MOBILE INFORMATION MANAGEMENT: INVESTIGATING PACES OF INTERACTION USING MOBILE DEVICES

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF SCIENCE

AT THE UNIVERSITY OF CAPE TOWN

IN FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Richard Schroder

November 2005

Supervised by

Professor Gary Marsden

Abstract

We feel that designers of mobile software are too intent on converting functionality directly from desktop systems to mobile systems, ultimately neglecting an important consideration – the mobile nature of the devices. Mobile computing, by necessity, introduces its own paradigm of interaction and design complexities. Rather than trying to re-create miniature desktop systems, designers should focus on creating *mobile software*: software that can tolerate intermittent connectivity and does not require full cognitive focus from the user. But once users have information on their mobile devices, what do they do with it? How do they work with it? How do they search for information whilst mobile?

In this dissertation we introduce the concept of *laid-back interaction* to handle the user's varying attention levels as they use the mobile device within their every-day working world. *Information packages* are used to manage the information stored on the device and to manipulate search queries, their results and any other data users may wish to access. Not only will we consider the role of information access within the user's personal world, but also the role of the mobile device itself: How do users perceive these mobile devices? Do they have an effect on the user's work patterns? Do users see them as "personal" devices, like cellphones? A search query management system was integrated into the information packages, enabling us to investigate how users work with their information whilst mobile. These systems were implemented on both the desktop computer and the mobile device with seamless synchronisation taking place, allowing users to work with information anywhere.

Through heuristic evaluation, task-based observation and real-world observation we evaluated our system's suitability to mobile devices in terms of both interface and interaction design. These results were then used to investigate the questions posed and test our belief that laid-back interaction is valuable to the designers of mobile software. From these experiments we found that while users feel laid-back interaction is beneficial, the concept of information packages is not.

Acknowledgements

I would like to thank Professor Gary Marsden for not only providing the inspiration for this research, but also for his tireless assistance, guidance and patience during the long process. Thanks must also go to Dr Matt Jones of the Waikato University Computer Science Department, New Zealand. To Marshini Chetty, David Nunez, Dynal Patel, Professor Edwin Blake and my colleagues in the Collaborative Visual Computing Laboratoy in the University of Cape Town Computer Science Department, thank you for your input, ideas and criticism.

Acknowledgements must also go to those who provided funding and equipment for this research, namely: Siemens, Telkom, THRIP, HP, Microsoft, Bridges.org and the National Research Foundation. A big thank you to you all.

To all the friends and family who provided valuable proof-reading services – thank you for your support!

Richard Schroder
February 2005

Contents

Chapter 1

Introduction

With the rapid development and evolution of network technologies, information is fast becoming available in ways and places that people could not have imagined even 10 years ago. Coupled with this growth in information availability has been the development of search engines and search technologies designed to make finding and visualising information easier and quicker. At the same time, mobile technology has been rapidly evolving with "smart" devices being developed to allow people to access and work with their information in almost any environment. Devices such as Personal Digital Assistants (PDAs), cellphones and Tablet PCs are all designed to allow remote access to information through some wireless medium.

In particular, the development and spread of wireless technologies, such as 802.11b [IEE04] or even GPRS [Buc04] can be credited for the growing popularity of mobile computing. As people realise that they are able to access their information remotely, so they begin to expect and demand these capabilities. However, a concern is that designers of mobile software are concentrating on directly converting the functionality of desktop computers to mobile devices, rather than realising that mobiles introduce their own paradigm of interaction. Mobile devices introduce their own constraints that arise, not only from their physical size and processing limitations, but also from their personal nature as they become part of the user's everyday life [HR02]. In particular, mobile software for the most part fails to take into account that wireless network access is often sporadic at best due to external factors (such as bad weather or obstructions) and that connectivity is not a given. Software that requires and assumes the presence of a network connection becomes terribly limited once that connectivity is lost. We believe that for mobile software to become truly

effective and "usable" for the average user, the influence of connectivity upon functionality should be as limited as possible.

A question that needs to be asked by developers is "how can we take advantage of mobile devices to provide greater ease of information access?" Rather than being seen as a limiting factor, device mobility should be seen as an advantage – new functionalities and roles now become available. An example of this can be seen in the Short Message Service (SMS) provided by cellular phones; originally seen as an experimental messaging method, it has established itself as a communications channel in its own right[Rhe04].

Developers should rather focus on new ways of using these devices and to think "outside of the box" that is the current desktop computing model. Not only do we need to consider how to provide novel forms of information access to mobile users, but also how these users would *manage* this information once they have it. With the growing storage capacity of mobile devices in the form of "flash" memory, device capacity is no longer a concern for the developer and the potential for carrying large amounts of information exists. However, new problems, such as visualisation and manipulation of this information on a small screen device arise. As the hardware of mobile devices evolves, developers need to begin concentrating on interface design and how users interact with these devices, as opposed to concentrating purely on system design.

An example of the information need that could be associated with mobile devices is presented:

> "John is attending a conference where wireless access is available, but only in certain selected "zones." Whilst attending a session which is not within one of these wireless zones, the speaker makes a comment about a new technology that is currently under development. Using his PDA, John enters a search query to search for that technology and resumes listening to the speaker. Later, whilst in the foyer area, he realises that he walked through a wireless access zone on his way out of the session. Pulling out his PDA he sees that several results have been retrieved – he selects a few of the results and instructs the PDA to base more searches on these results. He then resumes his conversation with a colleague."

The scenario presented above is an example of what we feel is the ideal interaction between a user and their mobile device. At no time was John's interaction method dictated by the

state of the system – the only limiting factor here was the presence of the network. John was always in control of his *pace of interaction* with the device and was able to resume listening to the seminar without interruption. With existing systems on mobile devices, a more realistic scenario would probably have been that he would have made a note on a piece of paper to search for the information. Later, after the presentation he would have headed across to a wireless zone and begun searching using the PDA's web browser. However, the time wasted whilst entering the queries and waiting for search results meant that he was unable to converse with some colleagues by the coffee table, thus possibly missing out on some valuable information.

Furthering the scenario, John would want to *work* with the information that he has retrieved using his mobile device. Owing to the constraints of the mobile device he would prefer to work with this information on his desktop computer at home and thus uses some synchronisation mechanism to transfer the information. As he works with this information his mobile device is kept up-to-date, allowing him to carry the most current version of his information with him.

Whilst potentially extreme, the above scenarios illustrate the issues that can be associated with mobile device usage in the everyday world. It also introduces an interaction concept adopted for this dissertation, termed "Laid-back interaction" [JJBC04]. This style of interaction allows the user to interact with a system when the *user* desires, as opposed to when the system *demands* it. We feel that this method of interaction is suited to mobile devices as it allows users to work as they wish, when they wish, with minimal distraction. Further, it highlights the need to provide some form of information transport or synchronisation mechanism between the mobile device and the desktop computer. This interaction style not only empowers the user to overcome technological problems (such as connectivity), but also *social* issues, such as the appropriate moment when to use the device.

In the scenario, it was mentioned that John creates some new queries based on the results returned to him by the original search. This illustrates an important point – very rarely is a user's information need satisfied by a single query. Rather, search queries tend to evolve over time with a variety of information sources eventually satisfying the user's information need. What users need is a way of managing these "evolved" searches and their results. Information packages, as presented by Twidale et al [TNP97], would appear to be the best solution for this problem. Users can then store multiple search queries and their results in a single location for easy access. It is important to note that this research

is not focussed on developing new web search technologies or new web search algorithms – these are well researched topics that form a field in their own right. Rather, the focus is on how mobile users can work with information from existing search engines.

Combining the concepts of information packages and laidback interaction gave us the inspiration for our information management system design. We visualised a system that allows users to store and transport a mixture of information types on their mobile device, whilst also working with that information on their desktop computers. The system would provide some form of synchronisation mechanism to allow users to seamlessly work with information on either the mobile or the desktop, as they wish. Our rationale behind integrating mobile devices and their desktop systems is that people want to *work* with information, not simply view it. The physical limitations of mobile devices simply do not allow users to effectively view and edit information. More importantly, this system would work to minimise problems caused by sporadic network connectivity and would be designed to function as normal regardless of the connection state. Mostly, the focus of the system would be on performing searches in an on- / off-line manner, allowing users to carry on with their tasks at hand in the most transparent manner possible. This will allow users, such as John in the scenario presented earlier, to create searches and manage information on their mobile devices without paying undue attention to other issues such as connectivity. This, in our opinion, will help to reduce the burden placed upon the user by these external influences.

Not only do we aim to integrate the ideals of the information package concept and laidback interaction, but also concepts from the field of information management. These concepts include those of how users manage their documents, how users search for information and how users' work patterns are influenced by the tools that they use on a daily basis. Inspiration is drawn from many diverse fields, including research into user search behaviour whilst using Online Public Access Catalogs, document management and the influence of email on users. Examples of these research works will be presented in the next chapter.

Once developed, the system was be exhaustively tested to support our belief that laidback interaction is indeed suitable for mobile computing. We proposed three stages to test this system:

- Heuristic evaluation – ensures that the interface is consistent and doesn't spring any unexpected surprises upon the user

- Task-based analysis – to check that the system is "usable" and performs as expected
- Observation-based analysis – to determine just how useful the system is and the validity of our claims regarding laidback interaction

Each stage of testing was designed to support the following stage, with the final stage indicating whether or not people find laidback interaction useful when working with mobile devices. The first stage, heuristic evaluation, was designed to iron out any potential inconsistencies within the user interface. Once the user interface was consistent, it was then tested on users through task-based analysis to ensure that the interface is indeed usable for first-time users with no help or support. Thus, during the final stage it can be assumed that any problems or benefits the system presents arose from the interaction style as opposed to the interface itself.

The rest of this dissertation is composed of 7 parts – in the next chapter, background research will be presented. Following that the system design and ideas will be presented. The actual implementation of the system will be presented in a chapter of its own. The experimental design and then the results will follow, with our conclusions being presented in the last chapter.

Chapter 2

Background Information and Research

While much research has been performed in the fields of desktop-based search and mobile search, it has largely focussed on always-on network connectivity and has ignored the uncertainty of the mobile user. To date, no in-depth research could be found that investigated search using mobile devices and considered the issues posed by sporadic network connectivity. Also, mobile search solutions that have been presented have neglected the role of the desktop computer, providing no simple and effective way of sharing those search results between the mobile device and the desktop computer.

Mobile computing technology has evolved in two directions – hand-held devices and laptop or notebook computers [LK93]. Through the rest of this dissertation the term "mobile device" will be used to refer to hand-held devices, such as PDAs.

Over the rest of this chapter, the issues surrounding the usage of mobile devices, mobile search and desktop search systems will be presented.

2.1 Ubiquitous computing and its implications

Ubiquitous computing was first described by Weiser (1991) [Wei91] whilst researching at Xerox PARC. The main focus of his work was to replace the personal computer with embedded devices within the spaces that we inhabit, thus moving interaction with digital information into the personal lives of users. It quickly became apparent that as devices moved into the personal realm care would have to be paid to their design in order to avoid

being obtrusive within the user's environment. This led to the concept of *calm technology* [WB96], which is the notion of technology that continuously presents information to us in an unobtrusive manner. To illustrate this concept, Weiser uses the example of car engine noise. Whilst driving a car we are focused on the road but should there be an unexplained change in the sound of the engine we would immediately become aware of it [WB96]. Weiser & Brown [WB96] describe this as something that engages the *periphery* of our attention. Calm technology, they say, engages both the center of our attention and the periphery, continuously moving between the two as appropriate. The goal of calm technology therefore, is to integrate technologies into our lives in such a fashion that they remain unobtrusive whilst constantly feeding us with information.

Small screen device research has tended to concentrate on innovative ways of interacting and displaying information to the user and furthering the spread of ubiquitous computing. However, mobile device design differs from desktop-based design in that mobile devices are private and personal items – only one person uses a mobile device at a time and very rarely is it shared. Over time, due to their small size and portability, mobile devices become integrated into a user's life, or become part of their *lifeworld* [HR02]. Hallnas & Redstrom (2002) [HR02] argue that as computing devices become integrated into people's personal lives and as people *invite* things into their lifeworlds, so designers must not only focus on usability, but also on *presence*. Presence in this instance refers to the aesthetics of the device, how well it blends into the user's world and what *expressions* it gives. All objects present themselves through expressions, or physical appearance, and the first invitation by a user must have something to do with its appearance [HR02]. This is also a reference to Weiser's concept of calm technology and the need for unobtrusive integration of technologies into our lives. As Weiser & Brown (1996) state:

> "If computers are everywhere they better stay out of the way, and that means designing them so that the people being shared by the computers remain serene and in control. ... But too much design focuses on the object itself and its surface features without regard for context. We must learn to design for the periphery so that we can most fully command technology without being dominated by it. [WB96]"

These concepts as presented are valid not only for physical device design but also for mobile software design. Software that causes a device to beep for attention or which requires

continuous monitoring can quickly become irritating when the users feel that the demand is unwarranted. As mentioned in the quote above, context of use must be taken into account by the software. This includes not only the physical context of use, but also the infrastructural and system context of use [DRD+00]. Software on a mobile device that is unaware of its network connectivity status and continuously requests some network resource of the user is not of much use and becomes disruptive – anything but calm. This forces the software to remain at the center of the user's attention when they may be attempting to continue with some other task at hand. A well-designed user interface will be able to cope with the uncertainty introduced by wireless communications and manage the user interactions as appropriate [DRD+00].

Ebling *et al* (2002) make the observation that the demand a mobile (or indeed, any) system places on a user's time is a threat to the system's usability [EJS02]. This time is not only the time spent using the system, but also the time taken to learn how to use the system and the time spent waiting for the system to produce the desired results.

2.2 Mobile software design considerations

Interface design for mobile devices is further complicated by their physical restrictions. Limited screen area, slow data entry methods and reduced processing power are just some of the issues that face designers over and above the design ideals presented in Section 2.1. Part of the interface problems, we feel, stem from the attempt to directly translate functionality from the desktop computer to the mobile. As early as 1993, Landay & Kaufmann [LK93] pointed out that although running the same applications on the two systems may be useful and desirable, running the same environment is not only undesirable but also mostly impossible due to the processing limitations of mobile devices. Mobile devices typically present interfaces that are specialised for a small number of tasks, and if well designed, will consist of limited feature sets [LEF+00]. The greatest challenge facing designers in today's wireless world, however, is that of handling communications and maintaining the illusion of connectivity even when it is poor or non-existent [EJS02].

Distributed systems have been accorded a lot of attention with regard to mobile devices in recent years and one of the leading file systems in this field is Coda [SKS87]. Developed at Carnegie Mellon University, it is designed to provide remote access to files in environments where connectivity can be poor or even non-existent. The illusion of connectivity

is maintained through sophisticated caching and document pre-fetching methods, which store the files on the mobile device. Its main design goal is to make usage of the system as transparent as possible and to remove connectivity concerns from the user. Ebling *et al* point out, however, that complete transparency is not necessarily a desirable goal [EJS02]. Sometimes, due to circumstances such as varying connection speed, the system may behave in a manner that conflicts with the user's expectations and causes confusion. As a compromise and a solution to this problem, Ebling *et al* propose a system design of *translucence* as opposed to *transparency* [EJS02]. With translucent design, the system exposes critical information to the user whilst continuing to hide non-critical information. This affords the user an extra degree of control and helps to reduce any potential confusion.

Compounding the issues of wireless computing and mobility is the concern of power management. Mobile devices, by nature, have to conserve battery power as much as possible in order to extend their time "on the road." Whilst manufacturers strive to produce devices that are as energy-efficient as possible, software design also has a large influence. Software which prevents the device from powering off or makes excessive use of the wireless network capabilities of the device can greatly reduce the battery life, necessitating frequent recharging and thus reducing the usefulness of the device to the user. Unlike desktop systems, mobile devices are never truly "switched off." Rather, when the user presses the power button the system is placed in a frozen (or doze) state, allowing it to perform an "instant switch on" when the user switches the device back on. With a device that is not allowed to completely drain its battery, it is entirely conceivable that software on the device could run for days, weeks or even months. Thus, long term software stability is crucial to the success of mobile software.

2.3 Search using mobile devices

Most research relating to mobile devices and search has tended to concentrate on the interface design, input methods and how to visualise the results, with very little focus on what to do with these results once they are fetched. The most notable work regarding mobile search has been conducted by Buyukkokten *et al* (1999) [BGMP99] and Buchanan & Jones (2000) [BJ00]. Buyukkokten *et al* are part of the Stanford Power Browser Project [POW04] which aims to address the problems of interacting with the World Wide Web through handheld devices. Whilst both of these projects focussed on web interaction using mobile devices, they

have been modified to take into account web-searching through mobile devices. Research into web interaction using mobile devices has followed two main approaches: reformatting the page to ensure that it fits the small screen, or parsing the page and creating a hierarchial view of the page structure. The work presented by Buukkokten *et al* and Buchanan & Jones has followed the latter approach, as it allows users to customise their view of the page as they see fit. Chang *et al* (2002) presented a system where not only were spoken queries accepted, but automatic title summarisation was employed to shorten document title lengths [CMLF02]. This summarisation was introduced to reduce text wrapping and the amount of scrolling that users have to perform.

Buchanan & Jones found that users of mobile devices show a significant preference for "direct access methods" (or searching) as opposed to following long browsing paths [BJ00]. In other words, when using mobile devices, users preferred to make use of search functionality rather than exploring a hierarchy. This finding applied not only to web site browsing but also to file manipulation on the mobile device and offers some indication to the importance of search on mobile devices.

2.3.1 The Power Browser project

The Power Browser project utilises a combination of tools to ease web browsing using mobile devices. The system aims to decrease the amount of navigation that users must perform when searching for information [BGMP99]. This objective is realised through three main tools:

- **Link Navigation:** This tool provides the ability to navigate without viewing the full page. To achieve this, all the links in a page are extracted and displayed in a tree form. Users can then navigate through this tree and once they are reasonably certain that they have found the page they need, they can switch from the link view to a text view [BJ00].

- **Local site search:** The system provides local site search, even for web sites that do not provide this facility. This local search facility generates an index that is increasingly complete as users spend longer periods of time visiting a site [BJ00].

- **Keyword entry support:** This is one of the main interface enhancements that the Power Browser system offers. As the system indexes the site currently being visited,

it also indexes the words within the site. Whilst the user enters search terms, site-specific keyword completion is offered. Also, the system provides a real-time estimate of how many hits the search term can expect to return.

Whilst the Power Browser project achieved significant improvements in web access through PDA's [BGMP99], it does have the drawback of requiring a proxy server. The proxy performs the caching, fetching and indexing of pages on behalf of the mobile device. The requirement of the additional hardware and software reduces the *transparency* [MNS00] of the system, decreasing its appeal to novice users.

Figure 1: Screenshots of the Powerbrowser system in use

Figure 1 shows the three main aspects of the PowerBrowser system. From left to right: the link navigation aspect, local site searching and word completion. All these operations were being performed upon the Stanford Database Group website.

2.3.2 Visualisation and presentation of search results: WebTwig

Buchanan & Jones' work focuses on the *visualisation* of the search results, rather than the speeding up of the search process as a whole. Their work formed part of the development of WebTwig [JMMNB99], a project which has a similar approach to the link navigating aspect of the Power Browser project. WebTwig creates tree-based outlines of web sites, in a similar fashion to the outline view a word processor creates of a document. This tree representation of the web site is text-based, which suits the typically limited graphical abilities of mobile devices.

It is recognised that searches for information usually generate long lists of results, which are impractical to view on mobile devices due to their limited screen size [BJ00]. WebTwig was adapted to deal with explicit search queries and their results. Top level nodes within the

tree are hits returned by the search engine – expanding these nodes reveals the alternative options within these hits to the user. However, WebTwig does have problems extracting an outline from sites that do not have a meaningful structure and rectifying this problem is ongoing work [BJ00].

Not only do mobile devices present a challenge for the visualisation of results, but also for the entry of search queries. Limited input facilities makes data input a cognitively demanding process, which could potentially distract the user from their task at hand [HR02].

2.4 Document management on mobile devices

Due to their small screen size and limited input capabilities, mobile devices are unsuited to the creating or editing of large documents. However, they are capable of being used as document viewers, although new visualisations are often needed, such as when viewing web pages. Complicating matters for users is the complexity of using the mobile file system as opposed to that provided by desktop systems – only one directory can be viewed at any time without any display of the directory hierarchy and the distinction between different store types (main memory, storage card or static memory) only serves to confuse users further.

2.4.1 The Satchel System

A frequent problem facing mobile users is the wide variety of document types and environments that they may encounter. Satchel aims to provide "streamlined access to documents and document services" through "Satchel-enabled" devices [LEF+00], as opposed to actual accessing of documents. Satchel-enabling devices simply involved fitting them with InfraRed receivers to enable the mobile device to communicate with these devices (such as printers, scanners, faxes or even desktop PCs) in order to share documents and services. Straightforward viewing of documents is possible using the system should the user wish to do so, although it is not the main focus of the system. Due to the limitations of the mobile device used (a Nokia 9000 Communicator cellphone), the document being shared is never stored on the device itself. Instead, *tokens* are passed, detailing the location of the document on some PC [FPJ+00]. These tokens are encrypted using public/private key algorithms to ensure the validity of tokens and to reduce the possibility of unauthorised access to documents.

The service is context sensitive, adapting to the device it is interacting with. For example, retrieving services provided by a Satchel-enabled printer will return a list different from that returned by a Satchel-enabled fax machine. The need to physically modify hardware in order for it to interact with the Satchel system can be seen as a limiting factor, although these modifications were not expensive. A 6-week trial was conducted using 26 users and it received a positive response from the users involved [LEF+00], which consisted mainly of Xerox PARC employees.

2.5 Mobile network connectivity

Increasingly, mobile devices are being produced with access to remote data in mind. Technologies such as 802.11b [IEE04], Bluetooth [BLU04] and even InfraRed are allowing mobile devices to perform remote operations with greater ease. However, the portable nature of mobile devices means that these communication media are inconsistent in connectivity and behaviour. As such, the systems designer must regard them as unreliable and at all times consider the presence of a network connection as being questionable.

Mobile devices are continuously evolving, although devices being produced nowadays are typically fitted with up to four different communications media:

- InfraRed – operates on line of sight, slow speed
- USB/Serial docking interface – used to synchronise device with its host PC, slow speed
- Bluetooth – Used for ad-hoc Personal Area Networks (PAN), fast
- Wireless LAN – can be used for any communications, fastest

Unlike *fixed hosts*, or devices physically connected to a network, mobile devices are not simply either connected or disconnected – they can be *fully connected*, *partially disconnected* or in *doze mode* [PB94]. The device enters a partially disconnected state when its battery power is running low or the wireless signal strength is very low. Doze mode is entered to conserve battery power when the device is not busy sending or receiving data. Due to the frequency of partial and total disconnections, they should not be treated as failures, but rather as temporary outages [PB94]. Indeed, Pitoura & Bhargava [PB94] point out that a mobile host should be capable of operating with a weak or non-existent network connection without excessively inconveniencing the user.

2.5.1 Interaction pace and connectivity

Jones *et al* put forward the idea of using *laid-back* interaction whilst searching for information with mobile devices [JJBC04]. They define laid-back interaction as being an interaction pace which is somewhere between *sit-forward* and *lean-back*. Sit-forward is interaction where the user is actively engaged in any activities, such as browsing web pages or using a spreadsheet. Lean-back, on the other hand, is at the opposite end of the interaction spectrum. The user is simply a passive observer and is not involved in the task at hand as information is "pushed" to them [JJBC04]. Laid-back interaction would allow the user to interact with a device as they wish, when they wish as opposed to when the device decides it is appropriate.

An implication of this style of interaction, particularly when applied to searching, is that the user need not explicitly worry about network connectivity. In fact, the only criteria that will limit their access to data (and indeed, to the interaction pace they desire) will be the availability of that data.

Given the mobile nature of handheld devices and the personal and infrastructural contexts in which their interactions take place, laid-back interaction is more than suitable. In the scenario of use presented in the previous chapter, the user, John, can be observed moving between the two extremes of interaction pace in a fluid manner. His attention was fully focussed upon the device whilst entering the search query (sit-forward) and he then ignored the system whilst waiting for the results (lean-back). Later, when the results had been retrieved, he both interacted with them and ignored the system at his own leisure (laid-back).

2.6 Search on desktop systems

Web and network-based search on desktop systems has been the subject of intensive research for the last 10 years, with many searching algorithms and strategies being developed. More recently, research in this field has shifted towards a collaborative focus with emphasis on sharing search results and search queries. Although the focus of our work is not collaborative, the potential for *self collaboration* by users makes these collaborative systems of interest. Self collaboration takes place when users review what they have previously found and assess how useful it is for their current information need [MNS00]. This follows a process similar to that of "normal" collaboration, with the exception being that the user keeps

the results for their own usage.

Whilst investigating Online Public Access Catalogues (OPACs), Twidale & Nichols (1998) observed that information searching forms a part of people's larger work activities and thus generally involves interactions with colleagues [TN98]. These interactions, or collaborations, are largely co-present in nature and include recommendations, sharing of search tactics and informal explanations of how the search system works. Indeed, although most existing search systems ignore the collaborative aspect of search, people use them in a collaborative fashion nevertheless [TN96]. As the amount of online information continues to grow and access mechanisms evolve, it becomes apparent that many (novice) users will need help finding information [TN96]. This information overload is taking place due to the amount of information growing at a rapid pace, along with the variety of information types and forms [TN98].

Collaboration between users can involve not only the sharing of results, but also filtering of results, recommendations of results and the indexing of results. Collaborative filtering of search results has become a widely researched field in its own right and has resulted in projects such as Tapestry [GNOT92] and Porqpine [PSB03].

	Co-located	**Remote**
Synchronous	book issue	telephone
Asynchronous	post-it notes	email

Table 1: Activities within a library in different spatial and temporal dimensions [TN96]

These collaborative actions can take place in one of four different "spheres," as highlighted by Twidale & Nichols (1996). Table 1 illustrates the collaborative channels that they observed people making use of within a library. The researchers believe that with the development of web technologies most collaborative search behaviour will move from being co-located and synchronous to being asynchronous and remote [TN96, Bat89]. Twidale *et al* (1997) point out that these developments are the most critical – remote searching will make traditional collaborative activities rarer due to the loss of physical proximity to other researchers [TNP97]. However, whilst traditional collaborative activities may be threatened, digital libraries open up new opportunities for collaboration. These include the ability to share the complete search process, collaborate with other researchers world-wide and to share the results of searches with other colleagues in a quick and easy fashion.

As presented by Bates (1989) [Bat89], searching is no longer a simple linear activity driven by an information need. Rather, it is similar to the browsing process, with users retrieving information from a variety of sources [Bat89, TNST95]. Bates goes on to liken the process to *berrypicking* in that the user collects small pieces of information from a range of sources to satisfy their ever-changing need [Bat89]. An important idea here is that a user does not have a fixed information need – as they uncover information, so their needs and search processes change [Bat89, TN96]. Thus, search tools need to take into account that a particular search for information is not fixed and that the user's queries and goals are constantly shifting. The challenge for designers is to create systems which not only acknowledge the presence of collaboration, but also the berrypicking behaviour of the users.

2.6.1 The Ariadne project

Twidale & Nichols (1998) recognised this berrypicking behaviour of users and noted that not only do users gather information from a variety of sources, but also their search process *evolves* based on the results of previous search actions [TN98]. This evolution affects not only formation of the search query, but even the user's search goals. Bates (1989) states that "information seeking is a multi-staged process" [Bat89] and this is highlighted by the evolutionary process of information seeking. However, it is difficult for users to remember actions taken when they are focussed on their search goals [TN98]. This led to the creation of the Ariadne project [TNST95], which aims to create an easily-shared record of a user's search process, along with the results of that process. This project was developed with OPAC accessibility in mind, although its principles can be applied to web-based searching. It ties together two approaches: visualisation of the search process and enabling collaborative browsing / searching.

The system captures input from the user and the resulting output from the database and creates a search history made up of these *command-output* pairs [TNP97, TN96]. This system works for any text-based interface and allows users to proceed as they normally would, with the new method of working only becoming apparent when the user "plays back" their search history [TNP97]. Visualisation of the search process is graphical, using playing card-size thumbnails of the outputs. Clicking on a card expands it to full size. The cards are arranged on three levels, with the vertical position of the card representing the semantic of the card (menu choice, specifying a search or looking at a search result). The horizontal position of a card gives its sequence in time.

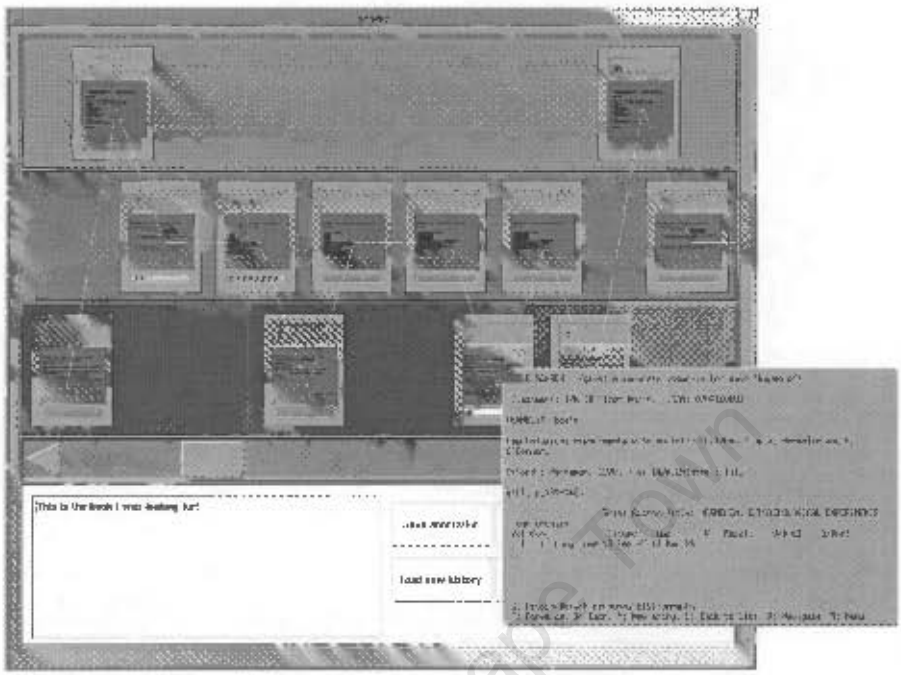


Figure 2: Screenshot of search being performed using the Ariadne system

Figure 2 shows the "three tiered" approach used by Ariadne. The top level represents main menu screens. The middle level shows the search result lists generated, with the bottom level displaying an individual search result. Clicking on one of these results displays a pop-up picture of the result.

