-user extensibility: letting the user extend the language to new forms
- Nesting of expressions: allowing for the embedding of one statement within the clause of another
- Reorderability/Optional clauses: supporting reordering/optional clauses
- Mandatory delimiters: whether to require the user to adhere to strict syntactic rules
- Syntactic ambiguity: whether the strings in the language can have multiple valid interpretations
- Command or data description: does the language take a noun-oriented perspective (“there is a meeting”) or a verb-oriented perspective (“create a meeting”)?

The preceding study identified a number of these features as essential if not highly convenient for most users. In particular, supporting familiar references to entities and properties and literal type deduction appear to be of particular importance. There is often a delicate trade-off between naturalness and unambiguous interpretability. For example, syntactic de-

limiters were seen in general to be somewhat onerous (e.g. requiring quotation marks around literal expressions); however, relaxing the syntactic requirements too far often resulted in an explosion in syntactic ambiguity – which is perceived to be far worse.

The Inky user study reported earlier demonstrated that being flexible regarding word/phrase order in expressions might be important as well; the study revealed that nearly 50% of expressions entered by users in their sloppy-programming [101] based language were out of order in some way, despite immediate graphical assistive feedback.

5.9.2 Design Axes

We can likewise characterize the design choices of the tool. Though there are many such axes, a few particularly prominent ones are:

- Embedded or independent: is the language embedded in an application (as is Pidgin in Jourknow), or is it an independent interface?
- Disambiguation: how does the interface support disambiguation? Is disambiguation performed by shuttling through full parses, or by fixing individual problems?
- Ephemeral or permanent: do the statements continue to exist after they have been issued? Are they kept up-to-date as the referenced PIM items are changed?

5.9.3 Pidgin Languages and Natural Language Processing (NLP)

A potential criticism of our approach is that it is simply “NLP-lite” — that as soon as computer scientists solve the problem of recognizing arbitrary natural language, the Pidgin approach will not be needed. However, there are several reasons to doubt such an argument. First, a complete NLP solution remains distant at this time, and is more than is needed for the simple data capture we support. Thus, the Pidgin approach offers many of the benefits of NLP input, sooner. Second, individuals’ jotted notes are generally not in “natural language.” They include personalized language, abbreviations, ungrammatical constructions, and a variety of other language hacks to make entry more efficient. By contrast, most NLP corpora are trained on well-formed English sentences common across individuals. Users do not want to waste time crafting grammatically complete sentences to record information fragments. While NLP might ultimately be able to handle this unnatural language, shorthand is highly individualistic and requires a solution that learns for each user differently – an even more challenging problem than standard NLP. Third, Pidgin emphasizes the value of bridging from a more naturalistic input framework to a tra-

ditional GUI output environment, in contrast to many natural language systems that assume natural language is the right modality for both directions.

5.10 Conclusion

Pinky is the first step toward a new type of information scrap manager — one that preempts the need to create information scraps. As such, it eschews the typical tropes of notebook applications for a command-style interface. I believe that it is such an approach — paying attention to the design needs behind information scraps and not necessarily to the scraps themselves — that will lead to the most innovative and successful designs.

Pinky accomplishes this break with tools like OneNote and Evernote by focusing on a constrained subset of the problems: information scraps that could in principle live in common applications, but in practice do not. Our investigations uncovered lightweight capture as a powerful reason moderating capture of this data, and thus Pinky is designed as a capture tool.

6. DESIGN PROCESSES FOR INFORMATION SCRAP

The design process is rarely a wholly scientific effort. For most design problems, process exists to lend guidance to an inherently entropic set of tools and techniques. These techniques are aimed first at *getting the right design* (generating as many ideas as possible and selecting the best one), and then at *getting the design right* (executing the chosen idea) [129]. These techniques begin with sketching and ideation [49] and transition to refinement and a focus on usability.

Not all design problems are well-suited for a process, however. Rittel identified a subset of design problems he termed *wicked problems*: those with particularly messy, circular, or contradictory requirements [115]. When attempting to solve a wicked problem, it is most likely that a designer will unintentionally uncover or create further issues. For such design tasks, process is less of an aid: a designer may enter late stages of usability testing, only to discover that the design has a fatal flaw in its fundamental assumptions.

My experiences with information scrap design suggest that the design of an information scrap manager is indeed such a wicked problem. As discussed in CHAPTER 4, the Jourknow prototype faced unanticipated feed-

Work presented in this chapter is a collaboration with Max Van Kleek, David Karger and mc schraefel. It is published as a CSAIL Technical Report [37] and a CHI 2007 work-in-progress paper [39].

back at the conclusion of its longitudinal evaluation. I had been interested in the usefulness of contributions like context-based re-finding, Pidgin input syntax, and the mobile Jourmini client. Instead, I found that many participants gave up on the tool early in the trial, and gave unanticipated reasons for doing so:

- “It didn’t integrate with Outlook.”
- “I didn’t like that it was an extra window open on my Taskbar.”
- “I didn’t *get* it.”

This chapter is a characterization of the nature of such design problems associated with designing for information scraps. I will begin by reviewing the Jourknow design process and recounting how the design and research team rationalized its investigative approach. I then look at four points in the process that reflection suggests might have impacted our outcome: scale of prototype, effective prototyping techniques, choice of participants, management of participants in a longitudinal study. I interrogate these points against known design methods. I conclude with a consideration of how to move forward, and reflect on implications of our experience for design practice.

6.1 The Design Process

The following section traces out the evolution of the Jourknow prototype, from early design exercises, a first prototype, ethnography and a revised prototype. The “we” voice in this section refers to the research team.

6.1.1 Early Ideation and Design Space Exploration

Having identified a problem space of interest, the design team began with an exercise: if you had a magic text file that could do whatever you wanted, what would you do with it? Our team of four researchers spent a week interacting with this “fake computer.” We were interested in considering: what kinds of creative uses could we come up with for this tool?

We observed a number of interesting characteristics in our logs, for example deliberate ambiguities such as “do ___ stuff” or “remind me” notes without any mention of when the reminder should actually occur, and the use of commands such as “open cal.” Structure ranged from very orderly notes to almost unparseable text. Verbosity also varied between clearly explicated sentences and very condensed text, even within the same log. Two researchers explicitly recorded contextual information like date, time and

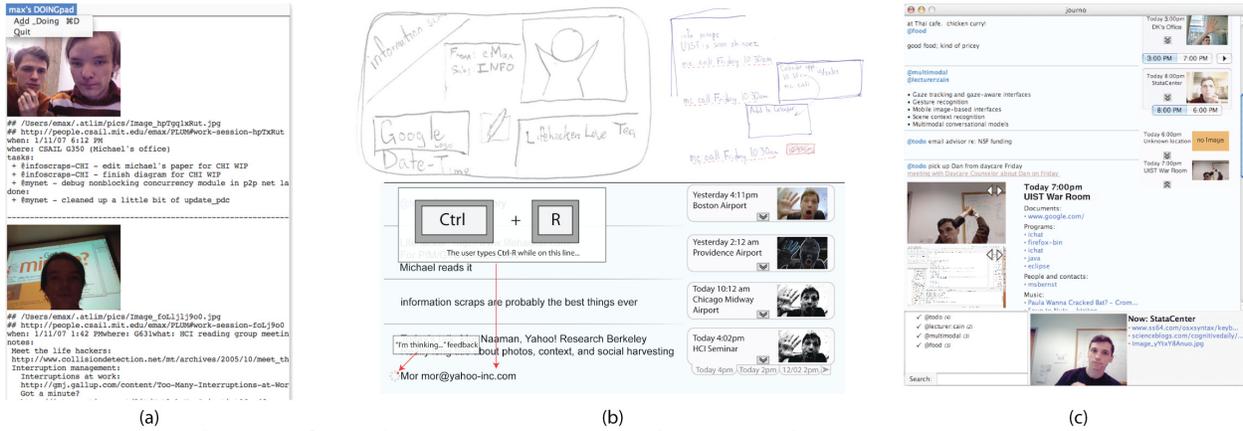


Figure 6.1. Evolution of Jourknow prototype: (a) the original DOINGpad prototype, (b) sketches and storyboards for our functional prototype, and (c) the final version.

location into the text file, and a third, reflecting on his failing memory for the note, remarked that he wished he had done so as well.

