Visualizing and Analyzing Human-Centered Data Streams

by

Michel Joseph Lambert

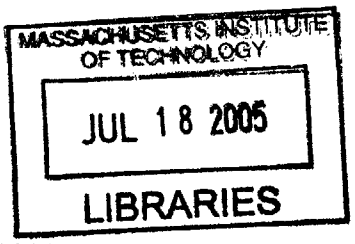
Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2005 [June 2005]



Author .....
Dep[illegible]omputer Science
May 19, 2005

Certified by..
.....
[illegible]andy) Pentland
Profe[illegible] of Electrical Engineering and Computer Science
isor

Accepted by.....
....
[illegible]nith
Chairman, Department Committee on Graduate Students

Visualizing and Analyzing Human-Centered Data Streams

by

Michel Joseph Lambert

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2005, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Computer Science and Engineering

Abstract

The mainstream population is readily adapting to the notion that the carrying of mobile computational devices such as cell phones and PDAs on one's person is as essential as taking along one's watch or credit cards. In addition to their stated and oftentimes proprietary functionality, these technological innovations have the potential to also function as powerful sensory data collectors. These devices are able to record and store a variety of data about their owner's everyday activities, a new development that may significantly impact the way we recall information. Human memory, with its limitations and subjective recall of events, may now be supplemented by the latent potential of these in-place devices to accurately record one's daily activities, thereby giving us access to a wealth of information about our own lives.

In order to make use of this recorded information, it must be presented in an easily understood format: timelines have been a traditional display metaphor for this type of data. This thesis explores the visualization and navigation schemes available for these large temporal data sets, and the types of analyzation that they facilitate.

Thesis Supervisor: Alex (Sandy) Pentland
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First, I would like to thank my advisor, Sandy Pentland, for his patience and support in helping me during my graduate year. He has been a great source of ideas and a vision for my work, and I am grateful for his feedback on work. I am thankful for the opportunity to work with the Human Dynamics group at the MIT Media Lab.

I would also like to express my gratitude toward my research supervisor, Nathan Eagle, who met with me regularly about the project, providing feedback, suggestions, and direction, all despite having his own PhD thesis work to complete at the same time. I am grateful for his reasoned approach to everything. He managed to be realistic in his expectations of my work, while at the same time providing a motivating vision of what was possible, pushing me to excel.

Of course, I want to thank God for his help in getting me through my graduate year. I can always rely on him to make sure things work out for the best, regardless of how terrible things may have seemed.

I would like to thank my family, for giving me the support and encouragement needed to make it to, and through, MIT. They supported my hard decision to change research advisors midway through, despite the hardships in funding that I faced.

I would also like to share my gratitude with my brother, for his incessant fixation on cell phones and mobile technology. Despite my skepticism for much of his purchases, enough of his enthusiasm must have rubbed off on me to interest me in this area of context-sensitive mobile computing.

I am grateful to those who helped me write my thesis: Clyde Law and Lee Lin for our discussions about thesis writing, Vito Miliano for our lengthy discussions about wearable and ubiquitous computing, and Jessica Young for her semi-infectious love of writing. I am eternally indebted to those who helped proofread the drafts of my thesis, including my parents, my sister, and Dick Dowdell.

I would also like to thank everyone that kept me sane all these years: The wrestling team for providing an outlet for the pressure of MIT, J-entry for their constant source of wacky humor and fun times, and both I-entry's for the amazing energy and activity.

Contents

List of Figures

Chapter 1

Introduction

The human mind is a fickle thing. Its processing capabilities are limited, it does not remember everything, and what it does remember can be changed, forgotten, or otherwise modified without leaving a trace. If the human mind attempted to process all sensory inputs, it would not be able to focus or concentrate, and the information overload would be devastating. Instead, the human mind filters out what it deems unimportant. A person is likely to remember the ordering of cars at a race, while forgetting the details of cars seen in everyday life. The benefit of recording information we deem important is obvious, and functions as a filter to avoid remembering unimportant details. Unfortunately, many times, details that we deem unimportant later turn out to be relevant to our lives, and our selective memories are of no help in recalling those details.

Fortunately, computers can complement the human mind quite well. They excel at computational tasks, have near perfect memories, and the amount of storage they can utilize is constantly increasing. Computer storage is increasing at a geometric rate, much faster than the rate of increase of the human lifespan. Recording 10Hz 640x480 video for 100 years would utilize about 180TB [Cla02]. If current trends continue, by 2015, 180TB will cost about $200 and occupy a few cubic inches [GH03]. Given that current portable MP3 players can cost more, and occupy roughly the same amount of space, it is not too difficult to imagine a human of 2015 carrying around a portable device that is capable of recording their entire life.

1.1 Motivation

However, recording the data is only half the battle: a usable interface for accessing the recorded data is equally important. After all, the human mind's selective memory allows it to avoid the clutter of irrelevant details when retrieving memories. Because computers of the future will be able to store *everything* about our lives, the ability to perform meaningful queries and navigate through this enormous mine of raw data will become essential. Time will obviously be one important dimension of navigation, as will traditional searching approaches. But what other types of questions will people want to ask of these computerized memories? The types of questions we can answer are limited only by the quality and information we can record.

The interface, and its power to enable users to extract meaningful data, will be the major factor limiting the usefulness of this recorded data. Questions that ask who, what, where, or when can often be satisfied by simple SQL database queries. Whether it is natural language, a mini-language similar to SQL, or a more interactive hypertext-based query interface that acts as a form of query constructor, there are many questions that can be asked:

- When did I last eat lunch with Dave?
- How many times have I been to a Red Sox game?
- Where did I eat dinner when my cousin came to visit?
- ...and so on.

But there are many forms of more complicated queries that can be asked, that instead of being answerable by a simple SQL query, rely on the processing of collected data to generate less discrete, more fuzzy results. Asking questions about life's patterns and trends, questions such as:

- How regular are my daily cycles and sleep patterns?
- What percentage of the time do I go in to work on weekends?

- Which of my friends do I hang out with most?
- Which restaurant do I prefer most for lunch?
- ...and other similar analytical questions

Some of these questions don't have a simple answer: The regularity of one's circadian rhythm does not have a single, simple answer. These types of questions require a more visual answer, an answer that allows the user to *see* the various gradations of regularity, allowing the user to answer his own question by just analyzing the visual display.

1.2 Background

The Reality Mining group at the MIT Media Lab has been performing an experiment involving roughly 90 people. Each person is running software on their Symbian Series 60 phone that records a variety of data about that person's use of their cell phone. Written by the University of Helsinki, the Context Log software records a variety of information about each participant's phone usage, including timestamped logs for all of the following[UoH05]:

- phone applications used
- plugging or unplugging from a charger
- nearby phones and computers detected from periodic Bluetooth scans
- current cell tower, and cell tower switches
- phone call information, including phone number called, call duration, and incoming/outgoing status
- screensaver activity (is the phone idle or not)

To date, it has recorded roughly 1GB of data, spanning a total of 350,000 hours, or around 40 years of continuous data across all participants in the subject [Eag05b]. For

more details on the software, the problems encountered in recording and retrieving data, please see [Eag05a].

The phone software is able to record a great deal of data without interrupting the phone's owner. The one exception is information about the phone's current location. While the phone software is able to determine the current cell phone tower, the information is in an unhelpful form akin to "24127, 2421, AT&T Wirel". With these strings, the software knows neither the precise location in terms of latitude or longitude, nor the contextual location of the owner: home, work, airport, and so on. To achieve something useful, the phone software prompts the cell phone owner every time it encounters a previously-unseen cell tower, asking the owner to name the current location, in a general sense. Through this, the phone is able to create a mapping between cell towers and location information in a form that is, in theory, more relevant to the cell phone owner.

With time, the sheer volume of collected text logs will become quite daunting for a participant. A proper interface is critical to extracting any value from the recorded data. This thesis examines a variety of interface approaches, comparing existing approaches to the specific needs of our application utilizing the cell phone data. A major focus will be that of the Diary, an application I developed to provide an interface to the cell phone owner's collected data. The Oxford English Dictionary[oed05] defines "diary" as:

1. A daily record of events or transactions, a journal; specifically, a daily record of matters affecting the writer personally, or which come under his personal observation.

2. A book prepared for keeping a daily record, or having spaces with printed dates for daily memoranda and jottings; also, applied to calendars containing daily memoranda on matters of importance to people generally, or to members of a particular profession, occupation, or pursuit.

Our Diary application has similar aims to a hand-kept diary, providing a "daily record of matters affecting the person" as well as a "daily record of [cell phone]

events." The Diary application should provide a means for a user to reflect on his recorded diary, even facilitating some of the functions of a traditional diary itself, supporting hand-entered descriptions of activities performed.

1.3 Outline

Chapter 2 provides an overview of the related work and discusses the various existing applications available for diary-like services. Chapter 3 details the design choices made in creating the Diary application. Chapter 4 describes the final Diary application and the functionality it provides, while Chapter 5 relates the implementation of the Diary application. Chapter 6 considers possibilities for future work, and Chapter 7 concludes.

Chapter 2

Related Work

In 1945, Vannevar Bush wrote an article, "As We May Think," describing his vision of the future of information management [Bus45]. Bush envisioned a person of the future using a recording device attached to his forehead, snapping pictures throughout the day whenever he would find something interesting. That person could transcribe any thoughts by speaking into a recording device. Employing analogies to the computers of his day and extrapolating forward, he imagined a device called a *memex* that would allow a person to easily navigate data by association (hyperlinks), searching the memex contents, and adding comments in the margin of documents as thoughts came to him. The device would support the creation of associative trails that connected documents together as they were traversed, allowing the user to recreate or share his train of thought. Similarly, encyclopedias could be created with various associative trails already established, enabling later researchers to "view" the thought processes of their predecessors.

Aside from many striking similarities to the hyperlinked web of today's internet, Bush also described an automated assistant that recorded a person's activity and facilitated search, navigation, and reflection after the fact. Bush's vision is reflected in many of today's research projects attempting to build variations on the memex concept.

Researchers building projects to capture information generally tend to focus on the participant's actions in one of two domains. The first group focuses on following a

participant throughout the day as he goes about his normal activities, recording continuous data about their actions. The second group is more focused on a participant's actions on a computer. This dichotomy is likely due to the difficulty of following a participant around with sufficient recording abilities. The advent of PDAs and cell phones makes this possibility more accessible, but restrictions on battery usage still limit the extent of data recording that is possible. Computers, on the other hand, are limited only by the software side, thus many software developers have focused their efforts to this end, because this solution does not require the participant to purchase new hardware. In addition, with the greater role computers play in a person's personal and professional life, the details and history of that person's computer usage become more relevant.

2.1 Recording Normal Activity

2.1.1 BodyMedia

BodyMedia is based on an arm-band device worn by the participant, recording a variety of metrics about the wearer. It uploads data to a PC periodically, allowing the participant to view historical graphs of their skin temperature, step counter, acceleration (longitudinal, transverse), heart flux, and galvanic skin response [LWS+]. The software can analyze these inputs to deduce information about the current activity and caloric expenditure, among other things [KSSF03].