For users working alone, this record of their search allows them to uncover errors that they may have made whilst searching, to re-use queries and it also serves as a reminder function [TN98]. Collaboratively, it aids co-located collaboration in that it provides an overview of the search process followed by another person. Remote collaboration can be achieved simply by sharing the command-output definitions.

Studies with the Ariadne system have found it useful for teaching novice users how to use the search system, as well as informing their understanding of the search process [TNP97]. Twidale *et al* (1997) have only tested the system for use in synchronous, co-located collaborative scenarios to date, although they are investigating remote, asynchronous possibilities for the system. In the context of web searching, the system is of interest due to the manner in which it allows users to review and modify their search processes. Its simple layout allows novice users to immediately visualise the processes taking place and to review their progress

in satisfying their information need.

2.7 Information packages

As user's search goals and process evolves, their behaviour becomes increasingly one of *browsing* as opposed to simple search [TNST95]. The visualisation presented by Ariadne supports this browsing process in that it allows users to return to their original search path should they become distracted. Twidale *at al* (1997) listed some of the ways that people can share the results of searches or obtain help searching for information:

- Saving the search result and sending it to someone directly
- Annotating stored searches
- Notifying searchers with similar goals
- Creating an *information package*
- Rating a search query and / or result set

Of these methods, the information package concept is one of the most interesting. Information that is retrieved during the course of a search or a number of searches can be made available for retrieval for others. This information could consist of not only bibliographic references, but also multimedia objects [TNP97]. Such an approach also raises the possibility of browsing for not only information, but also for people as sources of information [TNP97].

Potentially, an information package could be seen as a complete solution to a user's information need. Contained within the package could be search queries and their associated results and documents or multimedia objects. This concept would allow users to not only have all the information that they require at hand, but also a way of determining how they found that information. From a collaborative perspective, information packages have great potential in that users could simply share a complete package with other researchers as opposed to sharing the search process, requiring the work to be repeated.

2.7.1 Organising information

In a vein similar to the information package idea, Lifestreams [Fre97] aims to provide a new organisational structure to people's documents. The system arranges every document

a user creates in a chronological "stream," replacing the conventional hierarchial file and directory structures. The "tail" (or furthest end) of the stream consists of documents from the past and progresses to more recent documents and eventually into the future [FFG96]. Future streams contains documents that the user *will* need, such as reminders, calendar items and to-do lists. An interesting aspect of the system is its ability to create *substreams* from the main stream – these substreams can be results of searches through another stream, filters applied to a stream or simply organisational structures created by the user [FFG96]. The Lifestreams system's concept is very similar to that of information packages presented above in that it provides a way for users to group and easily access their information.

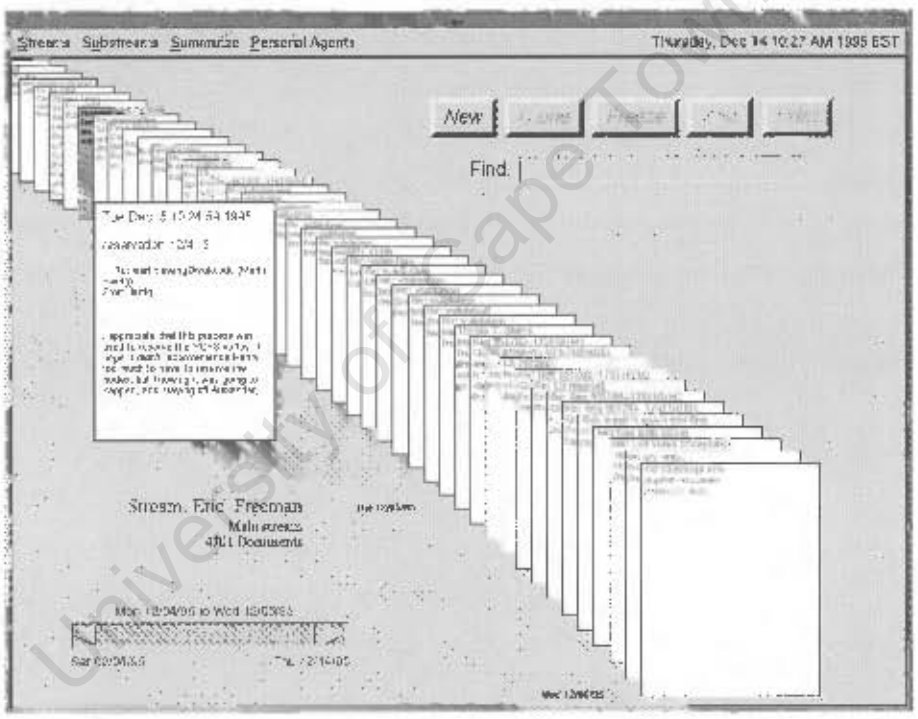


Figure 3: Screenshot of Lifestreams interface

Figure 3 shows a user's main stream with all their documents. Documents at the "tail" end are older, with newer documents near the "front" of the stream.

Studies by Cole (1982) [Col82] and Malone (1983) [Mal83] into how office workers organised their desks offer some insight into user behaviour in today's computer systems. They found that the way that people organised their desks often served as a reminder function

[Mal83] – in a similar fashion, people often leave files clustered on their electronic desktops in order to remind themselves to work with them and to make them easily available [RSK04]. Ravasio *et al* (2004) found that people frequently struggle to find and manage information within their personal space and thus tend to spatially groups files where they can be seen (ie. on the desktop) [RSK04]. A mitigating factor here is that users find directory hierarchies confusing and difficult to learn [MC03] and correspondingly, often have difficulty classifying their information [Mal83, RSK04]. Users are used to working within a document space and struggle with the file space structure that is imposed upon them by the operating system [DELS99].

2.7.2 Document space versus file space

An ideal system would provide two working "spaces" - a user space that would contain all their documents and personal files and a separate system space that contains all the system files [RSK04]. This distinction would help to prevent the confusion that users often experience in trying to find their documents (which are organisational entities with social meaning and practical use) among files (which are simply computational artifacts with some technical structure) [DELS99]. Presto, a document management system forming part of Xerox PARC's Placeless Documents project, aims to enable people to work with their documents as they do within the real-world [DELS99]. The system allows users to assign attributes to documents, to search by those attributes and to place documents in "collections" which are analogous to Malone's physical file *piles* [Mal83]. Collections can be *open* or *closed*, with a closed collection being represented graphically by an icon showing a pile of files, the height of which is relative to the size of the collection. The management of objects is coherent in that collections are treated as documents, allowing collections to be subjected to all the operations that can be applied to documents. Using document attributes, interaction by users is based on meaningful properties as opposed to those of the hierarchy the document is stored within [DELS99].

Freeman argues that storage and visualisation of documents should be transparent, removing the need for users to work with file systems and file naming conventions. Just as people don't expect to name or title a piece of paper when they make a note, so people shouldn't have to name documents that they work with [Fre97]. Raskin (2000) goes as far as stating that the need to name and create files interrupts users' workflow and creates potential distractions from the task at hand ([Ras00], *pgs 117-118*). According to Ravasio

et al (2004), all information should be handled in the same manner due to users perceiving different information types such as files, emails and bookmarks as one body of information. Stuff I've Seen (SIS) is a system that was developed at Microsoft Research to provide a unified index of all information that a user has accessed [DCC+03]. This index is fully searchable and because the person has seen the information before, is full of contextual cues, such as screenshots or chronological order. Search results from queries are displayed in a listing that is similar in structure to the email message list displayed by Microsoft Outlook (see Figure 4). At present they are experimenting with timeline visualisations of results and one imagines that this would result in functionality similar to that provided by Lifestreams.

Figure 4: Stuff I've Seen interface with some search results

Figure 4 is a screenshot of the Top View offered by the Stuff I've Seen interface. Another view, called the Side View which moves the filter controls to the side, allowing more results to be displayed at a time.

2.7.3 The influence of email

Due to their regular use of email systems, many users regard email as a habitat, a facility that organises many aspects of their daily work and forms a major part of their workday

[DB01]. An indication of how pervasive email has become in everyday work is given by user's complaints that the Satchel system (see Section 2.4.1) didn't provide them with access to email – something they had come to regard as a document store in itself.

Marsden & Cairns (2003) point out that although an email message listing is hardly an advanced visualisation, it is an effective one due to the user's ability to easily filter and sort the display of results [MC03]. This manipulation can take place in the form of sorting by different attributes, marking messages or moving messages to different locations. An interesting finding by Ducheneaut & Bellotti (2001) was that whilst users make frequent use of the sorting capabilities of email clients, they very rarely make use of their search facilities or filtering [DB01]. Only experienced users made extensive use of folders within their email systems, as many users found navigating folder structures time wasting and also feared "losing" their documents [DB01].

The flexible nature of email systems has led to many users regarding email as a task management resource [BDHS03]. Taskmaster, developed by Bellotti *et al* (2003) aims to provide users with task management capabilities in conjunction with email services [BDHS03]. An important feature provided by Taskmaster is that it treats all content equally – attachments are "broken" out of emails and linked to tasks accordingly, enabling users to collate data more effectively. The system takes advantage of users' familiarity with email systems to provide sorting and filtering capabilities, along with other email-related activities, to create an intuitive experience for users.

2.8 Sharing Data between the Systems

What many of the research projects presented here do not take into account is the sharing of data between the mobile and desktop systems. Primarily this is because the systems described here have been designed as stand-alone systems – they do not have some complementary partner on the opposite platform. However the sharing or *synchronisation* of data between mobile and desktop systems is a critical activity as it allows users to not only travel with their information but to also effectively work with it when back at the desktop.

For the PocketPC platform, a default synchronisation system is provided – Microsoft's ActiveSync[1]. ActiveSync handles the synchronisation of data between the desktop-based applications and those on the mobile device with examples including calendar, task and

[1] `http://www.microsoft.com/ActiveSync`

contact information. For the Palm OS platform, this function is performed by the HotSync software.

The synchronisation software has to consider not only the versioning information of the data being transferred, but also the *priority* of the system. If the same piece of data is modified on both the desktop and the mobile systems, which version is kept? Ideally, these changes would be merged but document management technology is only just beginning to employ such advanced techniques. Three general solutions exist: to give one system preference, to only consider the newest data or to even prompt the user for action. However synchronisation software should be non-preemptive owing to the background nature of synchronisation, meaning that prompting the user should only be a last resort to avoid data loss. From these few problems presented here, it can be seen that the synchronisation process does not simply consist of copying some data between two platforms – it also means ensuring the *integrity* of the data.

2.9 Summary and Research reference

In this chapter we have presented various research projects into information and search management on both mobile and desktop systems. Whilst the work presented above appears disjoint and to have very little relation or relevance to the goals presented in the Introduction, we feel that all these systems are related. The common denominator tying this work together is the need to allow people to manage their information in an intuitive, context-aware manner, regardless of whether they are mobile or seated at their desktop computers. The research in this chapter broadly covers the three main "problem" areas that we feel faces our proposed system: interaction with ubiquitous devices, user searching behaviour and management of search results by users. Where appropriate, the ideas presented within these works have been adapted to better suit mobile computing.

First of all, we presented ubiquitous computing and its implications, along with the need for careful mobile interface design. The research in this section was presented in order to gain insight into the challenges facing the design of mobile interfaces. An important highlight here is to think "out of the box" and to avoid the temptation of replicating conventional desktop interaction designs. Moving on to search behaviour and how it ties in to information management, the Power Browser project and the Satchel system were presented. The features of these systems illustrated the the problem of mobile connectivity

and provided us with some insight into how to deal with it. We feel that a broader solution here is laid-back interaction, particularly in the context of mobile devices.

Moving on, search systems and research into user behaviour when managing search results whilst searching on desktop-based systems was presented. The concept of user searches evolving and berrypicking behaviour was discussed, along with the Ariadne project and how it encourages this kind of behaviour. While this work is desktop-based, it is influential to our work as we would like to allow the user to work in a fashion that they are familiar with. Information packages, which are central to our system were presented and the Lifestreams research project was used to illustrate the concept. The idea of information packages only serves to enhance the issues people tend to have when working with file systems as opposed to document spaces. In this vein, the influence of email systems and the familiarity people have with email interfaces was presented. The problems behind sharing data between mobile and desktop systems were also highlighted and the default synchronisation provider for the PocketPC platform, ActiveSync, was introduced.

In the next chapter we will document the design of our system, showing the influence of the ideas and work presented here.

Chapter 3

System Design

From the research presented in the last chapter we have drawn inspiration for the design of our system, whilst at the same time working to overcome what we feel are negative aspects of these systems. First, let us restate the goals of our work in order to provide some direction to our design decisions that are presented:

- Create an information management framework that will allow users of mobile devices to manage their information regardless of connectivity status
- Allow users to work with their information in a consistent and intuitive fashion
- Implement the ideals of laid-back interaction, allowing the user to interact with the system as they wish
- Investigate the suitability of the laid-back interaction style to users of mobile devices

The first of these goals is central to the laid-back interaction style in that it allows users to work *when* they wish – necessitating the removal of user concerns for connectivity. Connectivity plays a central role in the design of mobile software owing to the portable nature of the mobile device and the level of uncertainty surrounding network access. We feel that this effect needs to be minimised as far as possible in order to create a consistent experience for the user.

Consistency in software design and behaviour is also considered important in a general usability perspective but it becomes even more so when one moves into mobile software design. *Calm technology*, as described by Weiser & Brown (1996) [WB96], must remain

unobtrusive whilst providing the user with information. If the behaviour of the software is inconsistent, it forces the user to focus on the interface and thus moves the software from the periphery of the user's attention to the center. By enforcing consistency, we can ensure that the system only moves from the periphery of attention when the user desires so. This consistency characteristic is one of the key elements of heuristic evaluation (see Section 5.2 for more details on heuristic evaluation) and is thus easily enforceable.

Intuitive design involves designing interfaces that quickly "make sense" to the user. That is, they are consistent, not only with the operating platform's conventions, but also with the user's *expectations* of the system. These expectations arise from a mental model that the user may form of the system and its behaviour. For example, with a document management system that presents a hierarchy similar to that of a file system, users would expect to be able to drag and drop items between levels of the hierarchy, just as they can with files within a visual environment.

Implementation of the laid-back style of interaction is central to our work and the aims of this dissertation. Several challenges are presented by this goal, such as allowing users to dictate the pace of interaction whilst maintaining a *non-preemptive*[1] interface. A partial solution here is to ensure that each object works independently from other objects in order to remove dependencies and thereby ensure that each object is fully in control of its own behaviour. The investigation of the effectiveness of the laid-back interaction style has little to do with the actual design of the system and more to do with the experimental process. As such, it will be detailed in Chapter 5.

For the remainder of this chapter we will present a high-level view of the system's design and present our rationale for the design decisions taken. This view of the design will be outlined from a global view – that is, it will apply to not only the mobile version of our system but also to the desktop implementation.

3.1 Global system design

From the outset it was apparent that our system could not simply be based on the mobile device alone – at some stage users will want to modify, work with or more effectively view information that is stored on the mobile device. Our solution is to allow users to work with

[1] A non-preemptive interface doesn't demand input from the user, blocking all work until it receives some input [Thi90].

the information seamlessly between the desktop-based computer and the mobile device, with synchronisation of information taking place in the background. Whilst the underlying functionality offered by both the desktop and mobile systems will be the same, in order to preserve consistency, the interface design and appearance (by necessity) will differ greatly due to the differences between the platforms.

The overall aim of our system is to allow users to manage search queries and their associated results whilst mobile. Thus, the main focus of the system's design will revolve around the design of the search interface and its needs. The primary need for conducting searches is connectivity and we agree with Pitoura & Bhargava (1994) [PB94] in that the mobile device and indeed, the desktop system too, should minimise any inconvenience caused by a weak or non-existent network connection. Thus, the system needs to make connectivity concerns as transparent as possible and, if necessary, become translucent in order to keep the user informed. This is in keeping with the findings by Ebling *et al* [EJS02] regarding excessive transparency and the possibility for confusion when unexpected behaviour is encountered.

Systems such as WebTwig, Satchel and PowerBrowser all achieved their goals of providing effective access to and visualisations of documents and web pages for mobile users but with one common drawback – all assume the continual presence of a network connection. Realistically, this assumption is not valid as it is not possible for a mobile user to remain continuously connected particularly whilst mobile.

Thus, we propose a framework (or an "engine") that will manage the information contained within the system and the needs of the various information types. It is this engine that manages the objects and provides the laid-back interaction mechanism to the user by ensuring that operations are kept consistent regardless of connectivity. For example, search queries will need network connectivity and should it not be available, the queries will be notified when it is. The engine can then instruct a search to "pend" until a connection materialises, at which point the search is instructed to resume.

Enforcing modularity between the system components will not only make it simpler to extend the information types supported by the system but will also allow the developer to have full control over the behaviour of the user interface (UI). The system is designed in a modular fashion, with clear distinction between the UI, the engine and the individual information objects. Each information type (such as a search query) will be represented by a unique class that will manage the data associated with the information type. This object-based design allows for easy extension of the system and for each information type to

work independently of others. In essence, the engine forms an abstraction layer that is used by the user interface to manage the objects and the information stored within them. All information management operations including synchronisation and the persistent storage of objects are handled by the engine.

While each mobile platform differs in terms of the hardware and software deployed, all mobile-desktop data synchronisation configurations comprise the same basic 3-tier arrangement of a host PC, the mobile device and a middle tier formed by the synchronisation software. With the PocketPC platform and a Windows-based host, this middle tier is formed by Microsoft ActiveSync, which provides synchronisation services, application programming interfaces (APIs) for managing the mobile device and connection *conduits* from the host to the device. These conduits are used to synchronise different types of data (such as calendar information) between the desktop PC and the mobile device. Microsoft ActiveSync currently supports all the media types in the list of communications media presented in Section 2.5.

3.2 Managing information

We feel that there is a need for information management (and indeed, search management) tools to enable the berrypicking behaviour noted by Bates (1989) [Bat89]. Users often begin a search for information with uncertain goals and tend to *browse* through searches until they find a suitable result [TNST95]. Thus, an information management system that manages a user's search process needs to support this browsing behaviour. Whilst the Ariadne interface, as designed by Twidale *et al* [TNST95], provides the user with a rich visual representation of their search history and patterns that supports browsing, it utilises too much screen space and detail to be an effective solution on a small-screen mobile device. Lifestreams, on the other hand, provides a visualisation that could not only be effective on a mobile device but its timeline metaphor is also intuitive and an easy-to-grasp concept for new users. However, we feel that for users working to satisfy some information need, information packages as proposed by Twidale *et al* (1997) [TNP97], provide us with the most complete solution for both mobile and desktop-based systems.

One could argue that an information package created within a document space utilising effective filters and visualisation techniques could offer functionality similar to that of Lifestreams and Ariadne. The document-space architecture allows users to have quick

and easy access in much the same way Lifestreams allows users to easily access documents they have been working on. An advantage of the information package concept over the process-path visualisation used by Ariadne is that the information package allows the user to access all aspects of the solution: the successful search results *and* relevant documents. Ariadne, on the other hand, only presents to the user their search process and results with no scope to add other data types that may also satisfy the individual's information need. Information packages are capable of providing users with functional aspects of both these systems – easy access to documents and a method of tracking multiple search queries.

Document space-oriented design as opposed to file space-oriented design benefits the user in that it allows them to work logically with *information* in any form rather than *files* or groups of files. This removes the need for them to remember file names or locations within hierarchies or even to save their progress. Freeman (1997) [Fre97] and Raskin (2000) [Ras00] both feel that the demands placed by file space systems are unnatural and that they interrupt users' workflow, creating distractions. An information package could negate this problem, holding all the information contained in a document space and treating all the information types in a coherent manner.

Coherent behaviour is particularly important, especially given the need for our system's interface to exhibit consistency. If different information types require different operations upon them to achieve a common task, it will require extra cognitive effort from users in that they will have to focus on what *type* of information they are working with. This, in turn, pushes them towards thinking within a file-based model and potentially limits their scope for working with the information *as information*. Thus, all the information type interfaces and dialogs need to exhibit similar behaviour and appearances where possible (for example, when creating new information objects). The simplest manner of implementing this is through the use of class inheritance – this will be covered in Chapter 4, System Implementation.

3.3 Interface design

Whilst designing the visual interface for their distributed file-system Ebling *et al* (2002) posed several design questions regarding the interface and the level of transparency employed by their system [EJS02]. We feel the two most critical questions asked by Ebling *et al* for designers are:

- What system behaviour must be brought to the user's attention?
- What aspects of system do the users need and want control over?

We propose a third question that is equally important and central to the design of the system:

- What is it that users are actually trying to achieve with the system?

These questions are central to the creation of a laid-back interaction system in that they address the primary needs of the user, avoiding excess and unnecessary functionality. Systems that alert the user to all aspects of their behaviour can potentially create an information overload and overwhelm the user with data. The reverse, however, is just as damaging – systems that provide little or no feedback about their behaviour can confuse both novice and experienced users with unexpected behaviour. Similarly, systems that provide too much control create opportunities for users to confuse themselves with a multitude of options and those with too little control can engender frustration on the user's behalf.

The behaviour of information within the system will need to be brought to the user's attention – for example, if a search is in progress, the user must be informed in some manner to avoid confusion as to why the results are not available at that moment. Similarly, should a connection be absent and the search is placed into a "pending" state, this state change needs to be conveyed to the user.

Aspects that the user may desire to have control over will vary according to the information type being used and the actions that information may perform. For the system as a whole, generic information control, such as edit or delete, is needed for all information types. However, for "active" information types (e.g. searches), the user may desire to have further control, such as the ability to halt a search that is in progress. A modicum of user control over the system's actions is desirable, yet the developer has to avoid providing *too much* control – in such a situation the user could potentially disable some functionality of the system and not be aware of this, leading to confusion.

The answer to our third question, particularly with reference to information management is relatively simple: users are trying to *work* with their information. For users to work in an effective manner, systems need to support their existing behavioural patterns rather than forcing them to comply to some pattern that they may be unfamiliar with. Thus, for a system that manages searches and their results, a primary need would be to support

the observed berrypicking behaviour of users. This implies not only providing quick and easy access to the all the information, but also the ability to *manipulate* the display of the information to fit some criteria in order to avoid an information overload. Email systems conform to this statement very well in that they typically provide users with a variety of filtering, sorting and searching tools to manage their emails and associated information.

One of the clearest examples of an existing document space environment has to be that of email. Email systems can contain a mixture of information types, ranging from simple emails through to contact information or calendar appointments. This information is all stored within the email system and users are never bothered with the details of how or where the data should reside. This document space metaphor provides a smooth and consistent working environment for users and allows them to work under the assumption that their data will be persistently stored at all times. Coupled with the familiarity of users with email systems as outlined in Section 2.7.3, this makes email systems and their associated interfaces a compelling metaphor for an information management system. Presenting users with a familiar interface and its associated functionality will enable rapid adoption of our system and intuitive understanding of what it can offer.

Of particular interest is the associated ability for users to mark emails and filter them accordingly – with appropriate filtering in place, an information package can provide users with effective and flexible views of their information at hand. Thus, not only can an information package provide users with a record of their search process, but also with the resulting documents and web pages that satisfy the original information need.

By designing our system interface to be similar in appearance to modern email systems we will also create *expectations* of similar behaviour from users. Conforming to a user's pre-conceived mental model of how the system should behave and work will not only reduce the cognitive effort required from the user when using the system, but will also introduce an element of consistency to their work pattern. Email systems are well suited to mobile use – the user can perform operations regardless of the connectivity status and the interface is usually non-preemptive.

Figure 5 shows the basic interface layout proposed for our desktop systems, with the mobile system following a similar platform convention for the mobile device. Building on the email interface metaphor, the object listing is analogous to the list of emails a user may have received and the viewing pane on the desktop system performs the same information display task as with email. On the mobile device the details of each object is displayed on

another screen – just as emails are viewed in a display of their own. Widgets for filtering or sorting the view of the object listing are also provided, allowing users to control what and how they view their information. The toolbar provides the user with shortcuts to commonly used commands, such as creating objects, editing objects or deleting objects, allowing more experienced users quick access to these frequently used functions, providing macro functionality.

While designing the mobile system, a basic prototype that had simple search functionality was created using the interface guidelines outlined here. This prototype was given to four volunteers within the department who already had mobile devices. Their criticisms of the interface were used to make improvements and also provided some insight into what users generally expect from mobile devices – responsiveness and multiple access paths to functions (short-cut menus or buttons). The users were happy with the basic email layout employed and it was decided to retain this interface.

3.3.1 Information types

The core information type to be handled by our system will be search queries – however an information package, by definition, can contain a mixture of data types and it would be limiting to restrict our system's support to only search queries. Generically, the results of searches tend to point towards web pages or to actual documents on the web (such as presentations or images) and it is these objects that are typically of use to the user. Thus, we feel, our information package system should provide some sort of support for these information types as well.

Adding generic support for any file-type will allow the system to potentially provide specialised information context-based services. An example of this could be support for image file-types: when the user selects an image object that is contained in the package, a preview of the contents is displayed without any need for the file to be opened in its associated application. In a similar vein, supporting links to web documents will allow users to store shortcuts to useful resources, providing them with quick and easy access to relevant results. Potential extension of this functionality could entail the inclusion of a "web-spider" to retrieve and locally store the actual web document hierarchies themselves.

For our system we will only be providing basic support for these information types and will thus only support search queries, web-page links and files of any type. Utilising these basic information types will allow us to perform our intended research goal of determining

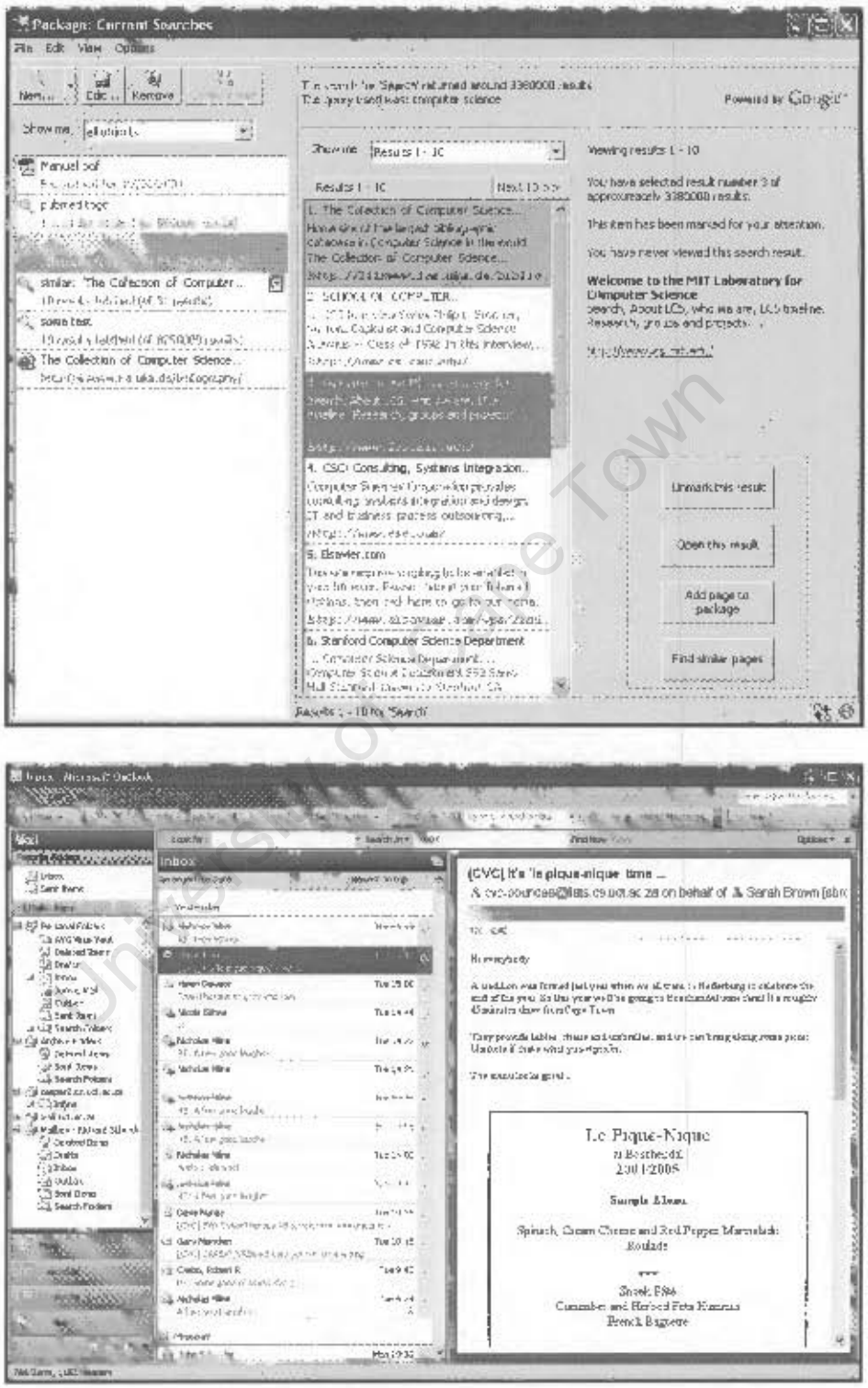


Figure 5: The basic email-inspired interface layout for the desktop system

The left-hand figure in Figure 5 shows the basic interface layout for the desktop system. Shown for comparison on the right is an email system's interface layout.

the usefulness of laid-back interaction and the extension of these information types is a subject of future work. By restricting information type support to these base types, we can also minimise the potential complexities in interaction that may arise from more specialised objects.