Based on our experiences, we built a first prototype system called the DOINGpad (Figure 6.1a), so named because it captured what the user was doing whenever he or she recorded a note. The DOINGpad (“doing-pad”) was intended as a functional sketch [49] intended to explore an idea space -- implemented in four hours, we built it to elicit feedback amongst the design team as we used it. DOINGpad recorded the following whenever the user begins to write a note: current date and time, friendly location name (from the wireless access point; e.g., “max’s office”), a webcam photo of the user and his/her surroundings, and a Uniform Resource Identifier (URI) which could be linked to other concurrent system activity such as window switches and music being played. Below these system-generated fields is a free text area for recording the note itself.

6.1.2 Involving Related Work, Functional Prototyping

With DOINGpad allowing us to reflect on our approach, we began to iterate upon our ideas. Our explorations spanned several research domains; from PIM research we took note of the inherent tension between a need for lightweight entry and a desire for structured representation later [30, 41, 86]. Studies of remembrance habits then informed us of the various mechanisms our users might utilize to re-find information, such as relevant people and situations [54] or pictures [122]. Here we drew on systems such as ChittyChatty [86], and Stuff I’ve Seen [60] for design inspiration.

Given this variety of research recommendations, we set out to incorporate them into our tool to see if their insights would positively impact our own work. Through design iteration (Figure 6.1b), we developed the first version of Jourknow, a journal that “knows.” Jourknow represented our first foray exporting our own ideas into the functional prototype space for feedback. Its main design points were automatic context capture and association with notes in support of re-finding (e.g., “it was that note I took down when I was at Starbucks”) and lightweight structured expression parsing, which we called Pidgin. We employed first-use studies and design critiques in order to get first-contact feedback on our prototype. The prototype was, however, still too slow and too brittle to be used on a regular basis.

6.1.3 Expert Feedback

With the Jourknow prototype demonstrable but not yet stable or polished, the research team decided the next appropriate step would be to put the Jourknow interface to expert critique. We headlined information scraps and Jourknow as a work-in-progress poster at CHI 2007 to gain feedback from the attendees [39]. We received a much more positive response than anticipated (first place award, people’s choice!), much positive feedback, and many requested features.

At this point we also received expert reviewer feedback on Jourknow, and acceptance of the prototype into a top-tier computer science and HCI conference [131]. Reviews indicated support for our direction but a need to test our ideas on real users:

- “The authors have implemented a reasonably complex system to try to address this well-motivated problem. [...] Since it was informally evaluated with CS students in a lab, how can we know if this is even reasonably usable for non-techies?”
- “There is a need for longitudinal testing to establish how such a system would fit in with people’s working practices: who does such a system actually suit, and why?”
- “I agree with the other reviewers that this paper describes a cool system. Certainly I want to use something like this. [...] but, I’m also not sure what we learn from this work without evaluation.”

6.1.4 Needfinding and Ethnography

Before incorporating the expert feedback into our prototype, we decided first to hone our knowledge of the domain of information scraps. To this point we had based much of our research on existing literature

informing information scrap management (as detailed in CHAPTER 2). However, we found that the literature left unanswered questions: what kind of data is kept in information scraps? What kinds of tools are generally used? What do they look like? What factors affect their creation and use?

Thus, in order to more fully understand the makeup, contents, and needs of information scraps, we performed our own investigation as detailed in CHAPTER 3 and in an upcoming journal article [35]. Our study consisted of semi-structured interviews and artifact examinations of participants' physical and digital information scraps across physical and digital tools. For details on the study, see CHAPTER 3.

6.1.5 Scoping and Research Specification

At the conclusion of our study, we reflected upon lessons learned and how we might apply our new knowledge to Jourknow. During a two-day caucus, the researchers attempted to scope the project to areas of interest in need of evaluation. We grounded our hypotheses firmly in our own work as well as related research — each hypothesis needed to be justified by observations from our ethnographic work. For each feature, we examined whether leaving it out would significantly harm the overall effectiveness of the system.

We began with our two hypotheses from the previous prototype: context capture and Pidgin structured language input. An object of discussion was: should we leave the work at those two hypotheses for evaluation, or add something new? We foresaw that users who did not always carry laptops may see limited use to the system, just as users of existing digital tools in our study found mobility a major inhibitor. Furthermore, our participants reported that their tools were often rendered useless when they were not accessible when a note was needed, for instance when away from their desks, driving to work, or at home. Thus, we hypothesized that supporting mobile note-taking might greatly improve the overall experience and usefulness of our system. Thus, we decided to focus our prototype on the following three improvements to existing information scrap practice: context capture, structured capture (Pidgin), and mobility.

6.1.6 Jourknow Client Redesign

At this point the research team took the opportunity to redesign the client based on knowledge gained from our previous iterations. We began by generating a large number of basic interface approaches for information scrap management, then built paper prototypes [113] (Figure 6.2) to investigate the most promising directions: an inbox metaphor, a note-

6. DESIGN PROCESS

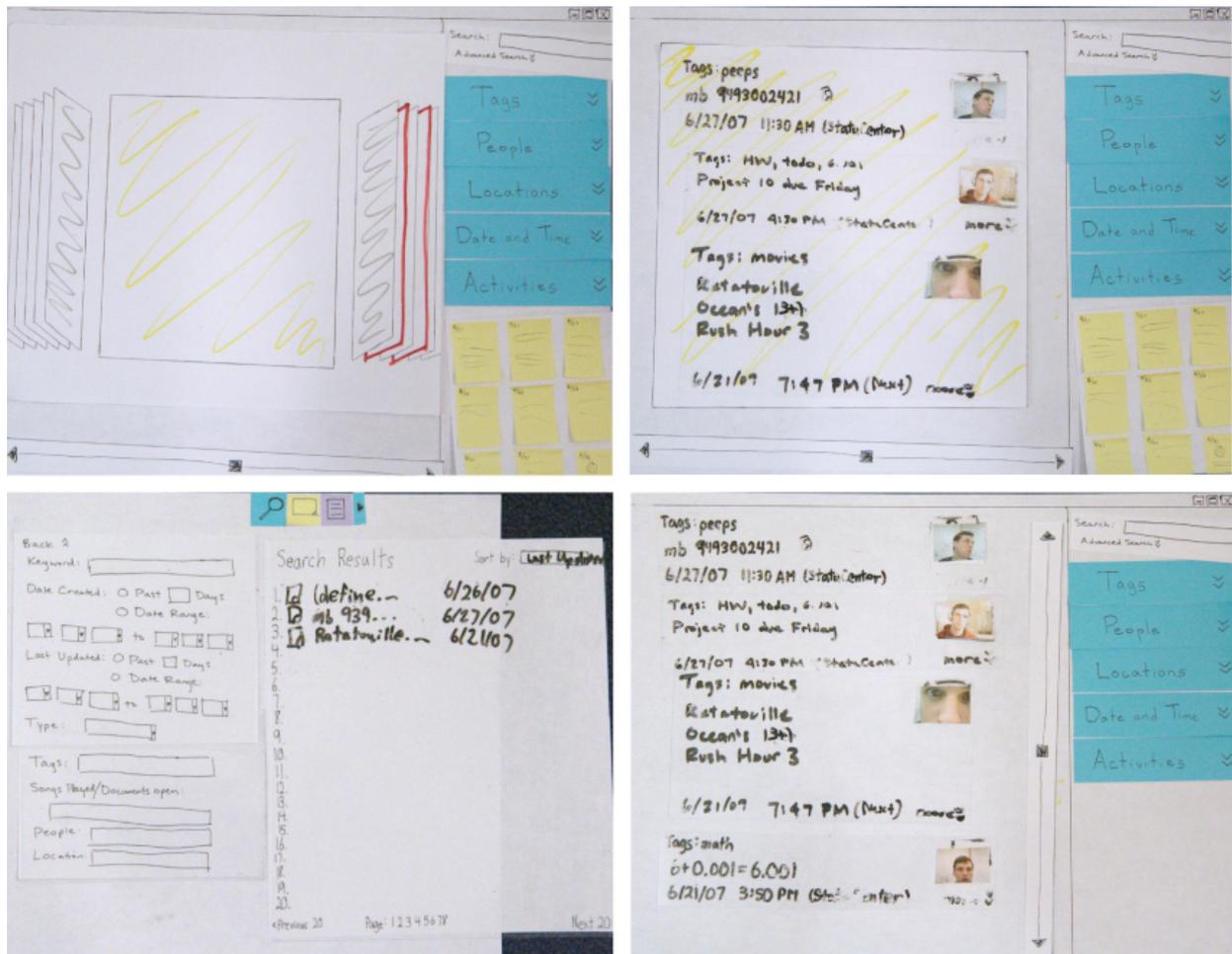


Figure 6.2. Paper prototypes of the revised Jourknow interface, exploring notebook, list and search approaches.

book metaphor, and a search-only interface. We recruited participants from the lab to interact with the paper prototype, which had already been populated with notes; we found that the notebook metaphor afforded a level of spatial memory that participants generally preferred. However, the list interface also seemed to have merits: physical resemblance to a word processor, a logical place to start capturing (i.e., at the end) and an easy metaphor for supporting both automatic and manual arrangement (i.e., sorting). Thus, we brainstormed and designed the remainder of the interface, relying heavily on existing interface paradigms in faceted retrieval (e.g., [139]) to reduce risk. Over a period of the coming weeks, we continued to refine of the design, focusing on Pidgin and context facet panel.

