RHODES UNIVERSITY
Where leaders learn

AN INVESTIGATION INTO THE USE OF A MOBILE PHONE PLATFORM AS A CONVERGENT TECHNOLOGY FOR TEXT BASED COMMUNICATION.

A Thesis submitted in partial fulfilment
of the requirements of the degree of

BACHELOR OF SCIENCE (HONOURS)
of Rhodes University

Curtis Lee Sahd

November 2008

Acknowledgments

I would like to thank my supervisor Dr Hannah Thinyane for guiding me through my honours project with positive criticism and encouragement. My honours project has taught me a lot in the areas of time management and perseverance. Many an hour was spent making sure that my project was bug free, and I would like to thank my family for constantly reminding me that there is no substitute for hard work. I would also like to thank my friends, Ray Musvibe, Takayedzwa Gavaza, Shange Ndakunda, Terence Foxcroft and Misty Appel for their advice with regard to user interface issues, and program functioning. Last but not least I acknowledge the financial and technical support of this project of Telkom SA, Comverse SA, OpenVoice, Stortech, Tellabs, Amatole, Mars Technologies, Bright Ideas Projects 39 and THRIP through the Telkom Centre of Excellence at Rhodes University."

Abstract

Mobile devices such as cell phones and laptops are increasingly becoming part of todays lifestyle, and so too are the applications that go along with these devices. Apart from the normal voice services provided on mobile phones, text-based instant messaging applications have greatly enhanced our communication capabilities with others while on the move. Global mobile phone usage has grown to 50% and is steadily rising with the current number of subscribers topping 3.3 billion. With the population of the world being 6,634,294,193 and the number of mobile phone subscribers being 3.3 billion, this is by no means an indication that half the people in the world have a mobile phone. Some 59 countries have a mobile penetration of over 100%, where some owners have more than one mobile phone [Virki, 2007]. This project aims to investigate inexpensive means of communication through the use of instant messaging, text messaging over Hyper Text Transfer Protocol (HTTP), and mobile email. All applications have been designed in such a way so as to make best use of the intermittent data connections on cell phone networks in South Africa. Mobile users have access to much fewer services and computing power than their wire-connected counterparts. Since the convenience of mobility is fast becoming a reality, it is important to pay considerable attention to the design and ease of use of software designed for such devices. [Landis and Vasudevan, 2002]. With this in mind, all applications developed in this project have been designed in such a way that they make use of minimal computing power, minimal disk space, and the least amount of money per data packet sent.

Contents

List of Figures

List of Tables

List of appendices

Chapter 1. Introduction

1.1 Background

When cell phones initially came on to the market, speech quality was a major deciding factor for consumers when choosing a cell phone. Since then there have been rapid advancements in cell phone technology, the differences in speech quality on various handsets is now less noticeable [Mattila, 2003]. As well as this growth in usability of cell phones, the services offered on mobile phones have also shifted to include data services [Bedlow, 2008]. Text messages are the most widely used data service, with a study conducted in 2006 indicating that 1.8 billion users generated $80 billion of revenue. Many cell phones also offer instant messaging services, and in countries like Japan, China and India, text messages can be sent via e-mail [answers, 2007]. These convergent technologies blend multiple streams of information into a single presentation on a single device, are central to the future growth of the information technology (IT) industry worldwide [Jussawalla, 1999]. To date, convergent technologies have generated an increasing demand in the region for the broadband spectrum and applications for its use [Jussawalla, 1999]. According to a survey conducted by J.D Power and Associates the average cellular customer uses their cell phone for 6 hours every month. Customers receive an average of 7 text messages per month and 60 percent of cellular calls are made outdoors, of which 62 percent of these outdoor calls take place in cars [CNN, 2003].

With the aforementioned points, it can clearly be seen that cell phones are being used as more than mere devices for placing calls. People spend a lot of time in vehicles and this makes the mobility of the cell phone and hence the vast array of communication tools that come with the cell phone vital for a fast paced lifestyle. Naturally, people want to minimize expenditure, and this is where text messaging, instant messaging and e-mail truly show their benefits.

1.2 Problem Statement

The primary aim of this project is to investigate the use of a mobile phone platform as a convergent technology for text based communication. A prototype will be developed and user tested to determine the viability of the platform as compared to already existing separate text based communication applications.

1.3 Research Outcomes

In completing this investigation, this research will produce the following deliverables:

- Mobile phone based prototype applications collectively known as ThEm that implement three text based communication technologies, namely: Text messaging (SendEm), Instant messaging (JabberEm) and E-mail (MailEm)
- User tests of application user interface efficiency
- Tests comparing message sending and receival times between JabberEm clients and Mxit clients
- A comparison between the actual user timings obtained and the times obtained from the Keystoke Level Model For Advanced Mobile Phone Interaction.

1.4 Structure of the Document

Chapter 2 outlines the the Java Mobile (J2ME) platform and its use for development on mobile devices as well as Java cards. Chapter 2 also describes persistent storage in J2ME as well as the publication and packaging of applications into JAR files and the data security of information contained within these JAR files. An outline of instant messaging, short message service and electronic mail for mobile devices and a brief introduction of each is also found in Chapter 2. The importance of usability principles for mobile devices is outlined, and the various components of mobile device user interfaces and error prevention techniques for these interfaces is introduced in Chapter 2. Chapter 2 also describes the Keystroke Level Model (KLM), and its components for mobile device user interface evaluation.