2.1.2 I Sensed

The *I Sensed* project, developed by Brian Clarkson at the MIT Media Lab, utilizes a backpack, augmented with cameras, a microphone, and a 3D orientation sensor. Continuously recording both audio and video feeds from the participant's life, it recorded 5GB of data per day, and provided a visualization of the data from a PC. Utilizing a horizontal timeline to present information, Figure 2-1 shows the presentation of images from the front and rear video streams, an audio spectrogram, orientation

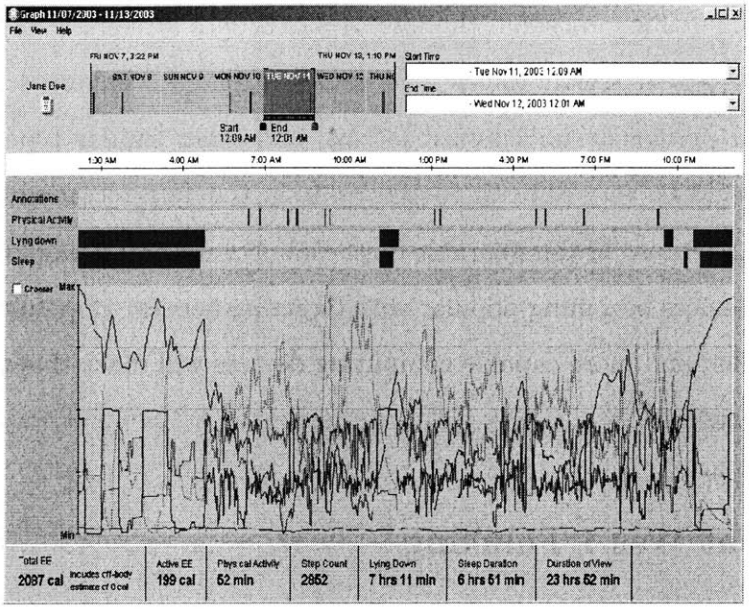


Figure 2-1: BodyMedia Screenshot

data, as well as labeled data indicating what the participant was doing at each point in time. Clarkson developed successfully developed classification algorithms to deduce the wearer's current position and/or activity by comparing data to other similar instances of recorded data [Cla02] .

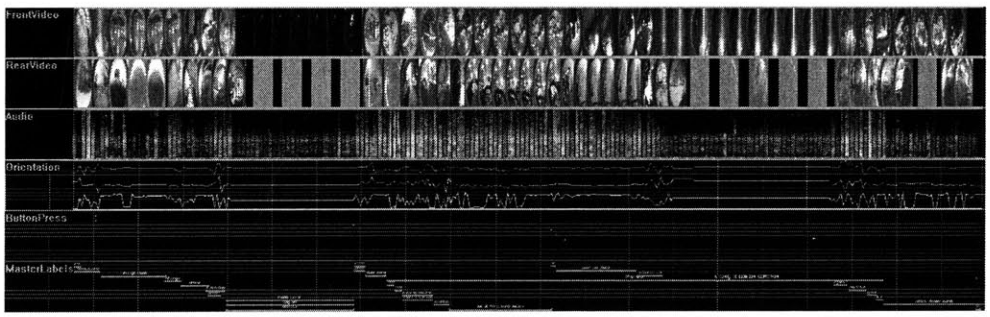


Figure 2-2: *I Sensed* Screenshot

One limitation of this project's approach lies with the raw nature of the data it is recording. The video and audio streams are useful, but the lack of any meaningful information makes it difficult to search for particular events or keywords, limiting the interface to browsing and navigating by time. While the location data added by

classification algorithms is useful, the unstructured data streams make it difficult to extract other types of information. The names of people conversed with, location information for previously unclassified locales, and other similar types of high-level information would be very difficult to extract from the data recorded by *I Sensed.*

In addition, because of the wearable device's bulky nature, it would be difficult to imagine such devices becoming popular with larger audiences. Hopefully the industry trends for smaller and more capable computing devices will make this approach more feasible in the future.

2.1.3 What Was I Thinking

The *What Was I Thinking* project, developed by Sunil Vemuri for the Electronic Publishing group at the MIT Media Lab, explores the recorded audio stream in more detail. Utilizing an iPaq with a microphone, the system applied voice recognition software to the continuously recorded audio streams, creating an easily searchable text archive. The PC portion of the system presented conversation transcripts, with brighter words corresponding to greater confidence in the word's transcription accuracy [Vem05]. The PC application supported a few different searching approaches, even allowing one to search on similar-sounding words, as Figure 2-3 demonstrates. Finally, because of people's ability to recall events relative to other memorable events, it supported the searching of news headlines, letting one search for a conversation that occurred "just after the SuperBowl" or other noteworthy event [Edi04].

2.1.4 Forget-me-not

Forget-me-not, a project by Mike Lamming and Mike Flynn at the Rank Xerox Research Centre, was developed with different goals in mind. Instead of collecting data on a portable device for navigation and searching on a desktop computer, they targeted their software to run directly on the PDA, creating an iconic language to represent actions and the people involved, to maximize the use of screen real estate. In addition, they augmented their workplace to be able to observe higher-level se-

Figure 2-3: *What Was I Thinking* Screenshot

mantic events, such as the transmission of documents from one coworker to another, the participants in email and phone conversations, and even information regarding the particular room in which a certain activity takes place [LF94b].

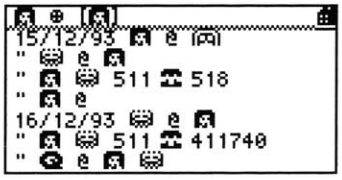


Figure 2-4: *Forget-me-not* Screenshot

By augmenting the workplace with devices to passively record all events that take place, the system is able to provide the users with a view of history that is more closely aligned with the way they think. In order to locate a document, the user is able to utilize memorable items/events related to "where," "when," and "how" the

user received it. As demonstrated in [LF94a], when asked for Richard's sketch, Mike is able to recall Richard giving it to him in the conference room, and is able to filter his event listings, narrowing the amount of data to be searched. The next step would be to assign the sketch as the subject, thereby enabling him to view the history of the sketch and all related events. Finally, Mike is able to print the sketch out to satisfy the original request.

While the system is very powerful, especially for a 1994 implementation, it requires an augmentation of the user's workplace to facilitate the recording of these higher-level events. Phone systems must log all phone calls and make them available to the system, and the workplace must have its rooms outfitted with the ability to identify themselves to the PDAs. While *Forget-me-not* is an significant demonstration, it places an undue burden on the environment itself. An additional liability of this system would manifest itself in the following scenario. If information begins to leak from the closed system, and a document transfer is not recorded within the bounds of the system, a user relying on it may find that the system's recording of inaccurate data would be worse than no data at all.

2.1.5 Blogger Mobile

Launched in May 2005, Blogger Mobile supports a different type of daily recording. Instead of a continuous record of activity, Blogger Mobile operates on-demand. When a cell phone user determines that some event or item is worth remembering, the user is able to take a photo and write a short snippet of text on the phone, and email it to a special email address. Google then publishes the text and image as an entry on a digital journal or diary. Marketed as a blog that can be updated on-the-road with a cell phone, this product provides the ability to caption your day with photos and text [Blo05].

At first glance, Blogger Mobile may appear to be out of place in this survey of classical attempts to record personal activities. It does not support any form of continuous recording, merely recording snapshots of time that the user finds interesting, and has the opportunity to capture. Often times, however, people want to recall

things that seemed important only in hindsight, and Blogger Mobile is unable to help recall that kind of information. The snapshot nature of the recording device also leaves many gaps in the blog's "memory", making it unreliable in determining what one was doing at a given date or time. Additionally, one may have forgotten to record anything at all.

However, Blogger Mobile is worth noting because of its ubiquitous availability. Requiring only a cell phone with a SMS text messaging capability (and optional camera), and each posting costing the price of an SMS message, it has the potential to reach and attract a large population, something the other systems can not hope to achieve in the near future. The system's on-demand nature that only records information explicitly requested by the user allows it to sidestep issues of privacy, secrets, or other information that people may not want recorded.

2.1.6 Nokia Lifeblog

Nokia's Lifeblog, like Blogger Mobile, attempts to record the cell phone user's life on-demand. It records all photos and images taken, as well as any messages sent or received, storing the time and place the data was recorded, based on the current cell tower. Lifeeblog stores the information on the cell phone itself, thereby avoiding the cost of SMS messages. The user is required to periodically download to a PC in order to avoid filling up the internal memory of the user's phone [Nok05]. Lifeblog creates a timeline of the recorded information, viewable from both the PC and phone. Aside from the delay before information can be made public, it offers a superset of the features found in Blogger Mobile. One drawback of this solution is the requirement to use a newer version of Nokia's Series 60 phones [War04].

2.2 Recording Computer Activity

2.2.1 Classifications for computer-based recording devices

Many systems prefer to reside on the computer, analyzing the participant's computing habits. With the increased frequency of interaction between computers and employee's lives, this turns out to be a remarkably efficient avenue for analyzing one's life and has the added benefit of providing higher-level semantic actions. Instead of relying upon error prone audio and video recognition or classification algorithms, the software is able to process what the user types into the computer. Similarly, there is a plethora of information available through application and system APIs. For example, information such as windows opened by the user, websites visited, email utilization, and other such computing tasks, are available for computer resident systems.

Numerous applications include temporal navigation features as a means of accessing their data. Many email applications permit the user to sort or filter by time, allowing the user to see what particular conversations he was participating in at the time. Even Google's GMail (a web-based email client) allows you to perform date-range-based searches on email, i.e. looking for email that meet certain search criteria and lie within a certain span of time. Many other applications (including Windows Explorer and a web browser's History listing) include primitive temporal sorting mechanisms that order a list of items, allowing one to backtrack and find recently modified files or visited websites.

2.2.2 Desktop Search

One major area that has been receiving recent widespread attention is that of Desktop Search. As part of the battle for people's attention occurring between Google, Yahoo, and Microsoft, all three companies have released desktop search engines aimed at assisting users in finding all the information stored on their computers. Assuming that the user never permanently deletes any files, this provides a rudimentary search of that user's history in computing activities. Unfortunately, these desktop search offerings

provide little in the way of date- or time-based navigation features, providing only a simple keyword-based search engine. For example, while Google Desktop Search indexes the modified times for files, email send/receive times, and even the times one visited a website, it does not allow the user to search on this stored data.

2.2.3 Time-Machine Computing

Jun Rekimoto's project at Sony Computer Science Laboratories takes a rather unusual approach to storing and searching a computer's history. The *Time-Machine Computer* project changes the nature of the computer desktop, replacing it with an application called "TimeScape." The TimeScape operates like a normal desktop, allowing the user to save items on the screen, but differs in that the desktop has a third dimension of time that can be navigated. If you use your desktop to store what you are currently working on, items of note, and so on, TimeScape then allows you to rewind your computer to see what you were occupied with at any point in time, even rewinding the contents of documents to allow you to read older versions. To avoid getting over-cluttered, items on the TimeScape desktop fade out over time with disuse, automatically removing old items from the current desktop as they become irrelevant. The time navigation features of TimeScape are impressive, providing calendar views, timeline views, and even filtering based on keywords. In Figure 2-5, some of the items on the desktop are fading out from being recently removed, others appear as small messages akin to Sticky Notes. Note the interface located across the top which provides a means of searching as well as navigating through time.