6.1.7 Development

A team of four researchers tasked themselves with implementing this new version of Jourknow over an approximately ten-week period. The first four weeks were concentrated on implementing the general client, including the dashboard mechanism, reminders, the basic user interface in a notebook metaphor, and internal logic and representation. Much of the code from the original Jourknow prototype was rewritten to support the new design. Throughout the design and development process, the research team held weekly design reviews with a larger group of students and researchers to get feedback on progress and design decisions.

As described above, at the end of the fourth week the client entered a design review and came out with a revised specification. From this point on, our focus was in completing the prototype in time for the summative evaluation to come. Midway through development, a fifth researcher joined to implement the mobile client. Implementation fell behind schedule and the researchers made value tradeoffs concerning features to cut. Various core and auxiliary features were cut in the last weeks of development, including automatic transactional saving and integration with existing office applications. Cuts were made carefully avoiding features that we believed would compromise our ability to test the main hypotheses of the project. The final prototypes are shown in Figure 6.3 and Figure 6.4 — for further discussion of the prototype see CHAPTER 4.

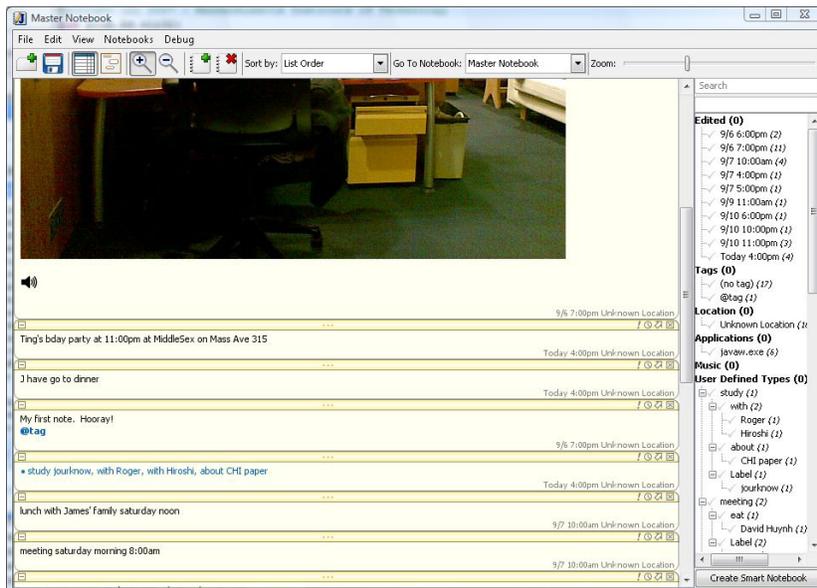


Figure 6.3. The final Jourknow prototype.

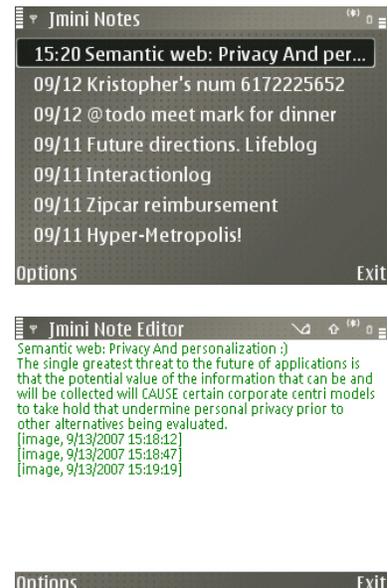


Figure 6.4. The final Jourmini prototype.

6.2 Study Design and Execution

The evaluation of our design may have had as strong an effect on the results as the design itself. In this section I detail the evaluation approach we took toward understanding Jourknow’s successes and failures.

6.2.1 Method

Concurrently with the research scoping meetings, the research team deliberated on an evaluation approach for Jourknow. The two study types we considered were laboratory and longitudinal evaluation. A laboratory study would have allowed us to directly examine particular aspects of the interface, such as the design of the Pidgin language or the facet panel, whereas the latter (what Kelley and Teevan term a combination of longitudinal and naturalistic studies [89]) would give us feedback on the integration of the tool with users’ lives.

We viewed a laboratory study as too artificial and controlled to reveal how Jourknow might be used to capture information scraps in real situations. Furthermore, our previous reviewer feedback indicated a need for longitudinal evaluation of the system. We thus opted for a longitudinal study to give Jourknow a chance to integrate itself into our participants’ information management practices so that later we could observe its impact. Our decision carried an implicit assumption that Jourknow would achieve basic uptake, and thus that real-world observation of its research features was a useful next step.

Our study and the results are detailed in CHAPTER 4. In brief, we recruited 14 participants from our university (ages 18-41, median 26), external to our research group. Following standard practice (e.g. [60, 117]), we installed Jourknow on participants’ computers, and instructed them in the use of the interface. We also described several of the shortcomings of the current version of the research prototype — slow loading and saving, occasional GUI bugs, and a remaining server bug that was patched near the beginning of the study. Participants were instructed to introduce Jourknow into their everyday note-taking practices, and to make extra effort to use the software to capture their thoughts and notes. They then used the Jourknow client for a period averaging eight days, including one weekend. Throughout the study, we used e-mail announcements to promote use of the tool, remind participants to integrate the tool into their lives, and keep in constant contact. This level of contact was fell short of other studies which made regular visits to participants (e.g., [31]), but was more direct than those with no reported communication during the study (e.g., [124]).

6.2.2 Study Results

The study results are reported in CHAPTER 4 — in this section, I instead focus on feedback that was unexpected and has bearing on the design process. As a brief review, we quickly found that participants were not making regular use of Jourknow. In response to an e-mail suggesting everyone synchronize their notes with the server, two participants e-mailed us admitting that they had not yet opened the tool, with a third participant experiencing trouble starting the tool on his computer. We helped the participant debug the problem, then sent an e-mail reminding all participants that we had asked them (as per the study agreement) to make daily use of the tool. At the conclusion of the study, three participants (P2, P3, P11) had never launched the client after their initial installs. A fourth participant only used the client once right before his exit interview (P1). Others' usage varied significantly, though there were usage spikes in response to our reminder e-mail.

During the closing interview, we scheduled each participant to spend an hour with two investigators (one acting as facilitator and one taking notes), where we planned to have participants first provide their general impressions of using the system, and then to walk-through the notes they took using the system, to allow participants to recount their experiences with it.

It took little time to discover that this protocol would need to change due to anemic tool adoption. With our first exit interview (participant 1), we discovered that he had not used the system at all during the week, and had only 2 notes (one of which was created on the day of the install, and one created on the day of, and shortly prior to his exit interview). When asked why he had not used the system despite requests and the terms he had agreed to in the study, he responded “It didn't become part of my routine, I had to be conscious of it; I'm not accustomed to doing this kind of thing, and it required too much effort for me to bother with it.” Other participants who did not use the tool responded similarly; adopting the tool seemed to require more effort than they wanted to invest. Participant 9 had a slightly different explanation of why he didn't adopt the tool: “Your tool is just not useful to me. You said that this tool was designed to help people whose ideas just ‘pop’ into their heads, who need a place to write them down. Well, it occurred to me that this just never happens to me! Either I have a lot of ideas that are just not worth writing down, or I just have one good one that I hang on to [in my head] and I don't need to.”

A majority of the remaining feedback we received focused on highly specific, particular characteristics of the system and of the user interface

that they did not like, found annoying or “broken.” These included synchronization “just not working,” complaints about lengthy save/load/launch times, various note views “not working” and being confusing, frustration from the rendering, and issues with ordering and presentation of notes, including font and icon sizes.