Chapter 3 gives an overview of how each application in ThEm was implemented, and what tools were used throughout the implementation. The basic design and problems overcome for JabberEm (Jabber IM Client), SendEm (Text messaging via HTTP client) and MailEm (E-mail client) are discussed. A description of each application accompanied by excerpts of pseudo code can be found throughout this chapter. The various error prevention and user validation techniques are also described throughout Chapter 3.

This is followed by a detailed discussion of user interface evaluation methodologies and results, as well as timings for sending and receiving messages between JabberEm clients and Mxit clients in Chapter 4. The structure of the experiments and the findings of each experiment are discussed and analysed in Chapter 4.

This document is concluded with a discussion on how Java Mobile has simplified application development for mobile devices. Chapter 5 also outlines the networking capabilities and user interface design considerations for application development using the Java Mobile platform.

Chapter 2. Related Work

This chapter introduces the Java Mobile platform and its suitability to application development on mobile devices. The Java Card technology is also outlined and compared to the ease of application development in J2ME. Persistent storage, publication and packaging and network data and security are introduced which will be further elaborated upon in Chapter Three. This introduction to the various components of mobile device application development is followed by Instant messaging, text messaging and e-mail application considerations and the usability design principles involved in the development of mobile device applications.

2.1 An overview of Java 2 Mobile

Java 2 Mobile provides a flexbile, robust environment for applications running on cell phones, personal digital assistants and printers. Java 2 Mobile includes flexible user interfaces, security, built in network protocols, and support for networked and offline applications that can be downloaded dynamically [Sun, 2008].

There are two types of J2ME configurations: the Connected Limited Device Configuration (CLDC), and the Connected Device Configuration (CDC) [CLDC, 2008]. CLDC is the smaller of the the configurations, and the Mobile Information Device Profile (MIDP) is built on top of this configuration. CLDC was designed for devices with limited system resources such as cell phones, personal digital assistants and two-way pagers. These devices usually have a minimum of 128 KB to 512 KB of memory. MIDP offers the core functionality to mobile devices such as the user interface, networking, storage, and application management [Longoria, 2004].

2.2 J2ME Application Considerations

As mentioned in the previous section, J2ME has two main configurations, namely: CLDC and

CDC. The characteristics of CLDC devices can be summarized as follows:

- Memory of between 160 and 512 kb
- A 16-bit or 32-bit processor with a clock speed of 16 MHz or higher
- Low power supply and consumption
- Little or no user interface
- Low bandwidth usage and intermittent network connection

Most CLDC devices consist of average cell phones, low end PDA's, and Radio Frequency Identification RFID devices. It can be seen that CLDC devices are constrained in terms of power, processing capability, memory capacity and network bandwidth and connections, and thus the standard Java Virtual Machine (JVM) cannot be used. Instead a cut down JVM is used, known as the Kilobyte Virtual Machine (KVM). These KVM's run on a number of platforms, such as Symbian OS, Palm OS, Windows Mobile, and Embedded Linux. [Zheng and Ni, 2006]

The characteristics of CDC devices can be summarized as follows:

- Larger memory of 2 MB Random Access Memory (RAM) and 2.5 MB ROM available
- Longer battery life
- Reliable network connectivity
- A number of user interface designs
- A 32 bit processor

Devices that make use of CDC include high-end PDA's, and cell phones. Clearly it can be seen that CDC is considerably more powerful than CLDC. Unlike CLDC, CDC makes use of the standard JVM. Since certain mobile devices are more powerful than other, profiles exist and enable a set of standard Java API's to be targeted at a narrow class of devices with the J2ME configuration.

There are three profiles that exist for CDC, namely: Foundation Profile; personal basis profile; and personal profile, where the foundation profile acts as a core for the other two profiles. There are two profiles available for CLDC, namely: Mobile Information Device Profile (MIDP); and PDA profile, both of which are depicted in Figure 1 below.

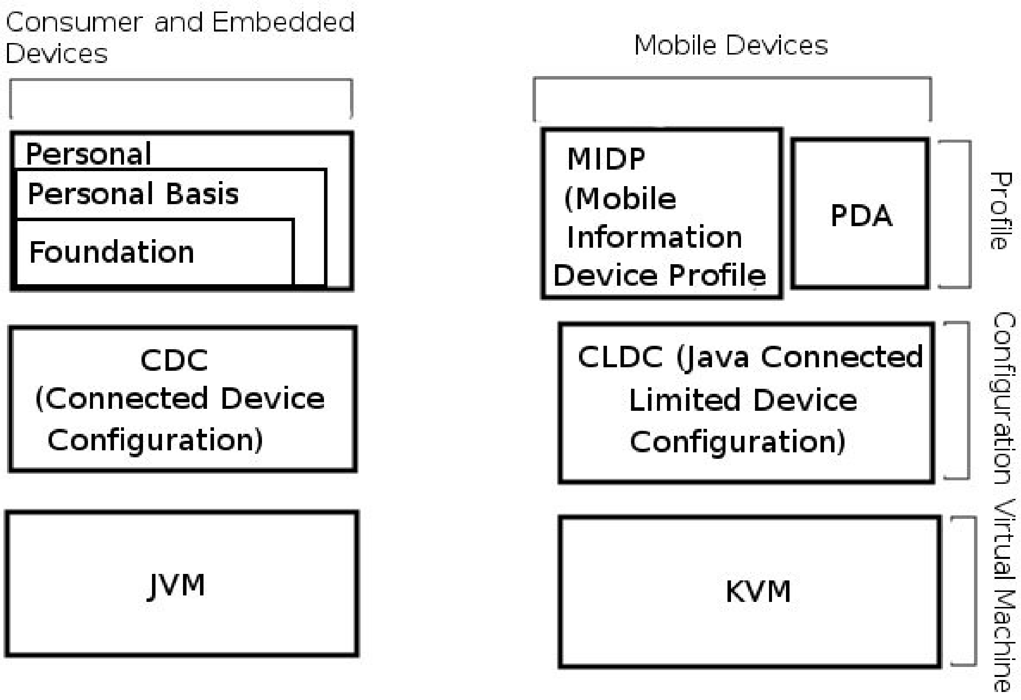

Consumer and Embedded Devices
Mobile Devices
Personal
Personal Basis
Foundation
MIDP (Mobile Information Device Profile
PDA
Profile
CDC (Connected Device Configuration)
CLDC (Java Connected Limited Device Configuration)
Configuration
JVM
KVM
Virtual Machine