2.2.4 Stuff I've Seen

Microsoft's *Stuff I've Seen* project, a precursor to MSN Desktop Search, contributed many ideas to its successor. *Stuff I've Seen* indexed the user's data in a variety of formats, such as email, web pages, documents, calendar appointments, etc. The team experimented with a variety of interfaces for navigating and browsing the collected data, permitting the user to filter on a variety of criteria. See Figure 2-6 [DCC+03].

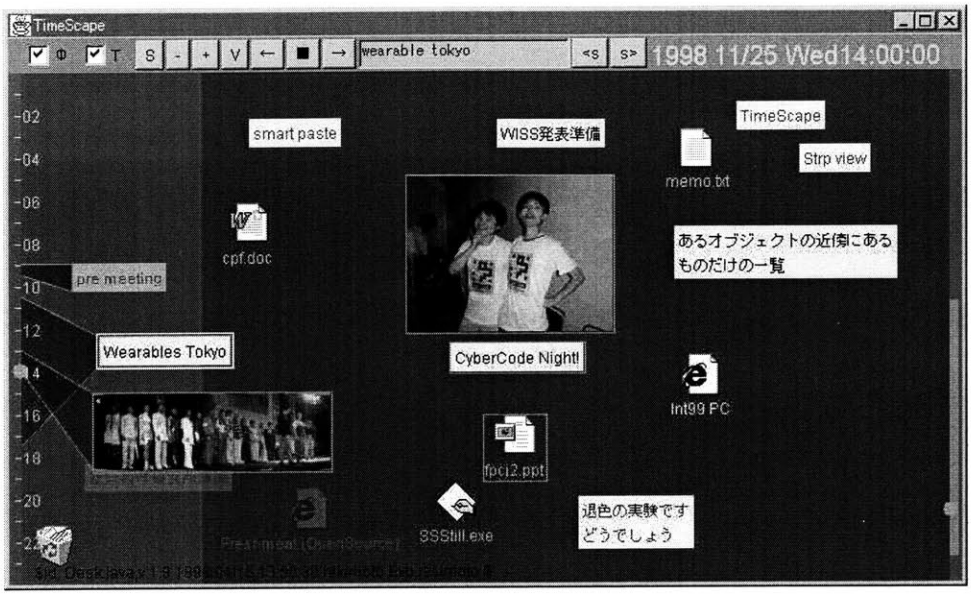


Figure 2-5: *TimeScape* Screenshot

Its indexing of calendar appointments allowed it to find and return results from future calender entries as well.

The project also experimented with more useful ways of presenting displaying history. In [RCDH03], Meredith Ringel found that adding landmarks to a timeline display helped people effectively navigate to find the documents they were interested in. Similar in concept to *Forget-me-not*'s use of news headlines, temporal landmarks like holidays, news headlines, and time-stamped digital photographs were shown to help jog the user's memory, significantly reducing the time taken to locate a particular event. A screenshot of the interface can be found in Figure 2-7 [RCDH03].

2.2.5 MyLifeBits

Microsoft's *MyLifeBits* is another project at Microsoft Research that attempts to directly realize Bush's vision of a memex [GBL+02]. Gordon Bell, one of the researchers, had the events of his entire life scanned in for the project. This included articles, books, photos, home movies, and all other media relevant to his life. To facilitate this task, the team developed software to support annotation, indexing, querying,

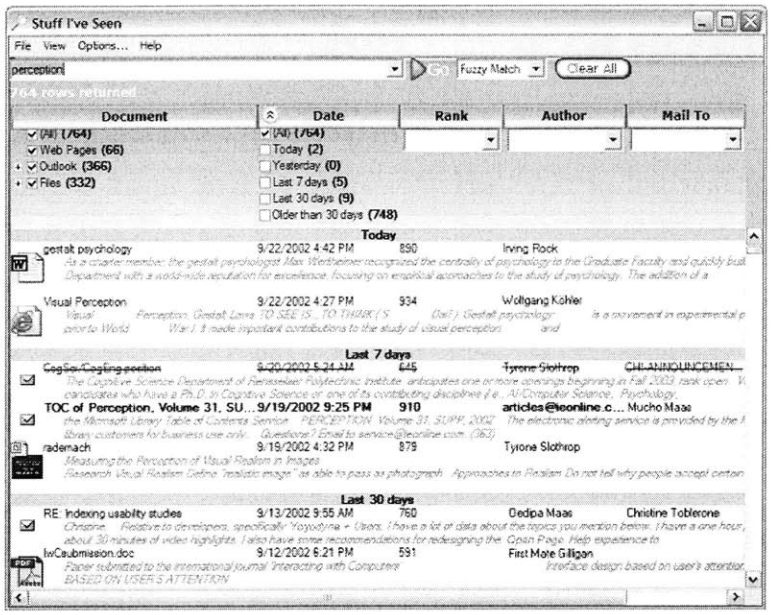


Figure 2-6: *Stuff I've Seen* Screenshot

and navigation by hyperlinks. The software also records a variety of user activities, including visited web pages, IM transcripts, as well as logging data regarding listening and viewing habits for radio and television.

There are a variety of navigation and viewing methods available for photos in *MyLifeBits*. Figure 2-8 demonstrates a timeline view, clustered-time view, map view, and even an interactive story view. These views offer the user an assortment of methods for navigating the user's digital archives. Users are able to retrace their steps on a journey to find a picture taken at a particular time (timeline view or clustered-time view) or location (map view), or search for user created stories written through the use of the *MyLifeBits* software itself (story view).

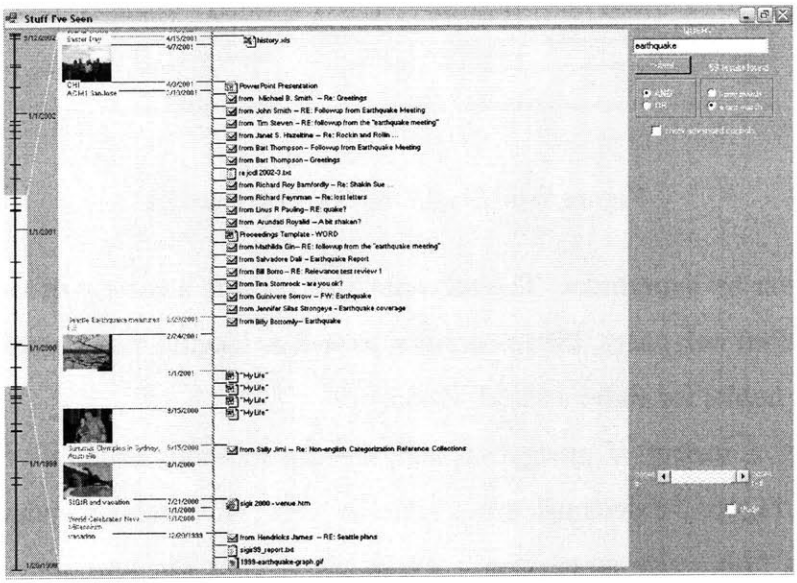
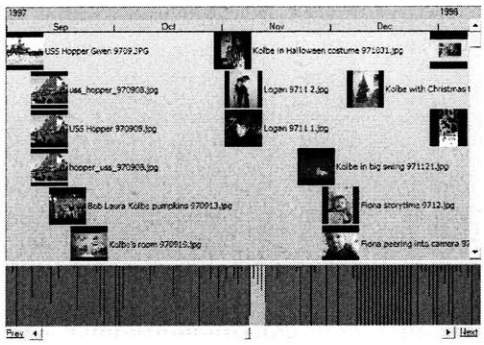
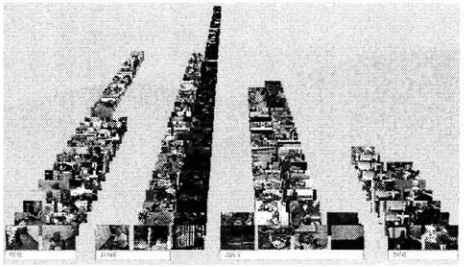


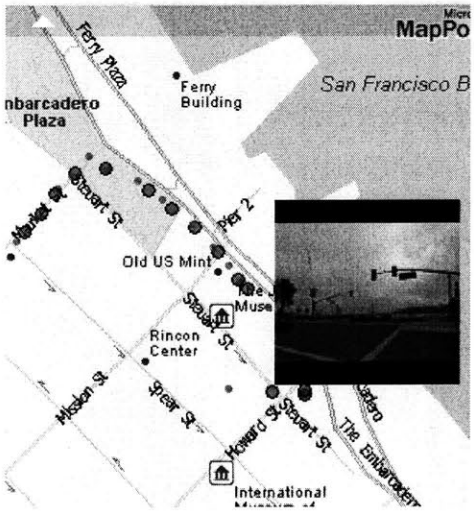
Figure 2-7: *Stuff I've Seen* Milestones Interface Screenshot



(a) Timeline View



(b) Clustered-Time View



(c) Map View

(d) Story View

Figure 2-8: *MyLifeBits* Photo Views

Chapter 3

Design Choices

In deciding how to provide an interface for Diary, there were a number of considerations to be made. Some of the more important design requirements included:

- Ease of use. Downloading and installing an application requires a lot of trust and faith in the application provider. It also required a certain amount of work that people may not have been willing to do for a non-essential application.

- Cross-platform. The participants were using a variety of operating systems, including Windows, Mac OS X, and Linux variants. Any developed application would need to run on all of these platforms.

- Appealing interface. While a purely utilitarian design may work for a personal project, an application's appearance must be compelling when advertising a project to others.

- Speed. Since the data is centrally-hosted, it is important that the display of data occurs in a reasonably efficient manner. Bottlenecks can occur in either CPU or network transfer that could result in unacceptable latencies.

- Extensibility. A powerful application is severely limited if it's programming environment does not facilitate modifications.

3.1 Data Storage Backend

The log files created by the Context Log software are stored as simple ASCII timestamped text files. The breakdown and format for the log files can be found in Appendix A. In preliminary prototyping, it was observed that providing an interface using these text files as the backend data repository provided satisfactory performance with a few days worth of data. Unfortunately, with one subject's data requiring more than 40MB of storage for log files, it is difficult to imagine processing the entire dataset in under a few seconds, a requisite for useful graphical displays.

Given the limitations of the input text format, a better format was needed: one that could filter and analyze data, returning data which met certain time or type criteria. A properly designed SQL database was more than capable of handling these needs. Details on the chosen SQL table schema can be found in Section 5.1.

3.2 Data Querying and Processing Approaches

Because of the constraints on the Diary data, the system needed to be centrally located. Because the participants did not have the technology necessary to collect their data on their own computers, all data collection was either accomplished through early-morning data transfers using the cell phone's data plan, or physical delivery of the phone to an SD card reader that would copy the latest set of files. In both cases, the data was centrally collected, thereby suggesting a development methodology based on this central collection paradigm. In addition, plans for collaboration with the MIT Media Lab's LifeNet project[EMSW05], required that some component of the system be centrally located, in order to leverage one person's common sense knowledge for others.

