GurunGo: Coupling Personal Computers and Mobile Devices through Mobile Data Types

Iván E. González [1]
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052 USA
ivangonz@microsoft.com

Jason Hong
Human Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15217 USA
jasonh@cs.cmu.edu

ABSTRACT
Networked devices like desktop computers and mobile phones make it possible for people to access any of the billions of web pages available on the Internet. However, mobile devices are fundamentally different from desktop PCs in terms of input speeds, screen size, and network speeds, making it harder in practice to find information when on the go. In this paper, we introduce GurunGo, a system that monitors a person's activities on their PC for mobile data types—kinds of data likely to be useful to a person when mobile—and then proactively copies these snippets of data onto his mobile device, thus making it easier to find that information when mobile. Our initial prototype finds and extracts mobile data types from web pages that are browsed on a desktop computer, annotates it with additional relevant information, and copies it to a mobile device in the background. We discuss the design and implementation of GurunGo, as well as some of the tradeoffs and design rationale.

Categories and Subject Descriptors
H.5.2. Information interfaces and presentation: User interfaces — *Interaction styles*.

General Terms
Design, Reliability, Human Factors

Keywords
GurunGo, web, mobile data type, data type detector

1. INTRODUCTION
While people have easy access to a large amount of information when at their personal computers, the same cannot be said when those people are mobile. Mobile devices may have limited or no network access, slow text input, and small screen sizes, making it hard *in practice* to find relevant content when on the go. Examples of such information that can be useful when mobile include maps, driving directions, movie show times, product price comparisons and reviews, flight times, locations and review of both stores and restaurants, phone numbers, weather, traffic information, and social events (e.g. concerts, book signings, etc).

HotMobile 2010, Feb 22–23, 2010, Baltimore, MD, USA.

There is also a chasm between one's personal computer and mobile device. If a person finds useful information on their desktop computer, there is usually no easy way for typical users to transfer that data to their mobile device. For example, it is more likely that a person will print out a map or print out details about a product they want to purchase, than to copy that information onto their mobile device, despite the fact that there are multiple tools for copying and synchronizing data. Furthermore, most of the synchronization tools that are available focus on syncing data such as email, contacts, and calendar information. These tools overlook the fact that people encounter a great deal of information in the course of their regular web browsing, information that may be valuable to them later on when they are away from their PC.

Rather than copying data from the desktop to one's mobile, an alternative approach would be to find the relevant content again on the mobile device itself. However, there are several weaknesses here. First, the mobile device may have slow or no wireless network available. Second, finding the desired information may require the user to input a great deal of text. Third, it might not be easy to find the information again. Past studies have shown that re-finding information is quite common, but often difficult to accomplish in terms of determining the correct search terms or remembering the path that they followed to get to the information in the first place [2, 10, 14]. Combined, these factors make searching and browsing for information while mobile potentially slow and tedious.

We address these problems with *GurunGo*. GurunGo makes it easy to acquire, annotate, and share web data from a person's desktop computer with their mobile device. GurunGo acquires data *implicitly* by monitoring the stream of web pages a person visits and looking for *mobile data types*, which are kinds of data likely to be useful for a person when mobile. GurunGo can also acquire data *explicitly* by letting people use the familiar copy-and-paste metaphor to copy data from their desktop onto their mobile device. Next, Gurungo annotates the data to make it more useful when mobile. For example, for driving directions, GurunGo includes synthesized speech output. Finally, GurunGo automatically transfers the annotated data with the user's mobile device using a direct network connection like Bluetooth or USB. On the mobile device itself, GurunGo provides a user interface that lets people browse through the copied mobile data types. In this sense, GurunGo can be seen as a customized version of one's web browser history, made available on one's mobile device.

[1] This work was conducted while a student at Carnegie Mellon University.prior to joining Microsoft Corporation.

In this paper, we present the design and implementation of GurunGo, and show how GurunGo supports two different mobile data types. We also discuss our design rationale and some of the tradeoffs involved.