Figure 1: J2ME Configuration
[Zheng and Ni, 2006]

With the amazing evolution of mobile wireless technologies, there has been significant development in the area of mobile applications, especially in countries where mobile phone and Personal Digital Assistants (PDA) penetration is high. Service providers are also encouraging developers to develop applications that provide value added service and make full use of 3G networks and emerging technologies. Traditional mobile applciations such as voice and data services are simply not enough anymore, since consumers are looking for mobile devices and applications that provide rich content entertainment, ubiquitous information access and agile business operations [Zheng and Ni, 2006].

When designing applications for mobile devices, the following factors need to be considered and understood in order to develop a successful application: Bandwidth limitations; intermittent connection; limited memory; limited CPU; and limited user interface. Mobile devices have much less bandwidth available for the exchanging of information over the Internet. Applications therefore need to be designed to minimize the use of information sent and received and hence the minimize the bandwidth utilized. Internet connections on cell phones are unreliable and intermittent, and there are often areas of varying signal strength which compounds the unreliability of the connection. Mobile devices typically have limited memory for the storage of applications and limited CPU's which decrease the amount of processing power available on the cell phone. Limited battery life also contributes to the limited processing capabilities of cell phones. Since the screen size of cell phones is so small, and a pointing device is non-existent, the user interface is limited in its freedom navigability and in terms of the amount of information conveyed to the user [B'Far **et al.**, 2008].

The following subsections describe mobile development using Java Card technology. Persistent storage, publication and packaging, and network and data security for mobile application development is described. Instant messaging, text messaging and e-mail for mobile devices is introduced and usability of mobile user interfaces is discussed. The evaluation of mobile user interfaces using the Keystroke Level Model is also covered in the following subsections.

2.2.1 Java Card mobile development

Java Card technology enables smart cards and various other devices with limited memory to run small Java applets. This allows manufacturers to develop applications on a secure interoperable platform which is compatible with existing smart card standards.

Java Card accelerates development since developers are able to build, test and deploy applications and services much more rapidly than before. Java Card is complementary to Java Standard Edition (J2SE), Java Enterprise Edition (J2EE) and Java Mobile Edition (J2ME).

There are a number of industries that benefit from Java Card technology, including the telecommunications industry which benefits from the deployment of Java Card technology on SIM cards. Commerical or financial sectors also benefit from this technology as can be seen in the use of bank cards and various other financial cards which often have Java Card technology embedded in them, which allows for both online and offline transactions. The security industry benefits greatly, since only people with valid security access cards can gain access to certain parts of the premises. In the same way that the security industry benefits from Java Card, so too does the transportation sector, since Java Card technology can be embedded in smart tickets which then allow passengers to make use of certain transportation facilities [Sun, 2002].

Java Card has a number of benefits, namely: Interoperability; security; multi-application capable; compatibility with existing standards.The main benefit from Java Card technology is that it is interoperable which means that any application/applet developed in Java Card will run on any other Java Card enabled smart card regardless of the vendor or underlying hardware. In the same way that J2ME has built in security which relies on the Java language, so too is Java Card a reliable and secure platform for application development. The compatibility of Java Card with existing standards such as the International Smart Cards standard (ISO7816) greatly increases the versatility and popularity of Java Card for application development [Sun, 2002].

Java Card is composed of three main components, namely: The Java Card Virtual Machine Specification; The Java Card Runtime Environment Specification; and the API for the Java Card Platform. The Java Card Virtual Machine Specification defines the features and services, as well as the behaviour that Java Card should support. This specification should also include the instruction set of the Java Card Virtual Machine (VM), the supported subset of the Java language, and the file formats needed for installation of the applets on smart cards. The Java Card Runtime Environment Specification defines the behaviour of the Java Runtime Environment (RE), and it must include implementations of the VM and the Java Card API classes. The API for the Java Card Platform contains the class definitions required to support the Java Card VM and the Java Card RE. This API is also compatible with international standards as well as industry-specific standards [Sun, 2002].

Now that the benefits of Java Card technology as well as the components of Java Card have been explained, the advantages of Java Card over Java Mobile (J2ME) can be mentioned. Java Card can be installed and executed on low level cell phones which don't have support for J2ME. There are still a large number of cell phone users who have non-J2ME compatible cell phones in which case the interoperability of Java Card is best demonstrated. Not only do Java Card based applications target a much wider array of mobile devices, but the development of such applications under the Java Card platform is considerably less complex than developing the applications using the J2ME platform. Many mobile devices are still constrained in terms of memory and processing power, and Java Card applets are ideal for such devices since they target CLDC devices and other devices with a much lower memory footprint [Sun, 2002].