Because it was determined that the data was to be located on the server, it was necessary to resolve the question regarding where to perform the filtering and processing of data in order to return the results to the user in an accessible form. Options ranged from sending the user's entire dataset over the network for local

processing on the client, to performing the processing on the server and sending only the completed results to the client. There are essentially two operations that needed to be performed: filtering the data set to select the subset of data relevant to our display, and processing that data in order to present the results in a form that would be intelligible to the user.

3.2.1 Local Querying, Local Processing

The first solution required that the entire dataset be transferred to the client, where it could be filtered and processed without any intervention on the server's part. In the previous section, the reasons for centrally locating the data repository were explained. However, one could store the master copy of the user's dataset on the server, and make a read-only copy for the client that would be used for filtering and processing only, with changes being propagated back to the server's version. This approach would have a number of advantages. The server would not be burdened with expensive CPU calculations, and the system could better scale to handle many clients, as the computational load would be distributed to the clients.

One user's data occupied 40MB in the source ASCII form of log files. This was reduced to around 3MB when it was stored as a SQL table in binary form. Sending the 3MB of data to the client would take a non-trivial amount of time, especially on a remote or off-campus connection. If the data could be stored locally on the client to avoid retransmission, this one-time download would be acceptable. Unfortunately, because of the requirements that the system operate within the web browser without any download-and-install component, it would have been impossible to store 3MB of data on the client's machine. Given a relaxation of this particular constraint, this approach may become more viable.

3.2.2 Remote Querying, Local Processing

Because it was not feasible to send the entire dataset to the client, a second solution would be to query the server as a database through some form of Remote Procedure

Call (RPC), and then process the result data on the client's display. This approach was taken in some initial prototypes, where XML-RPC was used to retrieve a subset of the data from the server. The client would be viewing a period of time, and all the data within that time period was requested and displayed locally. While this worked quite well in the initial prototypes involving a week or two of data at most, it would certainly not scale. In the worst case, the entire dataset would need to be transferred to the client, offering a solution no better than the full data transfer mentioned earlier. In fact, due to the verbose nature of XML-RPC calling conventions, or the overhead in de-normalizing the SQL data in a query result, the total data transferred would exceed that of the full data transfer in the worst case. For an early prototype utilizing XML-RPC, it took over five minutes to convert the data to conform to the calling conventions, not counting any network bandwidth overhead.

3.2.3 Remote Querying, Remote Processing

The final solution considered pushes the full computation load to the server. The server stores a SQL database with the full dataset, and the client requests processed data for a particular period of time, and other conditions. The server selects the appropriate data and processes it into a form for the client. For the initial prototype demonstrating this particular approach, the processed form was delivered as a compressed image. For the prototype involving a few months worth of data, there was a noticeable amount of delay before getting the image back. Because no optimizations had been performed yet, this was deemed acceptable at the early prototype stage, and this approach was utilized in the final application.

3.3 Diary Development Platform

Because the server's implementation platform was never visible to the user, Python was chosen for a variety of reasons. Other members of the team had prior experience working with Python, so completed work would not be orphaned after the original developer had left. The Python language, comparable in power to Perl or Ruby, also

offered significant development advantages over Java or C++, in the areas of developer productivity. Given these factors, Python was chosen for the implementation language for the server as well as various other tasks that needed to be performed behind the scenes, such as importing or transforming data.

However, this left the choice for the implementation of the client platform undefined. The user would be interacting with a local system to view their Diary, interacting remotely with the server. There were a few choices available for implementing the client Diary application.

3.3.1 Native Application

A native application developed in Java or .NET would offer the a wide suite of functionality, including the ability to store the entire dataset locally. However, this would not be a feasible solution since it would require that the user perform additional steps (such as downloading and running/installing the application). While a Python client implementation with a wxWindows GUI would have provided a standard language for Diary development, it also would have required the user to download and install an application. If it was determined that the downloading of an application by the user could be a viable design decision, then this would be a fruitful area for further exploration.

3.3.2 Flash

Flash is a widely supported, web-based deployment platform, with a recent version available on 97% of web browsers [Mac05]. Flash may be most easily recognized for its use in displaying interactive web advertisements, but it provides an environment quite suited to vector-oriented graphics display. Unfortunately, the system has an expensive, quirky development environment that seems more targeted towards developing animations rather than applications. Its proprietary programming language, ActionScript, has very little support for the higher-level programming constructs which make programming easier.

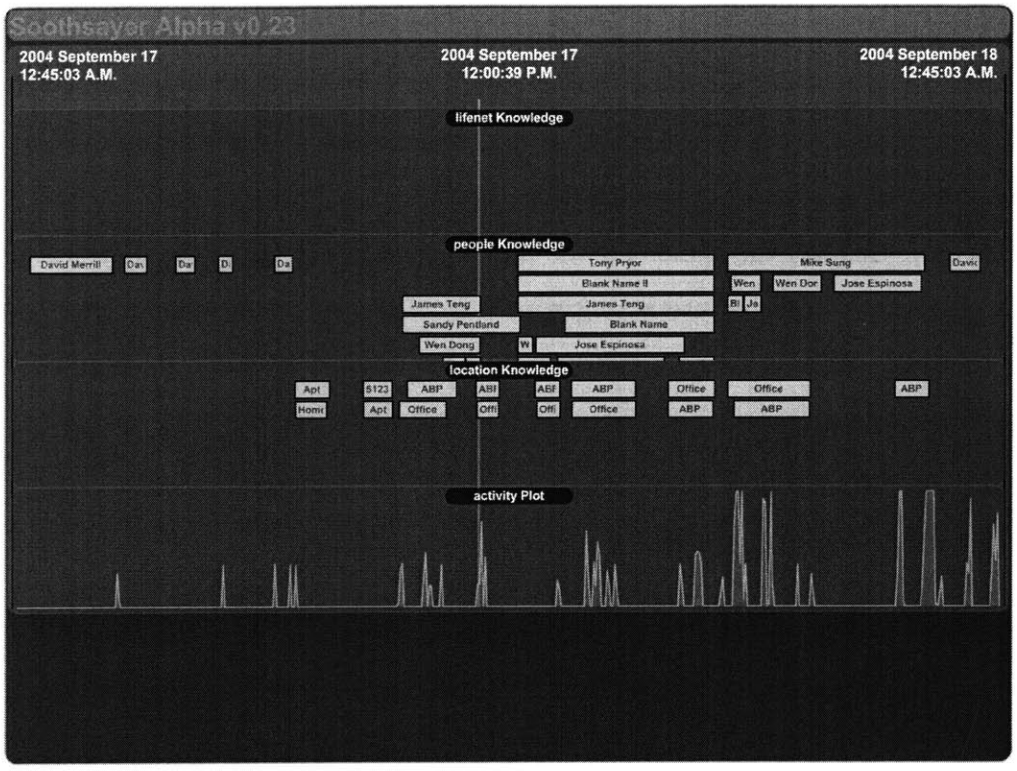


Figure 3-1: Screenshot of the prototype Flash interface, showing one day of data for a day at work with co-workers

While an initial Flash prototype showed promise, it was decided to not pursue it further. The most significant advantage of Flash, the GUI display, could not be utilized effectively because of the dynamic, data-driven approach to design. Instead, the GUI had to be constructed "on the fly" in code, leading to rather verbose code that was difficult to work with. A screenshot of the Flash prototype can be found in Figure 3-1.

3.3.3 Java

Java, with a slightly lesser market penetration than Flash, is market positioned as a sweet spot of ease-of-use and power lying somewhere between C++ and the more dynamic languages like Perl and Python. It has rather extensive support for GUI development, although some critics take exception to the Swing UI that is Java's

GUI toolkit of choice. A Java prototype utilizing a vector-based visualization toolkit was developed and a screenshot of this prototype can be seen in Figure 3-2. The toolkit, while providing powerful dragging and zooming capabilities, relies on the concept of drawn objects, which does not scale well with large sets of data in zoomed out views. Sections 4.1 and 4.2 present detailed views of this particular limitation.

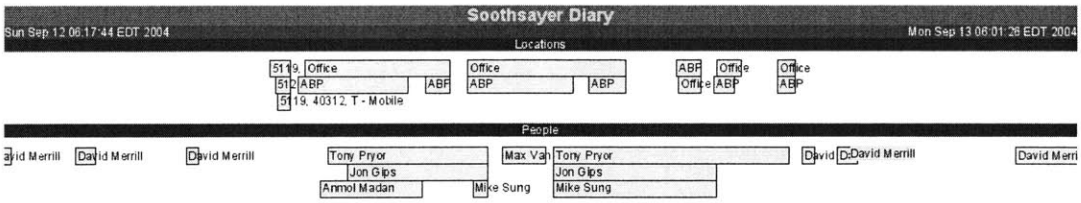


Figure 3-2: Screenshot of the prototype Java interface showing roughly one day, and the spans of time for which the user was near co-workers and at the Office

3.3.4 Web Application

Web applications are essentially "applications" that utilize the user's web browser to provide an interface. Their functionality derives from a combination of interaction with the web server and the user's browser. Although occasionally limited by the constraints of the web browser's capabilities or incompatibilities, powerful applications can nevertheless be created entirely within the web browser. This approach toward application development often requires a range of languages, such as HTML, CSS, JavaScript, as well as the server-side programming language. Since web applications utilize the web browser's interface, they share wide market penetration via compatibility with the three major web browsers: Internet Explorer, Firefox, and Safari.

Ultimately, the final iteration of the Diary project was developed and released as a web application implementation. The ability to develop in a powerful and concise language like Python for the server-side components proved to be quite valuable. Because web browsers are quite efficient at displaying graphics, a web application implementation also worked well with the graphical results produced by the "Re-

mote Querying, Remote Processing" approach described in Section 3.2.3. There were certainly limitations with the decision to go with a web application, including the difficulty in developing client-side interaction and GUIs, as well as problems ensuring that the application worked identically in every browser. These issues are discussed more thoroughly in Section 5.4.

Chapter 4

Diary Overview

The Diary application was launched on May 13, 2005 to the roughly 100 participants of the cell phone study. Each participant was provided with a username and password that allowed them to log in and view the data collected from their cell phone. After logging in with a username, the diary application presents a quick help page before dropping you into the application itself.

Like many other diary-style applications, the Diary operates off a timeline view of the user's data. For the Diary, the timeline allows the user to select any span of time, and look at the collected data within that period. Because of the widely supported range of time scales, the data view must gracefully handle both a day view where you can see the individual five-minute increments used in Bluetooth scanning, to the year view where each day is allocated only a two pixel vertical slice.

One of the fundamental units of the Diary is the timespan. A timespan represents an activity's temporal duration, consisting of a start time, and end time, and information about what that span of time represents. Most event types recorded by the cell phone can be represented easily with timespans. A comprehensive list of events recorded by the Context Log software can be found in Appendix A. For example, the duration of time for which the user is near a cell tower, near another person, or on the phone, are all capable of being stored as timespans. Similarly, the time in which the phone is active, charging, or has network service is also easily represented by timespans.