We feel that this system will support the working practices of users for the three major aspects of information management – searches, access to web-links and access to local files. As no search management systems are available that could be considered part of users' standard working practice, we feel that the search aspect of our system will aid users. At present users only have access to web browsers that provide very little or no search history management features. The information package concept provides users with access to web-links in a similar fashion to a web browser's "Favourites" list, allowing quick and easy access to frequently used or important links. Similarly, the file management aspect of our system allows users to keep a collection of short-cuts for access to documents considered important – providing an alternative to existing options, such as the desktop itself or even some folder hierarchy. The assertations made here will be investigated during the experimental stage.

3.3.2 Data entry

Whilst data entry on mobile devices is an issue in terms of the effort and cognitive focus required, we feel that it is a separate issue on its own and is beyond the scope of our work. The PowerBrowser system worked to alleviate this input problem by creating an auto-complete function that utilised a vocabulary based upon the content of the website being searched. This solution may be effective for quickly entering words but it introduces extra computational and resource overhead in the form of proxy servers and site indexes. Also, it fails to address the problem faced by users of how to actually physically enter the text – it simply speeds up the word creation process, as opposed to the character entry process. We have decided not to implement any input assistance functionality within our system and will instead make use of the basic auto-complete functionality provided by the PocketPC platform.

As it stands, data entry needs on the mobile device will typically be very limited – in the simplest case, a user may only need to enter a query string. In a more complex scenario, they may wish to create and advanced search query and give the query some sort of descriptive title. By minimising the amount of data required from the user we can restrict the amount of time and cognitive effort involved in creating a new search query,

thus reducing any potential distractions that the device may pose to the user.

3.4 Design Summary

In this chapter we have presented our design ideas for our information management system that makes use of laid-back interaction. Outlined are our ideas for an email-influenced interface, along with our reasoning for how this supports the document-space metaphor used by information packages.

Our system will be based upon both the mobile and desktop platforms, allowing users to not only access their information whilst mobile, but to also effectively work with it when back at their desks. Our approach varies from that of Jones *et al* [JJBC04] in that we are applying the ideas of laid-back interaction to both the desktop and the mobile systems. Previous research into the concept of laid-back interaction has only focussed on its implementation on mobile devices and has failed to investigate its potential usage on desktop-based systems. Through our systems and their interfaces, we wish to show that consistency in *interaction*, not the *interface* is important when designing mobile and desktop systems.

During the course of the experimental stage we will be investigating the effectiveness of the laid-back interaction style in aiding mobile users. In the next chapter we present the actual implementation of our systems, based upon the guidelines laid out within this chapter.

Chapter 4

System Implementation

Using the design guidelines presented in the previous chapter, our system was developed for both the desktop and mobile platforms. First, we will present the development environment and the associated platforms before introducing the various aspects of our system.

4.1 Development platforms

A number of different hardware vendors have been developing their own mobile devices making use of proprietary hardware standards and the inevitable result of this is that a number of mobile platforms and operating systems[1] have been developed. Platforms available on popular commercial mobile devices today include PocketPC (Microsoft), SmartPhone (Microsoft), PalmOS (Palm Inc) and Symbian OS (Symbian). As our research is based upon HP iPaq devices, we have been working with the PocketPC 2003 platform running on ARMV4 hardware.

The PocketPC platform has been developed by Microsoft to mimic the desktop based Windows experience as closely as is possible. One of their rationales for this design decision is that by doing so they will enable users to quickly adapt to mobile environments. In practice, however, there are certain fundamental interface differences, such as the locations of menus and toolbars that can potentially confuse first-time users. Other differences in operating system design (such as memory management) mean that mobile-based programs behave differently from their desktop counterparts. For example, when the user taps the

[1] A platform is a combination of hardware and operating system – identical operating systems running on different hardware configurations may behave in a different manner and vice versa.

"X" icon to close a program, it doesn't actually close the program – it simply moves the application window to the back of the stack, hiding it from the user's view. On the other hand, if an "Ok" icon is displayed, then the program is actually closed. The intention here is that less time will be spent waiting for applications to load as they will simply be waiting in the background. The greatest failing of this mimicking of the desktop design is the underlying assumption that people will work on their mobile devices as they work on their desktop systems. In reality users cannot afford mobile devices the same level of focus as they do for desktop-based devices due to the operation of the mobile in the user's personal environment.

Microsoft's latest development technologies, .Net, are designed to provide a unified environment (or Framework) for a multitude of technologies to work together. A reduced-size version of the .Net Framework called the .Net Compact Framework (.Net CF) has been developed by Microsoft for mobile devices, or "Smart Devices" as they are termed. The entire .Net CF is approximately 1.5MB in size (as opposed to approximately 35MB for the desktop .Net Framework!) and it provides a device- and platform-independent development environment[2] for mobile devices. Other advantages include consistency in code behaviour and the ability to share libraries between the two different Framework types.

The desktop system was written and tested on Microsoft Windows XP. This is the latest version (2004) of Microsoft's popular operating system and is installed on nearly all new computers that are shipped today. The .Net framework is compatible with other Microsoft operating systems, such as Windows 2000, but we will not be making use of this operating system as it is nearing the end of its product life-cycle and is being phased out by Microsoft.

The entire system was written in Microsoft's C# using Visual Studio .Net 2003 to take advantage of the tools and technologies provided by the .Net Framework and .Net CF. One immediate benefit that arises from this development environment is that a large amount of code sharing is possible between the mobile and desktop systems, with very minor modifications to account for hardware differences. A counter-argument to this could be the need for the respective .Net frameworks to be installed on the platforms. However this is a trivial problem as once the Framework has been installed it will support any application created using the .Net environment.

[2]For a full comparison of the two development environments, see http://msdn.microsoft.com/library/en-us/dv_evtuv/html/etconNETCompactFramework.asp (Accessed August 2004)

4.2 The Laid-back engine

Not only is the engine responsible for the information objects but it also tracks connectivity and notifies the objects when a network connection is lost or found. The objects then adjust their behaviour accordingly and any changes in status are reported to the engine, which in turn relays salient information back to the UI. The mobile and desktop versions of the engine adhere to the same basic structure, with differences arising over how the data is stored and the data synchronisation process.

Three basic types of information are to be supported by the system: search queries, web links and files. These three information types were selected as they form the "base" of all data types that can support a user's information need. For example, "files" can serve as a base class to various document types, such as "picture" or "spreadsheet." Each information type is represented by a class that inherits its properties from a common interface, the `ILBEngineObject` interface (see Appendix A for a complete description of the interface). The interface provides methods, properties and events for the engine to manage and communicate with the individual objects.

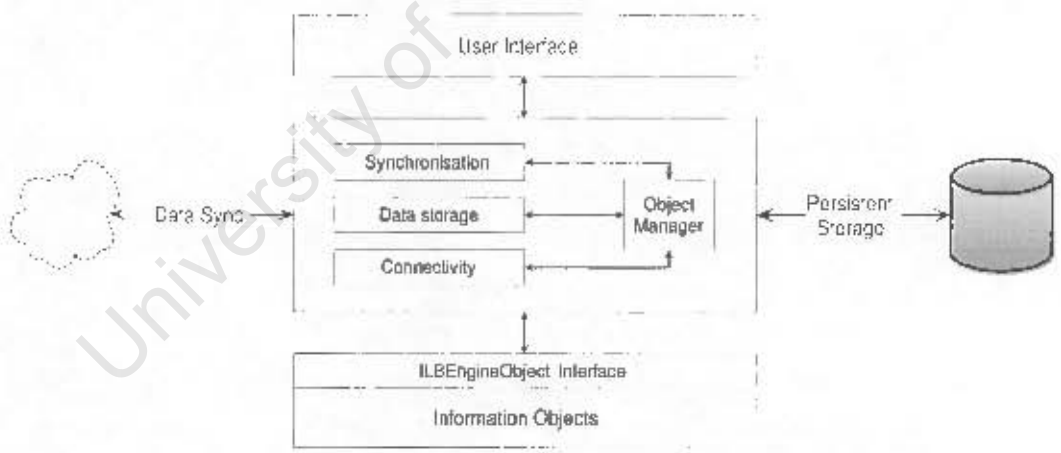


Figure 6: Basic structure of both the mobile and desktop systems

Figure 6 illustrates a high-level structure overview of the mobile and desktop systems. The engine passes appropriate information between the user interface and the individual objects as necessary, stores object data and handles synchronisation between the mobile and desktop systems.

Every object is assigned a unique key, generated from a MD5 hash[3], which is used to

[3]MD5 hashing is an encryption technique that generates a unique 128-bit string from some input string.

identify the object for the duration of its lifetime. When requesting data from the engine or receiving a report from the engine, the user interface will make use of these keys. If necessary, the engine will return the actual object in question to the UI, such as when updating the interface with new data (eg. search results have been returned). As such, the engine will be unaware of what type of object it is handling – they are all treated equally through the interface definition.

The laid-back engine is designed to be event driven, allowing multiple events to take place in a seamless fashion without delaying other processes. All the information objects have events that are registered with the laid-back engine and when fired, trigger specific handler methods. Similarly, the engine has events registered with the UI that are used for report-back purposes. It is this event-driven nature of the system that allows users to dictate the pace of interaction and thus enables the laid-back style of interaction. Rather than giving rise to events and forcing the user to react, the system *responds* to events.

The most important of these events employed by the objects and the engine is the `ObjectStatusChange` event, which returns a string containing the key of the relevant object. This event notifies the handler of the event that the specific object's status has changed in some way – a search query may have returned results, the object may have been synchronised or even that the user has edited the object details. When events are triggered, a new thread is created to run the associated method. Whilst processing events in this manner allows for easy handling of multiple events, it does mean that the developer has to be wary of race conditions or deadlocks forming. To ensure that code is "thread-safe," extensive use of *monitors* is made. Monitors are the .Net equivalent of semaphores and are used to synchronise access to objects or classes. A monitor creates a "lock" on an object and other threads are blocked from accessing the object for the duration of the lock. Once the lock has been released, the next thread in the stack is unblocked and allowed to access the object. Critical and frequently accessed data, such as the array of information objects inside the engine, is protected by these monitors.

Dialogs for adding or editing objects are to be provided by the objects themselves. This enforces the modularity of the system and enables the individual information objects to be almost completely self sufficient. In order for them to be completely self sufficient, they need to provide their own UI widgets as well, increasing the memory cost and development time of the objects. A compromise here is to have the UI provide the widgets and displays for the various objects and to then extract the information from the objects as needed. For

further information regarding the design of the UI, refer to Section 4.5.

The engine will also be responsible for the logging of actions taken by information objects. This logging is primarily used for experimental purposes but also serves as a debug tool. To preserve the user's privacy specific details such as the search query used are not recorded in this log of activity. Logging will be performed via a web-service located on a dedicated machine and will take place in the background without any user intervention. Should a connection be absent, outstanding logs will be cached and then sent once a connection is found.

4.2.1 Differences between the mobile and desktop versions

The desktop version of the engine makes use of direct serialisation to store data held by the individual information objects. Serialisation is the process of converting objects directly to a logical stream of bytes which can then be manipulated in the same manner as any other stream (ie. store to disk, copy or send via a network). This process is fully supported by the .Net Framework and is a fast and efficient way of not only storing data, but also of ensuring version control – if the structure of the class is changed in-between serialisation and de-serialisation, the data will not be loaded and an error will be raised.

Due to the size restrictions placed upon the .Net CF because of typically the limited storage space on mobile devices, serialisation support is not included. Thus, data is stored to binary files by the engine itself. This data is retrieved from the individual objects using the same methods that collect data for synchronisation with the desktop system. In essence, this data flow can be regarded as a stream of characters, with its destination being either persistent storage or the network. To reduce delays brought about by the slow speed of the device, data storage is handled in a separate thread from the main laid-back engine process. This thread is activated when an object indicates that it has completed some work and when the engine is shutting down.

For the synchronisation of data between the mobile and the desktop, the desktop system plays a client role (see Section 4.4 for more detail). The ActiveSync Remote API (RAPI) is used to monitor for the presence of a mobile device and when one is detected, the desktop engine initiates a connection, beginning the synchronisation process. The mobile device acts as a server, waiting for connections and serving requests. This design was decided upon for reasons of discovery – on the mobile device it is not possible to detect when a connection has been established between the mobile device and the desktop. However, this is possible

on the desktop using the ActiveSync RAPI and once a connection has been discovered, the address of the mobile device is known.

Other minor differences between the systems exist, such as the events provided and handled by the engines. These primarily arise from the difference in interface design between the two platforms and the storage mechanisms used. Appendix B contains a detailed listing and comparison of the methods and events offered by the two versions. Despite these differences, the basic functionality and operational process of the two versions is identical with both systems providing a consistent experience for the user.

4.3 The information objects

As mentioned in Section 4.2, all the information objects or "laid-back objects" as we have termed them, make use of a common programming interface. This interface provides the basic methods and events for the objects to communicate with the engine and ultimately, with the UI. Each object class builds upon the interface as appropriate to provide the class with the required functionality. This approach was followed in order to make it simplistic to extend the system and to enable the modular design.

The objects are treated as "blackboxes" by the engine in that they are presumed to be self-sufficient and able to work independently. All threads of running objects are tracked by the engine to allow it to terminate any running objects. This becomes necessary in one of two situations: the user cancelled an object's work or the user is closing the program. Once an object has completed its work and reported its status change to the engine, its thread is automatically removed from the thread-pool and discarded.

Whilst storage of data is handled by the laid-back engine, the laid-back objects are responsible for providing the data for storage. The data is returned to the desktop engine in a structured array which is then serialised directly. On the mobile device the stream of characters returned by the synchronisation methods is stored to a binary file. When re-initialising the information package, the engine creates instances of the information objects and passes them their data arrays, recreating the original object structure.

Every laid-back object within the system has two "timestamps" associated with it – one for when the object's data was last modified and one for when the object was last synchronised. Any changes made to the object's data results in an update to the relevant timestamp and begins the process of synchronisation. For a detailed description of the

synchronisation process and the rules employed, see Section 4.4.

Object status is represented by an enumeration, `LaidBackObjectStatus`, which contains the following definitions:

- `Pending` – The object has work to do and is waiting for the engine to "start" it
- `Working` – The object is currently busy with some work
- `Done` The object has completed its work and is idle

When an object reports back to the engine that an update has taken place, the engine checks the object's status and acts accordingly. Objects that are set to pending will be started up and their resulting threads are placed into the thread pool. Objects that are working are left alone and their status is reflected back to the UI. If an object is newly marked as `Done`, four actions are performed: its thread is removed from the thread pool, its timestamps are checked (for synchronisation purposes), the data is saved to the information package and the UI is notified.

4.3.1 Search objects

The `SearchObject` class handles search queries created by the user or by the system. These queries are resolved using the Google API that is provided by the Google search engine[4]. We make use of this API as we are not trying to develop our own search technologies – we are more interested in managing the results from existing search engines. At present the API has a restriction of 1000 queries per day and limits results retrieval to batches of 10 results. Thus, the `SearchObject` class stores its results in "buckets" of 10 results each that are displayed to the user. These buckets contain the search results as well as their associated meta-data, such as the number of times the user has opened a specific result or if the result has been marked by the user. Each individual result is stored within a `struct`, which in turn is stored within an array. These arrays form the buckets and are stored within a hashtable using the bucket number as an index to the table.

When a search object is initialised by the engine, a simple process is started. The following steps take place:

1. Check the object's status. If it is set to `Done` then there is nothing to do, stop.

[4]For more information on this API, see http://www.google.com/api (Last accessed August 2004)

2. Begin the call to Google Search Service and notify engine of new `Working` status.
   (a) If the call succeeded, set status to `Done`, generate the result bucket and notify the engine.
   (b) If the call failed, set status to `Pending` and notify the engine of potential lost connection.
3. Provide the engine with logging information.

While the reason for the call to the Google API can fail for many reasons, the foremost assumption made is that the network connection has been lost. This fact is reported back to the engine which then begins checking for network connectivity. Should a connection be found, the object will be notified that a connection exists and the process will repeat itself. On the other hand, should there be an error with the Google Search Service (eg. service not available), the object will simply set itself back down to a `Pending` status and will be initialised the next time the engine goes through its list of objects, or if the user manually "starts" the search.

Common to both the mobile and desktop systems, the dialog provided by the `SearchObject` class allows the user to create advanced search queries, selecting criteria such as specific phrases to use or words that must be excluded from the results. The dialog is re-usable, capable of being used for both creating and editing search queries. When editing, it performs a check to ascertain what was changed – if the search query was changed, this means that any existing results are now invalid. A dialog is displayed to the user alerting them of this fact and asks whether they would like to create a new query based on the changes, thus preserving the existing results. An alternative option is to accept the changes and lose any previous results fetched.

Advanced queries make use of formatting codes to aid the search engine in finding exactly what the user is looking for. These codes can, for example, be used to restrict search results to those within a web-site domain that contain specific terms. The advanced query dialog builds these search queries for the user with the supplied parameters. When the user creates a "standard" query (ie. free text entry) the advanced query entry controls are still visible but are disabled, as indicated by their greyed-out status (see Figure 7). The controls are left visible so as to avoid hiding functionality from the user and to also serve as a reminder to the user that the extra functionality exists. Both the desktop and mobile versions of the system display this query creation functionality in the same manner.

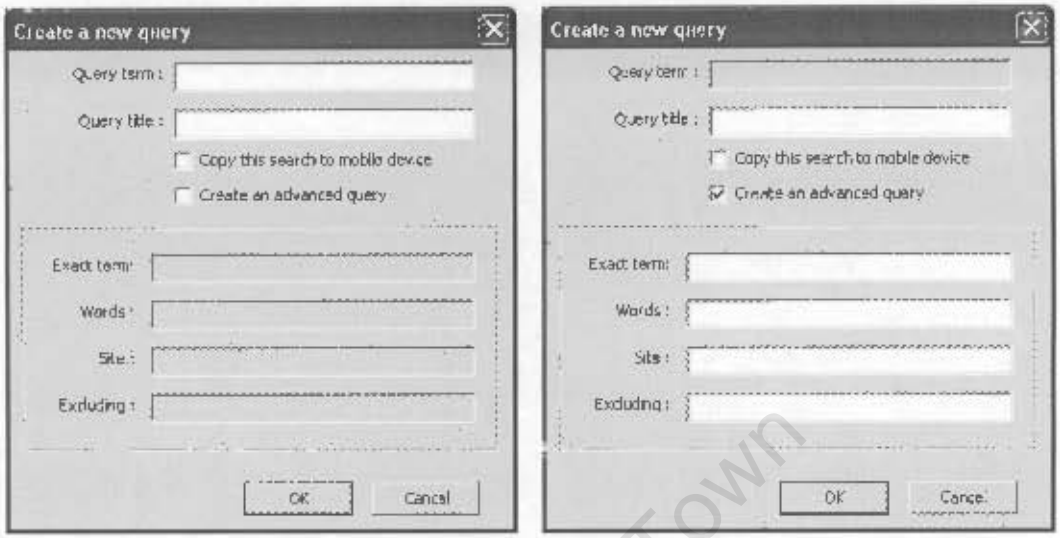


Figure 7: The dialog provided by the `SearchObject` class

Figure 7 shows the dialog used by the `SearchObject` *class to add or edit Search objects. The figure on the left shows the creation of a straightforward query. On the right the user has chosen to create an advanced query and their data entry options are indicated by the greying out of textboxes.*

4.3.2 Web link objects

Web link objects are represented by the `URLObject` class and are the simplest information object types currently supported by the system. The basis of this class is the link (or URL) to a webpage. Other meta-data, such as notes the user may have about the page and the number of times that the page has been viewed are also included. Including this meta-data provides the user with some context, allowing them to remind themselves why the page is important to them and whether or not they have actually visited the page.

As the `URLObject` class performs no work, its status is always set to `Done`, and it's controller methods, such as `Init()` and `Start()` have no effect. In essence, this class can simply be regarded as a shortcut to some web-based location.

The dialog provided by this class is very straightforward, simply providing text boxes for the user to enter the title for the object, the URL and any notes they may have. The desktop version of the class has two extra features above this: it provides an option for

object to be synchronised to the mobile device and it makes use of *autocomplete*[5] to aid the user with URL entry.

4.3.3 File objects

File objects are similar to web link objects in that they point to an information source with the exception being that they also manage the file in question. The `MetaDataFileObject` class stores meta-data associated with files by the Operating System, such as the time the file was last modified, in order to track changes made to the original file.

The desktop-based version of the class creates a copy of the file in pointed to so as to avoid access problems should the user move or delete the source file. In contrast, the mobile version of this class simply stores the path to the file on the mobile device. The file isn't copied to a cache due to the limited availability of storage space on mobile devices. Should the file originate from the desktop system and is now copied to the mobile, then it is copied to a "cache" directory on the mobile device. The class tracks the origin of the file (mobile or desktop) and performs its checks accordingly. As an example, if the desktop version of the system is in use and the file originated from the mobile device, there is no need to check for changes to the file as the mobile engine will do that.

When initialised or loaded from a store, file objects adhere to the following procedure:

1. Set class data – title, filename, etc.
2. Retrieve the file's icon for displaying to the user
3. If file originated from same platform, check the original file for any changes (Desktop)
   (a) If there are any changes, alert the user and prompt for action (update information package, restore file or delete entry in information package)
   (b) Update the cache directory accordingly

In a sense, file objects are simply pointers to files, just as the web-link objects are simply pointers to web pages. These object types serve as placeholders, allowing users quick and easy access to the most recent version of their information. As the file objects contain very little user-entered data, they have the simplest dialog screen with the fewest user options.

[5] Autocomplete is a feature where as the user enters a URL, the system provides a drop-down list containing previously visited URLs that match the one currently being entered.

The user can provide a title for the object, select the file contained by the object from a file selection dialog and, on the desktop version, select the synchronisation option.

4.4 The synchronisation process

Synchronisation of the data is achieved in an asynchronous client-server manner with a set of commands being used between the mobile and the desktop. Whilst it is possible to use the ActiveSync synchronisation system and its associated conduits for maintaining data consistency, it was decided to use our own synchronisation system due to problems arising from mixing managed code (C#) with unmanaged code (native C++). Communications are established using a standard network socket over a TCP/IP connection between the mobile and the desktop, utilising ActiveSync for device discovery.

All objects that are created on the mobile device are automatically synchronised to the desktop system as it is likely that information which is useful whilst mobile will also be useful whilst working at the desktop. On the other hand, not all information is automatically transferred from the desktop system to the mobile system – only the information that has been specified by the user is copied. This enables the user to avoid transporting information that they have no need for whilst mobile. Should they wish to, using the desktop system, the user may change the synchronisation setting on an information object at any time.

4.4.1 Connecting the systems

Ideally, the desktop computer would be the server in this scenario as it has greater computing resources at its disposal. However, it is not possible for the mobile device to be dynamically alerted to the desktop computer's presence (via ActiveSync), forcing the desktop computer to play a client role. Whilst the mobile device acts as a "server," all decision making regarding the direction of information transfer is performed by the desktop. This is to reduce as much as possible the load placed upon the PDA in its server role.

When connecting the two systems, there are two possible scenarios:

- The mobile system is running and the desktop system is started up
- The desktop system is already running and the mobile system is loaded

The first situation is trivial to deal with, as it becomes a straightforward connection between a client and a running server, with the server's address being provided to the client by

ActiveSync. The second situation, however, becomes a little more difficult to deal with. This arises from the fact that the client is already running when the server "comes up" and has no way of detecting this fact. A solution here is for the desktop system to "listen" on a specified port when it is not connected to the mobile device. When the mobile system loads, it can check for the last known ActiveSync connection's address and attempt to connect using that port. Should a connection occur, the desktop computer will then begin the synchronisation process as in the straightforward scenario. In the event that the desktop computer loses its connection with the mobile device for some reason (eg. mobile device moved out of the wireless network range) then it will resume listening for the mobile device on the "wait" port.

4.4.2 Rules employed

Once a connection has been established, the systems are ready to begin the actual synchronisation process. The rules governing the synchronisation process and management of the objects within the laid-back engine are presented below:

1. The desktop creates a connection to the mobile and requests a list of objects
2. The mobile returns a list of object keys, along with their update and synchronisation timestamps and comparison meta-data
3. On the desktop system, the object keys and timestamps from the mobile system are compared to those on the desktop system:
   (a) If the modification timestamps are the same, no action is taken
   (b) If the mobile modification timestamp is newer and the object contains more data, the object is copied from the mobile
   (c) If the desktop modification timestamp is newer, the object is copied to the mobile
   (d) If a key doesn't exist on the mobile, only on the desktop:
      i. If it has never been synchronised before the object is copied to the mobile
      ii. If it has a synchronisation timestamp the object is deleted
   (e) If a key only exists on the mobile:
      i. If it has never been synchronised before the object is copied to the desktop

ii. If is has a synchronisation timestamp the object is deleted on the mobile

4. Object's are received and added to the systems as they arrive

The aim of the synchronisation process is to seamlessly enable users to keep information current between their mobile and desktop systems with as little user interaction as necessary. However, an inherent danger here, especially without user intervention, is the possible loss of data due to synchronisation in the wrong direction. The rules have been constructed to show favour towards the mobile device as it is potentially more time consuming and disruptive to repeat work upon the mobile than on the desktop. As an example, when using only timestamps to control synchronisation, a potential loss of information could arise in this scenario:

> "Using his mobile device, John creates a query and the results are fetched. Later, while at home the results of the search are copied to his desktop PC. The next day at work he then fetches another 20 results on the mobile device under that query and marks several of them. Going home that evening he realises that he has forgotten his PDA at work and simply fetches the next 10 results from the original query that was on his desktop computer. Later, when he has the PDA at home, he synchronises the two systems and notices that he has lost some information."

The problem arises from the fact that whilst the desktop computer's data is *newer* than the mobile's data, it is redundant in that the mobile device has *more* data. Synchronisation would result in a loss of the extra 10 results retrieved by the mobile, as well as the information about which results are marked. Hence, Step (3b) in the synchonisation process checks not only the timestamps but also which system has the *most* data, using the `ComparisonData` property provided by the `ILBEngineObject` interface. The potential for losing meta-data such as the marking of objects still exists but we feel this is a tradeoff in that it presents less disruption than the actual loss of data.

Synchronisation is a continuous process and for as long as the mobile device is connected to the desktop update notifications will be sent between the systems. An example of this process is shown in Figure 8. The figure illustrates a desktop system connected to a mobile device with both systems synchronised and up to date. The desktop system then creates a `SearchObject` which fetches the results of the query. The mobile device is notified and it

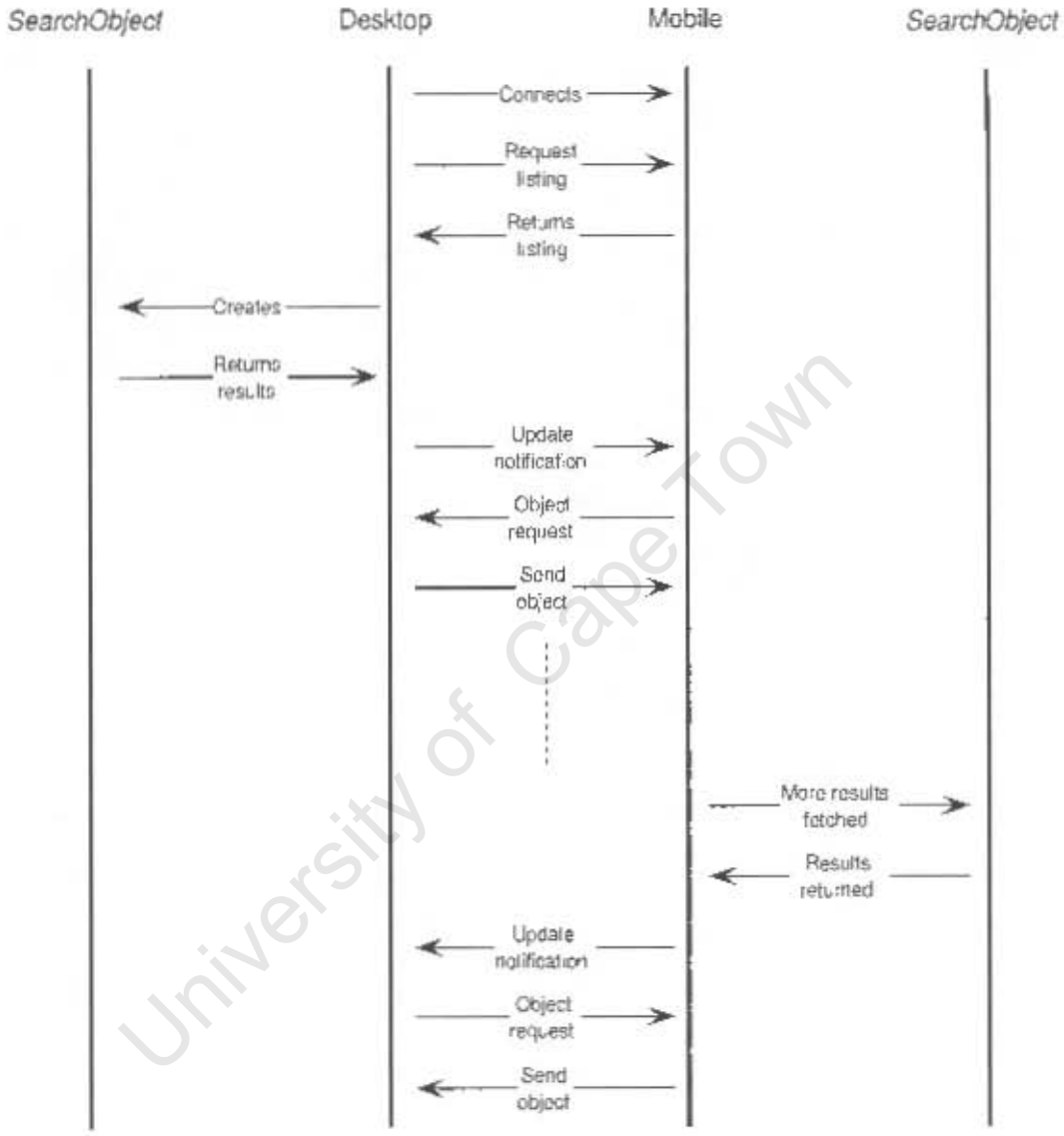


Figure 8: The synchronisation process

Figure 8 illustrates the steps of the synchronisation process between the desktop and mobile systems with one `SearchObject` *instance. Initially the desktop system creates the search query and fetches the first bucket of results. Later, the user fetches the next bucket of results with the mobile system. All these changes are seamlessly replicated between the two systems as the information retrieval process takes place.*

requests a copy of the search results. After they are relayed, the user fetches more results using the mobile device which in turn notifies the desktop system of the status change.