1.1 Use Scenario

Here, we present a user scenario to describe the functionality of GurunGo and how it would be used by people.

Alice is at home and plans to visit a local retail store to buy some repair tools for her house. She goes online and browses through several sites providing product reviews of various tools. As she does this, GurunGo detects each product page she views, clips out relevant portions, and then annotates the information with reviews from other sites. Then, in the background, GurunGo copies the annotated data plus the original web page to her mobile device. As the pages are sent to her mobile device, there is a small notification letting her know that the pages are being copied over.

Alice finds a tool that she is especially interested in, and manually copies and pastes the page to her mobile. While the page itself would have been copied just by browsing it, manually copying and pasting the page flags it and makes it easier to find on the mobile device.

Alice does not know how to get to the local retail store, so she uses her desktop web browser and looks up directions to the store. At this point, GurunGo detects that Alice is looking at a map web page. GurunGo then annotates the map web page with synthesized voice directions. GurunGo finally copies the annotated data, plus the link to the original page, over to her mobile phone as a background process. GurunGo also gives a small notification that it is copying data over to her phone.

Once she has finished browsing, Alice takes her phone with her knowing that it has directions that she can listen to while driving. Once at the store, Alice browses through a shopping list, which contains all of the product pages she viewed previously. Browsing through this list, Alice quickly finds the page she had previously flagged as being of interest. She looks over the reviews before purchasing what she was looking for.

2. RELATED WORK

We have organized related work into three categories: (1) user needs and user interaction, (2) data caching and synchronization, and (3) mobile web access.

2.1 User Needs and User Interaction

Sohn et al conducted a diary study examining user needs when mobile [13]. They found patterns in the kinds of information that people desired, with trivia, directions, and points of interest being the most common needs. Based on this work and our own experiences, we created a list of mobile data types (as described in the intro), focusing on those that would be (1) relatively easily detected and (2) things that a person would likely browse for beforehand on their desktop computers. For example, trivia would be difficult to detect and is unlikely that people would browse for it beforehand, whereas directions fits both criteria well.

From the perspective of web browser history, a number of papers have found that re-visiting past web pages is a common activity. One study found that revisiting web pages makes up 58% of all Internet browsing [14]. Another study found that revisitation is common, with 81% of web pages having been previously visited [2]. Obendorf et al broke down revisitation into four categories [10], with 72.6% of revisits being within an hour, 12% of revisits within a day, 7.8% of revisits within a week, and 7.6% revisits being after 1 week. These pieces of work suggested to us that being able to access data that had been seen on the desktop would likely be useful. However, it is important to note that these papers were with respect to desktop web browsing. To date, there has not been a study on revisitation patterns on mobile devices, or on the interplay between the desktop and mobile devices.

In a separate line of work, both Dearman and Pierce [3] studied work practices regarding multiple devices. Dearman and Pierce found that a common problem faced by people was the diffusion of information across multiple devices, including web bookmarks and histories. Karlson et al [5] also studied how people used smartphones with PCs, and found that web browsing accounted for 24.1% of all mobile activity and that people browsed far less pages when mobile than when at desktops.

Harding et al described how planning ahead can be used to facilitate the delivery of information when it is most useful [4]. People can use a copy-and-paste metaphor to prepare content, and then specify contextual triggers for when the information should be presented, for example using location or time. Our work with GurunGo has a similar rationale, though takes the position that people may not always know in advance what information they may want. For these cases, we opportunistically hoard data that a person sees in their regular use of their desktop computer, focusing on data types that are likely to be useful when mobile.

2.2 Data Caching and Synchronization

There are many commercial products that offer web content tailored for mobile devices. For example, AvantGo was a service that formatted and copied web content onto one's mobile device, which let people read web content when disconnected.

There has also been a great deal of past work on disconnected operation and caching in mobile and distributed systems (e.g., [6, 15]). Perhaps the closest work to ours is Komninos and Dunlop's work on pre-caching web content for mobile devices, based on entries on one's calendar [7]. For example, if a person had the name of a place in her calendar, one that was atypical for that user, then the system would try to pre-cache related content for maps, hotels, and so on.