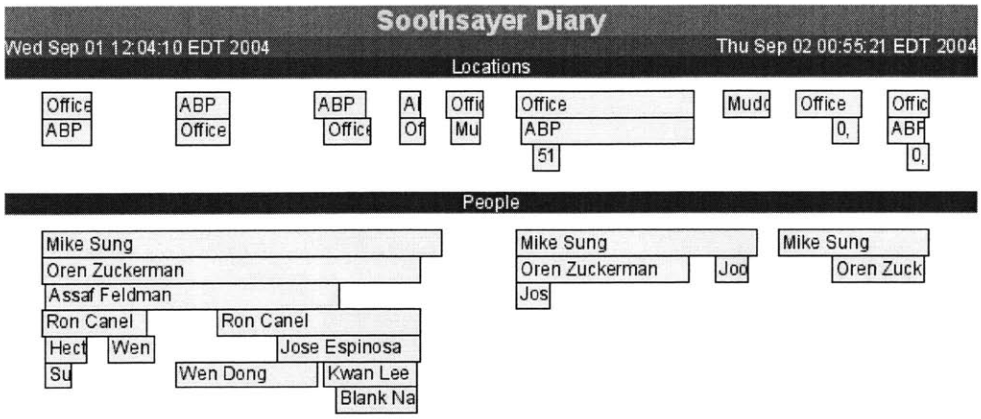


Figure 4-1: Box Spans: Zoomed-in view ($\frac{1}{2}$ day) view, boxes display both names and time-boundaries

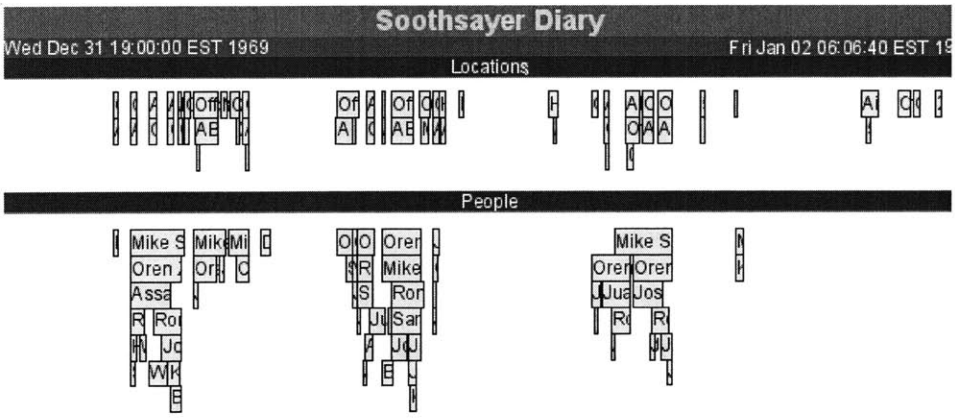


Figure 4-2: Box Spans: Zoomed-out view (2 days): boxes begin to lose their name, but retain time-boundaries

4.1 Box Spans

The easiest representation for displaying timespans is that of the box span, which draws a box symbolizing the temporal duration of the timespan.

As can be seen in Figure 4-1, the box view displays a box for times when the user is near that cell tower or person. This works quite well when the boxes are large, as it is easy to read the text within the boxes, but begins to break down as the boxes get thinner. If one zoomed-out further as in Figure 4-2, the boxes are thin enough that text within the box becomes illegible. If we were to add labels on the left-hand

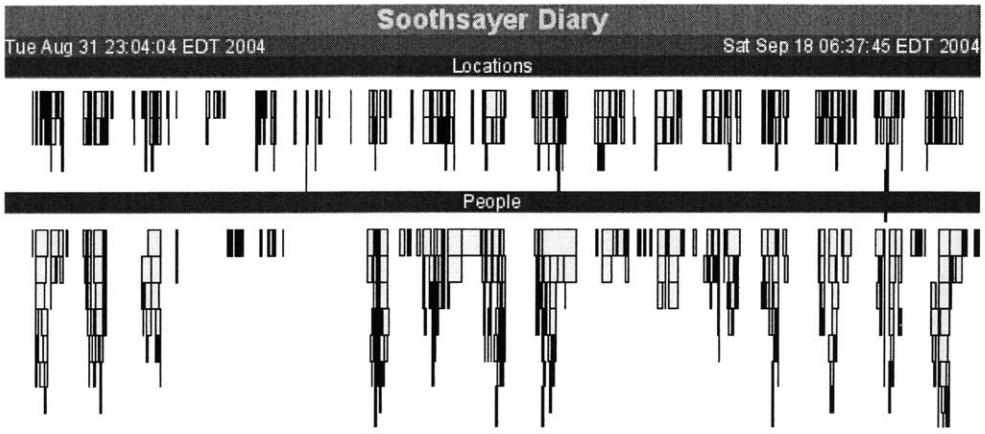


Figure 4-3: Box Spans: More zoomed-out view (17 days): boxes are now adjacent, making it difficult to discern time-boundaries

instead of inside the boxes, it would help with this problem. But if one zooms-out far enough, as in Figure 4-3, the boxes themselves shrink enough that the bounding boxes delineating them compete with the box size itself for screen real estate, making the boundary of particular timespans difficult to discern. As can be seen, this approach will not scale to seventeen days, never mind an entire year's worth of data.

4.2 Probability Spans

Probability spans is an alternate technique for displaying the spans of data that gracefully handles extremely zoomed-out views. Instead of representing timespans with a box, they are represented as a probability distribution over the timeline. In the zoomed-in case, this reduces to a timeline consisting of 1.0 where the box span would have been, and 0.0 elsewhere. The advantage of probability spans arises as the boxes begin to shrink. As soon as a box (or possibly even a box end) represents less than one pixel's worth of screen real estate, the probability distribution represents what percentage of that pixel is covered by the box span.

For example, Figure 4-4 demonstrates probability spans acting similarly to box spans. Each day shows a workaholic user alternating between their apartment and the office as they go to work every day (including weekends). The probability distribution

Figure 4-4: Probability Spans: Zoomed-in view (2 weeks): each day is represented by one cycle between Apt and Office

shows the user as using the apartment cell tower and office tower most of the time, with some of the office time broken up, probably to eat lunch or perform other errands. There's nothing here that fundamentally could not be done with the box spans approach.

Figure 4-5: Probability Spans: Zoomed-out view (8 months): each day is represented by two to three pixels

However, when looking at Figure 4-5, the advantage of probability spans becomes apparent. Each day in this heavily zoomed-out view is represented by two to three pixels, and any attempt to represent the timespans involved in work and office with boxes would run into the problems of Figure 4-3, but to a greater extent. Instead, Figure 4-5 represents the fraction of time spent at work and office during each vertical time slice. If one looks closely, a jagged sawtooth pattern can be recognized. This sawtooth pattern is the zoomed-out version of the daily patterns displayed in Figure 4-4.

4.3 Periodic Views

In addition, probability spans have the nice property of supporting periodic views. Periodic views can be useful when looking for regularity or periodicity in one's life, such as the patterns of sleeping, eating, working, and friends over the course of a person's daily or weekly schedule. To find these patterns, the Diary timeline accumulates each day's (or week's) data, displaying it on a timeline the size of a day (or week.)

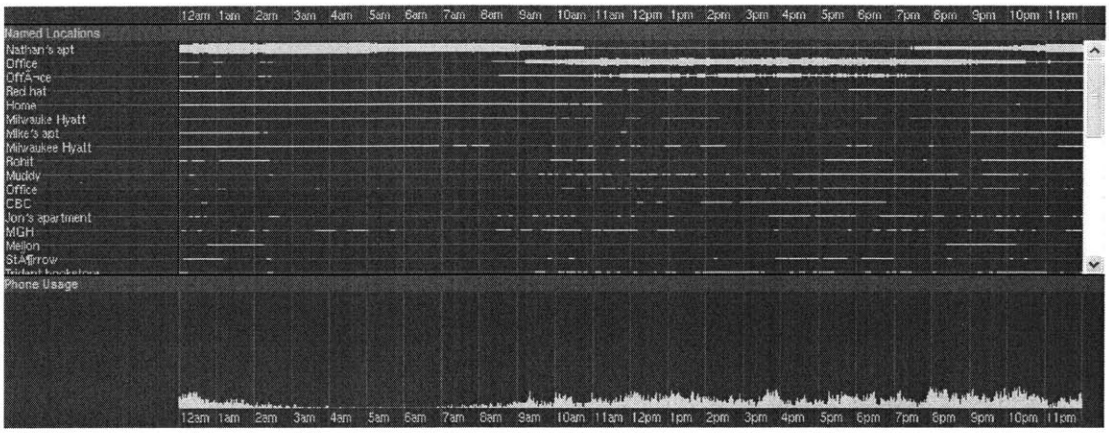


Figure 4-6: Daily schedule for two months of data

An example should make this concept clearer. In Figure 4-6, the daily schedule for two month's worth of data can be seen. Rather than the discrete box-looking spans we saw earlier, this shows much smoother and more gradual changes to the probabilities, as the probabilities are accounting for two months of data. In analyzing the above, we can see the subject spends a significant amount of time at home at night, and a lot of time at the Office during the day. His phone usage also shows a dip during the hours when he is likely sleeping. Other patterns can be found in the list of nearby people (not shown), and by selecting other periods of time to view.

4.4 Labeling Data

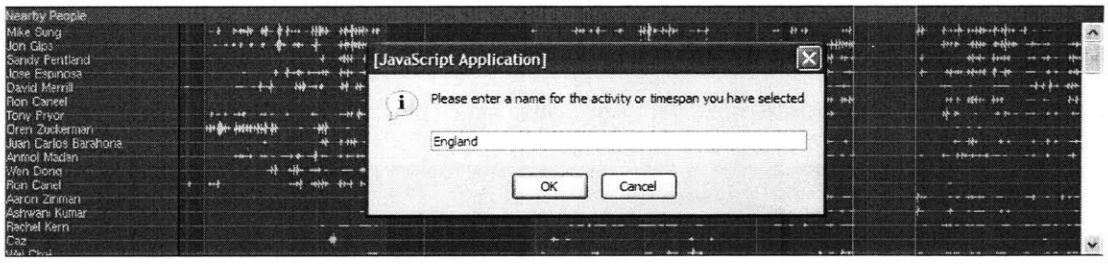


Figure 4-7: Labeling the selected data (vertical bars delimiting an empty span of time) as time spent in England

Of course, the cell phone cannot necessarily capture everything. Traveling while

on vacation may not register correctly if you are without cell service, or don't bother to classify the cell towers while on the road. Or maybe the unusual patterns before finals week is not immediately obvious unless you know what that week of time is, which the cell phone, nor the Diary application, can infer from the data. Instead, the Diary application allows the user to manually classify information. The user can select a span of time as in Figure 4-7 (the selection can be seen on the right of the dialog box), and label it appropriately. The labeled data can then be seen as a probability span with all other data. If the user spent the time to accurately label their sleep schedule (as opposed to inferences about when the phone is charging), then the periodic views would give an accurate probability of the user's chance of being asleep at a given point in time.

4.5 Interface Issues

A lot of revisions were made on the Diary application in the goal of finding a usable interface. Originally the interface supported the ability to drag-select a period of time and automatically zoom the timeline to frame that time, which made sense. But as the only navigation mechanism, it performed poorly. Applying the same mechanism to zooming out with the Control key (where the selected space was the location the current timeline should shrink to) proved to be counter-intuitive. The user would select a small span of time to zoom out a lot, and a large span of time to zoom out slightly, the opposite correlation of what was done to zoom in. It was also difficult to pan the timeline with this interface. When viewing a day and wishing to view the subsequent day, one would need to zoom out, and then zoom in on the desired day.