4.4.3 Data formatting

The synchronisation data sent between the desktop and the mobile is a character stream that has been converted to a byte stream. These bytes are re-assembled by the receiver and converted back into a string representation. A string representation is used as the Compact Framework is not capable of directly serialising and de-serialising objects from a byte stream. Thus, text-based strings are used as a method of "serialising" the objects. Delimiters within the stream are used to break the data down into increasingly finer granularity from a higher-level message down to low-level data (see Figure 9). The appropriate information objects work with the data when it has been reduced to object-based "chunks." For a complete description of the command codes, delimiters and message formats employed, refer to Appendix C.

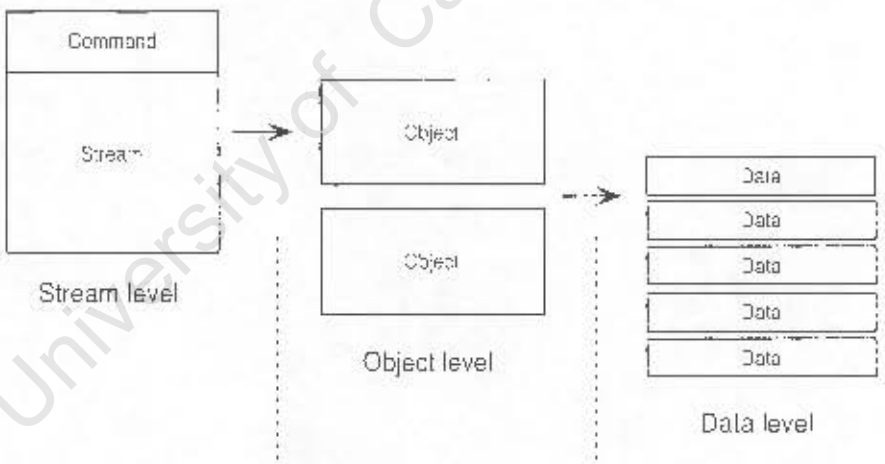


Figure 9: Granularity of synchronisation data

Figure 9 shows how the synchronisation data is broken down. Stream data is broken into objects by the engine. These objects are then broken down by laidback objects into raw data which is then stored inside the object.

The methods for exporting and importing this data are provided by the `ILBEngineObject`

interface and as mentioned, on the mobile system serve a dual-purpose of not only providing data for synchronisation but also for committing the objects to the persistent store. By keeping data management at object level, we not only reinforce the modularity of the system but also make it easily extensible for future work.

4.5 The User Interface design

The interfaces were designed with the structure of email systems in mind due to the familiarity users feel for them (see Section 2.7 for more detail) and for the mental model they provide users with. We feel that there are several parallels between the behaviour of email systems and our proposed information management system's laid-back interaction style. A comparison between the properties of typical email systems and our proposed information management system are presented in Table 2.

EMail System	Information Package System
Folders contain emails, attachments, etc.	Information package has mix of data types
Email listing with message view pane	Object listing with view pane
Flag emails for follow up	Mark search results for attention
Filter and sort view	Filter and sort view
Read emails whilst disconnected	Browse information whilst disconnected
Edit emails whilst offline	Edit search queries while disconnected
Compose emails whilst disconnected	Users can create searches whilst offline

Table 2: A comparison of email system properties to our information management system

As presented in Section 3.3, our system's user interfaces will be modelled to look and behave in a manner similar to contemporary email clients available on the desktop and mobile platforms.

For our system, where the primary work is in the creation and execution of searches, users need to be alerted to the information object's status: `Pending`, `Working` or `Done`. Logically, the status of an object can also bring to the user's attention the connectivity status. If a search query is pending then it is highly likely that the system doesn't have a network connection to work with (based on the assumption that the Google API is reliable). Synchronisation is another "invisible" activity that needs to be brought to the user's attention, albeit in a discreet and non-disruptive manner. We will consider the issues raised

by these questions as we discuss the design of the mobile and desktop interfaces.

4.5.1 Desktop interface design

The object listing makes use of a custom listbox[6] based upon the system listbox, the `ObjectListBox` class, which displays `ObjectListBoxItems`. Each `ObjectListBoxItem` has several associated properties such as the object's key, title, representative icon and object status. The listbox then renders the display of these items accordingly to reflect the object's current status. This rendering of the object listing items is performed in a similar manner to that of Microsoft's Outlook, with an icon to represent the data type, a title and other relevant meta-data.

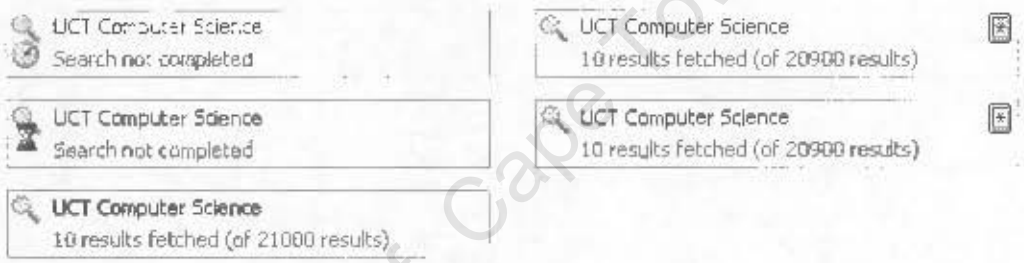


Figure 10: The rendering of `ObjectListBoxItems` in the object listing

Figure 10 illustrates the basic renderings provided by the `ObjectListBoxItem` *class. The left-hand items illustrate an item's appearance as its status moves from* `Pending` *to* `Working` *to* `Done`*. On the right, the upper item is marked for synchronisation but has not been copied. The lower item has been copied and is up to date on both the mobile and the desktop.*

Figure 10 shows the methods of displaying a search query's status used by the system. This same process is used for all information object types, with the object icon and the message displayed varying according to the object type. We feel that this representation provides us with a compact and verbose way of relaying information to the user. At a glance it is possible to determine the object type, its status, its title, its synchronisation status and the meta-data associated with it.

The layout of the viewing pane varies according to the currently selected object's type and the amount of data to be displayed. As mentioned earlier, the display of information is

[6] .Net allows developers to easily create *custom controls*, which are widgets that inherit their base functionality from some system-provided control.

not handled by the information objects themselves, but by the UI due to practical concerns. When an item in the object listing is selected the relevant object is retrieved from the laid-back engine and inspected. Depending on the object type, handler methods are called by the UI to update the interface with the object data. This design does violate our design goal of enforcing modularity within the system but we feel that it is an acceptable compromise for the performance and development time improvements it provides.

Just as desktop-based email systems support drag-and-drop operations, so does our system. The resulting information object that is created varies depending on the data that is "dropped" in – dragging in text will create a search query based upon the text. Similarly, dragging in a URL from a web-browser will create a web-page link and dragging files in will create a file object pointing to the file in question. These operations are not supported on the mobile system as it is not possible to drag data between applications due to the fact that the currently running application uses the entire display area.

4.5.2 Mobile interface design

Unlike the desktop system, the mobile design is completely modular in that the information objects provide their own information displays. This design is possible as information is displayed on a completely new screen due to the limited screen space. These displays are similar in layout and behaviour to the viewing panes employed on the desktop version of the system in order to create consistent behaviour and expectations for the user.

The main display, which consists mostly of the object listing, also contains widgets for manipulating and creating these objects. These widgets are in the form of toolbar buttons, which run along the bottom of the screen, as is standard on the PocketPC platform. Further options, such as the ability to view marked search results, are available from a tap-and-hold[7] context menu that the object listing provides.

The object listing uses a simple one-line list-based format as opposed to the desktop's custom object solution. With a single-line listing, more information can be rendered in one screen, maximising the usage of the limited area of the mobile device display. Three pieces of information are displayed in a single line: The object's type (via an icon), the object's title and the meta-data associated with the object. Figure 11 shows an example

[7] As the only input method on mobile devices is the stylus, conventional desktop operations such as "right-click" are not available. A work-around is the "tap-and-hold" menu – the user taps the stylus and holds it down. After approximately 2 seconds a context menu is provided.

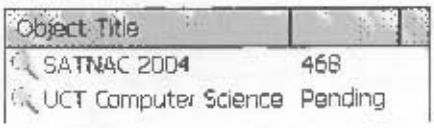


Figure 11: The display of objects on the mobile device

Figure 11 shows how objects are listed using a one-line format. This example has two search objects within it - one has completed its work and the other is pending for a network connection.

of this layout. The meta-data displayed varies according to the object type and status. If the object status is *not* Done then the meta-data displayed is the object status. Otherwise, the meta-data displayed depends on the object type: search objects display the number of results, web objects display the hyperlink and file objects display no meta-data.

4.5.3 The SearchObject interface

Before we could design the interface for the `SearchObject` class, we first had to consider some of the tasks that users wish to perform with searches and their associated results. The interface was then designed around this list of actions which includes:

- Opening a result's page
- Filtering the view of results to only show a sub-set
- Creating short-cuts to results for quick access
- Creating new searches based on a result
- Checking if they have viewed a result

Displaying the results of searches is difficult in that there is a large volume of information to be displayed yet one does not want to overwhelm the user with information. To combat this problem we re-created the object listing interface on a smaller scale, creating an element of consistency and familiarity within the interface. The same basic layout is presented, with a result listing, a filtering widget and a data display area. Individual search results are displayed in a custom listbox that shows the result's title, a short summary providing the context of the search terms and the URL of the result. Selecting one of these results displays

the information more clearly in the viewing pane, along with the result management tasks available to the user. These result management tools are also available in a pop-up context menu, providing short-cut functionality for more experienced users.

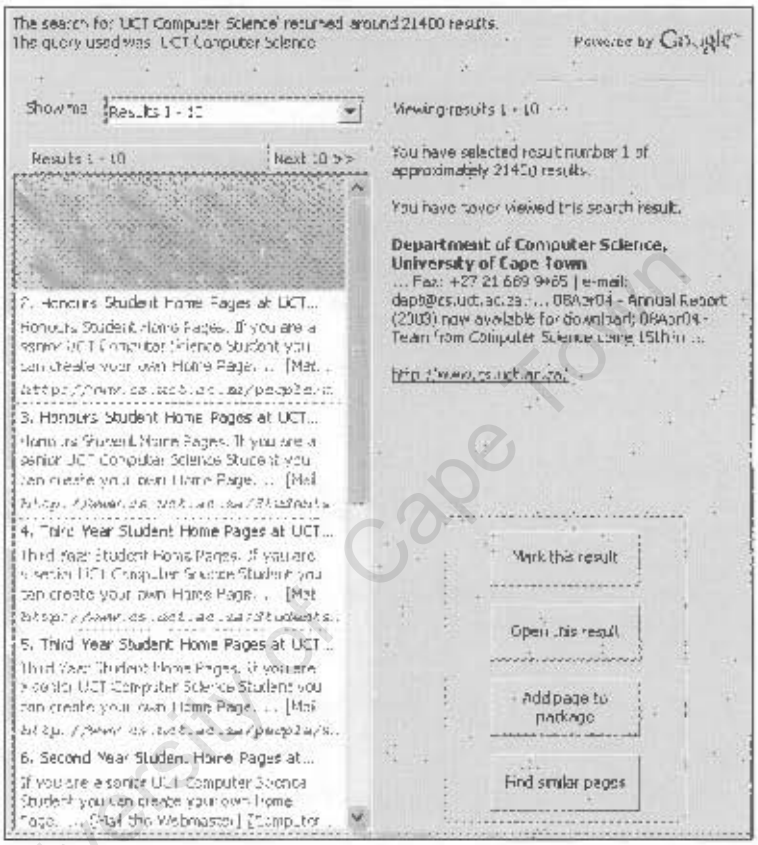


Figure 12: The `SearchObject` User Interface as seen on the desktop system

Figure 12 shows the `SearchObject` *interface with the first 10 results from a search for "UCT Computer Science." The first result has been selected and the associated tasks available are displayed in the lower right.*

These tasks include: "marking" or "unmarking" results, creating a web-link to a result, basing a new search upon the result or opening the result in the web browser. Marking of results is used to filter the view of the results – using the bucket selection drop-down toolbox, users can opt to only view marked results. This functionality allows quick and easy access to only the results that the user feels are relevant to the query.

Unique to the desktop system is the ability for users to cancel active search queries should they wish to do so. This option is not available on the mobile version of the system as it is not possible to terminate active threads using the .Net CF. The functionality was included on the desktop system as we feel that this is an aspect of the system that the user may wish to have some control over.

The Mobile `SearchObject` Interface Design

The mobile version of the `SearchObject` class has its own dialog for displaying search results. This dialog is linked to its owner class via events that are used to signal user actions, such as selecting the next result bucket or opening a result in the web browser. Information is displayed using a listing and a display area – similar in design to the desktop version of the `SearchObject` class user interface. Search results are displayed in the listing, with a text area below presenting the summary information to the user. Result management tasks are provided through a drop-down listbox that is utilised to conserve screen space. The resulting interface is displayed in Figure 13.

4.5.4 The MetaDataFileObject interface

The `MetaDataFileObject` class is simply a pointer to a file and as such has very little data to display to the user. Information such as the object's title, the filename and the file type are typically of most interest to the casual user. The interface displays other meta-data such as the file's size and the date it was last updated, allowing the more advanced user an easy means of viewing all the meta-data associated with a file.

Two file manipulation operations are provided – the ability to open the file and to copy the file to some other location. The copy functionality is provided for the scenario where a file contained within an information package does not originate from that platform but the user would like to store it in another location for some reason. This functionality is simply exposed using two buttons as shown in Figure 14. In the desktop version of this interface, should the file originate from the same platform, an option is provided to open the original file's containing folder. This is included due to the *spatial locality* of data – if a user is interested in a file, it is highly likely that they will be interested in other files stored with it.

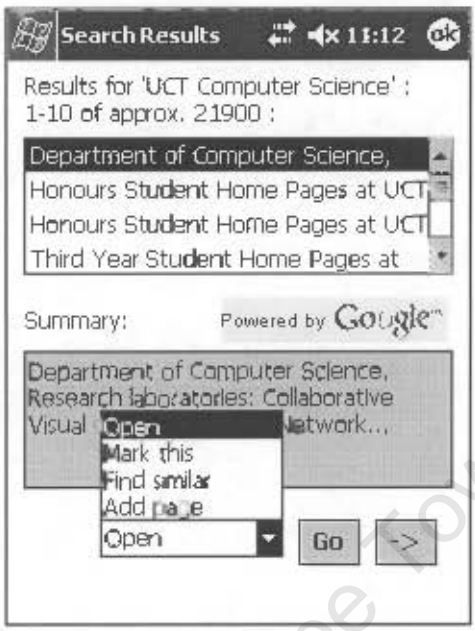


Figure 13: The `SearchObject` User Interface as seen on the mobile device

Figure 13 shows the interface used to display search results to the mobile user. The drop-down listbox has been expanded to show the tasks the user can perform upon the selected result.

4.5.5 The URLObject Interface

This interface is very similar to that of the `MetaDataFileObject` interface in that the class is simply a placeholder pointing to another information source. The only information available to display to the user includes: the object's title, the URL to the web page, the number of times the user has visited the page and any notes that the user may have made about the page. A link is also provided to allow the user to create a search query based upon the page linked to by the class. This functionality is included based on the assumption that if a user is interested in a particular page, then they may be interested in similar pages. The resulting display as seen on the desktop version of the interface is shown in Figure 15.

4.6 The design goals and our design

Referring to the design goals presented in Chapter 3, we feel that our system more than adequately meets these criteria. The management of information takes place in a consistent

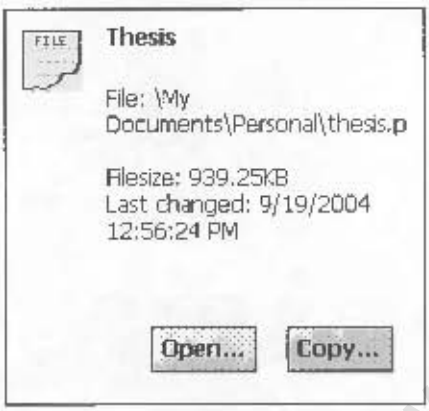


Figure 14: Part of the `MetaDataFileObject` User Interface as seen on the mobile device

Figure 14 shows a portion of the user interface provided by the `MetaDataFileObject` *class on the mobile version of the system. Displayed are the two buttons that allow the user to either open the file or to copy it to another location on the device.*

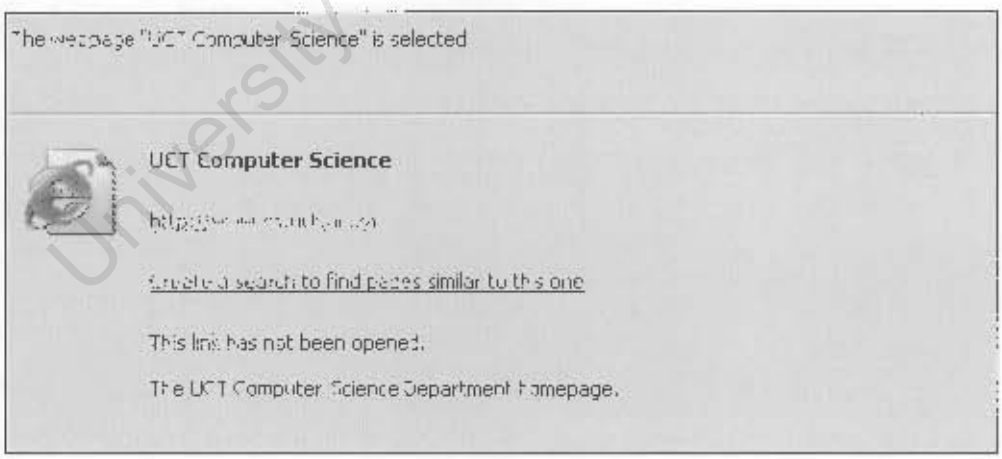


Figure 15: The `URLObject` User Interface as seen on the desktop system

Figure 15 shows the interface used by the desktop version of the `URLObject` *class. The link is for the UCT Computer Science Department - as shown by the note at the bottom of the image.*

manner with the same operations and interface styles being presented in both versions of the system. Extensive use is made of laid-back interaction in that the system never exhibits pre-emptive behaviour and responds to events, rather than causing events to take place. The mobile interface also aims to be unobtrusive, presenting information in as concise a manner as possible, allowing the user to work with the data regardless of the device's connectivity status. Management of searches and other information is consistent in that the connectivity status of the system does not prevent the user from accessing previously viewed data.

Inspiration was drawn from email systems, not only in the interface design but also in the actions that users may perform. These tasks include the ability to sort, filter or mark information contained within the package. We feel that users with previous experience in using email systems will expect this functionality and that it will provide a consistent experience. The validity of this assumption will be assessed when performing user testing upon the systems.

Synchronisation takes place in the background, with widgets being used to indicate to the user that the activity is taking place. Should no connection between the systems be available, the systems enter a "wait" mode pending a connection. Similarly, connectivity concerns are minimised for the user, with search queries being performed as a connection becomes available. When in an "off-line" environment, users are still able to create new queries and to browse existing results. The interface behaviour is more translucent than transparent in that the user is notified if a query could not be completed by its `Pending` status. Users of the desktop system are able to choose what information is copied to the mobile device, allowing them to only take what is needed when they leave their desktop environment.

As far as possible, the system has been designed in a modular fashion, with the laid-back engine managing the system as a whole. The use of the `ILBEngineObject` interface has made it very simplistic to add information types to the system, with large amounts of code-sharing possible between the two platforms.

4.7 A scenario of usage while mobile

Using the scenario presented in the Introduction of a mobile user called "John" at a conference with limited wireless connectivity, here is a synopsis of the steps John could take to make use of the system:

John is at a conference and the presenter from the UCT Computer Science Department catches his attention. While sitting in the conference hall where there is no connectivity, he decides to find the Computer Science Department web-site. John creates a new search query on his PDA by tapping the "new" icon on the taskbar and enters "UCT Computer Science" into the query term dialog box (Figure 16(a)). When he clicks on "Ok" a `SearchObject` is created by the system and it is added to the information package. Due to the lack of connectivity, this object has a `Pending` status (Figure 16(b)).

Later, after having walked through a wireless "hot-spot" John checks his PDA and sees that the search has been processed. He knows this because its status has changed from `Pending` to reflecting the number of search results found as can be seen in Figure 16(c). He opens these search results and sees that the first result is the Department's homepage and decides to create a shortcut to this page (a `URLObject`) using the drop-down menu, allowing him quick and easy access to the page in future (Figure 16(d)).

When he returns to his office, John's PDA automatically synchronises with his desktop PC. Looking at the information package on his desktop system he sees the search results for "UCT Computer Science Department" and this reminds him that he intended on following up on the research presented by the lecturer.

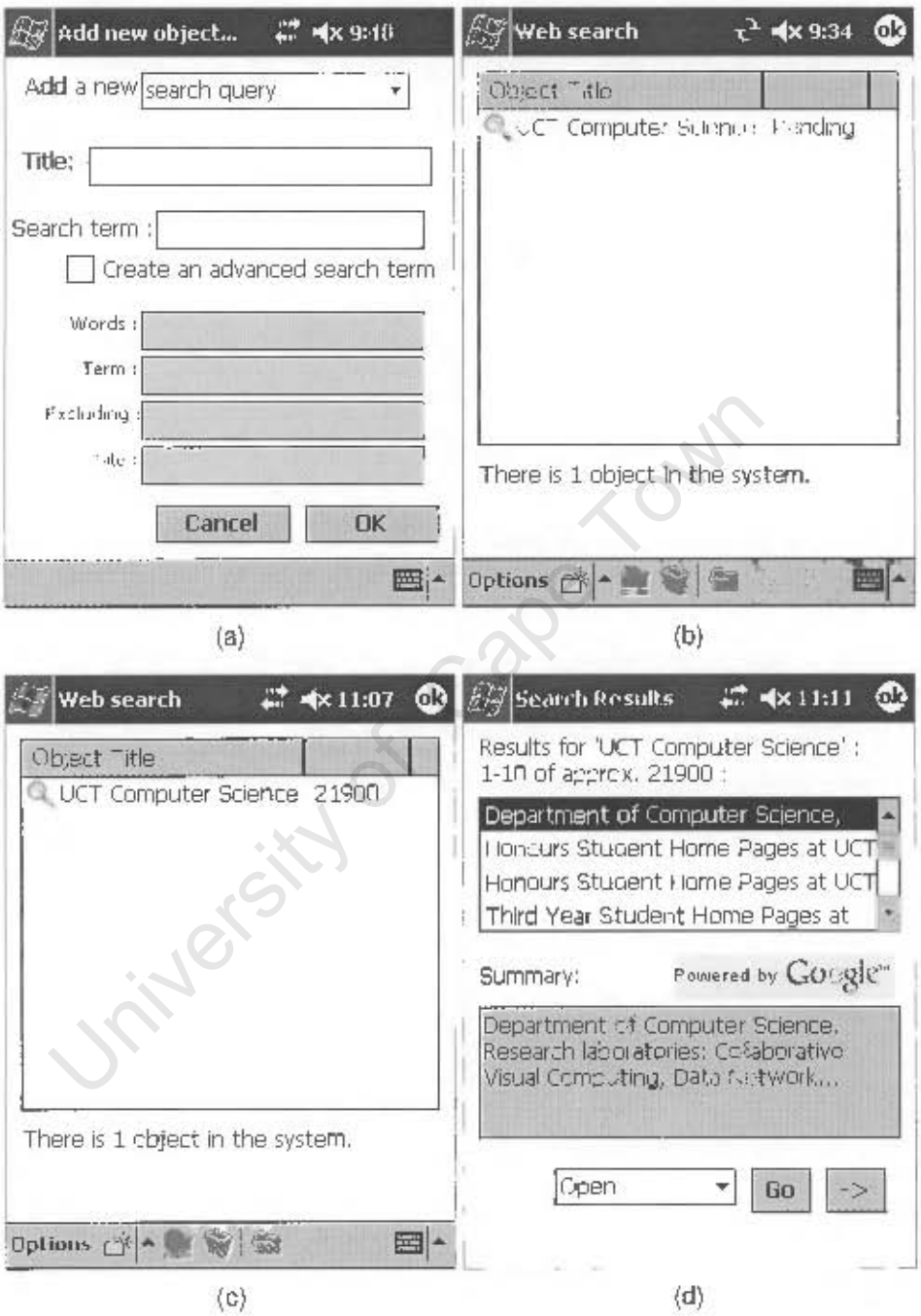


Figure 16: PDA search status

This figure shows the sequence of interface updates for the creation of a search query (a), its pending status (b), completed status (c) and the viewing of the results (d).

Chapter 5

Experimental Design and Procedure

In Chapter 3 we presented the design for our system. In this chapter we present the experimental process we will be following to validate that design and to support the assertations we have made about the system regarding its ability to manage information effectively and to support users' working practices.

Our experiments were designed with the ultimate aim of validating the usefulness of the laid-back interaction style for working with information, particularly on mobile devices. In order to do this we first needed to validate that any user problems encountered during our evaluations were arising from the interaction style itself rather than the interface design or interface behaviour. Thus, evaluation of the system takes place in three stages, with the first two stages being designed to eliminate or reduce any interface design or behaviour issues that may exist for the user. These first two stages are brief, expert studies with the third being a long term (1-2 months) study with real-world users. The stages along with a brief description are listed below:

- **Heuristic evaluation** – To ensure that the interface is consistent in behaviour and appearance and does not present the user with any unexpected "surprises."
- **Task-based evaluation** – Allows us to ensure that the system is "usable" by users and that all aspects of the system can be used with little or no external support. This ensures that the interface layout is consistent with the user's expectations.

- **Real-world observation** – Determines just how useful the system really is to the user and whether the laid-back interaction style is appropriate or not.

Each stage is designed to support the subsequent stages and to ultimately aid us in confirming the validity of our beliefs regarding laid-back interaction. The first stage, heuristic evaluation, will help ensure that the interface behaves in a consistent fashion and presents a consistent experience to the user. Thus, during the task-based evaluation, any problems that the user may encounter with the interface should be attributable to the interface layout rather than its behaviour. Also, task-based evaluation will allow us to confirm our assertation that our system supports users in their current working processes. Once task-based evaluation has been completed and any problems encountered rectified, one can hopefully assume that users with little or no experience can quickly and easily come to grips with the system. Real-world observation, the final stage of the evaluation process, will then expose to us the benefits or pitfalls of laid-back interaction that may arise over a period of prolonged usage within the system's target environment – the user's lifeworld. It will also allow us to investigate our assertations that our system will aid users in their working processes and in the management of the three information types supported. An overview of this process can be seen in Figure 17.

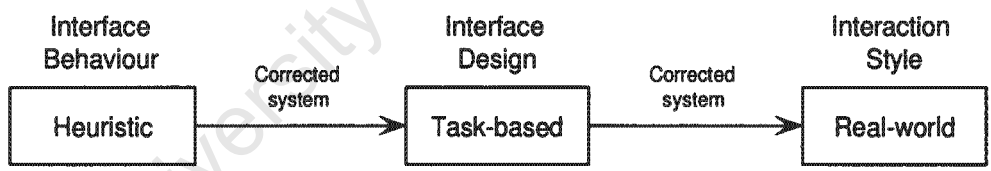


Figure 17: The experimental process

Figure 17 shows the general process used for our evaluation of the system interfaces. Heuristic evaluation is performed with corrections being made to the system before moving on to task-based analysis. Finally, real-world observation is performed.

5.1 Quantitative vs Qualitative measures

One of the challenges facing us was how to analyse the results of these evaluation stages: quantitatively or qualitatively? Quantitative data is useful in that it allows us to directly compare systems and results. On the other hand, qualitative data gives us a sense of context

and of the user's thought processes. The best solution, we feel, is to embrace a mixture of these two methods and to apply objective reasoning to the results obtained. By doing this we can not only provide figures concerning usage of the systems, but also keep these results in context, reducing the potential for drawing false conclusions. Concentrating solely on either of these two data analysis methods can distort the researcher's perceptions of the data at hand and thus needs to be avoided where possible.

Over-reliance on quantitative analysis, for example, can give results that are too narrowly focused and even potentially misleading [Nie04b]. The reasoning here is that the quantitative analysis will inform us that a problem exists but not *what* the problem is. Nielsen gives an example of too much focus on quantitative analysis regarding the usability of websites [Nie04b]:

> "...reporting that using websites is 206% more difficult for users with disabilities and 122% more difficult for senior citizens than for mainstream users. Of course, using bottom-line scores to summarize elaborate usability study outcomes neglects the details that take 273 pages to explain: *Why* are websites more difficult for these groups? *What* should you do about it?"

The heuristic evaluation comprises of quantitative data in the form of the heuristics employed and the severity ratings assigned to them. Data returned by the task-based evaluation, however, is a mixture of both quantitative and qualitative measures. We can derive quantitative values from the user's answers to the questionnaires and from analysis of their *actions* whilst using the system. This analysis could, for example, consist of efficiency considerations – how long it took them to perform a task or perhaps even how many button clicks the user employed. At the same time, qualitative values can be derived from monitoring the user's *behaviour* during the tests. The third and final stage, real-world observation, will consist primarily of qualitative data that arises from interviewing the test subjects. However, it will be possible to extract quantitative data by analysing the usage patterns of the systems and from a combination of these data to determine the user's attitude towards the system as a whole.