Feedback was also occasionally positive, but often inconsistent. Several participants reported liking features (such as the ability to keep notes on the desktop) but it was not clear that they had actually ever used the features (as they were unclear about how they worked); we also noticed that two participants contradicted themselves by first saying they liked something, and then saying they were annoyed by it or “couldn’t stand it” in another context.

After all of the negative and inconsistent feedback regarding the desktop client, we were surprised when 3 participants protested when began to delete the system from their computers. This was the strongest evidence we had that some participants had actually started to adopt Jourknow into their organizational practices.

6.2.3 Symptoms of a Wicked Design Process Failure

Our user study results exhibited a small number of generalizable characteristics:

Participants’ inability to articulate their critique. Whereas we intended to probe for feedback on the general design of our tool and on our research hypotheses, our users were unwilling to provide much feedback on them. Instead, we received very general responses, characterized by broad generalizations such as “I didn’t get it” or “I didn’t find this tool useful.” When pressed for reasons, participants (unable to articulate the causes of their disposition) usually paused briefly and then produced a reason which we believe constituted the first plausible justification they thought of. The range and types of reasons varied largely (as described earlier) but largely surrounded overly specific details, failing to provide any larger insight regarding the tool’s design.

Inconsistent feedback. When appraising the usefulness of various features of the tool, we often found both inter-participant and intra-participant disagreement. While the former could be explained by differences in individual preferences and practice; the latter, self-contradictions, are troubling — they suggest that appraisals were less reliable as a source of information regarding whether they would truly use the features being appraised.

Lack of adoption of the tool. We observed very little use of our tool amongst our participants. We had requested that our participants insert the tool into their everyday practice, but it was clear that existing practice proceeded with little effect by our tool. Several participants barely used Jourknow during the study period, and several more tried briefly and then ceased to use it.

Lack of coverage over users' varying habits. Though we dedicated a large amount of engineering and design work to covering the basic needs of the information scrap space, we were nonetheless unable to satisfy many of our users. In addition, we received seemingly inconsistent feedback that basic features, while critical to some participants' happiness with our tool, were highly undesirable to others.

6.3 Reflection on Practice: What Went Wrong?

From the above discussion, we see that despite strong momentum going into the final study, the study was unable to test the team's desired hypotheses. My goal in this section is to examine the various choices made in the process of designing and developing Jourknow. I first caveat by discussing the most straightforward solutions the user-centered design process suggests. I then propose four candidate moments for review, which I call *breakpoints* in the process. These breakpoints include the research scope, the design and prototyping process, the study type, and study population. For each of these particular breakpoints, I will reflect upon how the particular domain of study — personal information management (PIM) — played a role. My goal is to reflect on the methodology that informed our actions in each of these phases, and to investigate what other practices might have better informed the team's approach.

6.3.1 Considering the Obvious Solutions

The most straightforward critique of this process may be that we did not adequately follow the user-centered design mantra. These are the stock solutions that the interaction design process has to offer. Certainly these critiques are valid and following this advice would have abetted our process. However, they may be occluding other important elements of the story, and so I point out ways in which paying attention only to these stock solutions might be misguided.

More UI prototyping! One answer to the lack of user adoption might be that the team should have carried out more interface prototyping. To be sure, lo-fi and hi-fi prototyping would have revealed errors and missteps, for example to improve the visual representation and layout of the facet panel, and structure of the pidgin syntax. However, this prototyping may not have addressed the fundamental issues our study participants reported. For example, our business school participants almost unilaterally did not want to use a computer to take these kinds of notes; if they did, they needed it to be an extension of Outlook, not a separate tool. As we discuss in the breakdowns to follow, our prototypes may have simply been focused on the wrong aspects of the experience.

More system testing! Much feedback we received surrounded participants' perception of the client being buggy and too slow/unresponsive. We have no doubts that more time would have allowed for greater integration and performance testing using more client workstation configurations; which would have uncovered problems that could have lessened this perception. However, it is not clear that even testing our system until it was perfectly robust would have received substantially greater adoption, based upon feedback from the couple users who persevered through the glitches and still found many aspects of the tool useless. This suggested that the most important troubles with Jourknow were design-oriented, and that perhaps the glitches were partially a proxy to blame for these more latent underlying design problems.

More iterations! Assuming we had more time, more prototypes, and multiple rounds of quick and dirty feedback, the next question is: would our methodology have supported us then? In deference to the "wicked" nature of this problem, the answer is not clear. Why did we see fit to move from high-fidelity prototypes to a first client implementation? Design is a process of exploration and then refinement [49]; we had refined a prototype that was somehow locally optimal (based on positive informal feedback) but not globally so. Specifically, having employed multiple methods, from interactive sketches, lo-fidelity prototypes to hi-fidelity prototypes, our team felt that we had enough design feedback to proceed with an implementation. Our study results uncovered this error.

6.3.2 Breakpoints: Process Inspection Points

In this section I detail the elements of our process which may have unexpectedly impacted our experience. These elements are related to the wicked nature of designing for information scraps, and will challenge any designs in this space.

6.3.2.1 *Breakpoint 1: Scope of PIM Investigation*

The team planned to introduce a single tool to address the problems of information scrap capture and retrieval. Our approach to building this tool specified four pillars of design to meet the challenges we had identified in our research: a general note capture and manipulation interface, context capture to facilitate note retrieval, a lightweight structured data capture language (Pidgin), and mobile capture and access. In hindsight, we might ask: did we really have one idea, or four? Should each of these have been studied individually, or were they simply too co-dependent to do so? What gave us confidence that we could design, develop and evaluate them all together?

At the time of this breakpoint, there were two main factors that played into our decision: the power of the Gestalt in PIM, and positive feedback and inertia from our previous prototypes. We analyze each in turn.

Personal information management tools are such multifunctional devices that they necessarily encompass an entire ecology of use rather than a single research or design problem. This situation leads to two results: huge functionality requirements (resulting in large start-up design and implementation costs), and perception of the system as a Gestalt rather than as singularly differentiable features. Bellotti et al. describe one PIM application, e-mail, as “a mission critical application with much legacy data and structure involved in it” — and go on to report that several of their users dropped out from the study due to limitations of their research system to adapt to users’ complex usage habits [31]. Kelly and Teevan conclude that PIM prototypes must be more robust than typical research prototypes [89], and with both TaskMaster and Jourknow we also see that these tools must also support broad functional requirements in order to compete.

This situation placed the team in a difficult position: the system as a whole may not be useful unless we solved several problems simultaneously. Specifically, our inclusion of the mobile client was a response to strong motivation in our previous studies suggesting digital tools severely limit their own usefulness by being available only on a user’s workstation or laptop computer. However, in retrospect, the inclusion of the mobile client may have contributed to a prototype unable to anticipate the broad functional requirements supporting our ideas. It is thus questionable whether broadening our scope improved the situation, or simply left us unable to do any of the ideas justice.

A second factor in our decision to incorporate all four ideas into our design was the very positive response we had received from outside re-

viewers inspecting our work and ourselves. We implicitly took such feedback as design approval and cut down on usability studies of the client. We mistook expert inspection feedback for user feedback. In the space of personal information management, we also see that inspectors may have had difficulty projecting themselves into the use of the client, leading to overly positive feedback.

6.3.2.2 *Breakpoint 2: Prototyping and Interaction Design Process*

In designing a complicated system like Jourknow we faced a number of interaction design challenges. Here we examine some of the potential design missteps we may have made, including too few iterations and difficulty prototyping the experience rather than the interface.

The negative feedback we received on the basic design of some pieces of our interaction points to a need for more formative evaluations, earlier on in the process. Design reviews and adherence to research precedent were insufficient in our case. One possible solution may have been to use formative laboratory studies during implementation to investigate features in isolation before the longitudinal summative evaluation, or to have lab partners use half-functioning versions of the prototype for feedback.

Our prototypes also faced a challenge simulating the *experience* of recording an information scrap, rather than simply the interaction. This means that our prototypes succeeded at getting feedback on many interface design challenges, but were less successful at placing that interaction in a context of use. This effect may also have been amplified by our position in the personal information management space, where even small details can make impressive differences in behavior [118]. Our prototypes focused on the novel features — on being able to re-find information based on context and capture structured information with little effort. Here, we question whether our prototypes were truly effective experience prototypes [47], garnering feedback on the rich context surrounding notes' capture and context surrounding reuse. If we failed to prototype important parts of the experience, it is not surprising that user feedback concentrated on unexpected areas of the system.