Our work builds on this past work in three ways. First, we cache data that people directly interact with. Based on the papers mentioned in the previous section that people are likely to revisit web pages, we felt that caching could be a viable strategy for helping people revisit data. We acknowledge that this does not cover all scenarios, but feel that it covers enough interesting scenarios that it could be potentially useful. Second, GurunGo looks for mobile data types in the data it caches, snippets of data that are likely to be useful when people are mobile. Third, GurunGo annotates the data so that it is more useful on the mobile device, for example adding additional relevant information or improving its presentation.

2.3 Mobile Web Access

There has been a much work in improving the presentation of web content on mobile devices, all of which cannot be listed here due to space constraints. Two relevant pieces of work are Digestor [1], a proxy that performs semantic and layout compression on web pages for mobile devices, and m-Links [12], which reformats

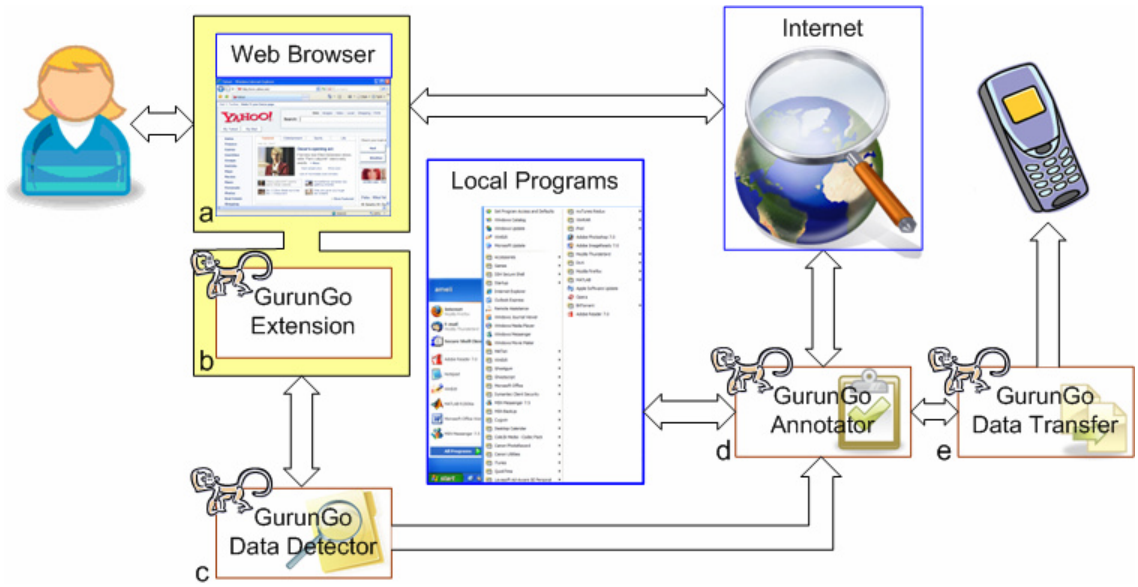

Figure 2: GurunGo system architecture: (a) The user visits a web page using a standard web browser with a GurunGo browser addon installed. (b) The addon sends newly loaded pages plus some meta-data to GurunGo's data detector. (c) The Data Detector identifies interesting data types and sends them to the annotator for further processing. (d) The annotator augments the data with extra information extracted by local programs, as well as web services, and passes the augmented data to the data transfer component. (e) The Data Transfer component copies the augmented data to the mobile device. The various GurunGo components do not obstruct the user's regular browsing behavior, providing a seamless interface with their mobile device.

content for small devices based on link navigation and on data detectors for phone numbers and street addresses.

The key difference here is that, reducing and reformatting web based content, GurunGo pre-fetches useful data based on the user's activity on her desktop PC. Additionally, Gurungo can be also useful without a wireless network connection, as it caches content locally on the device.