5.2 Heuristic Evaluation

Heuristic evaluation is the process of analysing an interface and ensuring that it conforms to a set of standard rules or *heuristics*. This evaluation is performed by people who are recognised as "usability experts" – that is, they have some sort of grounding in the basics of Human-Computer Interaction (HCI) and the challenges facing designers. For our heuristic evaluation, four evaluators were utilised. We made use of the 10 recognised heuristics as described by Nielsen [Nie04a]:

Visibility of system status The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

Recognition rather than recall Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use Accelerators, unseen by the novice user, may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Aesthetic and minimalist design Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help users recognize, diagnose, and recover from errors Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Help and documentation Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Whilst reviewing the user interface, the evaluators compare all the aspects of the interface and its dialogs to these heuristics. Should they feel that a heuristic has been "violated" or triggered, a *severity* rating from a scale of 1 (severe) to 5 (least severe) is assigned to the problem. For our evaluation, we associated a description with each severity level so as to provide the evaluators with a more concrete sense of scale. Table 3 lists the ratings along with their descriptions. For the full documentation provided for the heuristic evaluation to the evaluators, refer to Appendix D.

Rating	Severity	Description
1	Severe issue	The system is hampered by this and it must be rectified
2	Serious problem	The user is restricted by this problem
3	A problem	It is a major irritation to users, warrants further attention
4	Minor issue	It is a source of irritation to the user
5	Minor irritation	This is something which simply bugs the user

Table 3: Heuristic severity ratings and their associated descriptions

The severity rating assignations allow the developer to rank the problems found by order of severity, thus ensuring that the most critical problems are dealt with first. Depending on the frequency and severity of problems uncovered, another round of heuristic evaluation may be needed to ensure that all the problems have been dealt with adequately. The heuristic average for each major interface section of the system can be calculated, giving the developer some indication as to which portions of the interface are the most problematic.

This ability to pinpoint problems within the interface allows developers to ensure that they follow a critical path in terms of problem solution. By solving major issues immediately often several more minor problems related to this will "fade away," reducing the work load.

5.3 Task-based evaluation

Task-based evaluation takes place in a controlled environment and typically utilises "normal" users. A list of tasks are provided to the user and they are observed as they complete the tasks. This observation allows the system designer to monitor not only if users have problems completing the tasks, but also *how* they achieved the tasks. With knowledge of how users go about their tasks, one can begin to design systems that support the user's existing working patterns. No assistance is provided to the users and they are encouraged to explore the interface in order to discover the functionality of the system being tested.

Thus, the aims of the task-based evaluation is to determine if first-time users of a system will encounter any difficulties and to determine if users can discover system functionality quickly and efficiently. Due to the heuristic evaluation performed previously, it is already known that the interface behaves in a consistent fashion – therefore problems encountered by the user during this stage must be arising from the *design* of the interface rather than its *behaviour*.

5.3.1 Task descriptions

When describing the tasks to the subjects great care needs to be taken so as not to bias the user toward specific functionality that may exist within the interface. As an example, let us consider the case of requesting the user to create a web link object from a search result. Different ways of achieving this particular task could include: using the "add page" functionality provided, copying the page link and manually creating a web-link object or perhaps to even create a shortcut to the web-page on the desktop. Whilst the latter solution presented here is not the one intended by the software designer, it certainly is a valid solution to the user's goal of making the page easily accessible for future use. The objective here is to ascertain whether or not users can determine how to create an easily accessible web link from a search result. One possible way of phrasing this task would be:

> "View the most relevant search result and create a web-link from this result."

However, this statement exhibits bias in that it explicitly tells the user how to achieve the task as the *designers* intended it to be performed rather than in a manner that seems logical to the user. Alternatively, a more open-ended manner of stating the task could be:

> "View the most relevant search result and make it easily accessible for future use."

This second option provides the user with lee-way to achieve the task as they see fit. The tasks are created so as to encourage the users to explore all the functionality offered by the system. Thus the task descriptions are broken into two major sections: tasks to perform on the desktop system and tasks to perform on the mobile system. Within these sections tasks are provided relating to the three data types provided by the system – searches, web links and files. The order of the tasks, as well as the order of the systems tested was randomised so as to reduce any possibility of any "priming" occurring due to the task order. When a user performs a set series of tasks the possibility exists that the sequence of events may indicate to the user what the researcher is trying to evaluate. This in turn can possibly affect the user's behaviour and thus the validity of the experiment. When creating the task lists, the first step was to determine the possible actions the user may undertake apart from the basic create, edit and delete functions that apply to all object types. For searches, the following functionalities were listed:

- Fetch a bucket of results
- Mark individual results
- Filter view of results to show marked results
- Create new searches based upon a result
- Open a result in web browser
- Create a web link to a result

For file objects:

- Open the file in its associated application
- Copy the file to some location

- View the file's source directory (desktop system)

And finally, for web link objects:

- Open the link in the web browser
- Create a search to find similar pages

From this list of functions, tasks were then created for each aspect of the system. Where possible, the tasks were kept identical for both the desktop and the mobile systems in order to maintain consistency. For the full documentation provided to the test subjects, see Appendix E. For users that have never used a PDA before, a short "getting started" guide was created to introduce them to the interface of the mobile device's operating system. The guide makes no reference to our laid-back system and simply exists to inform users of the platform interface standards, as well as methods of interacting with the interface.

5.3.2 Evaluating user performance with The Observer

In order to effectively evaluate the actions of the test subjects during this stage all user actions are to be recorded through a combination of screen capture and usage of a video camera. Using a video mixer, these two sources can then be merged into a "picture-in-a-picture" format, allowing the reviewers to not only see the user's actions on the screen but to also monitor their facial expressions. For the mobile device a small "platform" with a fish-eye camera mounted above it on a flexible arm is used to view the user interactions with the touch screen.

To perform quantitative analysis of the user actions we will be using Noldus Information Technology's *The Observer* v5. This software allows observers to mark events or actions as they occur during an observation. Once the observation has been completed, statistical analysis can then be performed on these marked instances. There are four main parameters that need to be configured before one can begin observing sessions using this system:

Independent Variables These are the variables that stay constant throughout the observation session. Examples of this include the subject's gender, their age, etc.

Subjects For studies where interaction takes place between more than one person, different categories of person can be created. Examples of this could be Parent and Child for interactions between parents and their children.

Behavioural Classes A mutually exclusive and exhaustive group of related *behaviours* that are applicable to a user. Only one behaviour within a behavioural class may be active at any given time.

Modifier Classes Modifiers are used to limit the scope of a behaviour or to specify it more precisely. For example a modifier could be used to indicate *what* a user is clicking on or *who* a user is talking to.

For our system where we were only interested in the actions of one person, only one subject is defined – the User. Three behavioural classes were defined to cover the user's actions: Behaviour, Interactions and Task. These behavioural classes made use of five modifier classes that included Physical Objects, Software, Widgets, LaidBack Objects and System.

As mentioned, the behavioural classes must be mutually exclusive and exhaustive for the duration of the observation. For our observation, the three behavioural classes defined cover the three things that a user could be doing simultaneously: some sort of physical behaviour (eg. showing confusion), some sort of interaction with the system software (eg. selecting something) and a task that they may be currently performing (eg. Search task). These behavioural classes are then further specified using the modifiers provided. The modifiers have been created with these behavioural classes in mind and allow us to specify precisely what the user is doing. Short descriptions of each modifier are listed below:

Physical Objects Physical things that the user may exhibit some sort of Behaviour to.

Software What portion of the software the user is Interacting with – edit screen, dialog box, etc.

Widgets A screen entity that the user is Interacting with on some Software object.

Laidback Objects The object type that the user is currently working with.

System Specifies which system the user is performing a Task upon.

An example of an event that could be specified using these definitions could be "User selects check item in edit box" and this event could occur in parallel to "User is performing File Task on the Mobile System." For a full list of the behaviours, modifiers and independent variables specified refer to Appendix G.

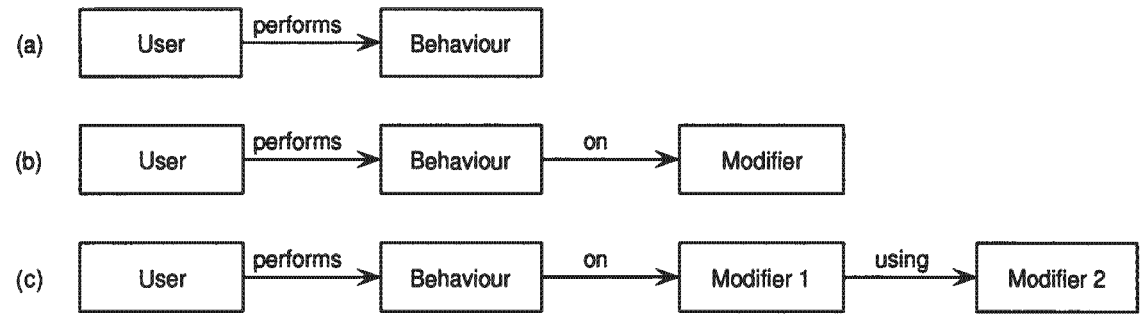


Figure 18: Three basic states that can be specified

Figure 18 shows the three basic states that can arise from the user-behaviour-modifier model. 18(a) shows a basic user-behaviour state. (b) illustrates the same state but with a qualifying modifier. 18(c) shows the most complex possibility – a state with two modifiers.

The "user-behaviour-modifier-modifier" model used by The Observer allows the evaluator to quickly and easily specify events and states as they take place during an evaluation. Each item (such as an individual behaviour) is assigned a keyboard shortcut that is used to signal it's state. This arrangement, along with the flexibility of specification, allows the evaluator to cover a wide range of scenarios. Each behaviour can have up to two associated modifiers, allowing for three basic interaction models. Figure 18 illustrates this concept.

The first case – a basic user-behaviour state is the simplest state available and is often used to signal an event. An example of this state within our configuration would be "User is confused." Adding a modifier, as shown in Figure 18(b), allows us to direct the focus of the user's action. For a user creating a new `SearchObject`, this state could be shown as "User is creating a SearchObject" with the `SearchObject` modifier showing us the focus of the User's creation behaviour. Similarly, two modifiers allow us to not only direct the focus of a user's behaviour but to also specify the context or even the recipient of this behaviour. "User selected a button in an edit dialog" would be an example of this, with the "edit dialog" modifier providing us with the context of the user's button selection operation.

5.4 Real-world observation

This is the most difficult aspect of the evaluation process to prepare and design due to the lack of research into this aspect of evaluating mobile systems. For this stage we were

influenced by the work of Hallnas & Redstrom [HR02] in their evaluations of everyday technologies within people's lives. This work was significant in that the authors were investigating how people interacted with their technologies and what influenced their actions whilst within their natural everyday environment. The researchers made use of incident diaries and frequent interviews with the subjects to monitor their progress.

In order to maintain the naturalness of people's usage of the system, we decided to avoid asking users to maintain a usage or incident diary. Previous research has shown that motivating users to maintain these diaries is problematic at best and we feel that their usage influences people's interactions with the system being tested.

One aspect that we are interested in here is how people's usage of a technology changes over time as noted by Hallnas & Redstrom [HR02]. Whilst their work focused upon the usage of physical devices within a person's *lifeworld*, we feel that the principles and ideas equally apply to the software upon these devices. A well-designed system that only engages users through the periphery of their attention and requires a minimum of cognitive effort is likely to cause very little disruption on the user's behalf and thus should be readily accepted by users. Initially, such a system is seen as a novelty and is treated as such due to its unfamiliarity and "newness." Gradually, over time, it is accepted by the user and becomes part of their normal working practice and this is when the usefulness (or otherwise) of the system becomes apparent. Our real-world observation aims to support this process and to enable us to determine the usefulness of laid-back interaction for mobile users. Should the reverse be true, however, the system will never be fully accepted by users and will remain unfamiliar to users.

As an alternative to usage diaries for providing us with usage patterns, we decided to create a logging system that would log user activities within our information management system. User actions such as the creation, modification or deletion of objects will be recorded. For search objects, further information in the form of the bucket depth retrieved will also be recorded, giving us an indication of people's search behaviours. By placing this logging activity in the background the system will thus be able to record the users' actions without introducing any extra disruption to their typical information management behaviour. In order to preserve the privacy of the users, actual data, such as the search queries used, will not be recorded. The logged information will form the basis of the quantitative analysis of the real-world observation stage and will be compared to our qualitative findings.

5.4.1 Interviewing subjects

The interviews will take place on a weekly basis with each test subject being given one-on-one interviews with the researcher. A semi-structured interviewing style will be used in order to fully explore the users' experiences with both the mobile device and our system. Of interest here is how their attitude towards the device changes over time – correspondingly, it is expected that their usage of our system will change as well. For each round of interviews, a few common questions will be asked of all the test subjects. The remainder of the interview will consist of questioning based upon the subject's answers to these questions and their relevance to the subject's previous experiences. Recording of the subject's responses will be performed through note-taking.

These interviews will provide us with qualitative insights into the users' perceptions not only of our system, but also into how they perceive the mobile device's role within their lifeworlds. We feel that should users be able to interact with the system as they wish (ie. dictate the pace of interaction as in laid-back interaction), it will lead to a more rapid acceptance of the system into their working practices and information seeking patterns.

As mentioned, the interviews will build upon each other, with the initial interview designed to be an exploratory one to establish the user's normal working patterns and need for information. Subsequent interviews will then be structured to determine if the subject's work methods and their attitude towards the device has changed and if so, in what manner.

A subjective and difficult question to answer is how long should the observation run for? Hallnas & Redstrom [HR02] performed their observation of users and their adoption of technology into their lives over a period of six months but felt that there was scope for similar studies to perhaps run for longer. However, time constraints do not allow us that luxury and a compromise needed to be reached. We have decided upon a provisional time period of one month and will continue for as long as we are able to, given device availability and time restrictions. Should some significant behaviour be observed and confirmed, then the experiment shall be concluded, otherwise it should be left to run for as long as possible.

5.5 Summary of the experimental procedure

Within this chapter we have presented the three experimental stages that form our investigation into our system and laid-back interaction. These three experiments, when tied together, provide us with a complete picture of the user's usage of the mobile device in

combination with the desktop system and how their interaction is affected.

In the next chapter we present the results from these experiments and our interpretations of these results.

Chapter 6

Experimental Results

In this chapter we present the results from our experimental process as described in the previous chapter and our interpretations of these results. Where possible, we have tried to keep these results as a mix of quantitative and qualitative analysis in order to retain some sense of perspective for the context of the user's interactions.

6.1 Heuristic results

The heuristic evaluation was performed by four evaluators from within the Collaborative Visual Computing (CVC) Laboratory research group. Each evaluator was presented with a scoring sheet and a set of suggested tasks to perform and was then asked to score the system. Suggested tasks were provided so as to ensure that the evaluators explored the full functionality, and hence the full interface behaviour, of the system. However they were not restricted to these tasks alone and were encouraged to explore the system and its interface at will.

Evaluation was performed in two distinct stages – one for the mobile system and one for the desktop system. No particular order of evaluation was forced upon the evaluators and they were allowed to begin with either system. In total, 47 heuristic "violations" were reported for the two systems by the evaluators, with an average heuristic score of 3.00 overall. Breaking the scores down, 30 issues were raised for the desktop system (average of 3.10) and 17 problems for the mobile system with an average of 2.82. However not all these violations reported were unique and removing duplicate results gives us a very different picture. The system-wide data is summarised in Table 4. For a complete description of the

severity ratings scores and their associated meanings, refer to Table 3 in Section 5.2.

	# Issues	Score	Avg Score	# Unique Issues	Score	Avg Score
Overall	47	141	3.00	41	122	2.98
Desktop	30	93	3.10	27	82	3.04
Mobile	17	48	2.82	14	40	2.86

Table 4: Breakdown of the heuristic scores for the system

Whilst approximately two-thirds of the reported issues occurred on the desktop system, the mobile system was considered to have more serious problems as evidenced by its lower average score. Although these reported problems were of a similar nature to those on the desktop system, it is possible that they were assigned more severe ratings by the evaluators due to the critical nature of the mobile device's interface. As mentioned in previous chapters, the mobile device interface cannot afford to provide any form of irritation or disruption to the user due to its personal and ubiquitous nature. Considering the disparity in the number of problems reported by the individual evaluators for the mobile device, two possibilities come to mind: the evaluators' limited experience with mobile devices and the "restricted" interface of the mobile device.

The evaluators were chosen because of two main criteria – they have experience in using mobile devices and they have experience in the field of HCI and computing in general. However given the relative newness of mobile technology, it is not realistic to expect their experience with mobile devices to be comparable to their experience with desktop systems. Thus they will be quicker to pin-point problems that may exist within the desktop interface as opposed to the mobile interface. In comparison to the desktop system, the mobile interface is restricted in that it makes use of far fewer widgets per display, reducing the opportunities for these widgets to provide conflicting messages to the user. This limitation of widgets is not a design decision – it is simply a result of the reduced screen area available.

6.1.1 Desktop system results

Of the 30 results returned from this heuristic evaluation 28 were unique, indicating a wide disparity in the interface issues found. This diversity primarily occurred due to the individual evaluators concentrating on various aspects of the interface of most interest to them – one evaluator, for example, was more concerned about the usage and layout of icons on

the toolbar than the usage of jargon, whereas another may be more concerned with the accessibility of functionality.

Many of the results returned were interface consistency issues as opposed to interface layout problems and were for the most part easily rectified. Of particular concern was the availability of the "Synchronise" button that allows the user to force a synchronisation operation should they feel it is necessary. Originally the button was unaware of the device availability and this created confusion for the evaluators as it created the misconception that data had been copied to the device even when the device was not connected in any manner. Other consistency issues included incorrect updating of the display as information arrived, incorrect display of new results and faulty rendering of the object listing.

Viewing the breakdown of the average heuristic scores by area supports the notion that many of the issues were related to information display, with the `Viewing Object` category having the most severe rating of 2.00, followed by the `Editing Object` category with 2.33. However, comparing the *number* of issues raised, it can be seen that these areas are not of major concern despite their severity ratings. The `Main Display` area is of most concern with a combination of a relatively high number of results and a high severity rating. These figures can be seen in Table 5.

	# Issues	Total Score	Avg Score
Main	12	32	2.67
Adding	7	29	4.14
Editing	3	7	2.33
Viewing	2	4	2.00
Other	3	10	3.33

Table 5: Breakdown of the heuristic scores for the desktop system

From this analysis it can be seen that the `Main Display` area of the desktop system requires urgent attention first, followed by the remaining areas in ranking of average severity. Many of the concerns centered around consistency in the display of data, although several interface layout issues were also raised. Chief among these were the usage of jargon, particularly within the search query creation dialog box, and the positioning of widgets.

It was felt that many users are unaware of how to create advanced queries using Google

and that advanced query creation options such as "term" would be confusing[1]. Other uses of jargon, particularly within the object listing widget were also modified to reduce the level of "technological intimidation" a new user may feel.

Widget positioning was adjusted in order to not only present users with a logical "flow" in operations, but to also reduce the potential for users to perform unintended (and potentially disruptive) operations. An example of this was the positioning of buttons on the toolbar - the "Synchronise" button was placed at the end of the bar as it is an operation that users are bound to perform frequently and accordingly, the user must be able to quickly and easily locate the button. However to its immediate left is the "Remove" button and a concern here was that the user may inadvertently click on this button when they intended to synchronise their data. This leads to a destruction of data rather than a dissemination of data – completely the opposite intended effect! A solution was to place a separator between the buttons and to change the button label from "Synch" to "Synchronise." This had the effect of increasing the distance between the buttons and also making the Synchronise button larger than the Remove button. The end result can be seen in Figure 19.

Figure 19: Changes made to desktop system toolbar

Figure 19 shows how the toolbar was modified to ensure that the user could easily select the "Synchronise" button without inadvertently choosing the "Remove" button. A separator was put between the buttons and the Synchronise button was made larger.

6.1.2 Mobile system results

In contrast, a large number of the mobile system's problems were centered around viewing objects and the creation of objects, with the viewing objects category considered to be more serious. The data for the mobile system's heuristic scores has been summarised in Table 6. Once again, a large number of the issues were display consistency problems and usage of jargon as opposed to actual interface layout problems. It is interesting to note that

[1] In this example, the label "term" was changed to "phrase," a simpler wording.

the remainder of the problems highlighted were related to the interface widgets themselves (which are platform-standard) and the evaluators' comparative experience with desktop-based systems and their associated widgets.

	# Issues	Total Score	Avg Score
Main	2	3	1.50
Adding	5	17	3.40
Editing	1	3	3.00
Viewing	5	14	2.80
Other	1	3	3.00

Table 6: Breakdown of the heuristic scores for the mobile system

An example of the problem that evaluators had relative to their desktop experience was that many had no idea of how to close a display screen. When a display on the PocketPC platform is closed, one of two things can take place: the screen is simply hidden from the user view and is retained in memory (in essence "minimising" the window), or it is fully closed and unloaded from memory. This difference is shown by the icon used to close the window – in the former case it is shown as an "X" and in the latter as an "Ok" icon. This behaviour is platform-default and is beyond the control of the developer. However, most of the evaluators thought it a concern and felt that the changing icon was confusing for new users. Placing a button to close the window upon the display is not a valid solution for two reasons: it is against the standard design practice for the platform and wastes valuable display space.

Two main facets of the interface were re-designed as a result of this evaluation. Initially, object creation was implemented through a drop-down list attached to an icon on the tool-bar, with the default action of the icon being to create a new search query. However once in the search query creation dialog, there was no way for users to change the type of object that they were creating. A solution here was to implement a drop-down list on the object creation dialog display that allows users to change the type of object being created. This also provides shortcut functionality for more advanced users and thus enhances the interface at the same time. Figure 20 illustrates these object creation options. This additional flexibility in object creation allows users to easily recover from mistakes made during selection operations (such as selecting the "New Object" icon instead of an item on the drop-down list). As the user is mobile and potentially not focusing completely on the device, it must be

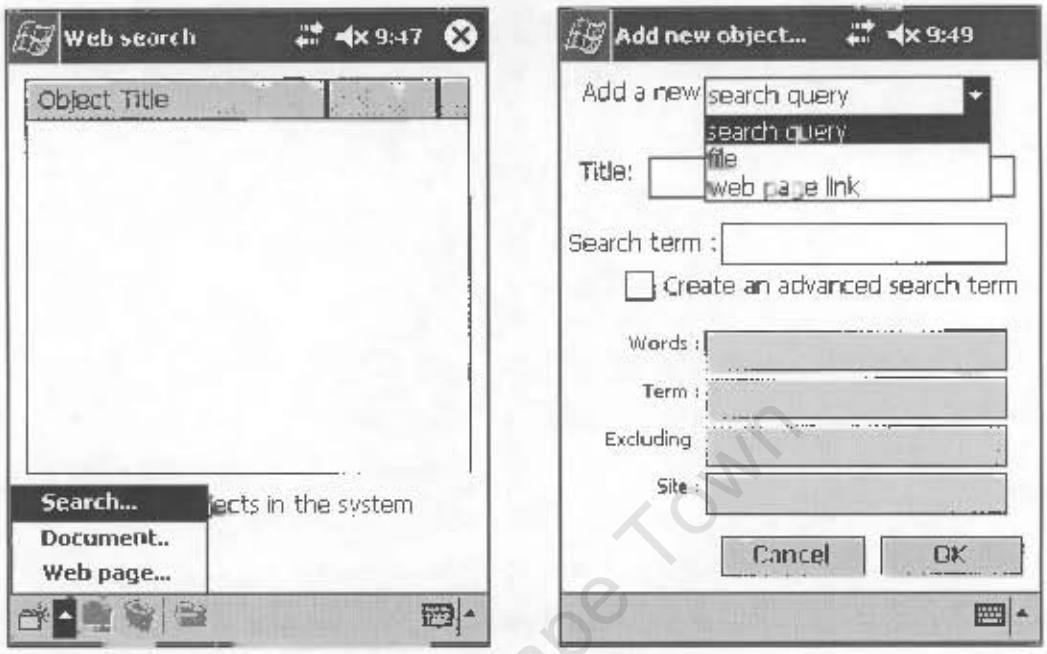


Figure 20: Different methods of creating objects on the mobile system

Figure 20 illustrates the different manners that a user may create new objects on the mobile system. The figure on the left shows how an object can be created from the system's main display. On the right is an object creation dialog (in this instance, a search query) showing the drop-down list the user can use to change the type of object that is being created.

assumed that the user will make these selection mistakes from time to time and provisions must be made to allow the user to resume working with minimal disruption. This follows the basic laid-back interaction principle in that the user is still very much in control of the interaction and its pace.

The second facet that was changed was the method of viewing the objects themselves. The default behaviour was that opening an object such as a web-link or file object would result in the actual contents pointed to by the object being opened. This behaviour was found to be inconsistent in that it does not provide the user with access to meta-data that they may have associated with the object, such as additional notes or the number of times the object has been accessed. To resolve this behaviour, all object types are required to load a display screen showing the meta-data associated with the object (or, in the case of the search object, a list of results for the current search bucket) with widgets provided to

view the actual data encapsulated by the object.

6.1.3 Comparison between the systems

Comparing the heuristic scores for each area on both systems reveals a very similar trend in the scores of the issues uncovered. Viewing a comparison of the *proportion* of the violation count each area provides supports our observations made above regarding the problems encountered on each platform. For the Desktop system, these were centered around the `Main` and `Adding` categories and the Mobile system provided most of its scores from the `Adding` and `Viewing` categories. A bar chart of the proportions is provided in Figure 21.

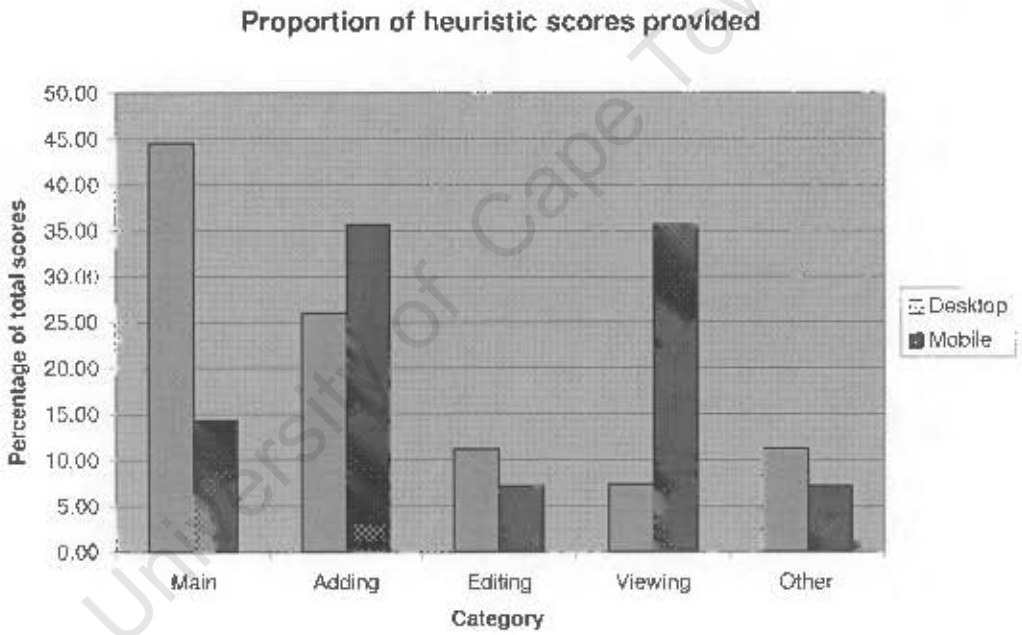


Figure 21: Bar chart showing proportions of heuristic scores

Figure 21 shows a graph of the proportion of heuristic issues each area provides for its system. Areas with the highest proportions are those that need the most attention from the designer.

With the issues highlighted by the heuristic evaluation having been resolved, we can now state that both the Mobile and Desktop system's interfaces are consistent in appearance and behaviour. With this assurance, we then began the Task-based Observation to determine

how "usable" the interfaces of the systems are for first-time users.

6.2 Task-based Observation

This observation took place in a closed office with only the research subject and the evaluator present. To reduce any "intimidation" that the users may feel, the evaluator was seated behind a partition, out of the subject's sight. The subjects were presented with a questionnaire to determine their perceived level of computing experience and their level of exposure to mobile devices. Those that had little or no previous experience using mobile devices were provided with a short "tutorial" sheet explaining the behaviour of the mobile platform and its interface widgets. The tutorial made no reference to the mobile version of the Laid-back Information Management system, ensuring that the users were not primed in the operation of the mobile laid-back interface.

The observation equipment consisted of a video grabber, digital camcorder, microphone and a video mixer. Using this equipment allowed us to not only capture the user's actions on the screen, but also their physical expressions. A frame from each of the outputs captured from the desktop and mobile systems can be seen in Figures 22 and 23 respectively.

The video was then recorded in a digital format using a video capture device. This system was designed to run in conjunction with the Noldus Observer (as described in Section 5.3.2), although scoring was not done in real-time. Real-time scoring was not utilised as we feel that it is far too easy for the evaluator to make mistakes and it also prevents the subject from communicating with the evaluator, should they need to do so. Instead, scoring of the resulting video was performed when the experiment was completed, allowing the evaluator to concentrate fully on the task at hand.