6.3.2.3 *Breakpoint 3: Study Methodology*

The choice of population implicitly assumes the question of the choice of study form: user-centered design promotes the use of multiple methodologies for evaluation, and recognizes the tradeoffs of different methods in

evaluating an interactive system. A point of reflection: was a longitudinal use study the best choice for Jourknow at its current stage of development, and could we have organized the study more in support of our goals?

To recall, we chose a longitudinal evaluation to give Jourknow a chance to ingratiate itself into our participants' practice, and to reflect on how that practice, once engaged, was or wasn't successful. Was this decision optimal, however? Should the team have adapted or combined longitudinal and first-use study methods, rather than using them in their typical formulation? For example, we might have begun with a shorter longitudinal study (2-3 days) to identify pain points with the application and then proceeded to use laboratory evaluation to further understand the results. It was potentially to our detriment that we chose the most ambitious study to begin with.

Given that we chose a longitudinal evaluation, did we design the study in such a way as to maximize our chances of success? For example, we chose to keep in contact with participants via e-mail rather than requiring further in-person interviews during the study. Plaisant and Shneiderman [111] and Bellotti et al. [31] report that their longitudinal efforts benefited from reappearances to remind each participant of processes in the software that he or she had forgotten about. In previous longitudinal studies of software, however, we see that researchers do often follow up remotely [60, 121] with success. In our case, keeping in close contact with participants would have increased social pressure to use the tool and allowed us to provide follow-up training; this is evidenced by our mid-week e-mail reminder causing a temporary spike in usage.

6.3.2.4 *Breakpoint 4: Choice of Study Population*

The quest for external validity [105] dictates that researchers and practitioners randomly choose participants from the target population, rather than form a hand-picked subset. Recently this issue was brought to a head by Barkhuus [26] with a call-to-arms for SIGCHI to stop using local participants (particularly HCI graduate students) in their studies. Thus, pressure from the CHI community to use a random population was a large motivator in our decision to give Jourknow to a group of consisting largely of business students. Here we examine our choice to follow this desire to achieve this new CHI goal for studies to get out of one's back yard rather than testing on participants closer to the research project, or even ourselves.

In the domain of personal information management, ironically, there are reasons why testing outside a friendly community might hurt a study.

Kelley and Teevan point out that recruiting PIM system evaluators is a particularly thorny issue: participants must be willing to grant access to personal information, overcome self-consciousness of messy practice, agree to a large time commitment, and commit to temporarily suspending their deeply-ingrained practices [89]. Kelley and Teevan also note that studies in this space, including Bellotti et al. [31] and our own, suffer from a degree of participant mortality (drop out prior to the conclusion). A third possible problem lies in community practices in PIM (for example, business students using Outlook) previously unknown to the experimenter. Finally, again due to the “mission critical” aspects of PIM, there is little room for error — while business students were excellent critics of the system and an appropriate user group, they were also unable or unwilling to overlook entry barriers to using the system such as outstanding bugs and performance issues.

PIM researchers are left with three main options, then: continue to pursue externally valid studies with outside participants, use insider participants who may be more pliable and willing to evaluate a system through its defects, or “eat their own dog food” and have the researchers themselves reflect on using the system themselves for a period of time. Jones [80] promotes this final option of self-study as a particularly useful tool in PIM research. However, the closer the study population to the research team, the less external validity the results carry. In our case we believed our tool was ready to demonstrate an improvement to a general audience, but this may have been a heavy investment with little return.

6.4 Outcomes for Design Methodology

While I have considered above how we might address next steps for my own process in this project, my overall goal here has been to reflect upon the methodological path or choices that lead my team to the decisions we made. Based on this experience, I suggest that the level of certainty various design methods instill in the practitioner or researcher may vary depending on the problem domain. Particularly in wicked domains such as information scraps and personal information management, applying a plurality of methods gave us false security that were prepared to build and evaluate a full research prototype — when in fact basic design elements of our system were still faulty.

I would suggest — though this proposal itself will need to be validated in some way — that the breakpoints we have identified in our process may indeed be generalizable breakpoints for others (a) working in PIM research in particular, (b) focused on wicked design problems, or simply

(c) using multiple methods in any artifact design. We must interrogate the process, and watch for warning signs that indicate a false positive. In our case, experience prototypes did not succeed in eliciting feedback on the full range of the experience of using our tool.

6.5 Conclusion

This chapter examines a negative research result in search of a new design practice for information scraps and personal information management in general. What went wrong, how is this common to other design difficulties noted in the PIM literature, and what can we do about it?

The final question — what to change — is the elusive one, and the one that is most potentially transformative. Our research response has been to scale down the design problem under consideration, focusing only on common PIM types in information scraps; this decision led to the Pinky prototype introduced in CHAPTER 5. Scaling down makes the research more incremental, but better-tuned to each problem.

7. CONCLUSION AND FUTURE WORK

The genesis of innovation often lies in identifying *breakdowns*: moments when existing tools and designs fail to live up to their promise. By definition, information scraps are such breakdowns: they represent moments when we deliberately choose not to use any of our existing PIM tools. Thus, information scraps reveal the most salient shortcomings in PIM tools today.

I have translated these shortcomings into research directions: lightweight capture, flexible content, flexible organizational strategies, visibility and reminding, and mobility and availability. Jourknow is a test bed for innovations in several of these directions; Pinky focuses only on lightweight capture. Jourknow looks much like PIM tools today; Pinky does not.

In this conclusion, I examine the future of information scrap management, and the ways in which this future may or may not look anything like our PIM tools today.

Lightweight Capture. Pinky represents a first attempt to re-envision the information capture process on the desktop. It is imperfect: it introduces ambiguity and has limited discoverability. How else could we design for fast and accurate capture when the user has limited time and attentional resources? How might these designs change when in a mobile scenario, with nothing but a cellphone?

Flexible Content. Currently, the long tail of personal information has little or no explicit support. How can I manage my college applica-

7. CONCLUSION

tions? My fantasy football team's lineup? My collection of guitar tabs or my personal restaurant review list? The TurtleDove language in Jourknow focused on a method for recording such information, but it proved too complicated for most users. Is it possible to design a generic tool capable of supporting this broad range of information? Of supporting its evolution from broadly scattered e-mails to structured spreadsheet? Of leaving the e-mails scattered but providing spreadsheet-like interfaces to the information?

Flexible Usage and Organizational Strategies. Jourknow attempted to provide multiple management strategies, from a reorderable list interface, text search, and a 2-dimensional canvas layout; it provided facilities ranging from reminding to desktop pinning. However, the result was a largely unfocused design. Instead of broad feature sets, we might instead consider ways in which we can support adaptability. This may require end user hacking — which is, as Bonnie Nardi points out tongue-in-cheek, just a small matter of programming [106].

Visibility and Reminding. Canonical ubiquitous computing demos typically fall into two classes: 1) museum tour guides, and 2) location-aware reminding services. It is this second class of applications that suggest new directions for visibility and reminding services — not just attaching information to physical locations, but attaching annotations to arbitrary windows and documents on our computer, or linking arbitrary personal information in other pieces of personal information.

Mobility and Availability. Mobile scenarios offer unique affordances and constraints. What are common mobile personal information needs, and how can we support them? How can we solve the capture problem when text entry on a keypad is prohibitively slow and other modalities like voice input introduce even more ambiguity into parsing?

The first response to an interest in information scraps seems to be to design a note management tool [5, 9, 17, 18, 22, 24, 86, 131]. However, through this work I have come to believe that it may be more important to direct our efforts toward the basic reasons that such notes exist — addressing the causes rather than mitigating the effects. This more targeted approach may be the real future of information scrap management.