2.2.2 Persistent Storage

There are three types of storage for mobile devices, namely:

- Record Management System (RMS): which is a tiny database on the phone [Giguere, 2004]
- File Connection API which provides access to the mobile device's file system [Giguere, 2003]
- Personal Information Management (PIM): allows manipulation of the device's contact list, calendar and to-do list [Mahmoud, 2005]

All user data such as email addresses, nicknames and passwords are stored to the persistent storage. This storage is know as RMS , and is in fact a minute database for mobile devices [Knudsen, 2007]. Problems with record stores arise when records are removed. Unlike a vector, when you remove a record, all records after the remaining record do not “shift up” to take the place of the removed record. So the way that this was dealt with is: Whenever a record is removed from the record store, then you also removed the record from the vector. The record store is then closed and removed. Of course, as was mentioned above, a vector shifts elements after the deleted element up in the vector, so we have the desired representation of the data that is to be stored in the record store. We then open/create the record store, and iterate through all our elements in a vector, and at the same time, we add those elements to the record store.

The advantage of using a record store is that it is small, and it is very secure. The reason for its security is that midlets cannot access and open record stores created and contained in other midlet suites [Topley, 2002]. Since mobile devices do not have the same computing power and space as a personal computer, space, performance and security are vitally important to developing a successful mobile application.

2.2.3 Publication and Packaging

MIDlet's consists of compressed Java Archive (JAR) files and Java Application Descriptor (JAD) files. The JAR file contains all Java class files, pictures and application specific files. The JAD file describes what is contained inside the JAR file, such as the size, author and version of the JAR file. In order to create the JAD and JAR files, the classes first need to be compiled, then preverified, and then the creation of the JAR and JAD files takes places respectively.

Preverification is the process of checking class files to ensure that no illegal operations are performed [Nylund, 2002].

2.2.4 Network and Data Security

The total number of Java mobile devices on the market currently exceeds 1.2 billion units worldwide [Debbabi et al., 2006]. With the large number of applications available for Java-enabled devices, security is vitally important, especially in the case where sensitive data is being transmitted over the Internet. Through a study conducted by Juan and Long, they evaluate the advantages of Java Mobile applications over their WAP and native application counterparts. They believe that the content of J2ME applications rather than the connection should be secured. There are a number of security issues on devices with the J2ME platform, namely: Low-level securityl application-level security; and end-to-end security. Low-level

security ensures that class files in the virtual machine do not perform illegal operations, whereas application-level security ensures that J2ME applications only access the libraries and resources which the J2ME application and mobile device permit them to access. End-to-end security ensures the protection of information sent between clients and servers.
On the J2ME platform, MIDP addresses application and end-to-end security, and CLDC address low-level and application security [Debbabi et al., 2006]. The simplest way to ensure the security of data in J2ME networking is to make use of the HTTPS or SSL protocols. Unfortunately, a limited number of devices support direct socket connections such as SSL, and thus the majority of devices use the HTTPS protocol.

Data persistence through RMS ensures that data stored by an application is only accessible from that application. Further data security measures can be implemented, such as encrypting the data before storing it to the record store, and then decrypting the data when it is retrieved from the record store. This can be cumbersome since record store operations have limited performance to begin with [Giguere, 2008].

2.3 Instant Messaging in J2ME

Instant messaging is defined as real-time interaction between two parties or or among a group of people via a computer network such as the Internet, enterprise network, campus network or even a small Local Area Network (LAN). According to Zheng and Ni [2006] an instant messaging system must provide two basic services, namely: Presence service; and enhanced messaging service.

Presence service is further divided into presence query, presence notification, and presence update, all of which allow users to be aware of other users (e.g online, offline, away, busy). A presence query occurs when a user queries a directory server for the status' of other users, whereas a presence notification occurs when contact A changes their status and the directory server notifies all contacts on contact A's contact list of his/her status change. Presence notification occurs when a contact's status changes and they inform the directory server of this change. Enhanced instant messaging service offers a variety of communication methods such as text chat, voice chat, multiplayer gaming and application sharing.

Most instant messaging protocols make use of a directory server which then keeps track of all status updates and notifications. Directory servers manage presence services and are also usually in charge of user authentication. In order to provide enhanced messaging services some instant messaging systems often also use directory servers as centralized servers which act as gateways which forward the data back and forth between contacts [Zheng and Ni, 2006].

Presence in communication is becoming a key issue in the wireless and wired world. At the elementary level, presence informs users when contacts are online or offline. Elementary presences are expanded from online/offline to include rich presences, such as moods, invisible presence. Presence determines who the user is, where the user is, when and for how long the user has had a presence, how they obtained that presence, and why they have a certain presence [Vogiazou, 2002].

Most instant messaging clients such as MSN Messenger and Yahoo! Messenger have the following basic features: contact list management; contact to contact communication; and presence management. Contact list management allows users to view other contact's presence and chat to their contacts. Contact to contact communication if simply the sending and receiving of messages between contacts and this can sometimes be done even if the contact is offline. Contact to contact communication often relies on presence management which provides and indication of whether contacts are available, busy, away, or offline and some instant messengers provide the option for custom statuses [Vogiazou, 2002].

The integration of presence and IM in the wireless domain will affect communications patterns and behaviour even more so than what desktop IM has already done. Desktop IM has had a significant impact on the way people interact socially, as well as professionally. Desktop IM has allowed professionals to exchange meeting arrangements, coordinate work, and has allowed for the inexpensive exchange of information between staff and colleagues. Mobile instant messaging provides more flexibility than desktop IM by allowing people to constantly be “in touch” with their contacts. This then allows people to exchange information while on the move, and this is convenient for both social purposes and the organization and coordination of business arrangements [Vogiazou, 2002].