6.2.1 User Performance

The test subjects for this experiment were randomly recruited using sign-up sheets within the Computer Science Department. While this restricted our sample pool to that of people who have greater exposure to computer systems than "normal" users do, it does provide us with a sample of people who typically have some information need and generally try to satisfy this need online. As the observation equipment was only available for three days, 24 time slots of 40 minutes each were created, providing us with 8 sessions a day. As an incentive to participate, research subjects were offered remuneration for taking part in the

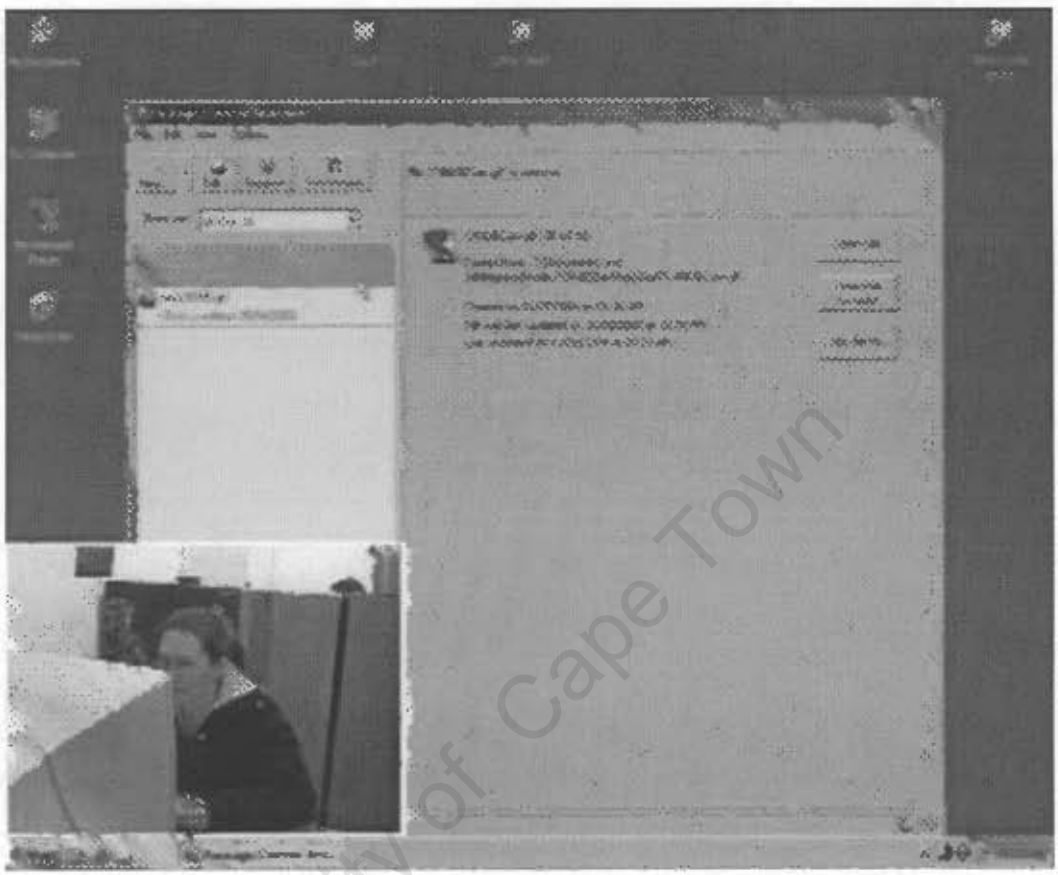


Figure 22: Screen capture showing the desktop view

Figure 22 shows the video capture taking place while the subject works with the desktop system.

experiments.

Before beginning the experiment, test subjects were asked to answer some questions regarding their computing experience, their perceived level of experience with mobile devices and their typical search practices. Where a quantitative answer is valid, a Likert scale of 1 to 5, similar to the heuristic severity rating scale, was made available for the test subject to select an answer[2]. The subjects were then provided with two sets of tasks: one each for the mobile and desktop platforms.

These tasks consisted of those listed in Section 5.3.1 and where possible, the same tasks

[2]A Likert scale is a rating scale that is used to measure an individual's strength of agreement towards a set of clear statements.

Figure 23: Screen capture showing the PDA view

Figure 23 shows the video capture taking place while the subject works with the mobile device.

were specified for both the desktop and the mobile systems. It was not feasible to provide identical task sets for both systems due to limitations, such as the inability to drag-and-drop data between applications on the mobile system. The order of the tasks within each system was randomised for every experiment and the system order was alternated for each experiment, resulting in unique experimental orders for each subject. One such task sheet is included Appendix E. For example, the task order for the first three subjects is shown in Table 7. Once all the tasks have been "rotated" for the desktop system, the mobile system task order is rotated and so the process continues. A full listing of the task orders can be seen in Appendix E.

The tasks were chosen so as to investigate how easily users could make use of the systems' functionality with little or no external help. Both systems had their entire functionality

Subject	System Order	Desktop Task Order	Mobile Task Order
1	Desktop, Mobile	Search, File, Web	Search, File, Web
2	Mobile, Desktop	Search, Web, File	Search, File, Web
3	Desktop, Mobile	File, Search, Web	Search, File, Web

Table 7: Task order for first three test subjects

sets exposed by the tasks provided and thus enabled us to determine which functions are superfluous. Primarily, the tasks were chosen to ensure that the process of creating and manipulating the different object types followed a pattern that is logical to the user.

Out of the 24 time slots created, 7 test subjects did not attend, leaving us with 17 subjects. Of those 17 subjects two were discarded: one was unable to complete the test and the Google Search API shut down during the other's evaluation. Each of the 15 evaluations that were accepted in our final analysis were unique in that no two evaluations had the same task order. On average, each evaluation required approximately 40 minutes, with a standard deviation of 0.006, showing that the evaluation times were consistent. An initial impression one can form from this is that all the test subjects had similar experiences while using the system.

Breaking the subjects' times down, it can be seen that the mobile system's portion of the evaluations took significantly longer than the desktop system. This is to be expected, given the subjects' typical unfamiliarity with the device and the relative slowness of data input using the touch-screen keyboard. The complete list of times can be seen in Table 8. Whilst it is possible to generate much more detailed quantitative data using The Observer, we wish to create a formative opinion of the data present, rather than a summative one. What is important to us is the user performance, not in the context of efficiency or speed, but rather in *how* they achieved the tasks. With this in mind, details such as the number of clicks each user needed to perform a specific task are meaningless. Far more relevant is how many users made mistakes during that task, why they made those mistakes and what steps they took to rectify them.

Comparing these results it is interesting to note that users typically had better relative performance in the second half of the evaluation – implying that they were able to relate to their experiences on the alternate platform. For users who began their evaluation with the desktop system, on average they spent 33.3% of the total evaluation time using the desktop

Subject	System Order	Total Time	Desktop Time	Mobile Time
1	Desktop, Mobile	40:03	11:25	28:38
3	Desktop, Mobile	51:34	17:07	34:27
6	Mobile, Desktop	56:59	14:28	42:31
9	Desktop, Mobile	33:28	11:52	21:36
10	Mobile, Desktop	40:18	06:59	33:19
13	Desktop, Mobile	43:43	14:49	28:54
14	Mobile, Desktop	34:56	05:20	29:36
15	Desktop, Mobile	40:26	12:00	28:26
16	Mobile, Desktop	40:58	09:42	31:16
17	Desktop, Mobile	36:40	15:20	21:20
18	Mobile, Desktop	46:33	08:17	38:16
19	Desktop, Mobile	DNC	DNC	DNC
20	Mobile, Desktop	39:14	09:58	29:16
21	Desktop, Mobile	47:39	15:12	32:37
22	Mobile, Desktop	25:04	03:34	21:30
23	Desktop, Mobile	DNC	DNC	DNC
24	Mobile, Desktop	55:52	09:13	46:39

Table 8: Time taken for each user to complete the evaluation (in minutes and seconds)

system. When the user began with the mobile system, the average time taken with the desktop system fell to 19.9% of the total experiment time. This confirmed our design goal of ensuring consistency in behaviour between the two systems despite differing interfaces. While we have not directly translated functionality, we have provided both systems with functionalities that achieve the same goals and create a consistent experience for the user.

6.2.2 Observations from the evaluations

For the most part, users were able to complete all the tasks set out in the evaluation sheet. Some users (particularly on the mobile device) struggled to complete certain tasks, such as the marking and filtering of results during the Search task.

The primary reason for failing this task on the mobile device is fairly straightforward - to view the list of filtered results, the user needs to call up a context menu. In the case of users who do not know about the mobile device's "tap-and-hold" functionality (see Section 4.5.2) this operation is never available. However an interesting behaviour was observed: once a user had "discovered" (through prompting, from reading the tutorial sheet or even

simply by accident) how tap-and-hold works, they began to attempt to call up context menus on every display. This *functionality seeking* behaviour is generally a time-wasting exercise for the user, although from a designer's perspective it is interesting. It indicates that users are aware of the existence of and the power that tap-and-hold exhibits – but they are not aware of *where* is it to be used. Indeed, there are no visual cues available, hence the seeking behaviour. On the desktop system, similar behaviour occurs, although it is not as disruptive as on the mobile device – for a tap-and-hold menu to pop-up the user has to hold the stylus down for approximately 3 seconds. On the desktop one simply clicks the right-mouse button, an action that takes very little time.

Translating functionality between desktop and mobile systems

While the design behaviour of context menus is the same between the mobile and desktop systems, it does highlight the fallacy of simply translating functionality directly from the desktop to the mobile. We are of the opinion that a "design re-think" is needed here and some sort of visual cue needs to be provided to the user indicating the presence of a tap-and-hold context menu. The difficulty here lies in providing a standard cue that does not conflict with the developer's design of the UI.

Also of interest is the shortcuts users attempted to make use of on the desktop system while working with the information types, highlighting the need for consistency with platform standards. In particular, they attempted to make use of two supplementary methods: the "send-to" file context menu and "copy-paste" behaviour. Figure 24 shows the send-to context menu that is displayed for a file object within the Windows Explorer. Of interest here is the *expectation* that the system will behave in a manner consistent with the Windows desktop environment and will thus support any functionality extensions available. Whilst this need for consistency with platform standards is one of the heuristics employed in the heuristic evaluation process (see "Consistency and standards" in Section 5.2), this observation reinforces the need for careful application of the heuristics. However no users attempted to discover if such functionality exists on the mobile platform.

Copy and paste behaviour was exhibited on both the mobile and desktop versions of the system. Common to both systems, users copied and pasted text using the standard Windows "Ctrl-C, Ctrl-V" keyboard shortcuts. This was noteworthy in that on the mobile device there is no indication of this functionality and users were drawing from their desktop experiences to aid them with the mobile system. This is expected as the PocketPC platform

Figure 24: The Windows "send-to" context menu on the desktop

Figure 24 shows the "send-to" shortcut on a file object's context menu. This shortcut or macro allows users to quickly copy or send a file to a variety of locations.

is modelled to look and behave in a similar manner to the Windows desktop. In the example of copy-and-paste, this direct translation of functionality is acceptable and works in that it was supported – confusion arises when some expected functionality is *not* supported, such as the convention of using "tab" to move the focus caret between widgets. For some reason this is not supported by the widgets provided for the PocketPC platform and can engender some user frustration: one user continued repeatedly tapping the "tab" key on the virtual keyboard despite observing that it was not having any effect.

Working with information

Our system promotes a document-space mental model, yet several users seemed to continue working with a file-based mental model. Several users wanted to know how they should save the search results – indicating that they were viewing the search queries as *files*, rather than as logical chunks of *information*. Our system makes use of an email-inspired interface in order to promote the document-space model that email systems utilise, yet it would seem that this connotation was not fully realised by the users.

The stumbling block here is that the users are seeing themselves as working with files, rather than information. When storage capacity was limited and the users were all trained experts, this model was more than adequate. However as capacity increases and the level of user knowledge decreases, the file-based model becomes a problem. The reasons behind this have been covered in Section 3.2, with the experiences of our users highlighting the difficulties of breaking this mental model.

An example of people thinking of their information in terms of files is highlighted by the actions users took when told to synchronise some information between the desktop and mobile systems. When the information concerned was a file object, 4 users attempted to directly copy the file to the mobile device as opposed to marking the information object for synchronisation. These users stated that they were unable to transfer search queries to the mobile device. Several other users began attempting to copy information directly to the device (using the Windows Explorer) but upon realising that the search queries were not stored in files, eventually "discovered" the synchronisation option.

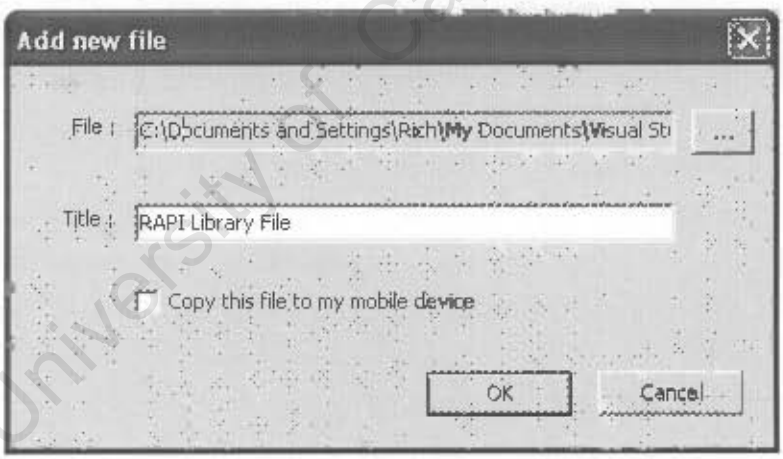


Figure 25: The object creation dialog for a file object

Figure 25 shows the object creation dialog box for a file object. Shown is the check-box widget that toggles the synchronisation status of the object between the mobile and desktop systems. Some users were unsure of this check-box's function and chose to simply ignore it.

When creating or editing objects the synchronisation option is available for the user to choose (as can be seen in Figure 25). Some users checked this option whilst creating objects, others created the objects and then later edited the objects and set the synchronisation

option. One of the users who was unable to synchronise the objects felt that editing the object in order to copy it didn't seem logical – a valid point. Perhaps it may have been better to also provide a context menu option that allows the user to toggle the synchronisation status of the object. However this solution skirts the design intention that the objects should be seen as information that persists between the two platforms rather than simply a file that is copied between the mobile and the desktop. Again, this highlights the problem of breaking through the file-space mental model into the document-space model.

This problem became more evident with the users that made use of drag-and-drop on the desktop to create their objects. Dragging and dropping information into the information package immediately creates a relevant information object and negates the need for an object creation dialog. Of the users mentioned above that had problems setting the synchronisation option, three used drag-and-drop to add files to the information package. Out of all the users, only four made use of drag-and-drop to add data to the information package – all of them adding files through this method. One of these four also added a web page link by dragging the link from the web page into the information package. No users discovered that dragging text into the package window will create a new search query based upon that text.

The mobile interface

Many of the users struggled with the layout of the mobile interface – in particular with the platform convention of placing the toolbar buttons and the system menus along the bottom edge of the screen. Six of the users taking part in the trial began using the mobile system by tapping on the Start menu icon (which resides in the top-left corner of the screen) and attempting to find menu options for the program there. When prompted that the toolbar is at the bottom of the screen, many of these users expressed surprise. This example highlights some of the problems that can arise from extending the Windows desktop metaphor to the mobile device – the users expect not only similar *behaviours* but also similar *designs*. Once users became accustomed to the locations of the toolbar buttons and system menus, they were able to manipulate the objects without any hindrance.

Feedback from the system regarding confirmation of events was of a major concern to the users. In this instance we are referring to system feedback rather than interface feedback, such as when creating a new search or web-link object from some search result. The user would select the option from the drop-down list and then tap the "Go" button to perform

the action. As per the button metaphor, the button surface inverts as the stylus taps on it, informing the user that the device acknowledges that they have tapped the button. However there is no indication given that the requested action was actually performed. The user was then unsure as to what had taken place and usually ended up repeatedly tapping the button, creating multiple instances of the required object in question. On the desktop system no indication was needed as the object listing is available at all times, unlike the mobile system that only shows one display at a time. The problem here is that displaying a dialog box to confirm the action not only violates the laid-back interaction principle (the user has to close the dialog box before they can continue) but it also creates a potential distraction to the user. We feel that some other visual cues need to be considered in this situation.

6.2.3 User Responses

Overall the users were positive about the system and its potential, with 75% stating that they would use the system if they had a chance to. It is an encouraging endorsement of the system that the users should feel this way toward the system with such brief exposure. Initial impressions of software and its capabilities go a long way in influencing the future use of a system and it is crucial that users feel comfortable with a system from the start of their experience.

Regarding the mobile system, many of the users commented that they found the mobile system easy to use once they were accustomed to the mobile widgets and their behaviours. Common requests made by the users included help functionality and the ability to work with more than one information package. The help functionality was excluded as part of the experiment – the objective here was to ensure that first-time users would be comfortable using the system and able to do so with minimal fuss. The second request is a logical extension to the functionality of the system, but was excluded due to the problems involved in synchronising between the desktop, the mobile and multiple information packages. It would definitely need to be considered as part of future work on the system.

Users who felt that they wouldn't use the mobile system stated that they didn't feel a need to search for information whilst mobile. It must be noted that these users mainly saw the system as being another method for transferring files between the desktop and the mobile device. When asked why they wouldn't use the system, they stated that it "is easier to wait until they're at a computer and to then use Google." It would seem that just

as users are conditioned to working within a file-space model, so they also feel that web searches are to be conducted through a web browser. Whilst searching using a web browser can be more simplistic in that only one window is used (the browser window to conduct the search and to open the results), it is limiting in that moving between different search queries requires repeating the query. Using the laid-back system, two windows are utilised (the application window and the web browser to view the result) but it allows the user to easily move between multiple search queries and their associated results.

Common to all the users was appreciation for the ability to browse through multiple sets of search results and to come back to those results later without repeating the search query. We feel that this appreciation stems from the laid-back interaction in that the users were no longer reliant on the network connection (for both accessibility and for speed of access to the results) for their interactions and were thus able to dictate the interaction pace completely.

With such positive user response and performance, we felt that the system was ready for the final stage of usability testing – real-world observation. During the course of the task-based observation process several minor bugs were identified in the system (such as failing to filter newly added objects correctly) and these issues were corrected before beginning the next stage.

6.3 Real-World Observation

The observation was conducted using 4 subjects, with this limit being imposed due to device availability. All the subjects were from the CVC Laboratory and were selected for the trial as they are active researchers and thus typically have current information needs that must be satisfied. Table 9 lists the basic details of the test subjects.

Subject	Gender	PDA Experience
A	Male	None
B	Female	Has used a PDA
C	Male	Has used a PDA
D	Female	Owns a PDA

Table 9: Basic details of the real-world observation test subjects

Each subject was leased a HP iPaq h4155, supplied by Bridges.org, for the duration of the experiment. The laid-back information system was then installed onto these devices as well as the subject's desktop computer. The subjects were then encouraged to use the system whenever they had some information need to satisfy whilst using either the mobile or the desktop system. It was stressed that the users should work in what they feel is a natural manner and that they should only make use of the system should they want to. This was done so as to ensure that user interactions with the system are of a "natural" manner and that they reflect the user's wants and needs.

Usage of these systems was logged and the subjects were interviewed 3 times over a 4 week period, with a questionnaire being given to them at the end of the experiment in order to assess their opinions of the system and the mobile device (see Appendix H for the questionnaire).

6.3.1 User responses

In this section, the responses made by users during interviews, and our interpretations of these responses will be presented. This will be done on a per-user basis in order to preserve the context of each user's experience. The first interview was mainly of an exploratory nature to determine the user's level of experience with the PDAs, how they felt about using the devices and to find whether they had encountered any problems or not. The following interview built upon this, aiming to find out how useful the users were finding the mobile device in general, as well as the laid-back information system. The third interview was designed to explore the user's attitude towards the device and their usage of it, with some questions regarding the laid-back system. Finally, the questionnaire closed off the experiment.

Subject A

A was unused to PDAs, having never used or owned one before. However he was very keen on the concept of a PDA and was eager to try out the idea. His experience of mobile technologies was limited to usage of his cellphone, which he used frequently for communication and occasionally for important reminders. He also has infrequently made use of his phone's WAP capabilities to conduct quick searches on Google. In most situations when experiencing an information need he prefers to use a "pen and paper" approach of making a note and then later following up on it when at a computer.

After his first week of using the PDA, A was slightly less enthusiastic about the PDA, citing a lack of software capabilities as a reason. He liked the search capabilities of the laid-back system, stating that it allowed him to carry on working with other things while the search was in progress. When asked what he meant by that, he said that "when using a web browser, I tend to watch and wait for the results because I'm not sure how reliable the mobile's connection is." On the desktop, however, he said that he had continued to use Internet Explorer and Google in conjunction from sheer force of habit. He has made occasional use of the laid-back search system, but feels that once he has a result, there is no need for him to have access to the original search query.

In particular, the device's organisational capabilities appealed to A – most notably the calendar and note-taking applications. Having previously used a paper-based calendar and occasionally his cellphone for reminders, he likes the ability of the PDA to allow him to view appointments for his upcoming week and to provide him with reminders for important events. Despite this, he doesn't always carry the PDA around with him as it is too big to fit comfortably in his pocket along with his cellphone. He feels that his cellphone is of far more importance and thus it gets priority over the PDA. When he had the PDA with him, he used it to create searches and to store documents from those searches. When asked about his usage of the web-link facility, he said that he "[doesn't] see the point of it because I can just use the Internet Explorer's favourites to keep links."[3]

By the end of his second week of using the PDA A was feeling ambivalent towards the device, finding it both useful and not useful. As mentioned the previous week, he finds the PDA useful for its organisational capabilities, but only when away from his desk. As he works at his desk most of the time, these moments of usefulness are few during the course of an average day. In fact, he has been spending so much more time working at his PC (conducting research) that he has lost the need for using his pen and paper approach of note-taking, preferring to simply use the PC directly.

Of the laid-back system, he has been making use of the PDA system, rather than the desktop system. He felt that there was no need for him to review or go back over his search process once he had found a relevant result and thus felt little need to use the laid-back system whilst using the desktop. A found the mobile laid-back system useful, but pointed out that he doesn't like creating advanced queries using this system as it takes too long

[3]When quoting the subjects it has sometimes been necessary to change the tense of some words in order to maintain grammatical correctness – these changes have been bracketed.

select field, enter text, select next field, enter more text and so on. Instead, when he has an advanced query need, he prefers to wait until he is back at his desktop PC. An interesting observation made by A was that he doesn't feel the need to synchronise search results between the PC and the mobile device, particularly once he has a link to a relevant result or document, preferring to use the device's "Favourites" folder. However on the mobile device he prefers to use the laid-back system rather than a web browser and Google (as on the desktop) as he prefers the way the mobile laid-back system lays out the search results.

Three weeks after receiving the device, A's attitude towards it has cooled considerably and he no longer finds it a novel idea or concept. He is almost solely using the device for its organisational capabilities now, as he finds it more suited to this than his cellphone or his previous paper-based diary methods. Having said this, he still finds himself having to consciously think about using the device, to remind himself that it exists as a solution to some situation. He has been using the laid-back system on occasion to make small search queries (queries where the search term is short and uncomplicated). As before, he hasn't been using the desktop system, preferring to use Internet Explorer to access Google and to then store the results in the Favourites folder.

When asked if he had been using the laid-back system for managing other information types, such as documents or web-links, he stated that he "[hasn't] used it for documents, although I think I would if I needed it." When asked when he would experience such a need, he gave the example of working at home and wanting to have some documents available. Interestingly, he also stated that he didn't think that the small screen size would be a problem for viewing documents. As for the web-link data type, he repeated his view from the previous weeks that he prefers to use Internet Explorer's Favourites folder.

In the questionnaire A make some interesting comments. Firstly he felt that the mobile device is useful to him from a working perspective, but not from a personal perspective. He stated that if software he found useful for personal use was available then he would begin using the device within a personal context. An interesting comment (from our point of view) is his assertation that he feels no need to integrate the mobile device with a desktop system and feels that it functions adequately as a stand-alone device. Unlike his cellphone, he only carried the PDA around with him "most of the time" and felt that his attitude toward the device did not change over the course of the four weeks. He made extensive use of the search functionality, mostly ignoring the document functionality. When asked what

functionality he would add to the system, he stated that he would like to be able to access his emails through the system.

He showed a definite preference for the mobile system, making the observation that since his desktop system is always online he might as well just use a web browser to access a search engine rather than make use of the laid-back system. However he liked the on/off-line capabilities that the laid-back system offered and felt that they made connectivity transparent.

Whilst he felt that the mobile device did not improve his productivity from a work perspective, he felt that he would still use one given the opportunity.

Subject B

Having had previous experience in using PDAs, B was eager to use one for organising her day-to-day activities and to assist her with research work. Her current working practice is to make notes either on pieces of paper or on research papers themselves and to then follow up on those notes, be it to read the document more fully or to conduct some sort of information seeking behaviour. At present she has very definite information needs as she is currently researching background information for her Masters thesis.

During her first interview, B revealed that she had been using the mobile device for a multitude of tasks – offline mail reading and composition, task lists and using the voice recorder to create notes and reminders to herself. Up until now she has occasionally used the reminder facility on her mobile phone but prefers to use the PDA as she finds text input using the PDA quicker. Both devices are carried in her handbag, so she feels it doesn't really matter which she uses as she always has both with her. She has been using the laid-back system, but not extensively. When asked why not, she gave an example where she "needed to look something up and used the mobile system. I couldn't find what I wanted, so left it for later and then forgot about it." Also, she was not sure about query creation whilst offline, citing uncertainty regarding the accuracy of the query regarding what she was looking for.

Using the PDA still has a "novelty" feel to it and she still has to remind herself to use it, unlike her cellphone, which she uses automatically without thought for making calls or for checking reminders. However she has been finding the PDA useful, particularly for note taking. Often she finds ideas regarding her research popping into her head and makes use of the voice recorder built into the PDA to "jot" down these sound-bites. Having said

this, she finds the lack of mobile software support for her needs frustrating – she feels that there "isn't much [she] can do with it with the default software installed." Also, she feels hampered by the need to continuously charge the device, comparing it to her cellular phone, which she only charges weekly. The need to continuously charge the device (or otherwise potentially lose data stored upon it) coupled with the thought that she won't be keeping the device permanently has made her reluctant to fully adopt it. "I don't want to come to rely on it [the device] and to then have to give it back."

She hasn't been making use of the desktop system, citing two main reasons for this. Firstly, it is force of habit for her to load a web browser and to then access Google from there. Secondly, her desktop is cluttered with icons and shortcuts, making locating the laid-back system application icon time consuming. When asked why she allows her desktop to become so cluttered she said that most of the icons are links to web pages and that she is aware of the locations of most of the icons as she put them there. It was then pointed out that a potential solution could be to assign a keyboard shortcut to the application icon – she expressed surprise, said that she had never thought of that option and that she would give it a try.

During the second week of usage, she forgot to charge the PDA for a few days and had trouble re-charging it and switching it on. Instead of working toward a solution, she left it thinking that the device wasn't critical to her work and that she would "sort it out later." We helped her to re-charge the device and to get it up and running again. She still hasn't been using the desktop system, preferring to stick with her "tried and tested" methods of working.

B then went away for 10 days and took the mobile device with her as she knew that she would have access to a wireless network where she was going. She was still having difficulty getting the PDA to charge itself and it appears that the cable connection is faulty – we suggested that she use the cradle to charge the device. Due to her difficulties in charging the device, after a few days she discarded it and reverted to using "Post-It" notes to serve as reminder functions to follow up on information needs. She did use the PDA for a few searches and said that she would have used it more (for conducting more searches, as well as for note-taking) if the device's battery hadn't been so problematic. When asked about her usage of the laid-back system's document facilities, she stated that she "[hasn't] really been using it, because [she] prefers to arrange documents in folders." She is very comfortable working within a file-based model and doesn't want to move to a document-oriented model

because she feels that she has less "control" over access to the data.

The PDA is now being kept in her laptop bag as opposed to her handbag because "it was taking up too much space and [she] [doesn't] really use it that much anymore." She does occasionally make use of the laid-back search system but has a similar complaint to A – creating advanced searches are "too much effort" and she prefers to use more direct methods. In B's case, she often searches for data within a specific website. Rather than using the laid-back system and restricting the search to that site, she prefers to go directly to the site and use its built-in search system.

In the final questionnaire, B stated that should she have to choose, she would prefer to use a desktop computer over and above a mobile device as it is faster (both in terms of processing power and network access) and she feels that she can do more productivity-wise. She does feel that over time the device could become useful to her and an object of a personal nature, but felt that knowing she had to return the device made her avoid relying on it. Initially she saw the device as some sort of "nifty toy" but over time came to see it as a useful tool. The mobile device was useful to her in both personal and working contexts, though she felt that there is a need for more software on the device. She feels that the PDA definitely enabled her to become more productive and organised through task-lists and note-taking via the voice recorder function.

Her usage of the system was restricted not only by the knowledge that she had to return the device at the end of the experiment but also by the poor reliability of the device. With the battery giving her so many problems she was worried about losing data and thus was reluctant to rely on it fully. Whilst she recognises that she could synchronise the device with her laptop computer, she feels that this is too much effort and would rather just work with one system. She mainly used the mobile device for note-taking and basic searching, with most of her search queries taking place on the desktop.

Most of her usage of the laid-back system was concentrated on the search functionality, with her mostly ignoring the web-link functionality. She feels that there is little need for this as her web-browser "remembers" links that she has opened – either through auto-complete or through the browser history. She liked the offline capabilities of the system and made extensive use of this ability. However she feels that Google returns too many search results and when composing search queries offline, she was not sure of the validity of the query and had no way of checking this. Also, she feels that the system should alert her when it has completed a search query that was created offline as she often found herself creating a

query while offline and then forgetting to follow up on it.

Subject C

Out of all the test subjects, C is the most active researcher, with several ongoing projects. Thus, he has a continuous information need and requires a high level of organisation with his information. He has previously used a PDA and is thus comfortable with the notion of using a mobile device as part of his day-to-day work practice.

C made extensive use of the system during the first week and then was away on a trip for two weeks and didn't make use of the system while away. During this time he also missed the first two interview sessions. However, on his return he made extensive use of the desktop version of the system, preferring it to the mobile device for several reasons. Firstly, he spends a lot of time at his desk and thus has little need for the mobile device. Whilst searching he likes to view documents and browse through the results returned – something that he feels is not feasible on the mobile device due to its limited form factor. Secondly, he feels little need to search for information whilst outside of his working environment. Should he experience some information need in this situation, he prefers to wait until he is at his desk and able to focus completely on the task at hand. Another contributing factor for choosing not to use the mobile system were the problems he experienced synchronising information between the two systems, particularly when including files within the information packages.