8. BIBLIOGRAPHY

- [1] *Anoto Digital Pen and Paper*. Anoto. <http://www.anoto.com/>
- [2] *ARToolkit*. Hirokazu Kato. <http://www.hitl.washington.edu/artoolkit/>
- [3] *E Ink*. E Ink Corporation. <http://www.eink.com/>
- [4] *Eclipse*. The Eclipse Foundation. <http://www.eclipse.org/>
- [5] *Evernote*. Evernote Corporation. <http://evernote.com/>
- [6] *GNU emacs*. Free Software Foundation, Inc. <http://www.gnu.org/software/emacs/>
- [7] *Google Calendar*. Google. <http://calendar.google.com/>
- [8] *Google Data APIs*. Google. <http://code.google.com/apis/gdata/>
- [9] *Google Notebook*. Google. <http://www.google.com/notebook/>
- [10] io2 Digital Pen. Logitech. http://www.logitech.com/index.cfm/mice_pointers/digital_pen/devices/408&cl=us,en
- [11] *iPhone*. Apple, Inc. <http://www.apple.com/iphone/>
- [12] *I Want Sandy*. values of n, Inc. <http://iwantsandy.com/>
- [13] *Jott*. Jott Networks Inc. <http://www.jott.com/>
- [14] *Jena*. <http://jena.sourceforge.net/>
- [15] *Natural Language Toolkit*. <http://nltk.sourceforge.net/>
- [16] *Lifehacker*. Gawker Media Network. <http://www.lifehacker.com/>
- [17] *OneNote*. Microsoft. <http://office.microsoft.com/onenote/>
- [18] *Post-it Digital Notes*. 3M. http://www.3m.com/us/office/postit/digital/digital_notes.html
- [19] *Protege*. Musen, M. <http://protege.stanford.edu/>
- [20] *Quicksilver*. Blacktree Software. <http://www.blacktree.com>
- [21] *Resource Description Framework (RDF)*. W3C. <http://www.w3.org/>

8. BIBLIOGRAPHY

- RDF/.
- [22] *Stikkit*. values of n, Inc. <http://www.stikkit.com/>
 - [23] *Web Clips*. Apple, inc. <http://www.apple.com/macosx/features/safari.html>
 - [24] *Yojimbo*. Bare Bones Software. <http://www.barebones.com/products/yojimbo/>
 - [25] Allen, D. *Getting Things Done: The Art of Stress-Free Productivity*. Penguin Books, New York, NY, 2001.
 - [26] Barkhuus, L. and Rhode, J.A., From Mice to Men - 24 years of Evaluation in CHI. In Proc. CHI 2007: ACM Conference on Human Factors in Computing Systems. 2007.
 - [27] Barreau, D. and Nardi, B. A. Finding and reminding: file organization from the desktop. *SIGCHI Bulletin*, 27, 3 (1995), 39–43.
 - [28] Barreau, D. K. Context as a factor in personal information management systems. *Journal of the American Society of Information Science (JSIST)*, 46, 5 (1995), 327–339.
 - [29] Bederson, B. B. Interfaces for staying in the flow. *Ubiquity*, 5, 27 (2004), 1-1.
 - [30] Bellotti, V., Dalal, B., Good, N., Flynn, P., Bobrow, D. G. and Ducheneaut, N. What a to-do: studies of task management towards the design of a personal task list manager. In Proc. CHI 2004: ACM Conference on Human Factors in Computing Systems. 2004.
 - [31] Bellotti, V., Ducheneaut, N., Howard, M. and Smith, I. Taking email to task: the design and evaluation of a task management centered email tool. In Proc. CHI 2003: ACM Conference on Human Factors in Computing Systems. 2003.
 - [32] Bellotti, V., Ducheneaut, N., Howard, M., Smith, I. and Grinter, R. E. Quality Versus Quantity: E-Mail-Centric Task Management and Its Relation With Overload. *Human-Computer Interaction*, 20, 1 (2005), 89-138.
 - [33] Bellotti, V. and Smith, I. Informing the design of an information management system with iterative fieldwork. In Proc. DIS 2000: ACM Conference on Designing Interactive Systems. 2000.
 - [34] Bergman, O., Beyth-Marom, R. and Nachmias, R. The user-subjective approach to personal information management systems. *Journal of the American Society for Information Science and Technology (JSIST)*, 54, 9 (2003), 872-878.
 - [35] Bernstein, M., Van Kleek, M., Karger, D.R., and schraefel, mc. Information Scraps: How and Why Information Eludes Our Personal Information Management Tools. To Appear in *ACM Transactions on Information Systems (TOIS)*. 2008.

- [36] Bernstein, M., Van Kleek, M., Karger, D.R., and schraefel, mc. Personal Information Management, Personal Information Retrieval? In *Proc. HCIR 2007: Workshop on Human-Computer Interaction and Information Retrieval*. 2007.
- [37] Bernstein, M., Van Kleek, M., Khushraj, D., Nayak, R., Liu, C., Karger, D.R., and schraefel, mc. Wicked Problems and Gnarly Results: Reflecting on Design and Evaluation Methods for Idiosyncratic Personal Information Management Tasks. *Technical Report MIT-CSAIL-TR-2008-007*, 2008.
- [38] Bernstein, M., Van Kleek, M., schraefel, mc, and Karger, D.R. Evolution and Evaluation of an Information Scrap Manager. In *Proc. PIM 2008: Workshop on Personal Information Management at CHI 2008*. 2008.
- [39] Bernstein, M., Van Kleek, M., schraefel, mc, and Karger, D.R. Management of personal information scraps. In *Proc. CHI 2007: Extended abstracts on Human Factors in Computing Systems*. 2007.
- [40] Blanc-Brude, T. and Scapin, D. L. What do people recall about their documents?: implications for desktop search tools. In *Proc. IUI 2007: International Conference on Intelligent User Interfaces*. 2007.
- [41] Blandford, A. E. and Green, T. R. G. Group and Individual Time Management Tools: What You Get is Not What You Need. *Personal Ubiquitous Computing*, 5, 4 (2001), 213–230.
- [42] Boardman, R. and Sasse, M. A. Stuff goes into the computer and doesn't come out: a cross-tool study of personal information management. In *Proc. CHI 2004: ACM Conference on Human Factors in Computing Systems*. 2004.
- [43] Boardman, R., Spence, R. and Sasse, M. A. Too Many Hierarchies? The Daily Struggle for Control of the Workspace. In *Proc. HCI International*. 2003.
- [44] Bolin, M., Webber, M., Rha, P., Wilson, T., and Miller, R.C. Automation and customization of rendered web pages. In *Proc. UIST 2005: : ACM Symposium on User Interface Software and Technology*. 2005.
- [45] Bowker, G. C. and Star, S. L. *Sorting Things Out: Classification and its Consequence*. The MIT Press, Cambridge, MA, 2000.
- [46] Bruce, H., Jones, W. and Dumais, S. Keeping and Re-finding information on the Web: What do people do and what do they need. In *Proc. of ASIST 2004: The American Society for Information Science & Technology Annual Meeting*. 2004.
- [47] Buchenau, M. and Suri, J.F. Experience prototyping. In *Proc. DIS*

8. BIBLIOGRAPHY

- 2000: *ACM Conference on Designing Interactive Systems*. 2000.
- [48] Butz, A. and Jung, R. Seamless user notification in ambient soundscapes. In *Proc. IUI '05: ACM International Conference on Intelligent User Interfaces*. 2005.
- [49] Buxton, B. *Sketching User Experiences*. Morgan Kaufmann, 2007.
- [50] Campbell, C. and Maglio, P. Supporting notable information in office work. In *Proc. CHI 2003: ACM Conference on Human Factors in Computing Systems*. 2003.
- [51] Victoria Chou. Inky: Internet Keywords with User Feedback. M.Eng. Thesis, Massachusetts Institute of Technology. 2008.
- [52] Connolly, D. RDF Calendar - an application of the Resource Description Framework to iCalendar Data. *W3C Interest Group Note*. 2005. <http://www.w3.org/TR/rdfcal/>
- [53] Csikszentmihalyi, M. *Flow: The Psychology of Optimal Experience*. Harper & Row, New York, NY, 1991.
- [54] Czerwinski, M. and Horvitz, E. An Investigation of Memory for Daily Computing Events. In *Proc. HCI 2002: International Conference on Human-Computer Interaction*. 2002.
- [55] Dahley, A., Wisneski, C. and Ishii, H. Water lamp and pinwheels: ambient projection of digital information into architectural space. In *Proc. CHI 1998: Conference Summary on Human Factors in Computing Systems*. 1998.
- [56] Dai, L., Lutters, W. G. and Bower, C. Why use memo for all?: restructuring mobile applications to support informal note taking. In *Proc. CHI 2005: ACM Conference on Human Factors in Computing Systems*. 2005.
- [57] Darken, R. P. and Sibert, J. L. A toolset for navigation in virtual environments. In *Proc. UIST 1993: ACM Symposium on User Interface Software and Technology*. 1993.
- [58] Dontcheva, M., Drucker, S. M., Salesin, D. and Cohen, M. F. Relations, cards, and search templates: user-guided web data integration and layout. In *Proc. UIST 2007: ACM Symposium on User Interface Software and Technology*. 2007.
- [59] Ducheneaut, N. and Bellotti, V. E-mail as habitat: an exploration of embedded personal information management. *interactions*, 8, 5 (2001), 30-38.
- [60] Dumais, S., 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 *Proc. SIGIR 2003: ACM Conference on Research and Development in Information Retrieval*. 2003.
- [61] Ennals, R. J. and Garofalakis, M. N. MashMaker: mashups for the