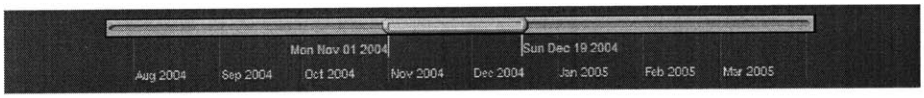


Figure 4-8: Timeline navigation bar, used to navigate the current timespan view in the displayed timeline

Instead, the interface was redesigned to support a timeline bar across the top of

the screen, as in Figure 4-8. The user could drag the bar sideways to pan the current time view, or drag the ends of the bar to grow or shrink it. It also provided immediate feedback as to the timespan that is selected via the tickmarks underneath. This UI mechanism turned out to be better in some ways, and worse in others.

Figure 4-9: Timeline navigation bar, used to navigate the current timespan view in the displayed timeline

In particular, the user often did not know they could drag the ends of the bar, especially when it was fully zoomed out as in Figure 4-9. And while this timeline navigation bar worked quite well for the largest data set recorded of about nine months, it would have trouble scaling in the future. Each day was represented by about two to three pixels of horizontal screen real estate, helping the user to accurately select a given point in time. But as the time represented by the timeline grows larger, the size allocated to days would shrink to a pixel, and then sub-pixel sizes, making certain days impossible to select. A better selection and display mechanism would be needed in the event of accumulating a few year's worth of data. Perhaps using a calendar-style view that users can drag across would be both more intuitive and more scalable.

Chapter 5

Diary Implementation

5.1 Database Design

In designing an efficient database system that would handle the data, it was important to consider both the structure of the underlying information, as well as how the data would be used.

5.1.1 People

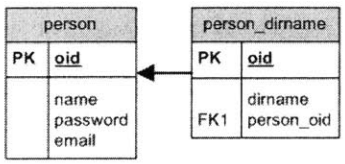


Figure 5-1: Entity diagram for person and person_dirname

The central data collection is organized in directories, one directory per phone. Occasionally, a person may have multiple directories due to the fact that users, for various reasons, may have switched phones during the experiment. This led to a rather simple database structure seen in Figure 5-1, where FK represents a foreign key, and PK represents the primary key.

The events in the log files, as described in Appendix A, represent individual points in time. But, as described earlier in Chapter 4, the timespan is a useful atomic element

for visualizing and working with the data. There were a variety of timespan type data that were useful in the Diary application.

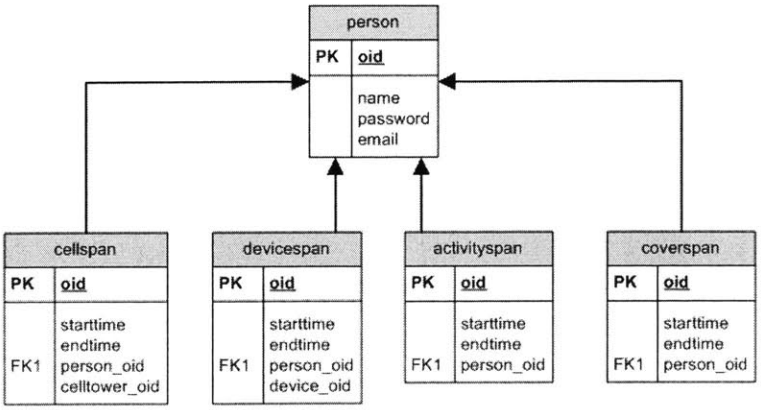


Figure 5-2: Entity diagram for cellspan, devicespan, activityspan, coverspan, and person

As demonstrated in Figure 5-2, there are various types of span tables to represent the different data types. While each span holds a starttime and endtime bounding the timespan, and a person_oid that identifies the person this timespan belongs to, they can optionally hold additional data unique to that type of timespan. The activityspan and coverspan tables shown in Figure 5-2 have no additional data, because the only interesting information is the length of the timespan itself.

Activityspan stores information for when the cell phone is active and not in screen-saver mode. Coverspan is used to determine what spans of time are covered by the log files. This information permits us to distinguish between a lack of data due to the cell phone being turned off, and the lack of any interesting events occurring.

5.1.2 Cell Towers

Stored information regarding the in-use cell towers as well as the user's descriptive names for the cell towers (as entered on the cell phone), is illustrated in Figure 5-3. The cellspan stores information about which cell tower the cell phone is connected to for that duration of time. The celltower's name field stores the self-reported name from the celltower itself, which is often a string such as "24127, 2421, AT&T Wirel".

Because the cell phone's software requests the owner to name their current location when it detects a previously unseen cell tower name, these user-entered names for cell towers can be retrieved in the cellname table.

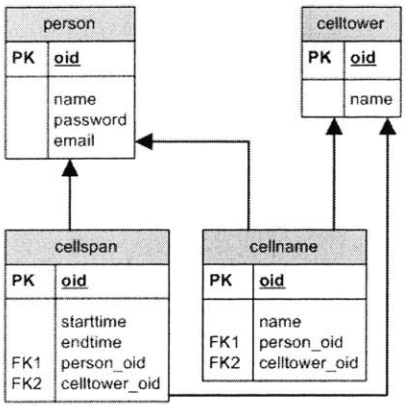


Figure 5-3: Entity diagram for cellspan, celltower, and cellname

5.1.3 Bluetooth Devices

Since other cell phone users advertise themselves through Bluetooth, storing information about the nearby Bluetooth devices is important for determining with whom the phone owner is proximate. Figure 5-4 details the relationships needed tos tore the information. Each unique device seen in the logs is represented by a unique device record, where the device's name field stores the self-advertised name received from the Bluetooth device. Sometimes, a Bluetooth device can represent another registered cell phone user in the study. In this case, the device record also has a non-null person_oid pointing to person the device represents.

5.1.4 User Labels

Finally, Diary's ability to let the user label activities required some tables on the database backend to facilitate that. These tables can be seen in Figure 5-5. The usertagspan table stores the traditional timespan information, and the usertag table represents the label itself. Multiple usertagspans referencing the same label will in

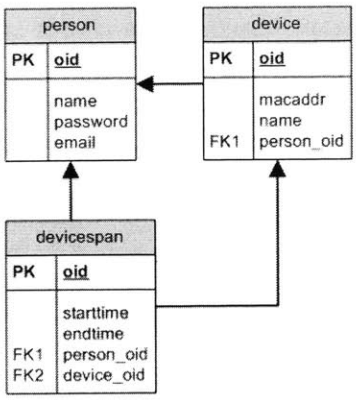


Figure 5-4: Entity diagram for devicespan, device, and person

fact reference the same usertag record. And of course, the usertags are themselves associated with the person who is labeling the timeline.

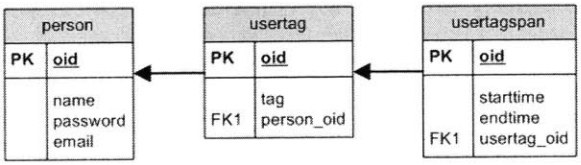


Figure 5-5: Entity diagram for usertag, usertagspan, and person

5.2 Importing Data

Of course, a database is useless without data. A custom script was written to import the data from the roughly 1GB of log files into the database. Each relevant event written to the log, the details of which are described in Appendix A, would begin or end a timespan.

The cell tower events recorded the time whenever the primary cell tower changed, and so each cell tower event would in turn create a cellspan representing the previously primary cell tower's timespan. The device events occurred every five minutes when Bluetooth scans were performed. As long as no more than 15 minutes passed between seeing a particular device in the device events, the code would continue to treat the device as if it were still nearby to the cell phone. So each device event could begin or

end multiple independent devicespans. Activity events of "active" or "idle" mapped directly to the start and endtime of an activityspan. Similarly, coverspans were created by recording the first and last time recorded in a log file, as the temporal extremes of data recorded by that log.

As expected, there were numerous corner cases and errors to handle. In the course of collecting 1GB of log data on cell phones and memory sticks, there were bound to be read/write errors. As the script parsed timestamps, it would frequently encounter malformatted timestamps that turned out to be single-bit errors. However, the import script undoubtedly encountered many other single-bit errors in the data set, errors that did not cause the script to fail outright. They might have changed the timestamp to form another valid timestamp, possibly leading to negative timespans, which were observed in the resulting database. It is also possible that a recording of a cell tower seen, a phone call made, or nearby Bluetooth devices would not import correctly because of data corruption. In the course of a year's worth of data, such errors would be difficult to spot, even for the original cell phone owner. It is believed that these errors are not worth excessive concern. The data corruption rate is extremely low, and as cell phone storage mediums become more reliable, the issue should become even less important.

5.3 Web Application Approach

In keeping with the design approach of many traditional web applications, the Diary application aims to display the results of querying data and operating on it within one screen, updating data dynamically as the user performs actions. The two main components of the Diary application are the overall timeline at the top, and the lower timeline below which displays the currently selected timespan.

As shown in Figure 5-6, the overall timeline at the top demonstrates the timeline in its maximally-selected state, along with the listbox for using periodic views. The lower display displays the detailed, currently selected view of the user's data. In this screenshot, the application displays the data loss for the month of October, as well

as a February trip to England (currently unlabeled) during which time the cell phone was not used (turned off). Finally, the screenshot demonstrates the user selection for a period of time over winter break.

The user interface of this web application required extensive JavaScript development. The timeline widget is entirely interactive JavaScript designed to appear like a growable slider. While somewhat visually unintuitive to work with, partially due to the lack of standardization on any sort of widget UI to perform this task, it presents a more visually appealing interface than the timeline widget UI of *MyLifeBits*, seen in Figure 5-7 [AGL04].

5.4 JavaScript Development

While one of the reasons for pursuing a web application implementation was the to exploit the advantages of Python on the server, it turned out that a significant amount of JavaScript development was needed to complete the client-side interface. While the JavaScript language itself is relatively simple to work with, it is burdened by a myriad of implementations and APIs. Formatting and layout is done with CSS and HTML. Much of the interaction with the web browser requires a combination of HTML and JavaScript. Both JavaScript and the browser's form submission and hyperlinks can be used to mediate communication between the client and the server. The different technologies all need to work together in harmony, despite overlapping in capabilities and responsibilities. Small changes to the web application can hit the limitations of one approach, requiring a major restructuring of the code to use another, thereby slowing the development process. The cross-browser compatibility requirements proved to be more difficult than anticipated: getting the application looking and behaving identically required considerable effort on CSS formatting and HTML layout code, as well as making allowances in the JavaScript code to handle the differences in the APIs offered by Mozilla Firefox versus Internet Explorer.

5.5 Drawing Images

As stated in Section 3.2, the project chose to utilize a server-based querying and processing approach. For the web application, this meant that the results would be returned as generated imagery. Throughout development, the image displays were under constant revision. Image processing and generation proved to be the primary performance bottleneck in the system and required many iterations of testing and code optimization.