Before making use of the laid-back system he made extensive use of Google through a web-browser but was frustrated by the need to repeat searches should he wish to go back and review search queries. He was delighted to give the laid-back system a try as it would allow him to quickly and easily go back and browse through search queries – a classic example of berrypicking behaviour in users. As he is always working at a desktop in a networked environment, he has never experienced connectivity problems and as such found the connectivity management aspect of the laid-back system of little use.

Despite his extensive use of the system and the usefulness he feels it offers him, he does need to remind himself to use the system, particularly the mobile version. He owns a cellular phone which is also used for the calendar and reminder functions. He has not been making use of the PDA for these functions, preferring to rely on his desktop computer, in conjunction with the cellphone.

C was most attracted by the ability of the system to keep his searches together and was appreciative of the information package concept. He often repeats searches for the

same terms over time and felt that this system would allow him to keep track of what he has already seen. However he felt that the ability to add *l*ocal documents to information projects was not useful but felt that being able to add remote (ie. web-based) documents would be a feature he would find useful. His reasoning here is similar to that of B in that he is used to working with a file-based model and organises his documents through the file-system hierarchy. Another factor here was the synchronisation issues he had been experiencing, as mentioned earlier. Investigating the issue found it to be related to the slowness of the mobile device – slowing down the transfer of information made the data transfer more reliable.

Integration of the PDA with desktop applications remains a problem for C, however, and he feels that it could be better integrated with all aspects of desktop computing. This, coupled with the relatively high cost of mobile-computing made him feel that the mobile device did not really have a place in his life, unlike his cellphone, which "does 90% of the stuff I need done when [I'm] away from my desk." He also cites a concern of losing the device, which meant that he did not carry it around as he does his cellphone. The need to carry two devices irks him and he feels that if properly integrated he would make use of a "hybrid" device.

He made use of laid-back the system whilst offline and found it useful as it removed the need for him to remember to search for some information. To quote: "If you're not in a rush for results, it's great." Interestingly, unlike B, he does not feel that the system should alert him when a search has been completed – this could be attributed to the fact that he makes more extensive use of the desktop system where it is possible to see the state of all objects at a glance.

Subject D

D is the only subject who has owned a PDA, in this instance a Palm V. At present she is not a busy person but realises that as her research work picks up, she will need to introduce some organisational capabilities to her work practices. She is the most interesting of our test subjects in that she owns a PDA, although she doesn't use it within her work environment.

One week into the experiment, D was not using the PDA for its organisational capabilities. She has been continuing her standard practice of relying on recall to keep herself organised, eschewing even pen and paper-based solutions, such as a diary. Should she experience some sort of information need, her practice is to make a note to remind herself and to

then follow up on it when at her desk. Primarily she is interested in the PDA for its ability to access her email and for software development towards her research work. She feels that if the input capabilities were "better" she would use the text editor for note-taking. When it was pointed out that she could use the voice recorder she expressed disinterest, stating that it "would take too long to listen to all of it." D has a similar complaint to B, namely that she feels that the default software is inadequate for her needs. When pressed as to what software she would need, she was unable to provide a definitive answer but felt that the existing software did not really aid her in working practices.

She has been using the mobile system for a couple of searches but said that she created them more out of curiosity to see how the system works rather than for some active information need. Whilst she carries the PDA around in her handbag, she hasn't been making much use of it, staying with her standard practice of relying on her memory to remember to look up things. She has, however, been using the desktop system for its document facilities, treating it as a method of providing her with "shortcuts" to her most frequently needed documents.

After two weeks she is still carrying the mobile device around with her in her handbag despite not making much use of it. When asked why, she pointed out that she used to do the same with her Palm V, carrying it around so that it "is available if [she] [needs] it." She still does not use the PDA for its organisational capabilities, preferring to rely on memory although she has begun using the calendar occasionally for reminders of appointments and meetings.

When asked how she sees the device as fitting into her world, she stated that she sees it mainly as an "information access device," rather than as something she would use to create information. Asked to clarify that statement, she gave the example of using the device to browse for some sort of information and then once she had found the information required, to later work with it at her desktop computer. She feels that the device is too small and its input capabilities too limited for her to effectively work with information on it.

During the course of the week she has been making use of the mobile search system and expressed surprise at the interface, stating that she found it very easy to use and learn. She said that she likes using it for browsing through search results whilst offline and to then mark relevant results for follow up and retrieval later. When asked how she sees the system, she said that she sees it as a "browsing system" – something that allows her to browse through information – rather than as a "searching system" which would allow her to

directly specify and find information that she needs. This is an interesting statement as it supports the notions of Bates (1989) and Twidale *et al* (1995) [Bat89, TNST95] that search is primarily a browsing process, with Bates describing it as berrypicking. It can be argued that how directly users can access their desired information is related to the "quality" of the query that they use, but query creation and formulation is a research field in its own right and is beyond the scope of this work.

Despite her offline usage of the system, she was not aware that she could create search queries whilst offline and expressed surprise when this was pointed out to her, saying that she would definitely find that functionality useful and that she would give it a try. However she has been forgetting to use the system, mainly through force of habit and still needs to remind herself of its existence and capabilities.

After three weeks she has been using the device more and more outside of her development needs. She has begun to rely on the calendar application as she is now much busier. To a limited extent she has been using the notepad application, though she prefers to use 'Post-It" notes as she tends to forget the existence of notes on the PDA. Also, she really dislikes text entry on the PDA, finding it slow and cumbersome. When enquired about how she used the Palm V, she stated that she also disliked the text entry on that device but that after time she "got used to it." She feels that she could get used to the text entry methods on the iPaq given enough time.

D has been making use of the search facilities on both the mobile and desktop systems, using what is available with no particular preference for either system. She says that her work practice and pattern has more or less remained the same despite her beginning to rely on the PDA for personal information and organisation. She no longer uses the laid-back system to manage her documents because she wanted to work with information on the PC and view/edit this information on the mobile device but feels that she cannot do this. When it was pointed out that viewer applications (such as Adobe Acrobat Reader) exist for the mobile device, she expressed surprise and said that she was unaware of this.

Unlike the other subjects, D made extensive use of the web-link functionality, especially making use of the meta-data facility associated with this information type. She used the search system to find links that she wanted and then created entries for those links within the information package, adding notes to the links to remind her why they were useful to her. However she made very little use of the file management aspect of the system.

She felt that over time the system would become more useful to her, especially if she

began to use the PDA more frequently. However she feels that she doesn't form part of the "target group" for mobile devices and that this impacts her usage of the device. From her comments in earlier interviews, we assume that she is referring to the fact that she typically has very little need for the device due to her low organisational needs. Correspondingly, she mainly made use of the desktop system to conduct searches. She did feel that if her life became busier then she would make more use of a PDA.

D makes a very valid and interesting observation during the questionnaire: she states that as the PDA has its own functionality and own operating environment, it needs to "stop trying to be a copy of the desktop." This reflects our observations that PDAs and mobile computing in general present their own interaction needs and interface requirements, and hence a corresponding interaction metaphor or paradigm is needed.

She carried the device around with her "most of the time" but still had to consciously think about using it to perform tasks. When asked if she would continue to use the device in the near future, she was ambivalent, stating that her response would depend on how busy she is in the future. It is interesting to note that she views the device primarily as an organisational tool, rather than as a generic computing device.

6.3.2 Conclusions from the real-world observation

As can be seen from the user responses, search was by far their most common mobile information management activity. It would seem that users experience very little need for access to documents or web pages whilst "on the road." However they do experience some sort of information need – hence their search activity. Of interest is that the results of searches are typically web pages or documents themselves, yet the users generally don't feel inclined to access these while mobile! While this may seem contradictory, it makes sense if one considers it from a time-management perspective: people generally feel more organised and ready to work if they sit down at their desktop computer knowing that the resources and information that they need are at their fingertips.

Users definitely found the system's laid-back style of interaction to their liking, with two of the four users in the experiment explicitly stating their appreciation for the interface design and the ability to work offline. Indeed, all of the users browsed through search results whilst offline, with some creating more searches based on these results. The users found the transparency of the system's connectivity to be useful, with one user stating that it allowed him to "just create a search without any worries." Interestingly, none of the users

found the system's document and web-link capabilities of any great use, although one user did state that if the web-link function "pre-fetched" the web-page in question it would be of more use. One possibility for users not finding the document facility of great use is that document access is typically laid-back in nature – documents are usually stored locally and users work with them as they wish, when they wish. Thus our system has very little impact in this field, particularly as mobile devices are unsuited for large-scale viewing or editing of documents. The web-link functionality did not appeal to users, with many preferring to continue working with shortcuts and bookmarks in their web browser of choice. One reason, supported by the user's comment that he would prefer it to pre-fetch web pages, could be that by themselves web-links do not really aid users that are offline. Should those pages be retrieved, however, the web-link would suddenly be of much use to the mobile user.

However despite the users' appreciation for the laid-back system and its capabilities, they were generally reluctant to fully adopt the device. This reluctance stems from two main sources: the perceived lack of reliability of the device and the users' knowledge that they would have to eventually return the device. The different users all experienced varying levels of reliability with issues spanning both hardware and software domains. Considering the personal information generally handled by a mobile device it is little wonder that users were reluctant to fully adopt it, as they have cellular phones.

As for the adoption of the laid-back system into the users' working practices, there were mixed results with two of the users preferring the mobile system to the desktop system and the remaining two preferring the reverse. Users had mixed feelings regarding the information package concept, particularly for managing documents. However they were enthusiastic about the ability to store searches and their associated results despite having to remind themselves that the system was an available option. It would seem that just as users are "locked-in" to a file-based mental model, so they are to a "web-browsers-are-for-searching" model. Many users not only acted from force of habit when using their web-browsers to conduct searches but also from the notion that to search the web they need to use the tool that they browse the web with. We feel that user attitudes towards searching and search tools could be the subject of further investigation.

6.4 Chapter Summary

Through heuristic analysis we managed to iron out most of the "bugs" in the system and to remove any interface consistencies that were present. The problems experienced by the two systems are similar in nature, although a proportion of the mobile system's issues arose from the mobile device's interface standards rather than the laid-back system's interface. With the interface behaving in a consistent manner we were able to proceed to the task-based observation stage assured that the interface's *behaviour* was correct.

During the task-based observation stage some issues regarding data consistency and "special-case" situations were uncovered and these problems rectified. With a random sample of people taking part in the experiment many user behaviours were observed along with varying degrees of difficulty in using the system. The subjects all had similar performance times, indicating a consistent experience among the users. Typically, the second half of the experiment was relatively faster, indicating that the interface and system behaviours of the two systems was as expected by the user. Due to the always-on connectivity of the experiment, the laid-back interface never really came into play. However the experiment and its results allows us to state that first-time users are capable of using both the mobile and desktop systems with very little or no help whatsoever. Thus we can state that at this stage the interface behaves and appears consistent and that it is "user-friendly."

The final stage, real-world observation, served to evaluate the laid-back interaction style and it was commented on by the users involved. Generally speaking, the mobile devices were well-received although the users felt reluctant to fully accept the mobile devices into their lifeworlds, with some reasons being a perceived unreliability, input difficulties and the knowledge that the device does not belong to the user.

Adding problems such as the limited availability of productive software applications for the mobile device and the issues of data input quickly makes the mobile device seem little more than a novelty toy that will only realise its potential once the technology matures. At present mobile devices would seem to require too much cognitive effort to use effectively within a day-to-day environment and developers of both the hardware and the software would do well to learn from cellular handset design: keep interactions as simple as possible!

The users found the search aspect of the laid-back system to be of most use, followed by the web-link functionality. Most users didn't find the file-handling capabilities of the system to be of any use at all, feeling that the file-based model works best for them. We

feel that this resistance is due to the users' familiarity with the file-based model – at present there is no representation for search as a unit of information and thus users are content to work within a document-space model for this type of information. A possible compromise here would be to modify the information package display so that it can *appear* to be file-based or document-based, dependant on the user's preference. An object hierarchy could be displayed or alternatively "flat" object listing filtered by some criteria, giving users further control over their interaction with the system.

Chapter 7

Conclusions

In our introduction we presented our views that mobile devices require new interface designs and new ways of thinking about how users interact with the devices and their data. We outlined our idea for a system that not only embraces laid-back interaction but also the notion of information packages to allow users to keep their searches and associated data in one place. With this system we aimed to reduce external influences that may affect users of mobile devices (such as connectivity concerns) and to investigate the influence of laid-back interaction on user interaction with mobile devices.

From the results presented in the previous chapter, several patterns emerged. Below we list some of the conclusions that arise from our system and the experiments performed upon it. This list is split into two parts – those conclusions that apply to mobile devices in general and those that apply to our system specifically.

7.1 Mobile device-specific conclusions

The conclusions presented in this section apply to mobile devices in general, rather than specifically to our system and its design. These conclusions arise from our observations of the users' attitudes towards and usage of the mobile devices during the course of our experiments.

Users need to trust the systems they use

From the real-world observation many of the users felt that the mobile device was unreliable and that they could not trust it with personal information. This lack of trust did not

arise from any form of security concern – rather from concerns over the persistence of the information and thus, the accuracy and validity of the information presented to them. Users were unwilling to begin to rely upon a device that they felt would let them down, particularly those users that were using the device for its organisational capabilities.

Similarly, users need to feel a similar level of trust for the software present on the system. None of the users experienced problems with the software that they felt were serious issues, although two users did experience difficulties with synchronisation. Coupled with their inclination to prefer the desktop-based system, one could conclude that these users were less inclined to use the mobile system as they felt it was not fully trust-worthy.

People treat PDAs as they do cellphones

A problem here is that users expect the devices to have reliability levels similar to those of cellphones. The issue here is that PDAs are *not* cellphones – they are computers and should be treated as such. Cellphones are specialised devices that perform one primary function – to enable communications. PDAs, on the other hand, are multi-purpose devices that have a vastly different primary purpose, namely information access. It is only recently, with the advances in mobile technology, that the distinction between the two device types has begun to blur.

However the design of the devices has remained constant throughout this shifting definition, giving rise to new problems as users begin to form new expectations. These problems are multi-faceted and include interaction issues, data consistency issues and device reliability issues.

The user experience needs to be kept consistent between platforms

The distinction here is that the user *experience* needs to be consistent, not the user interface. During the task-based observation it was noticed that proportionally, users spent less time on the second half of the experiment than they did on the first. This indicates that they learned from their experiences during the first part of the experiment and were able to apply them to the second despite vastly different interfaces. A consistent experience not only makes learning how to use the system that much easier, but it also reduces the general cognitive overhead for the user as they do not need to wonder if actions that are valid on the desktop system are also valid for the mobile system.

It is not always feasible to directly translate functionality to mobile platforms

This was seen during the task-based observation stage when some users were confused by the mobile device's adaptation of the desktop Windows computing environment. This confusion largely arose from an incomplete adoption of the platform conventions (such as the ability to TAB the focus caret between widgets) but also from the vastly different input mechanisms employed.

Whilst the tap-and-hold method of calling up a context menu is a good compromise to the right-click used on the desktop, it doesn't take into account users' exploratory behaviour. Simply put, searching for functionality using tap-and-hold can result in a large amount of wasted time and creates frustration for the user. In this instance the direct translation of functionality is that akin to the desktop system, there is no indication of the existence of the context menu. On the mobile system where screen space is limited and the context menu thus has a central role for access to functionality, some sort of indication of the presence of such a context menu is needed.

Similarly, when creating mobile versions of software that exists on the desktop, one needs to take into account the limitations of the mobile device – in terms of both input and the screen space available. Where necessary, different widgets need to be employed to either overcome the mobile device's limitations or to provide extra short-cut functionality in order to save time. An example of this short-cut functionality can be seen in the drop-down list used in the object creation dialog to select what type of object is being created.

Users don't explicitly require integration of desktop and mobile systems

We would expect users to require that a mobile system integrate with some corresponding desktop system in order to allow them to work with their information. However from our study it would seem that the presence or lack thereof of such functionality does not affect a user's decision to use a system. Whilst some users did comment that they would like a more complete integration of the mobile device with their desktop system, it did not dissuade them from using the device.

7.2 System-specific conclusions

Presented here are our conclusions that apply specifically to our laid-back information management system.

Laid-back interaction is beneficial to mobile users

All the users commented on how useful they found it to be able to browse their search results whilst offline – the only factor limiting the user's pace of interaction being the presence or absence of data. The degree of control users were offered was suited to the unique requirements of the mobile device in that the system never demanded the user's attention, nor did it behave in a pre-emptive fashion. With a very short learning curve, users were very quickly familiar with the system's interface and able to make use of all of its functionality. The unobtrusiveness of the interface enabled the user to focus on their task at hand without undue distraction and provided for a smoother user experience.

Users have a very strong file-based mental model

This was highlighted during the task-based evaluation phase particularly by the users who attempted to save their searches. We feel that document-space information management presents users with an information interaction opportunity that is far more flexible than the current file-based metaphor. However users prefer to "work with what they know" and this mental model needs to be "broken" or shown to be limiting before users will readily adopt a document-space metaphor.

A counter-argument is that email systems are document-based yet they are readily adopted by users. However these systems have *always* been based in a document-space and thus users have had no exposure to a file-based email model, hence their adoption of the document-space model. By introducing our search management tool within a document-space model we feel that users will be more inclined to work within this space. With more powerful filters and visualisation tools it is likely that users could be persuaded to move all their information management tasks to the document-space realm. These visualisation and filtering techniques could form the focus of some future work.

There is a need for search management tools

From the users' reactions to and usage of our system it is apparent to us that there is a need for a search management tool – for both mobile and desktop systems. We feel that a solution, such as the system presented here, that shares this information seamlessly between the desktop and the mobile is most ideal as it truly allows users to access their information regardless of where they are. However it would seem that this management need does not

extend to other information types, such as web-links and files. Users appear to be content with solutions that are available to them at present. It must be noted that simply because users are content to work with current solutions, it does not imply that these solutions are necessarily the best options available!

This need for a search management tool is especially relevant for the mobile device where repeating processes is time consuming and may not be a viable option to the user due to time or cost constraints. This is particularly true in the situation where the user may be paying for bandwidth or data consumed. On the desktop system users are typically less pressured by time constraints and are thus able to repeat search queries if necessary. However a search management tool will allow users to make more effective use of the search results returned in that they can return to them and browse them at any time. Other scenarios, such as external ranking of results or even automated pre-fetching of results now become possible with search management tools.

7.2.1 Final Summary

From the experiments one crucial fact was shown – that laid-back interaction is of definite benefit to mobile users with their continuously changing environments and attention levels. Sit-forward solutions to searching, such as web-browsers, are not suitable as they force the user to consider external factors such as network connectivity and satisfying such a concern may distract the users from their task at hand. However this only applies if the user is happy to be using the device in question: factors such as the perceived device reliability and data input difficulties affect the user's attitude towards the device.

Unlike laid-back interaction, information packages were not as well received by users. Their usage for collating searches and their associated results was appreciated, but users did not see much need for them to manage mixtures of information – such as web-links and files. In part this resistance can be attributed to their familiarity with file-space models as opposed to the document-space model promoted by information packages. Future work within this area should include focus on more powerful filtering and sorting tools than those provided by our system. With effective information management techniques information packages have the potential to provide users with far more control over their information than that offered by file-space models.

In short, from a technological perspective, mobile devices are an immature technology and it will be some time before users come to accept them into their lifeworlds while they

have more mature technologies available, such as cellphones. We feel that laid-back interaction has a role to play in the development of mobile software and that once users come to accept these devices into their lifeworlds, it will form a crucial part of their experience.

Developers seeking to create information management tools for mobile systems need to take into account user resistance to change – any system created will need to be very closely integrated into any existing systems so as to minimise disruption to users' established working patterns. This practice, combined with the non-preemptive principles of laid-back interaction, will greatly increase the likelihood of the system being accepted by users into their lifeworld.

Appendix A

The ILBEngineObject Interface

The `ILBEngineObject` interface is implemented by all objects that wish to be managed by the Laidback Engine. The properties, methods and events presented by the interface allow the engine to manipulate the objects and to pass data from the objects back to the user interface. There are minor differences between the interface implemented on the mobile system and that on the desktop system – these arise from the different object management methods that the engines were forced to employ due to limitations. Listed below are the events that are common to both platforms:

ConnectionLost :: `LaidBackObjectHandler`

Indicates to the engine that a network connection has been lost

ObjectStatusChange :: `LaidBackObjectHandlerReturn`

Notifies the engine that the object's status has changed

ObjectStatusLog :: `LaidBackObjectHandlerReturn`

Instructs the engine to log some event in the activity log

Listed below are the methods and properties common to both systems:

ComparisonData :: `string`

Data used for comparing object types during synchronisation

Key :: `string`

The object's unique identifying key

LastMobileUpdate :: `long`

FILETIME object was last synchronised

LastUpdate :: `long`

FILETIME object was last modified

LogData :: `string`

Object status that is stored in actvity log

ObjectStatus :: `LaidBackObjectStatus`

Enumeration representing the object's current state

ObjectType :: `string`

String representation of the object's type

Picture :: `Icon`

Icon used to represent object

StatusMessage :: `string`

Object's current status message to be displayed in object listing

Title :: `string`

The title of the object

Destroy() :: *no return type*

Destroys the object and removes any resources it may have been using

Init() :: *no return type*

Method to start the object and begin its work

ShowDialog(`LBDialogType`) :: `DialogResult`

Display the object's dialog and return the result of that dialog (ie. OK or Cancel)

ToString() :: `string`

String representation of object

A.1 Desktop implementation

One extra event is added in this class:

CreateNewInstance :: `LaidBackObjectHandlerReturn`

Instructs the engine to create a new object of some type (eg. new search based on some result)

Listed below are the additional properties and methods:

ObjectStopped :: `bool`

Indicates if the object's work was manually halted (eg. cancelled search)

ObjectSyncType :: `LBObjectSync`

Enumeration controlling the object's synchronisation

GetObjectData() :: `ArrayList`

Returns the object's data within an `ArrayList` for serialisation

GetSyncData() :: `string`

Returns the object's data in a `string` for synchronisation

SetObjectData(`ArrayList`) :: `string`

Initialises the object with data, returning the object's key

SetSyncData(`string`) :: `string`

Initialises the object with data from a synchronisation, returning the object's key

Start() :: *no return type*

Starts the object after some work has been stopped

Stop() :: *no return type*

Halts the object's current work and kills the processing thread

A.2 Mobile implementation

The differences between the mobile and desktop versions largely arise due to the mobile platform's lack of serialisation support and its limited thread-handling capabilities. Two events are added over and above the base implementation of the `ILBEngineObject` interface:

ObjectAction :: `LaidBackObjectHandlerReturnData`

Instructs the engine to create a new object of some type

ObjectThreadExit :: `LaidBackObjectHandlerReturn`

Indicates to the engine that a thread has completed and should be removed from the thread-pool

The mobile system requires very little extension from the base interface:

DisplayingResults :: `bool`

Informs the engine if an object currently has a result display open

HasMarkedObjects :: `bool`

Indicates to the engine if the object contains any marked items. Used to reduce computations needed

GetObjectData() :: `string`

Returns the object data in string form for either storage or for synchronisation

SetObjectData(`string`, `DataLocation`) :: `string`

Initialises the object with data from some source

Appendix B

LBEngine description

As with the `ILBEngineObject` interface, two versions of the `LBEngine` class exist, one for each platform. Again, these differences mainly arise due to the thread-handling and serialisation capabilities of the respective platforms. Listed are the events common to both platforms:

FoundConnection :: `LaidBackHandler`
: Indicates to the UI that a connection has been found

LostConnection :: `LaidBackHandler`
: Informs the UI that a connection has been lost

MessageReceived :: `LaidBackHandlerReturn`
: The engine wishes to display some message to the user

ObjectAdded :: `LaidBackHandlerReturn`
: Alerts the UI that a new object has been added, returning the object's key

ObjectDeleted :: `LaidBackHandlerReturn`
: Informs the UI that an object has been deleted, passing the object's key

ObjectStatusChange :: `LaidBackHandlerReturn`
: Passes on an object's notification that its status has changed

SyncEnd :: `LaidBackHandler`
: Informs the UI that a synchronisation operation has completed

SyncStart :: `LaidBackHandler`

Alerts the UI that a synchronisation operation has begun

The classes share no common properties as the mobile version does not make use of any public properties. Shared methods include:

addObject(`ILBEngineObject`) :: `string`

Adds object to the object listing, returning a unique identifier key

Dispose(`bool`) :: *no return type*

Shuts down the engine and stores all information in memory

getObjectByID(`string`) :: `ILBEngineObject`

Returns the specified object

removeObject(`string`, `bool`) :: *no return type*

Removes the specified object, sending a notification to the other platform if necessary

B.1 Desktop version

The desktop version of the laid-back engine (`LaidBackEngineDesktop`) adds four extra events:

DebugMessage :: `LaidBackHandlerReturn`

Used to pass information on to the communications debug window

DeviceConnect :: `LaidBackHandler`

Alerts the UI that a mobile device has been detected

DeviceDisconnect :: `LaidBackHandler`

Informs the UI of a device disconnection

SyncFail :: `LaidBackHandlerReturn`

Notifies the UI that some error occurred during synchronisation

The following properties and methods are unique to the `LaidBackEngineDesktop` class:

DeviceIP :: `IPAddress`

`IPAddress` pointing to connected mobile device. `Null` if no device connected

InformationPackageFile :: `string`

Path to current information package

InformationPackageTitle :: `string`

Title of current information package

getListObjects() :: `ArrayList`

Returns a listing of all objects contained within the engine

getListObjects(SyncType) :: `ArrayList`

Returns only those objects with specific synchronisation parameters

getListObjects(string) :: `ObjectListBoxItem`

Returns only the requested object

getObjects(Type) :: `ArrayList`

Retrieves all objects of type `Type` and returns them inside an array

Init() :: *no return type*

Initialises the engine, resetting all variables

RemoveDeviceIP() :: *no return type*

Forcibly disconnect the mobile device

SetDeviceIP(IPAddress) :: `bool`

Set the IP address of the mobile device

ShowObjectDialog(string, LBDialogType) :: `DialogResult`

Display an object's create/edit dialog

StartObject(string) :: `bool`

Starts up an object that was halted by the user

StopObject(string) :: `bool`

Forces an object to stop any work it is currently performing

SynchroniseWithDevice() :: *no return type*

Begin the synchronisation process with the mobile device, if connected

B.2 Mobile version

The mobile version of the laid-back engine class (`LaidBackEngineMobile`) is of a much simpler nature due to the fact that it does not handle the synchronisation of data. Listed below are methods that are unique to the mobile class:

connFound() :: `no return type`

Tell the engine that a connection exists and instruct all objects to begin working

displayObjectResults(`string`, `bool`) :: *no return type*

Displays the data screen of the specified object

getObjects() :: `Hashtable`

Returns all the objects contained within the engine, paired with their keys in a hashtable

Load() :: *no return type*

Instructs the engine to load the information package from persistent storage

Store() :: *no return type*

Instructs the engine to commit the objects to storage

Appendix C

Synchronisation codes

Synchronisation is governed by a small set of commands followed by their parameters. Each message string is delimited using `TAB` characters. Listed below are the commands employed along with a short description of each. For brevity the list of parameters for each command has been omitted.

The synchronisation process takes place on the desktop system, although the mobile device plays a server role. To start the process, the desktop system requests an object listing from the mobile device and the process follows automatically as the object listing is returned. Once the object listing has been received, the mobile list is compared against the desktop system's list and objects are updated accordingly.

Command	Sender	Description
`CL`	Mobile	The mobile has completed the object listing
`DI`	Both	An object has been deleted remotely
`DISC*`	Both	The remote system is disconnecting
`L`	Mobile	The mobile is returning an object listing
`RL`	Desktop	Request mobile device to return object listing
`RI`	Both	Request a specific object from the other system
`R`	Both	An object is being sent to the system
`UI`	Both	An object has been updated by another system

Appendix D

Heuristic Evaluation sheets

Included in this appendix are the forms used by the evaluators for the heuristic evaluation stage of the evaluation process. The scoring sheets were broken down into areas of the program to which the particular problem description applies to. Scores and heuristic codes were then filled in the relative cells for that problem description.

LIM – LaidBack Information Management
Heuristic Evaluation

This system aims to investigate the usage of laidback interaction and its suitability for mobile devices, particularly when working with information. Central to this is the concept of an "information package" – a place where users can store a mixture of information types in order to satisfy some information need. In this system, information is handled in one of three different categories: search queries and their associated results, files or documents and web page links. This system has been designed to work both "online" and "offline," allowing users to create search queries even when no connection is available.

The aims of this heuristic evaluation step are to ensure that the interfaces of the desktop and mobile systems are "usable" according to the heuristics listed on the accompanying sheet. Explore the system, perhaps making use of the tasks below and record your observations.

Score sheets are provided – please note what the problem was and under the appropriate column, assign a severity score (on a scale from 1 – 5) and the heuristic number which is appropriate.

Thank you for your time!
Richard Schroder

Some suggested tasks for you to try out:

Working with searches

- Create a search query and retrieve the first 20 results
- Create an advanced search query, using phrases. Eg. Search for the phrase "laidback interaction." Give the query a meaningful title
- Select some text in a document and drag it over the LIM window to create a new search
- Using the mobile device, create a search query and synchronise it to the desktop system
- Create a search query related to one of the results from your previous queries
- Edit a search query – change the title
- Edit a search query – change the query itself
- Edit a search query – change the synchronization type
- Try create a query with no network connection (unplug the desktop or disable the WLAN on the device)

Working with files

- Add a file to the information package
- View the file, copy it to the desktop
- Drag a file into the information package

Working with web links

- Add a web page link to the information package
- View the web page
- Add some notations to the link
- Drag a link from a web page into the information package

Mobile Device

Interface / Problem	Main Display		Adding object		Viewing object		Modifying object		Other	
	Score	Heuristic	Score	Heuristic	Score	Heuristic	Score	Heuristic	Score	Heuristic

Desktop system

Interface / Problem	Main Display		Adding object		Viewing object		Adding object		Other	
	Score	Heuristic	Score	Heuristic	Score	Heuristic	Score	Heuristic	Score	Heuristic

Do you have any other observations / suggestions to make about the system?