- masses. In *Proc. SIGMOD '07: ACM International Conference on Management of Data*. 2007.
- [62] Fertig, S., Freeman, E. and Gelernter, D. Lifestreams: an alternative to the desktop metaphor. In *Proc. CHI 1996: ACM Conference Companion on Human Factors in Computing Systems*. 1996.
- [63] Fisher, D., Brush, A. J., Gleave, E. and Smith, M. A. Revisiting Whittaker & Sidner's email overload ten years later. In *Proc. CSCW '06: ACM Conference on Computer Supported Cooperative Work*. 2006.
- [64] Gemmell, J., Bell, G., Lueder, R., Drucker, S., and Wong, C. MyLifeBits: fulfilling the Memex vision. In *Proc. Multimedia 2002: ACM International Conference on Multimedia*. 2002.
- [65] Gonçalves, D. and Jorge, J. A. Describing documents: what can users tell us? In *Proc. IUI 2004: International Conference on Intelligent User Interfaces*. 2004.
- [66] Gray, W. D. and BoehmDavis, D. A. Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied*, 6, 4 (2000), 322–335.
- [67] Gray, W. D. and Fu, W. Ignoring perfect knowledge in-the-world for imperfect knowledge in-the-head. In *Proc. CHI 2001: ACM Conference on Human Factors in Computing Systems*. 2001.
- [68] Greenberg, S. and Boyle, M. Generating custom notification histories by tracking visual differences between web page visits. In *Proc. GI 2006: Graphics Interface*. 2006.
- [69] Halbert, D.C. SmallStar: programming by demonstration in the desktop metaphor. In *Watch What I Do: Programming by Demonstration*, Cypher, A. ed. MIT Press, Cambridge MA, 1993, 103-123.
- [70] Hayes, G., Pierce, J. S. and Abowd, G. D. Practices for capturing short important thoughts. In *Proc. CHI 2003: ACM Conference on Human Factors in Computing Systems*. 2003.
- [71] Hayes, G.R., Rea, A., Brunette, W., Abowd, G.D., Pierce, J.S., Truong, K.N., and Pering, T. Lightweight Note-Taking Tools Using a Confederation of Mobile Capture and Access Devices. *Workshop on Multi-Device Interfaces for Ubiquitous Peripheral Interaction at Ubicomp 2003*. 2003.
- [72] Hill, W. C., Hollan, J. D., Wroblewski, D. and McCandless, T. Edit wear and read wear. In *Proc. CHI 1992: ACM Conference on Human Factors in Computing Systems*. 1992.
- [73] Hodges, S., Williams, L., Berry, E., Izadi, S., Srinivasan, J., Butler,

8. BIBLIOGRAPHY

- A., Smyth, G., Kapur, N., Wood, K. SenseCam: A retrospective memory aid. In *Proc. Ubicomp 2006: International Conference on Ubiquitous Computing*. 2006.
- [74] Hsieh, G., Wood, K. and Sellen, A. Peripheral display of digital handwritten notes. In *Proc. CHI 2006: ACM Conference on Human Factors in Computing Systems*. 2006.
- [75] Huynh, D. F., Karger, D. R. and Miller, R. C. Exhibit: lightweight structured data publishing. In *Proc. WWW 2007: ACM International Conference on World Wide Web*. 2007.
- [76] Iannella, R. Representing vCard Objects in RDF/XML. *W3C Note*. 2001. <http://www.w3.org/TR/vcard-rdf>
- [77] Ito, M. and Okabe, D. Camera phones changing the definition of picture-worthy. *Japan Media Review*, 08/29/2003.
- [78] Jones, W. Finders, keepers? The Present and Future Perfect in Support of Personal Information Management. *First Monday*, 9, 3.
- [79] Jones, W. Introduction. In *Personal Information Management*, Jones, W., and Teevan, J., Eds. University of Washington Press, Seattle, WA, 3–21.
- [80] Jones, W. and Cronin, B. Personal Information Management. *Annual Review of Information Science and Technology (ARIST)*, 41 (2007), 453-504.
- [81] Jones, W. P. and Dumais, S. T. The spatial metaphor for user interfaces: experimental tests of reference by location versus name. *ACM Transactions on Information Systems (TOIS)*, 4, 1 (1986), 42–63.
- [82] Jones, W., Bruce, H. and Dumais, S. Keeping found things found on the web. In *Proc. CIKM 2001: International Conference on Information and Knowledge Management*. 2001.
- [83] Jones, W., Dumais, S. and Bruce, H. Once found, what then? A study of keeping behaviors in the personal use of Web information. *Proceedings of the American Society for Information Science and Technology*, 39, 1 (January 2005), 391–402.
- [84] Jones, W., Munat, C. and Bruce, H. The Universal Labeler: Plan the Project and Let Your Information Follow. In *Proc. of ASIST 2005: The American Society for Information Science & Technology Annual Meeting*. 2005.
- [85] Jones, W., Phuwanartnurak, A. J., Gill, R. and Bruce, H. Don't take my folders away!: organizing personal information to get things done. In *Proc. CHI 2005: ACM Conference on Human Factors in Computing Systems*. 2005.
- [86] Kalnikaité, V. and Whittaker, S. Software or wetware?: discovering when and why people use digital prosthetic memory. In *Proc. CHI*

- 2007: *ACM Conference on Human Factors in Computing Systems*. 2007.
- [87] Karger, D. R. and Quan, D. Haystack: a user interface for creating, browsing, and organizing arbitrary semistructured information. In *Proc. CHI 2004: ACM Conference on Human Factors in Computing Systems*. 2004.
 - [88] Karger, D. R. Unify Everything: It's All the Same To Me. In Jones, W. and Teevan, J., eds. *Personal Information Management*. University of Washington Press, Seattle, WA, 2007, 127–152.
 - [89] Kelley, D. and Teevan, J. Understanding What Works: Evaluating PIM Tools. In Jones, W. and Teevan, J. eds. *Personal Information Management*. University of Washington Press, Seattle WA, 2007, 190–205.
 - [90] Khan, F. A Survey of Note-Taking Practices. *HPL-93-107*. 1993.
 - [91] Kindberg, T., Spasojevic, M., Fleck, R. and Sellen, A. How and Why People Use Camera Phones. *HPL-2004-216*. 2004.
 - [92] Kirsh, D. and Maglio, P. P. On Distinguishing Epistemic from Pragmatic Action. *Cognitive Science*, 18, 4 (1994), 513-549.
 - [93] LaMarca, A., Chawathe, Y., et al. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Proc. Pervasive 2005*. 2005.
 - [94] Lanzenberger, M. and Sampson, J. ALViz - A Tool for Visual Ontology Alignment. In *Proc. IV 2006: International Conference on Information Visualisation*. 2006.
 - [95] Lamming, M., Brown, P., Carter, K., Eldridge, M., Flynn, M., Louie, G., Robinson, P. and Sellen, A. The design of a human memory prosthesis. *The Computer Journal*, 37, 3 (1994), 153–163.
 - [96] Lansdale M. W. The Psychology of Personal Information Management. *Applied Ergonomics*, 19, 1 (1988), 55–66.
 - [97] Lasila, O. Programming Semantic Web Applications: A Synthesis of Knowledge Representation and Semi-Structured Data. PhD Dissertation, Helsinki Institute of Technology. 2007.
 - [98] Lichtenstein, S., Fischhoff, B. and Phillips, L. D. Calibration of probabilities: The state of the art to 1980. *Judgment under uncertainty: Heuristics and biases*, (1982), 306–334.
 - [99] Lin, M., Lutters, W. G. and Kim, T. S. Understanding the micronote lifecycle: improving mobile support for informal note taking. In *Proc. CHI 2004: ACM Conference on Human Factors in Computing Systems*. 2004.
 - [100] Little, G., Lau, T.A., Cypher, A., Lin, J., Haber, E.M., and Kandogan, E. Koala: Capture, Share, Automate, Personalize Business Processes on the Web. In *Proc. CHI 2007: ACM Conference on Human Factors in Computing Systems*. 2007.
 - [101] Little, G. and Miller, R. C. Translating keyword commands into ex-