The main problem with desktop instant messaging and hence mobile instant messaging is that there are many different instant messaging protocols, some of which are open protocols such as the Jabber protocol, and others which are closed protocols such as MSN. With such a vast number of protocols and instant messaging clients, it is difficult to provide interoperability between these protocols and clients to standardize instant messaging. If one person is using the MSN protocol, and another is using the Jabber protocol, these people cannot communicate with each other. Some instant messaging clients allow users to login using different protocols simultaneously. This provides a level of abstraction for the user, since the client determines which protocol the contact you are conversing with is using, and then simply sends messages to that contact using the matching protocol that you are currently logged with. As far as the user is concerned, the instant messaging client is sending messages to contacts, but the matching of protocols is taking place in the background. This obviously means that users will have to have multiple instant messaging accounts in order to communicate with contacts utilizing different protocols.

Table 1: Comparison of the major instant messaging protocols

Protocol	**License**	**Asynchronous Message Relaying**	**Transport Layer Security (TLS)**	**Unlimited Number of contacts**	**SPAM protection**
Jabber/XMPP	Open	Yes	Yes	Yes	Yes
YMSG (Yahoo! Messenger)	Proprietary	Yes	No	No	Yes
IRC	Open	Yes	Yes	No	Yes
MSN	Proprietary	Yes	No	No	No
OSCAR (AIM, ICQ)	Proprietary	Yes	Yes	No	Client based
Skype	Proprietary	No	Proprietary	No	No

[Vogiazou, 2002]

The main advantage of the Jabber protocol is that it is an open protocol and that it has support for an unlimited number of contacts. With 12.4 billion spam e-mails sent everyday, the Jabber protocol clearly provides a valuable service in the protection against spam [Evett, 2006]. TLS support protects users from man-in-the-middle attacks, such as the interception of passwords and instant messages. It can clearly be seen that in terms of the services listed in Table 1, the Jabber protocol proves to have to most functionality when compared to the other protocols. Internet Relay Chat (IRC) is a close second to the Jabber protocol with the only disadvantage being that the number of contacts that a user may have is limited.

The Jabber protocol is based on the Extensible Markup Language (XML), and is an open, secure protocol with standards published by the IETF and XMPP Standards Foundation [Jabber, 2008]. Other instant messaging protocols such as MSN, YAHOO exist, but these are closed protocols, meaning that information pertaining to the protocol was not intended for the public. Chat clients currently making use of the Jabber Protocol include Google Talk, Mxit and 2Go.

The main advantage of Jabber is that anyone can start their own Jabber Server, and thus there are a vast number of developers currently focusing on the Jabber Protocol [Jabber, 2008]. Further advantages of the Jabber Protocol are that it is secure, and it is free. To connect to the Google Talk server (which communicates using the Jabber Protocol), a Secure Sockets Layer (SSL) connection is made to talk.google.com, thus enabling the protection of data, and in turn respecting the user's privacy. There are many Jabber Application Programming Interfaces (API's) for Java Standard Edition, but APIs for Java 2 Mobile are lacking. From the reasons above, it can clearly be seen that the Jabber Protocol is suited to the purposes of this project.

2.4 Short Message Service (SMS)

Short Message Service (SMS) has noticeable benefits in that messages can be exchanged between devices regardless of whether they are online or not. Even though the SMS is more convenient to use than other protocols, SMS is still expensive when compared to technologies such as Instant Messaging (IM), Mobile Instant Messaging (MIM), and E-mail.

SMS has allowed people to send messages between mobile devices, and has been in development since 1985. In 2006 there were 946 billion SMS's sent worldwide, and there are over 2.41 billion SMS ready devices at this point in time [Schwartz and Retford, 2007]. According to a survey conducted by TNS Technology, MIM is said to overtake SMS (texting), and eventually E-mail. The survey interviewed 17000 users across 30 countries, and concluded that once cell phone users adopt MIM, it overtakes other forms of messaging, to become the primary non-voice method of interacting, which could have devastating consequences for network service providers' revenue. The survey revealed that 61% of the users make use MIM, 55% of the users use SMS, and 12% use e-mail. Currently, 8% of global mobile users use MIM, with the highest number of users in Hong Kong (23%). Clearly this is an indication of a massive opportunity among the country's 500 million subscribers [Cellular-news, 2008].

Mobile messaging applications take on many forms and functions, such as instant messaging chat provided by service providers such as T-Mobile's sidekick. Other instant messaging applications include mobile instant messaging integration with enterprise applications such as customer relationship management (CRM) and enterprise resource planning (ERP), as well as subscription based information services such as weather alerts, news updates and traffic reports. Instant messaging can also be used for message based voting services and interactive mobile messaging services such as Google SMS search which allows users to search Google and get back responses via a text message [Zheng and Ni, 2006].

SMS is supported on a wide range of digital wireless networks such as GSM, CDMA and TDMA. A text message can contain a maximum of 160 alphanumeric characters in Latin alphabets such as English, and a maximum of 70 characters for non-Latin alphabets such as Chinese. Even though the standard length of an SMS message is 160 characters, many service providers have set their own length for text messages, ranging from 100 to 280 characters. SMS essentially turns a cell phone into a pager, which then eliminates the need for users to carry a pager and cell phone [Mallick, 2003].