2.4 Data Detectors

There has been a great deal of previous work in inferring structure in text, for example, Apple Data Detectors [9], the Selection Recognition Agent [11] and, Microsoft Office XP Smart Tags [8]. Our work with GurunGo simply makes use of these known techniques and applies them to mobile computing.

3. GURUNGO OVERVIEW

In this section, we present an overview of how GurunGo works, along with our design rationale.

3.1 Acquiring Data for GurunGo

Data acquisition in GurunGo is based on detecting and processing *mobile data types*. Mobile data types are snippets of data containing information that might be useful for users while they are on-the-go. Examples of mobile data types include street addresses, phone numbers, product information and reviews, driving directions, movie times, weather, stock quotes, social events, traffic information, and flight times.

While all of these could potentially be valuable to users, for our initial prototype we focused on two specific ones: driving directions, and product details and reviews. Note that these two data types are static, in that they can be cached for days or even weeks and still be useful. Other kinds of data types may be more dynamic and require periodic updates, such as flight times and traffic information. We opted to start simple and focus on static mobile data types.

Data can be shared between desktops and mobile devices either implicitly, by copying data from web pages that people browse (see Section 3.1.1), or by having the user explicitly copy-and-paste the data to the mobile device (Section 3.1.2). GurunGo copies both extracted mobile data types as well as the original web page. The mobile data type is used for browsing for relevant information, while the original web page is used for seeing the original context of the data.

3.1.1 Implicitly Acquiring Data from Web Pages

Data is implicitly gathered by GurunGo by monitoring the stream of web pages for interesting data. Here, we discuss two issues, namely how web browser activity is monitored and how interesting data is detected.

Web browser traffic can be monitored by using either a web proxy (either local or remote) or a browser extension. The advantage of using a web proxy is that it is web browser neutral. However, we identified two disadvantages. First, a key limitation in using a web proxy is that many web pages dynamically change their content through asynchronous calls (i.e., AJAX). There is also no standard data format for these asynchronous calls, in that the data can be sent as JSON objects, XML, HTML, or a custom format, making it difficult to find and extract data. The upshot is that there may be a discrepancy between the page that the proxy has analyzed and what is being displayed to the user. Second, we had concerns that it would be difficult for novice users to understand how to setup their browser to use a web proxy. In our initial prototypes, we used a proxy-based approach, but switched to a browser extension later on because of these limitations.

Currently, we have implemented GurunGo as a Mozilla FireFox addon (Figure 2b). When a user visits a web page, the GurunGo

extension relays the content of the page plus some metadata to the GurunGo Data Detector (Figure 2c), a web service running locally on the device. This web service takes as input the URL of the page and the DOM tree for that page. The service then parses the web page and applies a series of data detectors to find and extract potentially interesting data. The web service returns a list of extracted mobile data types. We chose to decouple the parsing of web pages from the browser extension, to make it easier to port the software to other web browsers and to extend GurunGo for other streams of data, such as email.

At this point, the data detector uses one of two approaches to extract snippets of data. The first approach uses a predefined list of domains known to contain mobile data types. Associated with each domain are one or more XPath expressions that specify the path to the data to be extracted. The second approach is to scan the HTML for mobile data types. We currently use regular expressions as well as keywords that suggest relevant data. We use XPath expressions for cases that are difficult to capture using regular expressions alone, for example movie times. We use regular expressions for better coverage across web sites.

3.1.2 Copying and Pasting Data onto the Mobile

GurunGo also lets people explicitly state their interest in a particular page, by leveraging the familiar *copy-and-paste* metaphor. Data that is explicitly copied is also flagged on the mobile device, making it easier to find. We chose this design because we felt that manually copying and pasting information is a stronger signal that the data is useful.