Scoring

1 **Severe issue**
The system is hampered by this and it must be rectified immediately

2 **Serious problem**
The user is restricted by this problem, it must be sorted out

3 **A problem**
It is a major irritation to users, warrants further attention

4 **Minor issue**
It is a source of irritation to the user and should be eliminated

5 **Minor irritation**
This is something which bugs the user but does not hamper usage

Heuristic Categories

H1 **Visibility of system status**
Is the user made aware of what the system is currently doing? In other words, does the system reflect its internal state?

H2 **Match between system and real world**
Does the system match its real world environment by avoiding technical terms and jargon?

H3 **User control and freedom**
Is the user able to reverse actions? Can they cancel unwanted events?

H4 **Consistency and standards**
Does the system follow platform conventions for the device, should any exist? Does it feel consistent with the rest of the device's systems?

H5 **Error prevention**
Are errors reported in a friendly fashion? Does the system rather prevent them than report them?

H6 **Recognition rather than recall**
Objects, actions and options should be visible – users should not need to remember information from one part of the dialogue to another.

H7 **Flexibility and efficiency of use**
Are users able to tailor frequent actions? Are there "shortcuts" for the more experienced users?

H8 **Aesthetics and minimalist design**
Do dialogues contain any information which is not relevant to the user's interactions?

H9 **Users aided in recognising, diagnosing and recovering from errors**
Are errors plainly expressed and are users aided in recovering from them?

H10 **Help and documentation**
In the event that help is needed, is it available to the user? Is any such information provided relevant and easy to read?

Appendix E

Task-Based Evaluation Documents

Presented here is one of the scoring documents used during the task-based evaluation stage. Each user was presented with a unique sheet – the order of the individual tasks and the system order was rotated. This ensured that while every user performed the same set of tasks, they were done in different orders, thus eliminating any bias or influence a particular task may have had on another. Presented in Table 10 below is a list of the ordering of the tasks for the test subjects.

Subject	System Order	Desktop Order	Mobile Order
1	Desktop, Mobile	Search, File, Web	Search, File, Web
2	Mobile, Desktop	Search, Web, File	Search, File, Web
3	Desktop, Mobile	File, Search, Web	Search, File, Web
4	Mobile, Desktop	File, Web, Search	Search, File, Web
5	Desktop, Mobile	Web, Search, File	Search, File, Web
6	Mobile, Desktop	Web, File, Search	Search, File, Web
7	Desktop, Mobile	Search, File, Web	Search, Web, File
8	Mobile, Desktop	Search, Web, File	Search, Web, File
9	Desktop, Mobile	File, Search, Web	Search, Web, File
10	Mobile, Desktop	File, Web, Search	Search, Web, File
11	Desktop, Mobile	Web, Search, File	Search, Web, File
12	Mobile, Desktop	Web, File, Search	Search, Web, File
13	Desktop, Mobile	Search, File, Web	Web, Search, File
14	Mobile, Desktop	Search, Web, File	Web, Search, File
15	Desktop, Mobile	File, Search, Web	Web, Search, File
16	Mobile, Desktop	File, Web, Search	Web, Search, File
17	Desktop, Mobile	Web, Search, File	Web, Search, File
18	Mobile, Desktop	Web, File, Search	Web, Search, File
19	Desktop, Mobile	Search, File, Web	File, Web, Search
20	Mobile, Desktop	Search, Web, File	File, Web, Search
21	Desktop, Mobile	File, Search, Web	File, Web, Search
22	Mobile, Desktop	File, Web, Search	File, Web, Search
23	Desktop, Mobile	Web, Search, File	File, Web, Search
24	Mobile, Desktop	Web, File, Search	File, Web, Search

Table 10: Task order for each test subject

LIM – LaidBack Information Management
Task-based Analysis

Thank you for taking the time to be involved with the testing of our system!

The LaidBack Information Management (LIM) system introduces the concept of "information packages" – a place where you can keep a mixture of information as a solution to a particular research or information need you may have. There are three types of information you can work with: searches (which are performed by Google), files of any kind and web page links. At the moment, we are interested in improving the interface to the system in order to create a more complete user experience.

Please perform the tasks laid out below to the best of your ability. When you have completed a task, tick it off on the accompanying sheet. If, for some reason, you are unable to perform any of the tasks, place a cross in the space provided and please give a reason. Do not feel pressured to do things the "right" way – we are simply investigating if the software is usable or not. If you have a problem, the fault lies with us, the designers! At the end of the analysis please answer the questions and provide us with any comments that you may have. There are two sections to complete: one section is based on the desktop PC and the other on the mobile device.

Note that everything you do during this experiment will be recorded by a video camera. Should you object to this, please let the observer know before the experiment begins.

First, some details about yourself (circle where appropriate):

How computer literate would you say you are?

1	2	3	4	5
Never used one		Some experience		Expert user

Do you frequently make use of a search engine to find information?

1	2	3	4	5
Never		2 or 3 times a week		Daily

How would you rate your level of experience in using mobile devices (PDAs)?

1	2	3	4	5
Never used one		Some experience		Expert user

Have you studied any Computer Science courses? If so, what is the highest level course you have studied?

Do you make use of a particular search engine? (circle one)

- AltaVista
- Excite
- Google
- MSN Search
- Yahoo!
- Other: _____

If you use more than one search engine, do you have any preference for a particular engine? Why?

_____

_____

_____

_____

You are now ready to begin the tasks. Please perform the tasks to the best of your ability. Should you have any difficulties, queries or problems, make a note on the following sheets, providing the task number and a description of your problem.

Let the observer know when you are about to begin.

Tasks

Desktop System
Please complete the following tasks using the desktop system.

1. Searching for Information

1.1. Create a search query for sport at UCT and fetch the first 20 results.

1.2. Find the first 10 results for "UCT Computer Science."

1.3. Find the top 40 search results for "world cup soccer 2010" and store these on the PDA for carrying around with you.

1.4. Open some of the results from the last search which are relevant to South Africa's 2010 World Cup soccer bid.

1.5. Manipulate the display of the results of the soccer query so that only the results you feel most relevant are shown (just choose to display 2 or 3 results).

1.6. Select the result you feel is most relevant and make it easily accessible for future use. At the same time, create a search to find similar pages.

1.7. Delete the search for "UCT Computer Science."

2. Working with files

2.1. On the desktop is a folder called "Stuff." Add the files that are contained there to the information package.

2.2. One of the files needs to go home with you – have the system copy it to the PDA for you. Make sure that the title reminds you why it's so important to you!

2.3. Copy one of these files to the "Other stuff" folder on the desktop.

2.4. Open some of the files that you have added to the information package, to see what they contain.

3. Working with web links

3.1. Create links to these web pages, adding titles and notes as you see fit:
`http://www.uct.ac.za`
`http://www.google.com`
`http://www.supersport.com`

3.2. Open the links in the web browser and add another link to the information package from one of the pages.

3.3. You would like to access this link at home – copy it to the PDA.

3.4. Delete one of the links.

Desktop System

#	Comments

Tasks

Mobile System
Please complete the following tasks using the mobile system.

1. Searching for information

1.1. Create a search query for sport at UCT and fetch the first 20 results.

1.2. Find the first 10 results for "UCT Computer Science."

1.3. Find the top 40 search results for "world cup soccer 2010"

1.4. Open the results from the last search which are relevant to South Africa's 2010 World Cup soccer bid.

1.5. Manipulate the display of the results of the soccer query so that only the results you feel most relevant are shown (just choose to display 2 or 3 results).

1.6. Select the result you feel is most relevant and make it easily accessible for future use. At the same time, create a search to find similar pages.

1.7. Delete the search for "UCT Computer Science."

2. Working with files

2.1. In the personal folder is a folder called "Stuff." Add the files that are contained there to the information package.

2.2. One of the files is important to you. Make sure that the title reminds you why it's so important to you!

2.3. Copy one of these files to the "Other stuff" folder in the personal folder.

2.4. Open some of the files that you have added to the information package to see what they contain.

3. Working with web links

3.1. Create links to these web pages, adding titles and notes as you see fit:
`http://www.uct.ac.za`
`http://www.google.com`
`http://www.supersport.com`

3.2. Open the links in the web browser.

3.3. Delete one of the links.

Mobile System

#	Comments

Follow-up questions

Would you make use of this system if it was made available to you?

Yes No

Please comment on your answer:

Do you feel that this system is a useful one? Why?

What additional features do you think would be useful for a system like this?

Are there any features that you think are not useful at all? If so, why not?

Was there any aspect of this system that you found difficult to use or confusing? Why was it so?

Can you think of any other way the system could be used? (Other than how you have used it today)

Do you have any other comments or suggestions to add?

Appendix F

Mobile tutorial

Included in this appendix is the short tutorial used during the task-based evaluation stage. This tutorial was given to test subjects who had never used a mobile device and was designed to introduce them to the widget and interface differences between the mobile and desktop platforms.

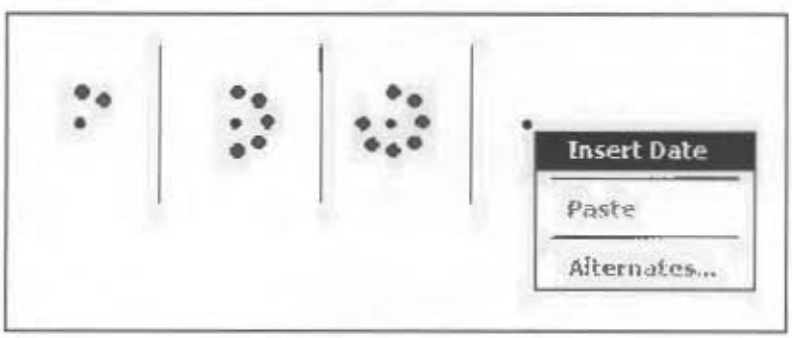
An Introduction to using PDAs

A Personal Digital Assistant (or PDA) is simply a computer that has been miniaturised to fit into the palm of its hand. The main difference, besides its size, is that it is specialised to work with personal information for a single person. Due to the small screen, several things are different from the Windows experience on a desktop computer. When the device is first switched on, the **Today Screen** is shown:

The display shows upcoming appointments that are in your calendar, outstanding tasks and information about any unread emails you may have.

At the top of the screen, in the top left, the Windows logo serves the same function as the *Start* button does on a desktop computer. The menu drops down to show you your options – a list of commonly used programs is shown. To access the programs installed on the device, tap on the *Programs* item. Along the top of the Start menu is a list of icons. These are the icons of the most recently loaded programs, with the most recent being the icon on the left.

When using a mouse on a desktop PC, often you can access extra functions by *right-clicking* to pull up a menu. However, when using a *stylus* to interact with a PDA, this isn't possible. Instead, the equivalent operation is known as *tap-and-hold*. Simply tap the stylus and hold it in place – as you do this, a circle of red dots forms to let you know that you are performing a tap and hold. When the circle completes, a pop-up menu is shown. You can then "release" the tap and select items from the menu.



Another important difference with PDAs is in text entry. The device has no keyboard and thus relies on a virtual keyboard or hand writing recognition for input. The handwriting recognition comes in three "flavours," one of which entails learning a new alphabet. As such, it tends to trouble new users and it is recommended that you stick to using the virtual keyboard instead. When text entry is appropriate, a keyboard icon is displayed in the lower right corner of the screen. Tapping this icon displays the keyboard. Simply tap the relevant keys and when finished, tap the keyboard icon again to remove it.

Another important note: unlike desktop computers, PDAs are designed to switch on and off immediately. Also, programs are designed to display their information. To achieve this, some programs are "minimised" instead of being closed when you tap the close icon. If the icon has a "x" in it, it will "minimise" (or hide) the program. To bring the program back, simply tap its icon in the start menu as though you were loading it. An icon that says "Ok" will close the program completely.

Lastly, the toolbar and menu-bar is in a different position from that on a desktop PC. As a user of a desktop PC, you may have become accustomed to the drop down menus and toolbars being at the top of the window. On Pocket PC they are located at the bottom of the screen and only the menus / icons for the currently visible program are displayed. In the picture showing the keyboard, the "New," "Edit," "View" and "Tools" menus can be seen. On the right of these is a toolbar icon for that application.

Appendix G

Noldus Observer Settings

Configuration Review - Laidback Interactions

Location : **C:\Documents and Settings\Rich\My Documents\Masters\Experiments\Full System\Task-based\Observer\Laidback Interactions.opp,.ocp,.opd**

Observation recorder: PC

Description

Experiments to investigate laidback interaction interface and to check that interface is "usable"

Settings

Setting	Value
Recording method	Continuous
Automatically generate key codes	No
Case sensitive	Yes
Duration of Observation	Open Ended

Independent Variables

Number of Independent Variables: 9

Independent Variable Name	Type	Values
Subject Number	Numeric	1 to 24
System Order	Nominal	Desktop, Mobile
		Mobile, Desktop
Desktop Task Order	Nominal	Search, Files, Web
		Search, Web, Files
		Files, Search, Web
		Files, Web, Search
		Web, Search, Files
		Web, Files, Search
Mobile Task Order	Nominal	Search, Files, Web
		Search, Web, Files
		Files, Search, Web
		Files, Web, Search
		Web, Files, Search
		Web, Search, Files
Subject Gender	Nominal	Male
		Female
Computer literacy rating	Numeric	1 to 5
Search frequency	Numeric	1 to 5
Experience with PDAs	Numeric	1 to 5
Highest computer science level	Nominal	CSC 100
		CSC 200
		CSC 300
		CSC 400
		CSC 500
		Inf

Subjects

Number of Subjects: 2

Subject Name	Code
Missing subject	?
User	u

Element Descriptions:

Subject Name	Description
Missing subject	
User	User of system

Behaviors

Number of behavioral classes: 3

Behavioral Class 1: Behaviour
Type: Nominal
Number of Elements: 3

Behavior Name	Code	Type	Modifier Class 1	Modifier Class 2
Confusion	C	State	(None)	(None)
Error	E	State	Software	(None)
Null behaviour	B	State	(None)	(None)

Element Descriptions:

Behavior Name	Description
Confusion	User showed confusion
Error	An error ocurred
Null behaviour	

Behavioral Class 2: Interactions
Type: Nominal
Number of Elements: 15

Behavior Name	Code	Type	Modifier Class 1	Modifier Class 2
Select	s	Event	Widgets	Software
Unselect	u	Event	Widgets	Software
Right-slick	r	State	Software	Widgets
Delete	d	Event	Laidback Objects	(None)
Create	c	State	Laidback Objects	(None)
Stop working	x	Event	Software	(None)
Drag	y	Event	Software	(None)
Drop	Y	Event	Software	(None)
Keypress	k	Event	Physical Objects	Software
Mark	m	Event	Widgets	(None)
Filter	f	Event	Software	(None)
Unmark	M	Event	Software	(None)

Null Interaction	v	State	(None)	(None)
Open	o	Event	Widgets	Software
Edit	e	State	Laidback Objects	(None)

Element Descriptions:

Behavior Name	Description
Select	Selected an object
Unselect	Deselected an object
Right-slick	or Tap-and-hold for mobile
Delete	Deleted an object
Create	Creating an object
Stop working	Stopped an object
Drag	Dragged an item
Drop	Dropped an item
Keypress	Used a keyboard shortcut
Mark	Marks a result using something
Filter	Filtered a view
Unmark	Unmarked a result using something
Null Interaction	
Open	Opened a file/link
Edit	Edit an object

Behavioral Class 3: Task
Type: Nominal
Number of Elements: 4

Behavior Name	Code	Type	Modifier Class 1	Modifier Class 2
Search Task	1	State	System	(None)
File Task	2	State	System	(None)
Link Task	3	State	System	(None)
Null Task	4	State	System	(None)

Element Descriptions:

Behavior Name	Description
Search Task	
File Task	
Link Task	
Null Task	

Modifiers

Number of modifier classes: 5

Modifier Class 1: Physical Objects
Type: Nominal
Number of Elements: 5

Modifier Name	Code

Missing Physical Objects ?	
Monitor	m
Sheet	s
Observer	o
Keyboard	k

Element Descriptions:

Modifier Name	Description
Missing Physical Objects	
Monitor	
Sheet	
Observer	
Keyboard	

Modifier Class 2: Software
Type: Nominal
Number of Elements: 7

Modifier Name	Code
Missing Software	?
System screen	m
Edit screen	e
Dialog box	d
Web browser	w
Desktop	D
Other screen	o

Element Descriptions:

Modifier Name	Description
Missing Software	
System screen	The main system screen
Edit screen	The edit screen for an object
Dialog box	A dialog boax that was displayed
Web browser	A web browser window
Desktop	A desktop element
Other screen	Some other screen

Modifier Class 3: Widgets
Type: Nominal
Number of Elements: 20

Modifier Name	Code
Missing Widgets	?
Button	b
Toolbar button	t
Text box	r
Drop down list	d
Check item	c
Search object	s
File object	f
Web link object	w

Search result	S
File contents	F
Web link content	W
List box	l
Next result set	n
Prev result set	p
Result set	R
Link	L
Pop up menu	m
Menu	M
Windows folder	[

Element Descriptions:

Modifier Name	**Description**
Missing Widgets	
Button	Button widget
Toolbar button	Toolbar button widget
Text box	Text box widget
Drop down list	Drop down list from button/box
Check item	Check item
Search object	Search object in listing
File object	File object in listing
Web link object	Link object in listing
Search result	Individual search result
File contents	Contents of file (opened in program)
Web link content	Contents of link (opened in browser)
List box	List box
Next result set	Chose to fetch next result set
Prev result set	Chose to fetch prev result set
Result set	Chose a result set
Link	Clicked on a link
Pop up menu	Clicked on a pop-up menu
Menu	Clicked on a window menu
Windows folder	

Modifier Class 4: Laidback Objects
Type: Nominal
Number of Elements: 4

Modifier Name	**Code**
Missing Laidback Objects	?
Search Object	s
File object	f
Web link	w

Element Descriptions:

Modifier Name	**Description**
Missing Laidback Objects	
Search Object	
File object	
Web link	

Search result	S
File contents	F
Web link content	W
List box	l
Next result set	n
Prev result set	p
Result set	R
Link	L
Pop up menu	m
Menu	M
Windows folder	[

Element Descriptions:

Modifier Name	**Description**
Missing Widgets	
Button	Button widget
Toolbar button	Toolbar button widget
Text box	Text box widget
Drop down list	Drop down list from button/box
Check item	Check item
Search object	Search object in listing
File object	File object in listing
Web link object	Link object in listing
Search result	Individual search result
File contents	Contents of file (opened in program)
Web link content	Contents of link (opened in browser)
List box	List box
Next result set	Chose to fetch next result set
Prev result set	Chose to fetch prev result set
Result set	Chose a result set
Link	Clicked on a link
Pop up menu	Clicked on a pop-up menu
Menu	Clicked on a window menu
Windows folder	

Modifier Class 4: Laidback Objects
Type: Nominal
Number of Elements: 4

Modifier Name	**Code**
Missing Laidback Objects	?
Search Object	s
File object	f
Web link	w

Element Descriptions:

Modifier Name	**Description**
Missing Laidback Objects	
Search Object	
File object	
Web link	

Appendix H

Real-World Observation Questionnaire

Included in this appendix is the questionnaire provided to all the test subjects in the real-world observation experiment. It aims to explore the user's experiences and opinions of the device and the laid-back system, complementing the interviews conducted during the course of the experiment.

Name: _____

Thanks for taking part in this experiment! Just to close it off, please could you return this questionnaire, along with your PDA to me as soon as possible.

The questionnaire is split into two parts – one deals with the Laidback Information Management system itself and the other with the device in general.

Many thanks,
Richard

Laidback Information Management System

1. What aspects of the laidback information system did you like or find most useful?

2. What aspects did you not like or find useful?

3. Which information type do you think you used the most?

Search query **File** Web-link

4. Which information type do you think you used the least?

Search query **File** Web-link

5. Can you think of any other information types that you feel the system should support? Why?

6. Do you feel that the system could become more useful to you over time?

Yes No

Why?

7. What would prevent or dissuade you from using this system?

8. What would encourage you to use this system?

9. Do you feel that you made use of both sides of the system (desktop and mobile) or had a preference for one of them? What caused this preference?

10. Would you use either of these systems on their own? If so, which one and why?

11. Did you make use of the laidback search system whilst offline? (ie. Did you create queries or browse search results whilst disconnected?)

Yes No

Did you find this functionality useful?

12. Did you ever find yourself with some information need whilst offline? Was the system able to help you in some regard? (eg. Perhaps create a search query while offline?)

13. Do you feel that mobile software needs to take into account the possibility of the device being offline?

Yes No

14. Do you feel that the laidback information management system handles this situation adequately?

Yes No

Please comment on your answer to the above:

15. If you could change anything about the system, what would it be?

16. Any other comments?

PDAs and Mobile Computing

1. Do you feel that the PDA has a place within your life – both in terms of your personal life and your working life?

2. If not, what would change this view?

3. Do you feel that there is a real need for mobile/handheld computing?

Yes No

4. Do you feel that it is possible to use the PDA as a stand-alone device? Or do you think it is necessary to have some form of synchronization between the PDA and a desktop PC?

5. Do you feel that the PDA could become an important personal device, such as the cellphone is? Or do you feel that this role is already filled by the cellphones/smartphones?

6. Did you carry your PDA around with you during the course of the experiment?

1	2	3	4	5
Never		Most times		All the time

7. If you currently own a cellphone, do you carry it around with you all the time?

1	2	3	4	5
Never		Most times		All the time

8. Did you have to consciously think about using the PDA, or did it just become second nature to you?

9. Would you continue to use a PDA for the near future if you had the option to do so?

Yes No

Why?

10. Do you feel that your usage of and attitude towards the PDA has changed over the last few weeks? If so, in what ways?

11. Ultimately, do you think that the PDA has enabled you to become more productive or organized?

Yes No

In what ways?

11. If you could change anything about the PDA itself, what would it be?

12. Any other comments?

Bibliography

[Bat89] M J Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Review*, 13(5):407–424, 1989.

[BDHS03] Victoria Bellotti, Nicolas Ducheneaut, Mark Howard, and Ian Smith. Taking Email to Task: The Design and Evaluation of a Task Management Centered Email Tool. In *Proceedings of the conference on Human Factors in Computing Systems*, pages 345–352. ACM Press, 2003.

[BGMP99] Orkut Buyukkokten, Hector Garcia-Molina, and Andreas Paepcke. Focused Web searching with PDAs. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):213–230, 1999.

[BJ00] George Buchanan and Matt Jones. Search Interfaces for Handheld Mobile Devices. *9th International World Wide Web Conference (Amsterdam)*, 2000. http://www9.org/final-posters/48/poster48.html.

[BLU04] "Bluetooth Protocol Specification", Last accessed August 2004. http://www.bluetooth.com.

[Buc04] Simon Buckingham. "What is General Packet Radio Service?", January 2000, Last accessed August 2004. http://www.gsmworld.com/technology/gprs/intro.shtml.

[CMLF02] Eric Chang, Helen Meng, Yuk-Chi Li, and Tien-Ying Fung. Efficient Web Search on Mobile Devices with Multi-Modal Input and Intelligent Text Summarization. In *Eleventh International World Wide Web Conference*, 2002.

[Col82] Irene Cole. Human Aspects of Office Filing: Implications for the Electronic Office. In *Proceedings of the Human Factors Society 26th Annual Meeting*, 1982.

[DB01] Nicolas Ducheneaut and Victoria Bellotti. Email as a habitat: An exploration of embedded personal information management. *Interactions*, 8(5):30–38, 2001.

[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 *Proceedings of 26th Annual international ACM SIGIR conference on Research and development in information retrieval*, pages 72–79. ACM Press, 2003.

[DELS99] Paul Dourish, W. Keith Edwards, Anthony Lamarca, and Michael Salisbury. Presto: An Experimental Architechture for Fluid Interactive Document Spaces. *ACM Transactions on Computer-Human Interaction*, 6(2):133–161, June 1999.

[DRD+00] Alan Dix, Tom Rodden, Nigel Davies, Jonothan Trevor, Adrian Friday, and Kevin Palfreyman. Exploiting Space and Location as a Design Framework for Interactive Mobile Systems. *ACM Transactions on Computer-Human Interactions*, 7(3):285–321, September 2000.

[EJS02] Maria Ebling, Bonnie John, and M. Satyanarayanan. The Importance of Translucence in Mobile Computing Systems. *ACM Transactions on Computer-Human Interaction*, 9(1):42–67, March 2002.

[FFG96] Scott Fertig, Eric Freeman, and David Gelernter. Lifestreams: An Alternative to the Desktop Metaphor. In *ACM SIGCHI Conference on Human Factors in Computing Systems Conference Companion (CHI '96)*, pages 410–411. ACM Press, 1996.

[FPJ+00] Mike Flynn, David Pendlebury, Chris Jones, Marge Eldridge, and Mik Lamming. The Satchel system architecture: Mobile access to documents and services. *Mobile Networks and Applications*, 5:243–258, 2000.

[Fre97] Eric Freeman. *The Lifestreams Software Architechture.* PhD thesis, Yale University, Department of Computer Science, May 1997. Accessed August 2004, http://www.cs.yale.edu/homes/freeman/dissertation/etf.pdf.

[GNOT92] David Goldberg, David Nichols, Brian Oki, and Douglas Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM*, 35(12), 1992.

[HR02] Lars Hallnas and Johan Resdtrom. From Use to Presence: On the Expressions and Aesthetics of Everyday Computational Things. *ACM Transactions on Computer-Human Interaction*, Volume 9(2):106–124, June 2002.

[IEE04] IEEE 802.11b Standard definition, Last accessed August 2004. http://standards.ieee.org/getieee802/802.11.html.

[JJBC04] Matt Jones, Preeti Jain, George Bucahanan, and Tzu-Chiang Cheng. Changing the Pace of Search: Supporting "Background" Information Seeking. To be published, 2004.

[JMMNB99] M. Jones, G. Marsden, N. Mohd-Nasir, and G. Buchanan. A site based outliner for small screen Web access. *Poster Proc. of 8th World Wide Web conference*, pages 156–157, 1999.

[LEF+00] Mik Lamming, Marge Eldridge, Mike Flynn, Chris Jones, and David Pendlebury. Satchel: Providing Access to Any Document, Any Time, Anywhere. *ACM Transactions on Computer-Human Interaction*, 7(3):322–352, Sept 2000.

[LK93] James Landay and Todd Kaufmann. User Interface Issues in Mobile Computing. In *Fourth Workshop on Workstation Operating Systems*, October 1993.

[Mal83] Thomas Malone. How Do People Organize Their Desks? Implications for the Design of Office Information Systems. *ACM Transactions on Office Information Systems*, 1(1):99–112, January 1983.

[MC03] Gary Marsden and David Cairns. Improving the Usability of the Hierarchial File System. In *Proceedings of SACSIT*, pages 122–129, 2003.

[MNS00] V. Menkov, D.J. Neu, and Q. Shi. AntWorld: A Collaborative Web Search Tool. In *Third International Workshop, DCW 2000.* Springer-Verlag, 2000.

[Nie04a] Jakob Nielsen. "Ten Usability Heuristics", 2004, Last accessed September 2004. http://www.useit.com/papers/heuristic/.

[Nie04b] Jakob Nielsen. "Risks of Quantitative Studies", March 2004, Last accessed October 2004. http://www.useit.com/alertbox/20040301.html.

[PB94] Evaggelia Pitoura and Bharat Bhargava. Building Information Systems for Mobile Environments. In *Third International Conference on Information and Knowledge Management*, pages 371–378. ACM Press, 1994.

[POW04] Stanford Power Browser Project, Last accessed August 2004. http://www-diglib.stanford.edu/~testbed/doc2/PowerBrowsing.

[PSB03] Josep Pujol, Ramon Sanguesa, and Juanjo Bermudez. Porqpine: A Distributed and Collaborative Search Engine. In *12th International World Wide Web Conference*, 2003.

[Ras00] Jef Raskin. *The Humane Interface.* Addison-Wesley Professional, 2000.

[Rhe04] Howard Rheingold. "Political Texting: SMS and Elections". *theFeature.com*, April 2004. Last accessed January 2005, http://www.thefeature.com/article?articleid=100479.

[RSK04] Pamela Ravasio, Sissel Guttormsen Schar, and Helmut Krueger. In Pursuit of Desktop Evolution: User Problems and Practices With Modern Desktop Systems. *ACM Transactions on Computer-Human Interaction*, 11(2):156–180, June 2004.

[SKS87] M. Satyanarayanan, James Kistler, and Ellen Siegel. Coda: A Resilient Distributed File System. In *IEEE Workshop on Workstation Operating Systems*, 1987.

[Thi90] Harold Thimbleby. *User Interface Design*, chapter 15, pages 345–361. Addison-Wesley and ACM Press, 1990.

[TN96] Michael Twidale and David Nichols. Collaborative browsing and visualisation of the search processs. *Aslib Proceedings*, 48(7–8):177–182, 1996.

[TN98] Michael Twidale and David Nichols. Designing Interfaces to Support Collaboration in Information Retrieval. *Interacting With Computers*, 10(2):177–193, 1998.

[TNP97] Michael Twidale, David Nichols, and Chris Paice. Browsing is a Collaborative Process. *Information Processing & Management*, 33(6):761–783, 1997.

[TNST95] Michael Twidale, David Nichols, Gareth Smith, and Jonthan Trevor. Supporting Collaborative Learning during Information Searching. In *Computer Support for Collaborative Learning (CSCL 95)*, pages 367–374, 1995.

[WB96] Mark Weiser and John Seely Brown. *The Coming Age of Calm Technology*, 1996. Last accessed August 2004. Available at http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm.

[Wei91] Mark Weiser. The Computer for the 21st Century. *Scientific American*, pages 933–940, 1991.