The basic algorithm for generating the probability span images follows. First a list of the timespans relevant for the probability span (the time, type, and subject of the timespan are all criteria) is generated from the SQL server. An array is then allocated, with each element corresponding to one pixel's data in the probability span image. Each element represents a span of time, with adjacent elements represent adjacent spans of time. The SQL timespans are then accumulated into the array, distributing the time in each timespan across the relevant timeslices in the array. The same process is repeated for the coverspan table timespans (which store the spans for which the phone was recording data). For each pixel of the probability span the element of the first array is divided by the second to generate the probability for that timeslice. This is then graphed horizontally on an image to generate the image's probability spans. Based on profiling performed while optimization, roughly half the time in the algorithm is spent calculating the probability spans, and the other half is spent drawing the pixels of the probability span on the canvas.

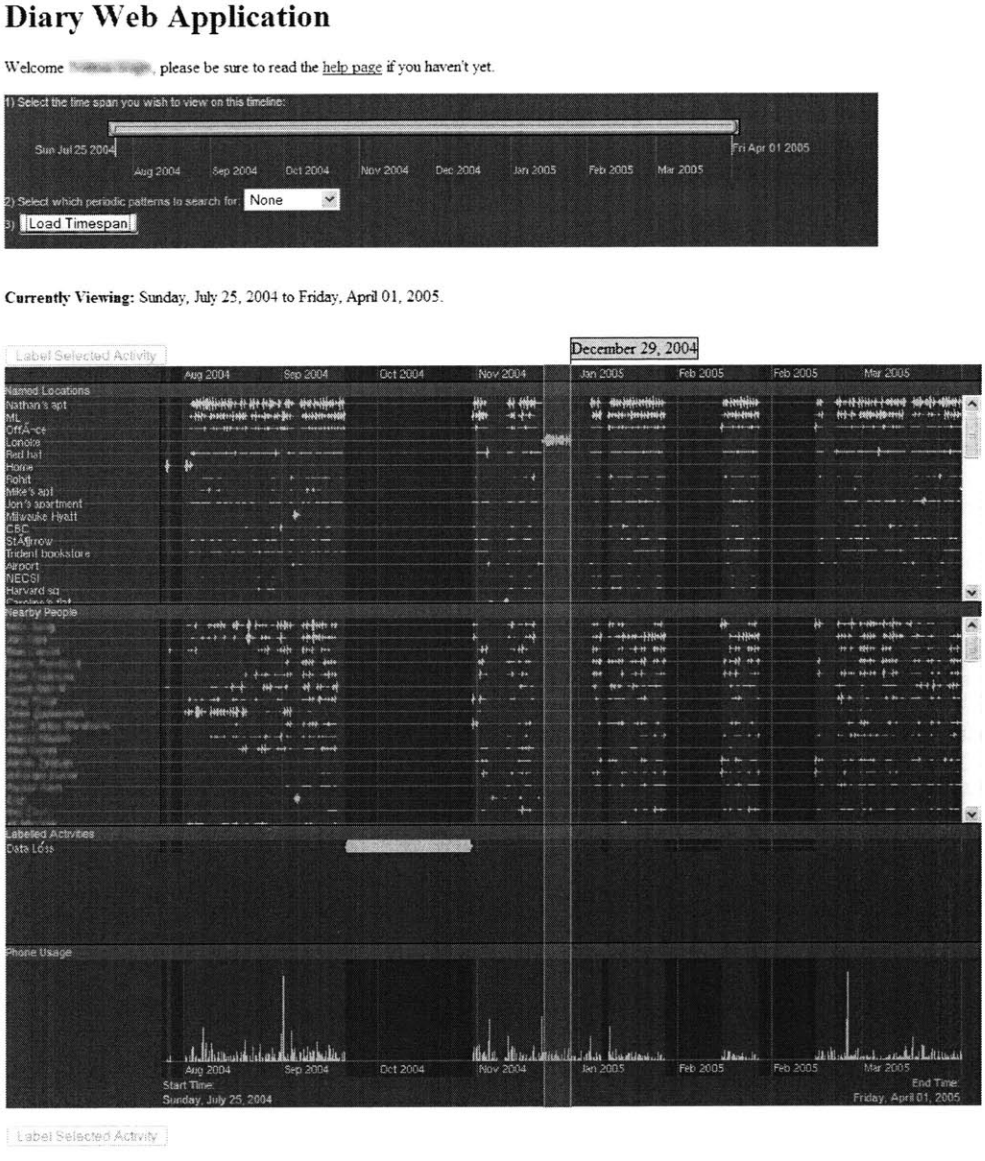


Figure 5-6: Screenshot of the Diary Application

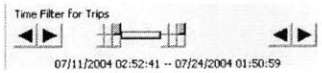


Figure 5-7: Timeline widget used in *MyLifeBits*

Chapter 6

Future Work

With an exploratory project of this scope, there are many directions for future work. Different design decisions might lead to a better Diary application, or more features might be extracted from the data sets. Other areas of future work include incorporating additional data points or data feeds into the application.

6.1 Lessons Learned

Many lessons learned during the development of the Diary application lead to potentially new avenues of development. Some lessons question the design decisions made, while others only suggest subtle changes to the application. All of these lessons learned may be explored in future work to potentially develop a better Diary application.

6.1.1 Data Visualization

The image visualizations of the user's data had a tendency to overwhelm users. Compressing an entire year's schedule and data required a rather dense visualization format. Many users were intimidated by the visualization, not entirely sure how to interpret the graph to get answers. It is possible that the visualization was too compact, and that display utilizing blank space effectively would be easier to work with.

Exploring other approaches to allocating visual space to the data is certainly an-

other area for further exploration. One problem with the compact data visualization is the lack of any use of space to indicate importance. Every cell tower and nearby person had an equivalent amount of room allocated to the horizontal probability span, regardless of importance. Depending upon the task at hand and the information desired, different cell towers or people may prove to be more important. Home and Office are probably the most popular cell tower descriptions, and enlarging them would show the finer details of the user's patterns. On the other hand, if the user is using the application to find a particular event that occurred in the past, he may be more interested in the rare cell towers and visited people. In a sense, patterns that diverge from a user's normal daily or weekly patterns are more interesting, and should be emphasized.

6.1.2 Navigation and User Interface

One of the limitations of choosing JavaScript to implement many of the user interface features of the Diary application was the time spent on the new user interface widgets. While the widgets could be made to work well enough, the nature of the interdependencies between JavaScript, HTML, and CSS made small modifications problematic, as they often caused cascades of changes through the dependencies. Because of this, the web application became resistant to change, which inhibited attempts to creatively experiment with the interface.

For example, instead of a draggable, resizable timeline across the top of the page, using a calendar may have been more effective. People are quite accustomed to dealing with calendar views, and may have found it easier to use. A sequence of months would be displayed across the top, and the user could drag the currently selected start and end dates, or even drag the selection itself, the same as with the current slider-based timeline. A calendar approach would have scaled just as well, and possibly better than the slider-based approach, but unfortunately because of the difficulty in making changes to the JavaScript user interface code or in developing new user interface widgets, this was never explored.

There are two avenues for future work here. The first, obviously, is to explore

more user interface approaches for controlling and navigating the data. To facilitate that, a second area of future work would be to re-implement the user interface in Java or some other GUI toolkit, with the intent of making interface modifications and experiments easier to develop and test out.

6.1.3 Speed and Responsiveness

For a typical user's data for a year, this process can take up to ten seconds to produce a full set of images for the selected timeline. While certainly not long enough to just offline computation, it appears to be just beyond the realm of a comfortable experience for real-time computation. This delay has the unfortunate consequence of making the user reluctant to navigate on the timeline, because of the delay involved in loading the new images. Instead of being a seamless interface that facilitates efficient navigation, it became something the user was hesitant to use. One very useful area of future work would be to eliminate or decrease this bottleneck. While the current Python code is close to optimal, greater gains may come from re-architecturing the algorithm or data structures, or even translating the code to a low-language like C that avoids Python's overhead.

6.1.4 Data Storage

One of the design constraints on the original system was the centralized data storage. All Diary users' data was located on the central server, and the Diary provided an interface to interacting with it. As discussed in Section 3.2, the server-hosted data set necessitated a decision to perform the querying and processing on the server as well, since the user would not install or download anything. If this particular decision were reversed however, and the user could download his data, it would trigger a cascade of changes in the design of Diary.

First, if the user is willing to download and/or install an application, it provides more flexibility in the implementation language and platform. Python with the wxWindows UI toolkit, or Java with the Eclipse UI toolkit could both be used

as an implementation language, providing cross-platform support along with native UI widgets. A downloaded Java application, bypassing the web browser's security model, would also allow for more capabilities such as filesystem access.

Second, this access to the filesystem now supports the possibility of a local copy of the user's dataset, supporting a local querying and processing approach to the data, as discussed in Section 3.2.1. However, instead of forcing the user to download their dataset every time they use the application, a downloaded application would be capable of storing the data set locally, and periodically getting the latest updates to the data set, either from the server or from the cell phone directly.

Third, as described in Section 5.5, the remote querying and processing of data proved to be quite slow as implemented in Diary. This was in part due to the stateless nature of web browsers. Every time a new timeline is requested, the server must start from scratch to generate the data visualization, filtering to find the correct timespans, accumulating time to calculate the probability span, and finally drawing it using the image libraries. With a local data set and all calculations being done on the client, more efficient algorithms can be utilized. Any requests to change the timeline can re-use work done to generate the visualization for the previous timeline. The performance improvement might be enough to support instant feedback and smooth scrolling, two things sorely missing from the current Diary application.

6.2 Querying Capabilities

As mentioned Section 1.1 on page 14, one of the main motivating reasons for a Diary-type application was the ability to answer specific questions about the user's personal history, and a variety of typical questions were given. Unfortunately, the current Diary application only supports a subset of these questions, relying on the user to answer the questions themselves by interpreting the visualizations provided. Strangely enough, while Diary easily answers the more complicated queries mentioned, it fails to answer the simpler questions that were presented.

For example, to answer a query about the number of Red Sox games attended,

the user would need to manually count the number of times activity was seen on the cell tower labeled "Red Sox". The user would need to ensure they were zoomed out enough to see all potential Red Sox games, while at the same time ensuring they were zoomed in enough to visually distinguish each Red Sox game on the timeline. For a question that has a simple numerical answer, Diary makes the process of answering the question more difficult than it needs to be.

Certainly, exploring the realm of question and answering facilities available here would be one important area of research. While many of the questions are answerable by simple SQL queries, the difficult part is in providing an interface that makes it easy for the user to obtain the desired answer. Due to the restricted domain of answering questions about events and time, it is possible that a natural language interface could be developed that helps answer many of the more common questions.

6.3 Alternate Data Sources

Another potential gold mine of future work involves extending the Diary application to support additional types of data. Currently, the system is limited to supporting timespans generated from the cell phone's event log files. There is a wealth of other information that can be explored for better visualizations, many of which were discussed in the related work of Chapter 2. Many of these types of data have been explored in different applications, but there has been little convergence to a common data format or application to allow side-by-side viewing of the various types of data on the same timeline.