Users are presented with a GurunGo icon, which resides in the system tray (see Figure 3). To make an explicit data transfer, users can highlight some text and do the standard copy command from their browser (or other application), and then use the paste command provided by the GurunGo system tray icon ("Send 'iPod nan…' as Shopping"). Once such an action is performed, the GurunGo data detector examines the pasted fragment for mobile data types. After these are detected, processing continues as in the case of implicit acquisition. If the user selects a fragment of text that does not contain any recognizable data types, the copied information fragment is treated as plain text

GurunGo also lets users send copied snippets as plain text over SMS if it is less than 160 characters (see Figure 3b). We opted to include this feature to let people send information to phones that do not have GurunGo installed, and to provide a basic level of compatibility in case the user does not have a smart phone.

(a)

(b)

Figure 3. (a) A user pasting the item "iPod nano" onto the GurunGo annotator. (b) The screen used to input the phone number to send SMS messages to.

3.2 GurunGo Annotator

The GurunGo Annotator is responsible for taking an extracted mobile data type and enhancing its presentation or finding complementary related information. The annotator can use web services or local programs to annotate mobile data types. Using local programs on one's PC is good for computational tasks that are too intensive for the mobile device.

We opted for this approach of augmenting mobile data types on the PC, because desktop computers have fewer constraints with respect to power and Internet connectivity, and tend to have faster microprocessors. Furthermore, since GurunGo already requires some software to be installed on the user's desktop computer, we felt that having additional software for annotating would not pose a barrier to entry.

As noted earlier, our current implementation of GurunGo supports two mobile data types: driving directions, and product details and reviews. For driving directions, we use a program on the PC to generate synthesized speech directions for playback on the mobile device. For product details and reviews, we use the Yahoo! Shopping API [16] to complement the product mobile data type with prices from multiple vendors, customer reviews and other information. Here, we implemented this service to retrieve details about the three most relevant matches to a particular item.

3.3 GurunGo Data Transfer

Once a mobile data type has been detected and annotated, it is transferred to the mobile device. In our current implementation, mobile data types and associated annotations are represented by a single folder containing the extracted data type, the original context of the data, and any annotations. We also make use of standard synchronization tools provided by Windows Mobile to copy that folder over. The specific tools used are described in more detail in section 3.5.

Currently, data is persistently stored when it is copied over and must be manually deleted. Automatic garbage collection of data is an important design issue to consider, since not all of the web content a person views will be of use. We discuss some of these issues in the future work section.

3.4 GurunGo Mobile Device Interface

Currently, GurunGo has a very simple user interface on the mobile device. GurunGo offers two levels of browsing. The first level of browsing lets users see what mobile data types there are, for example, driving directions and shopping list. Here, users can also see what items have been manually copied and pasted. After selecting a data type, users can then see what items are contained within, currently as a linear list.

Our current design is a proof of concept of an end-to-end system, as it is suboptimal and has difficulties scaling up to hundreds of entries. We discuss some possible solutions to making this user interface work well in practice in the future work section.

Figure 4 shows the user interface for driving directions. Here, GurunGo copies driving directions from Google Maps and Yahoo! Maps, and annotates the information with synthesized

Figure 4. The main screen of the Directions Application (left). A step in the directions, along with the option to play synthesized speech of the directions (right)

Figure 5. The main screen of the Shopping List Application (left), along with some more product details (right). Users can also choose to view price comparisons and reviews.

speech as an alternative to reading the directions. Ideally, this synthesized speech would be automatically announced when the user needs it, but this feature is not supported in our current implementation. Instead, a user can browse back and forth through the steps at their own pace, playing the step aloud if they so choose. Figure 4a shows the interface for selecting the directions. Figure 4b shows a step in the directions, with the option to play the text displayed as synthesized speech.

Figure 5 shows the interface for the shopping list in an emulator. Here, GurunGo has copied product information from Amazon.com and annotated the information with details from the Yahoo! Shopping service, which includes similar products, product specifications, price, and reviews from multiple online stores. This approach enables GurunGo users to walk into retail stores with a better idea of the product they wish to buy and the price range that the product is available for on the web.

GurunGo's phone interface lets users browse all products currently stored on the phone and see the details for any single one. Users can also view the specific product name, a summary description, prices, and average consumer ratings (if available).