SMS provides the following benefits to both the user and the service provider: Guaranteed message delivery using a store-and-forward approach; ease of use, without additional software or hardware; low cost method for information delivery; and a revenue source for service providers. The store and forward approach enables users to receive messages at a later time if their cell phones are turned off, or they are in an area with no signal coverage. If messages are stored for unusually long periods of time, then the service providers will remove the message from their systems due to load shedding and garabage collection considerations. Ease of use of text messages allows for the sending and receiving of text messages without any additional hardware or software (such as a WAP browser), since the SMS functionality comes pre-installed on the mobile device. Since SMS is a low cost method for information delivery, many people prefer to use SMS instead of making voice calls due to the considerably lower price involved in delivering an SMS when compared to that of voice calls. Apart from the revenue service providers generate through the delivery of text messages, they also use text messaging to inform subscribers of product updates and specials [Mallick, 2003]. SMS can be used for social, commerical and informational purposes. The most common use for SMS is peer-to-peer communication, which replaces the need for voice calls. SMS can be used for informational purposes such as informing subscribers of stock and weather updates. Commercial uses include advertising and vehicle positioning services. When a text message is sent from a cell phone it is carried over a wireless connection to the service provider's SMSC, and the service provider deals with all the lower level details such as delivering the text message to the appropriate service provider's SMSC and the transfer of the text message from one protocol to another. This, however, is not the case when a text message is sent from a device other than a cell phone, then it is the device's responsibility to communicate with the SMSC in order for the message to be delivered successfully. The problem with communicating with the SMSC is that most service providers don't expose their SMSC API. So to solve this problem, one has to establish a relationship with the service provider and then write to their proprietary interface, or one can use an aggregator that already has established connections with the service provider you are interested in communicating with. There are two main protocols that are used to communicate with SMSC's, namely: Telocator Alphanumeric Protocol (TAP) and Short Message Point to Point (SMPP) [Mallick, 2003].

Many service providers offer an extra interface to sending text messages such as SMTP access, which then enables users to send text messages through their e-mail client [Mallick, 2003]. Communicating directly with the SMSC allows tracking of the status of the message, whereas SMTP does not allow this.

Many experts thought of SMS as a means of alerting mobile users, while others envisaged SMS as a means of telemetry. However, few experts believed that SMS would eventually be used as a means of sending text messages between mobile users. The first SMS text message was sent by Neil Papworth of Airwide Solutions using a personal computer to Richard Jarvis of Vodafone on the 3 December 1992 [Short Message Service, 2008] . The traditional size of an SMS text message is 160 7-bit characters, 140 8-bit characters or 70 16-bit characters. This rule generally applies, except when Arabic or Russian languages have to be encoded, and this will result in a decrease in the number of characters an SMS message is able to contain. When a message exceeds the standard amount of characters of an SMS text message, then a multipart SMS text message will be sent. The header of each message in a multipart SMS message starts with a User Data Header (UDH) containing segmentation information. This then results in a decrease in the number of characters allowed in an SMS text message. The receiving mobile handset is then responsible for using the UDH's of each message to reassemble the messages into one long message displayed on the mobile handset. [Short Message Service, 2008]

The paths followed for sending text messages over GSM can easily be seen from the Figure 2: In this image a text message is sent from cell phone 1 to cell phone 2. Based on the assumption that Cell Phone 1's service provider is Vodacom and Cell Phone 2's service provider is MTN: The message goes from cell phone 1 to Vodacom's Short Message Service Center (SMSC). This message is then forwarded onto MTN's SMSC using the SS7 protocol. The message is then stored at MTN's SMSC until cell phone 2 is able to receive the message.

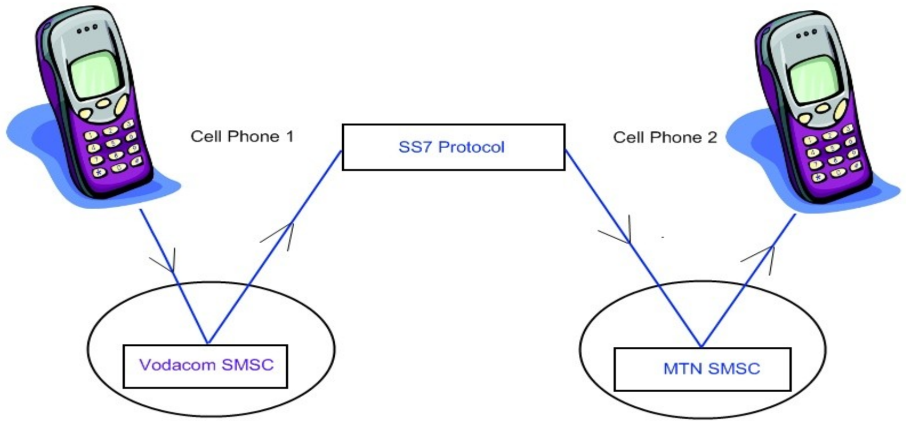


Figure 2: Short Message Service Architecture

2.5 E-mail for J2ME

Electronic Mail (e-mail) has been the most widely used Internet application since its inception in the late 1970's. When e-mail was first introduced, the number of Internet users was limited, and there was no need for a secure infrastructure. Since there are many more Internet users today than there were in the late 1970's, security of e-mail has become a vitally important aspect in maintaining the privacy of Internet users. Secure e-mail standards like S/MIME [Freed and Borenstein, 1996], PEM [Linn, 1993] and PGP [Elkins, 1996] were designed to work by making use of cryptographic algorithms at both the source and the destination. These secure e-mail standards have been used for decades, but suprisingly the majority of e-mail nowadays is still delivered using insecure transportation protocols [Narayandas **et al.**, 2006].