8. BIBLIOGRAPHY

- ecutable code. In *Proc. UIST 2006: ACM Symposium on User Interface Software and Technology*. 2006.
- [102] Malone, T. W. How do people organize their desks?: Implications for the design of office information systems. *ACM Transactions on Information Systems (TOIS)*, 1, 1 (1983), 99–112.
- [103] Mander, R., Salomon, G. and Wong, Y. Y. A “pile” metaphor for supporting casual organization of information. In *Proc. CHI 1992: Conference on Human Factors in Computing Systems*. 1992.
- [104] McAlpine, H., Hicks, B. J., Huet, G. and Culley, S. J. An investigation into the use and content of the engineer’s logbook. *Design Studies*, 27 (2006), 481–504.
- [105] McGrath, J.E. Methodology matters: doing research in the behavioral and social sciences. In Baecker, R., Grudin, J., Buxton, B., and Greenberg, S., eds. *Human-computer interaction: toward the year 2000*. Morgan Kaufman, San Francisco VA, 1995, 152–169.
- [106] Nardi, B. *A small matter of programming: perspectives on end user computing*. MIT Press, Cambridge MA, 1993.
- [107] O’Day, V. L. and Jeffries, R. Orienteering in an information landscape: how information seekers get from here to there. In *Proc. CHI 1993: ACM Conference on Human Factors in Computing Systems*. 1993.
- [108] Oulasvirta, A. and Sumari, L. Mobile kits and laptop trays: managing multiple devices in mobile information work. In *Proc. CHI 2007: ACM Conference on Human Factors in Computing Systems*. 2007.
- [109] Perlin, K. and Fox, D. Pad: an alternative approach to the computer interface. In *Proc. SIGGRAPH 1993: Computer graphics and interactive techniques*. 1993.
- [110] Pinhanez, C. S. The Everywhere Displays Projector: A Device to Create Ubiquitous Graphical Interfaces. In *Proc. UbiComp 2001: International Conference on Ubiquitous Computing*. 2001.
- [111] Plaisant, C., The challenge of information visualization evaluation. In *Proc. AVI 2004: ACM International Working Conference on Advanced Visual Interfaces*. 2004.
- [112] Popescu, A., Armanasu, A., Etzioni, O., Ko, D. and Yates, A. Modern natural language interfaces to databases: composing statistical parsing with semantic tractability. In *Proc. COLING 2004: ACL International Conference on Computational Linguistics*. 2004.
- [113] Rettig, M. Prototyping for tiny fingers. *Communications of the ACM*, 37, 4 (1994). 21-27.
- [114] Rhodes, B. The wearable remembrance agent: A system for augmented memory. *Personal Ubiquitous Computing* 1, 4 (1997), 218-224.
- [115] Rittel, H.W.J. and Webber, M.M. Dilemmas in a general theory of

- planning. *Policy Sciences*, 4 (2). 155-169. 1973.
- [116] Robertson, G., Czerwinski, M., Larson, K., Robbins, D. C., Thiel, D. and Dantzich, M. V. Data mountain: using spatial memory for document management. In *Proc. UIST 1998: ACM Symposium on User Interface Software and Technology*. 1998.
- [117] Rodden, K. and Wood, K.R., How do people manage their digital photographs? In *Proc. CHI 2003: ACM Conference on Human Factors in Computing Systems*. 2003.
- [118] Ross L. and Nisbett R. *The Person and the Situation: Perspectives of Social Psychology*. Temple University Press, 1991.
- [119] Salton, G. and Buckley, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24, 5. 513–523.
- [120] schraefel, m. c., Hughes, G. V., Mills, H. R., Smith, G., Payne, T. R. and Frey, J. Breaking the book: translating the chemistry lab book into a pervasive computing lab environment. In *Proc. CHI 2004: ACM Conference on Human Factors in Computing Systems*. 2004.
- [121] schraefel, m.c., Zhu, Y., Modjeska, D., Wigdor, D. and Zhao, S., Hunter gatherer: interaction support for the creation and management of within-web-page collections. In *Proc. WWW 2002: ACM International World Wide Web Conference*. 2002.
- [122] Sellen, A. J., Fogg, A., Aitken, M., Hodges, S., Rother, C. and Wood, K. Do life-logging technologies support memory for the past?: an experimental study using sensecam. In *Proc. CHI 2007: ACM Conference on Human Factors in Computing Systems*. 2007.
- [123] Sellen, A. J. and Harper, R. H. R. *The Myth of the Paperless Office*. MIT Press, Cambridge, MA, USA, 2003.
- [124] Smith, G., Baudisch, P., Robertson, G., Czerwinski, M. and Meyers, B. GroupBar: The TaskBar Evolved. In *Proc. OZCHI: Australasian Computer-Human Interaction Conference*. 2003.
- [125] Starner, T., Snoeck, C., Wong, B., and McGuire, R. Use of mobile appointment scheduling devices. In *Proc. CHI 2004: ACM Conference on Human Factors in Computing Systems*. 2004.
- [126] Stifelman, L., Arons, B. and Schmandt, C. The audio notebook: paper and pen interaction with structured speech. In *Proc. CHI 2001: ACM Conference on Human Factors in Computing Systems*. 2001.
- [127] Tan, D.S., Meyers, B., and Czerwinski, M. WinCuts: manipulating arbitrary window regions for more effective use of screen space. In *Proc. CHI 2004: ACM Conference on Human Factors in Computing Systems*. 2004.
- [128] Teevan, J., Alvarado, C., Ackerman, M. S. and Karger, D. R. The perfect search engine is not enough: a study of orienteering behavior in

8. BIBLIOGRAPHY

- directed search. In *Proc. CHI 2004: ACM Conference on Human Factors in Computing Systems*. 2004.
- [129] Tohidi, M., Buxton, W., Baecker, R., and Sellen, A. Getting the right design and the design right. *Proc. CHI 2006: ACM Conference on Human Factors in Computing Systems*. 2006.
- [130] Van Kleek, M., Bernstein, M., André, P., Pertunnen, M., Karger, D.R., and schraefel, mc. Simplifying Knowledge Creation and Access for End-Users on the Semantic Web. In *Proc. SWUI 2008: Workshop on Semantic Web User Interaction at CHI 2008*. 2008.
- [131] Van Kleek, M., Bernstein, M., Karger, D. R. and schraefel, mc. GUI — Phooey!: The case for text input. In *Proc. UIST 2007: ACM Symposium on User Interface Software and Technology*. 2007.
- [132] Van Kleek, M. and Shrobe, H. A Practical Activity Capture Framework for Personal, Lifetime User Modeling. In *Proc. UM 2007: User Modeling*. Corfu, Greece, 2007.
- [133] Venolia, G. D., Dabbish, L. A., Cadiz, J. J. and Gupta, A. Supporting Email Workflow. *MSR-TR-2001-88*. 2001.
- [134] Wagenaar, W. A. My memory: a study of autobiographical memory over six years. *Cognitive psychology*, 18, 2 (1986), 225–252.
- [135] Whittaker, S. and Hirschberg, J. The character, value, and management of personal paper archives. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 8, 2 (2001), 150–170.
- [136] Whittaker, S. and Sidner, C. Email overload: exploring personal information management of email. In *Proc. CHI 1996: ACM Conference on Human Factors in Computing Systems*. 1996.
- [137] Whittaker, S., Terveen, L., and Nardi, B.A. Let's Stop Pushing the Envelope and Start Addressing It: A Reference Task Agenda for HCI. *Human-Computer Interaction*, 15, 2 (2000), 75–106.
- [138] Yates, F. A. *The Art of Memory*. University of Chicago Press, Chicago, 1966.
- [139] Yee, K.P., Swearingen, K., Li, K., and Hearst, M. Faceted metadata for image search and browsing. In *Proc. CHI 2003: ACM Conference on Human Factors in Computing Systems*. 2003.
- [140] Yeh, R., Liao, C., Klemmer, S., Guimbretière, F., Lee, B., Kakaradov, B., Stamberger, J. and Paepcke, A. ButterflyNet: a mobile capture and access system for field biology research. In *Proc. CHI 2006: ACM Conference on Human Factors in Computing Systems*. 2006.