3.5 Implementation

GurunGo is implemented as a FireFox addon. The data detector web service is implemented using C# in combination with the open source HTML parser HtmlAgilityPack v1.3. The annotator was also implemented in C# and makes use of *2nd Speech Center* for speech synthesis and the Yahoo! Shopping API [16] for additional product reviews and price comparisons. Data is transferred to the mobile device using the *Microsoft Windows Mobile Developer Power Toys* utilities, which has a utility for copying files onto a device connected to ActiveSync over Bluetooth or USB. The mobile portion of GurunGo was implemented using Windows Mobile 2003.

4. MOBILE INTERACTION DESIGN

Here, we discuss some of the design rationale for GurunGo's mobile interface.

Due to the limited screen real estate of mobile devices, and since we felt that people may have limited attention while mobile, we opted to have a minimal amount of information on each screen.

To minimize navigation between screens, details were combined when they were related and short enough (e.g. prices and ratings are included in the same screen). Since navigation across several screens was still necessary, the item which they are all related to persists at the top of the screen (e.g. the start and end address in the directions application in Fig 4, and the product query and result number in the shopping application in Fig 5). All navigation can be done through the menus. Phone joystick/arrow keys can also be used for quick navigation. The user can also use the menu or joystick/arrow keys to go back to the previous screen, which we felt was made especially necessary due to the sensitivity of many phone joysticks and navigation arrow buttons.

We also used the conventional design of having the most likely action on any screen be the left button, and the menu be the right button.

ComboBox widgets were used on the main screen for each application to afford scrolling through items in the lists. They also provide users with an alternate view in which the whole screen is dedicated to the list, allowing for more efficient selection for longer lists. On the Directions application main screen (see Fig 4a), the start and end address of the currently selected item is repeated in the labels below. Most addresses did not fit completely within the visible bounds of the ComboBox. We chose repetition rather than decreasing the font size, as a cell phone screen is already quite small, and we did not want to further compromise legibility.

5. DISCUSSION AND FUTURE WORK

In the short term, there are three things we want to focus on. First, our implementation of GurunGo is currently limited to two mobile data types. There are many other mobile data types that also need to be supported, such as traffic information, social events, movie show times, and store locations. In particular, some of these data types are dynamic in nature, making it necessary to periodically update these kinds of mobile data types. It is also worth noting that different data types need different kinds of update rates. For example, traffic reports might need to be updated every few minutes, social events every day, movie showtimes every week, and store locations every few months.

Second, as noted in Section 3.1.1, GurunGo currently uses XPath for predefined sites to extract information, and regular expressions and keywords for all other sites. This approach makes it easier to get data, but is fragile when a web site changes its formatting or layout. To be effective in practice, GurunGo would need more robust techniques for extracting data, perhaps using a combination of our existing methods as well as basic natural language processing techniques. Alternative ways to extract content could include crowdsourcing extraction of relevant data on pages to a large scale of GurunGo users, or having content providers provide snippets of information in the form of microformats or RSS feeds.

Third, we want to improve the interface for browsing through dozens or hundreds of entries. Here, we outline two ways of addressing this problem. The first is to garbage collect copied data, removing old data that is unlikely to be used. The challenge is knowing what to remove, as it may be unclear which data is no longer useful, though in some cases, data has a natural expiration date (e.g. social events and movie show times). The second is to use contextual information to sort and filter the information. One approach is to use recency, making it easier for people to see what they recently viewed on their desktop. GurunGo could also make use of one's current location to filter out locations of stores and restaurants that are too far away.

In the long term, we want to expand GurunGo to support other sources of data beyond the web. Examples of other streams of data with mobile data types include email, calendar, and contacts. Each of these sources has unique challenges in extracting relevant data and finding the best way of presenting it to users. We also want to evaluate GurunGo, to assess how often people, when mobile, use data seen on their desktop, as well as how useful automatic copying and manual copy-and-paste are for users.

Another possible interaction we are considering is mobile to desktop, where the user's interaction on their mobile can provide a feedback loop to content detection or usage of a desktop. Another possibility is pairing location with content, to surface location relevant content, e.g. surfacing the latest flight status information when close to the airport.