6.3.1 Point Data

Timespans, the fundamental unit of data in Diary, deals with a block of time. But many types of data are not suited to this. Anything that is sampled periodically may be better suited to a representation that only stores an instant of time. This supports a much broader selection of data. Any numerical data sampled at multiple times would work, and line graphs work great for visualization. For example, a

person on a diet is likely to check his weight frequently, and Diary could display his weight alongside the other information collected, possibly supporting a search for correlations.

The data does not even need to be associated with a numerical value like weight. A single time value can work quite well when associated with all sorts of data. A person's personal diary or journal has timestamps associated with the entries. Google's recent project to store Search History [Goo05] is another example, storing each search query performed along with timestamps, providing a color-coded calendar view that shows the user's search activity. Integrating support in Diary for these individually timestamped events would allow for the exploration of various visualization techniques.

6.3.2 Photos

As demonstrated by *MyLifeBits* (Section 2.2.5), photos are another very interesting area to explore. Many digital photographs currently record the time they were taken, and some more advanced cameras even support GPS, storing the location they were taken as well. There are a variety of techniques for displaying images on timelines, from taking a representative sampling of photos to display, to scaling them down to fit as the user zooms in, or even a slideshow of photographs from a particular event displaying at a particular point on the timeline. Diary's current implementation in HTML could make support for this very easy, as web browsers are very capable image viewers.

Also, combined with journal, diary, or blog writing, this could utilize many of the same techniques from Nokia LifeBlog (Section 2.1.6) or Blogger Mobile (Section 2.1.5), but with the added advantage of including the other visualizations present in Diary.

6.3.3 Biometrics

Biometric data about the user is another interesting area for further exploration. BodyMedia (Section 2.1.1) records a great deal of information about the wearer, and supports displays in the form of line graphs. Other sensors such as accelerometers or step counters that record a specific type of biometric information are also useful, as they can be much cheaper (and therefore more accessible) than BodyMedia's product. BodyMedia is targeted more at a lifestyle analyzing product for people interested in looking at their stress levels, activity levels, or caloric expenditure. Integrating this biometric data with the various other types of recorded data could prove to be more valuable than the biometric data alone, as BodyMedia's visualization application provides.

In a similar vein, other methods for visualization can be explored. Accelerometer visualizations typically have been displayed as line graphs for each of the three axes, making it difficult to interpret the wearer's orientation without experience in reading the graph. Exploring better techniques for displaying orientation or other biometric data that makes it more intuitive and readable is certainly an important area of further research.

6.3.4 Desktop Computers

Finally, because computers play a large role in many people's lives, integrating information from computers into a Diary application would be very useful. A cell phone is limited in the amount of information it can gather, particularly if the user sits in his office all day. Gathering higher-level information from the computer itself about mouse and keyboard usage, or even information about the relative usage of applications (similar to the data gathered on the cell phone application), could be very illuminating. How much time is spent reading email, and how frequently does the user check it? How much time of the day is spent browsing the web, and how much is spent programming or writing a thesis? This type of information could be very valuable as a self-help approach for users interested in becoming more productive.

Chapter 7

Conclusion

This thesis discussed the need for applications that can augment our imperfect memories. Devices are now capable of recording massive amounts of data unobtrusively throughout our daily life. However, data alone is useless to us: we need interfaces to the data that let us navigate it quickly and effectively, assisting us in answering the questions we ask of our digital "memories."

The Diary application, the focus of this thesis, is one such interface for this type of data. Diary currently provides an efficient, compact representation of a year's worth of data recorded on cell phones, and allows the user to drill down to individual days and find more specific information. In this thesis, the application was explained from a number of perspectives, including an overview of the user interface, the reasoning behind the design decisions used in building the application, and some details on the implementation of the application itself. Finally, a variety of potential directions for the Diary application were discussed, ranging from revisiting fundamental design decisions in light of lessons learned in development, to ideas for extending the application to display additional sources of data.

Appendix A

Cell Phone Log File Formats

Each cell phone records the various interesting events to log files. The timestamps used to store these events are of the following format: `YYYYMMDDTHHMMSS`. The following substitutions are performed on the format:

`YYYY` = four-digit year

`MM` = two-digit month between 01 and 12

`DD` = two-digit date between 01 and 31

`T` = the character "T"

`HH` = two-digit hour between 00 and 23

`MM` = two-digit minute between 00 and 59

`SS` = two-digit second between 00 and 59

When the cell phone is turned on, three ASCII-formatted logs are created, using the current timestamp for the filename as follows:

starter-TIMESTAMP.txt A log for information about the logging application itself. It includes the starting/stopping status messages for the application, as well as other miscellaneous debug messages. It is not used when analyzing the participant's information, so we will not discuss it further.

call-TIMESTAMP.txt A log of the phone calls. This includes incoming/outgoing status, the phone number dialed, the duration of the phone call, and a few other statistics. It also includes information about SMS messages and data transfers.

log-TIMESTAMP.txt A log containing everything else: current cell tower, nearby Bluetooth devices, current phone application in the foreground, idle/active status, charging status, and so on.

A.1 "Call" File Format

All phone calls, SMS text messages, and data transfers are recorded in the `call-logs`. Each line is of the following format:

timestamp `EVENT ID:` *id* `CONTACT:` *contact-number* `DESCRIPTION:` *description* `DIRECTION:` *direction* `DURATION:` *duration* `NUMBER:` *call-number* `STATUS:` *status* `REMOTE:` *remote*

where:

`timestamp`	=	time at which the call took place
`id`	=	sequentially increasing id representing the call
`contact-number`	=	The ordinal index into the address book
`description`	=	`Voice call`: regular phone call
		`Packet Data`: data transfer, utilized by cell service
		`Short Message`: an SMS message
`direction`	=	`Incoming`: incoming call or message
		`Outgoing`: outgoing call or message
`duration`	=	duration of the call in seconds
`call-number`	=	for voice calls, the phone number connected with
`status`	=	`Sent`: a short message was sent out
		`Delivered`: a short message was remotely delivered
		`Disconnected`: utilized by packet data
`remote`	=	Person name or service currently connected with:
		For short messages / voice calls, the person's name as it is in the address book (or "Voice Mail")
		For packet data, should indicate the service provider, such as "TM" for T-Mobile.

A.2 "Log" File Format

All other events that the application records are stored in the `log-` files. Each event type will be described in turn.

A.2.1 Cell Tower Event

timestamp `area, cell, nw:` *unique-cell-tower-id*

where:

`unique-cell-tower-id` = a three-part value containing area and cell numbers, as well as a network name (such as "T-Mobile")

A.2.2 Bluetooth Device Event

timestamp `devices:` *device-mac-address* `[`*device-name*`]`
device-mac-address `[`*device-name*`]`
...

where:

`device-mac-address` = hexadecimal MAC address of the device

`device-name` = user-created name of the device

A.2.3 Activity Event

timestamp `UserActivity:` *idle-active*

where:

`idle-active` = `idle`: Phone is now in screensaver mode
`active`: Phone is no longer in screensaver mode

A.2.4 Application Event

timestamp `ActiveApp:` `[`*app-id*`]` *app-name*

where:

`app-id` = hexadecimal id representing the application

`app-name` = name of the application, for example:
`Phone`, `Phonebook`, `ScreenSaver`, `Menu`,
`ClockApp`, or `FileManager`

A.2.5 Network Status Event

timestamp `Network status:` *network-status*

where:

`network-status` = 0 or 1 value representing the new network status

A.2.6 Charger Event

timestamp `Charger:` *charging-status*

where:

`charging-status` = 0 or 1 value representing the new charging status

Bibliography

[AGL04] Aleks Aris, Jim Gemmell, and Roger Lueder. Exploiting location and time for photo search and storytelling in mylifebits. Technical Report 102, Microsoft Research, September 2004.

[Blo05] Google Blogger. On the go with blogger mobile. `http://help.blogger.com/bin/answer.py?answer=1131`, May 2005.

[Bus45] Vannevar Bush. As we may think. *Atlantic Monthly*, July 1945.

[Cla02] Brian Clarkson. *Life Patterns: structure from wearable sensors.* PhD dissertation, Massachusetts Institute of Technology, Media Arts and Sciences, September 2002.

[DCC+03] Susan Dumais, Edward Cutrell, JJ Cadiz, Gavin Jancke, Raman Sarin, and Daniel C. Robbins. Stuff i've seen: a system for personal information retrieval and re-use. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 72–79, New York, NY, USA, 2003. ACM Press.

[Eag05a] Nathan Eagle. *Machine Perception and Learning of Complex Social Systems.* PhD dissertation, Massachusetts Institute of Technology, Media Arts and Sciences, May 2005.

[Eag05b] Nathan Eagle. Mit media lab: Reality mining. `http://reality.media.mit.edu/`, May 2005.

[Edi04] MIT Technology Review Editors. 77 mass ave. *MIT Technology Review*, September 2004.

[EMSW05] Jose Espinosa, Bo Morgan, Push Sign, and William Williams. Lifenet. `http://xnet.media.mit.edu/LifeNetHome.htm`, May 2005.

[GBL+02] J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong. Mylifebits: fulfilling the memex vision, 2002.

[GH03] E. Grochowski and R. D. Halem. Technological impact of magnetic hard disk drives on storage systems. *IBM Systems Journal*, 42(2):338–346, 2003.

[Goo05] Google. My search history (beta) help. `http://www.google.com/searchhistory/help.html`, May 2005.

[KSSF03] Andreas Krause, Asim Smailagic, Daniel P. Siewiorek, and Jonny Farringdon. Unsupervised, dynamic identification of physiological and activity context in wearable computing. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, page 88, Washington, DC, 2003. IEEE Computer Society.

[LF94a] M. Lamming and M. Flynn. Forget-me-not: a portable context-sensitive diary for contextual retrieval. `http://www.xrce.xerox.com/programs/mds/videos/fmn-2-cd.html`, 1994.

[LF94b] M. Lamming and M. Flynn. Forget-me-not: intimate computing in support of human memory. In *Proceedings FRIEND21 Symposium on Next Generation Human Interfaces*, 1994.

[LWS+] Craig B. Liden, Mary Wolowicz, John Stivoric, Astro Teller, Chris Kasabach, Suresh Vishnubhatla, Ray Pelletier, Jonny Farringdon, and Scott Boehmke. Characterization and implications of the sensors incorporated into the sensewear™ armband for energy expenditure and activity detection. `http://www.bodymedia.com/pdf/Sensors.pdf`.

[Mac05] Macromedia. Macromedia - macromedia flash player : Version penetration. http://www.macromedia.com/software/player_census/flashplayer/version_penetration.html, May 2005.

[Nok05] Nokia. Nokia lifeblog. http://www.nokia.com/lifeblog/, May 2005.

[oed05] Oxford english dictionary. http://dictionary.oed.com/, May 2005.

[RCDH03] M. Ringel, E. Cutrell, S. T. Dumais, and E. Horvitz. Milestones in time: The value of landmarks in retrieving information from personal stores. In *Proceedings of Interact 2003*, September 2003.

[UoH05] Department of Computer Science University of Helsinki. Context project. http://www.cs.helsinki.fi/group/context/, May 2005.

[Vem05] Sunil Vemuri. What was I thinking? http://web.media.mit.edu/~vemuri/wwit/wwit-overview.html, March 2005.

[War04] Mark Ward. Log your life via your phone. *BBC News Online*, March 2004.