6. CONCLUSION

In this paper, we introduced GurunGo, a system designed to help users find relevant information while they are on-the-go, based on their interactions with their desktop personal computer. GurunGo acquires relevant information based on mobile data types that it detects in web pages that people browse. GurunGo then annotates that data with additional relevant information, and then copies the data from a person's desktop computer onto their mobile device. From this perspective, GurunGo can be thought of as a customized version of web browser history for the mobile device. GurunGo also lets people use a copy-and-paste metaphor to manually copy information to their mobile device.

7. ACKNOWLEDGMENTS

We would like to thank Kayre Hylton for her help in the initial prototype implementation of GurunGo.

8. REFERENCES

[1] Bickmore, T.W. and B.N. Schilit. Digestor: Device-Independent Access to the World Wide Web. In Proceedings of *6th International WWW Conference*. pp. 1075-82 1997.

[2] Cockburn, A. and B. McKenzie, What Do Web Users Do? An Empirical Analysis of Web Use. *International Journal of Human-Computer Studies* 2001. **54**(6): p. 903-922.

[3] Dearman, D. and J.S. Pierce. It's on my other computer! computing with multiple devices. In Proceedings of *The Twenty-sixth Annual SIGCHI conference on Human factors in Computing Systems (CHI 2008)* 2008.

[4] Harding, M., O. Storz, N. Davies, and A. Friday. Planning ahead: techniques for simplifying mobile service use. In Proceedings of *The 10th workshop on Mobile Computing Systems and Applications (HotMobile 2009)* 2009.

[5] Karlson, A.K., B.R. Meyers, A. Jacobs, P. Johns, and S.K. Kane. Working Overtime: Patterns of Smartphone and PC Usage in the Day of an Information Worker. In Proceedings of *Pervasive 2009* 2009.

[6] Kistler, J.J. and M. Satyanarayanan, Disconnected operation in the Coda File System. *ACM Transactions on Computer Systems* 1992. **10**(1): p. 3-25.

[7] Komninos, A. and M.D. Dunlop, A calendar based Internet content pre-caching agent for small computing devices. *Personal and Ubiquitous Computing* 2007.

[8] Microsoft, Microsoft Office XP Smart Tags. http://www.microsoft.com/technet/prodtechnol/office/officexp/maintain/xptags.mspx

[9] Nardi, B.A., J.R. Miller, and D.J. Wright, Collaborative Programmable Intelligent Agents, *Communications of the ACM*, vol. 41(3): pp. 96-104, 1998.

[10] Obendorf, H., H. Weinreich, E. Herder, and M. Mayer. Web Page Revisitation Revisited: Implications of a Long-term Click-stream Study of Browser Usage. In Proceedings of *CHI 2007*. pp. 597-606 2007.

[11] Pandit, M.S. and S. Kalbag. The Selection Recognition Agent: Instant Access to Relevant Information and Operations. In Proceedings of *International Conference on Intelligent User Interfaces*. Orlando, FL. pp. 47-52 1997.

[12] Schilit, B.N., J. Trevor, D.M. Hilbert, and T.K. Koh. m-links: An infrastructure for very small internet devices. In Proceedings of *MOBICOM 2001*. pp. 122-131 2001.

[13] Sohn, T., K.A. Li, W.G. Griswold, and J.D. Hollan. A diary study of mobile information needs. In Proceedings of *The twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI 2008)* 2008.

[14] Tauscher, L. and S. Greenberg, How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. *International Journal of Human Computer Studies* 1997. **47**(1): p. 97-138.

[15] Terry, D.B., A.J. Demers, K. Petersen, M.J. Spreitzer, M.M. Theimer, and B.B. Welch. Session Guarantees for Weakly Consistent Replicated Data. In Proceedings of *Proceedings International Conference on Parallel and Distributed Information Systems*. pp. 140-149 1994.

[16] Yahoo! Yahoo! Shopping Web Services. http://developer.yahoo.com/shopping/