In most cases e-mail clients and servers are on an intranet protected by a firewall which prevents e-mail from being accessed from outside the organization [Van Thanh et al., 2005]. This prevents spamming and unwanted relaying activity. Since many users on are the move, there is an increasing demand for mobile e-mail. This then leads to a compromise between security and availability. There are a number of challenges related to mobile e-mail, such as excessive bandwidth usage due to message exchange; size of e-mail attachments; unsupported presentation formats; complex e-mail client implementation due to various e-mail protocols needed for the sending and receiving of e-mails; and firewall configurations which limit e-mail access. If multiple e-mails are constantly being exchanged, this could contribute to excessive bandwidth usage, but since the average size of a text only e-mail is 18.5 kb, this is not really a large concern in terms of excessive bandwidth usage [Berkeley, 2000]. E-mail attachments can drastically increase the amount of bandwidth used, especially when their size is in excess of 500kb. Since the it costs roughly R1 per 1MB of data transferred on the Vodacom network, one can see that downloading e-mails with multiple attachments could become a costly exercise [Vodacom, 2008]. Most of the lower end cell phones do not provide support for certain file formats which were intended for personal computers. So e-mail on cell phones is generally limited to the exchange of plain text data. Because there are a vast array of protocols available, the developmental complexity as well as the application size of e-mail clients is increased considerably. Last but not least, e-mail clients on cell phones are difficult to integrate with existing firewall implementations and rules, since cell phones are considered external to the organization and the configuration of these firewall rules becomes cumbersome [Van Thanh **et al.**, 2005].

The primary concern with networking in J2ME, and thus the sending and retrieval of e-mail, is bandwidth. From Figure 3 it can be seen that there are two methods of sending and retrieving email, namely: POP3 and IMAP. POP3 is the older of the two protocols, and is more bandwidth intensive than IMAP. IMAP offers several advantages over POP3 such as having better functionality in manipulating the inbox, better functionality in being able to manage mail folders other than the inbox and more efficient bandwidth management than POP3. IMAP enables the user to read their emails directly from the mail server, without the need to download the email before reading it, and allows users to access their emails regardless of location. All emails remain on the mail server. POP3 on the other hand downloads emails to the PC or cell phone, which allows for the offline reading and manipulating of emails [University of Colorado, 2007].

When an email is sent it is completely transparent to the user. As far as the user is concerned, an email is composed, the recipient(s) is/are added and the email is arrives in the recipient(s) mailbox.

As shown in Figure 3, the life cycle of e-mail delivery from Yuan [2004] is as follows:

1. The user's email client contacts the SMTP server with the content of the message, and the recipient information.

2. The SMTP server discovers the recipient's POP3 or IMAP server through an Internet registry, and delivers the e-mail to that server.

3. The recipient's POP3 or IMAP server stores the e-mail until the recipient logs onto the server and obtains their email.

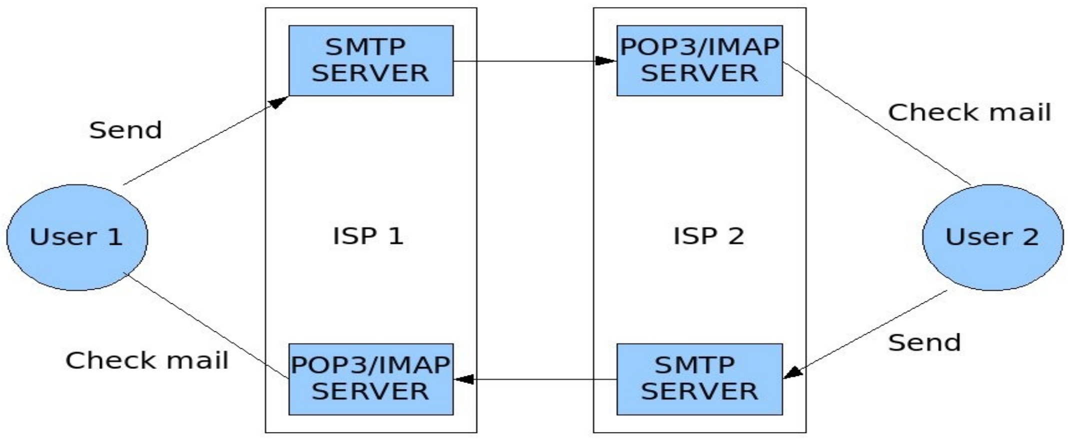


Figure 3: Processes involved in delivering an e-mail

2.6 Usability

Nielsen [1994] describes usability as being a multidimensional property of a product, consisting of the following five aggregate components:

- Learnability – How easy is it to accomplish tasks for the first time?

- Efficiency – How efficiently can the user perform a task once the design has been mastered?
- Memorability – After not using the product for a certain period of time, how easily can proficiency be re-established?
- Error recovery – How many errors does the user make, how severe are the errors, and how easy is it to recover from those errors?
- Satisfaction – Is it pleasant for the user to use the product?

With these five components of usability, applications and interfaces can be evaluated in a consistent manner and improved with the results of the usability study. Software quality evaluation and hence an assessment of usability can be performed by making use of the following practical process flow: Establish evaluation requirements; specify the evaluation; design the evaluation; and execute the evaluation

The above process flow can be diagramatically represented as follows:

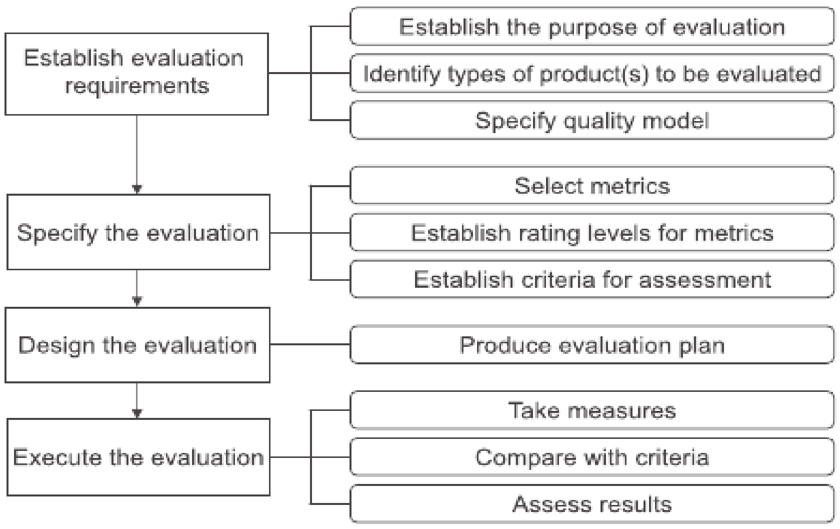

Establish evaluation requirements
Establish the purpose of evaluation
Identify types of product(s) to be evaluated
Specify quality model
Specify the evaluation
Select metrics
Establish rating levels for metrics
Establish criteria for assessment
Design the evaluation
Produce evaluation plan
Execute the evaluation
Take measures
Compare with criteria
Assess results

Figure 4: Software quality evaluation process of ISO/IEC 14598-1

The user interface for mobile devices is very different to that of desktop computers, since the screen size is smaller, and mobile devices do not usually have a pointing device, such as a mouse. Java 2 Standard Edition (J2SE) makes use of the Abstract Window Toolkit (AWT) which cannot be used on mobile devices because AWT is designed and optimized for personal computers with a poiting device, and AWT has a rich feature set, which is often not needed on mobile devices (such as screen resizing and layout managers). [Mahmoud, 2000]

The development and quality assurance process found in B'Far **et al.** [2008] recommends the following steps for developing applications in J2ME:

1. Select the primary set of devices that the application intends to support. Cell phones vary considerably in the configurations they support, i.e MIDP 1.0 or MIDP 2.0.
2. Specify which manufacturers your application is intended for.
3. Obtain emulators for the selected manufacturers.

2.6.1 Cards and Decks

Since the screen size for cell phones is limited, it is vitally important to convey as much information as possible in the limited screen space available, without overloading the user with information [Hays, 2008].

Cards can be compared to screens or forms on mobile devices. If the card is larger than the available screen size then most cell phones provide the user with a scroll bar in order to scroll through the remaining data on the screen. It is easier for users to move from card to card (i.e from screen to screen) than to use the scroll bar [Hays, 2008].

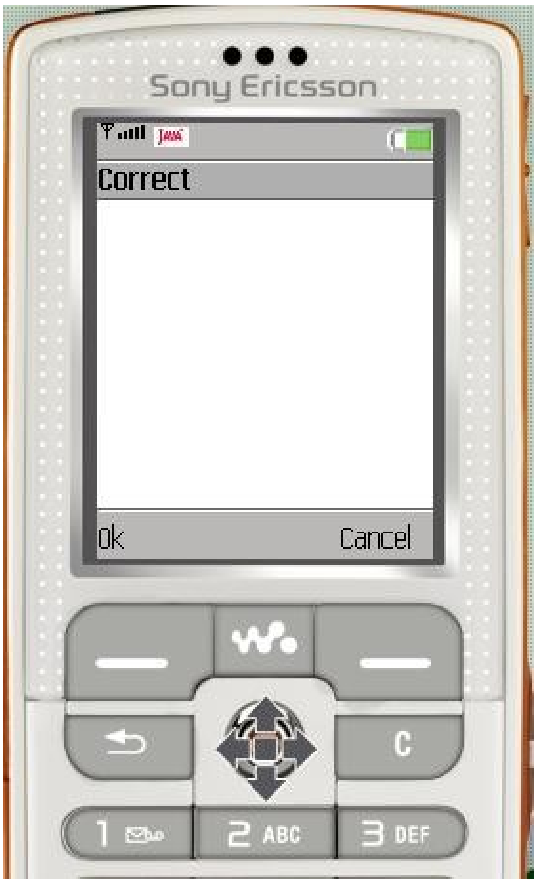
Since some cell phones don't have scroll bars, rows should never be left empty, instead a horizontal rule should be placed on the screen to indicate that there is more data to follow, or else users will not know that they are able to scroll down. The guideline is to have two soft buttons on a cell phone screen as shown in Figure 6, which then prevents the user from having to scroll though a long list of possible commands [Hays, 2008].

Decks contain cards, and allow the user to efficiently navigate through all the cards. An example of this would be to have a “Settings” deck (List in J2ME) and then put various options on the deck such as “Add Contact”, “Remove Contact”, thus simplifying and categorizing the menu options available to users [Hays, 2008].

2.6.2 Error prevention and interface considerations

Error checking is equally as important as error prevention, but it would make more sense to prevent errors altogether and save the hassle of informing users of errors which have occurred. One effective method in reducing the amount of errors on mobile devices is by logical layout of the user interface. Buttons which perform similar functions should be grouped together, and buttons with opposing functions should be grouped at opposite ends of the list. This is clearly demonstrated in Figures 5 and 6 below. Sometimes the placing of buttons and layout of a user interface on one cell phone model differs from that of another cell phone model, and this often results in buttons with opposing functions being grouped together. This unpredicatable grouping of buttons is demonstrated in Figure 7 below.

When the user chooses to delete an item, a confirmation card should be presented, thus ensuring that the user didn't press the delete button by mistake. Back or previous buttons should usually be on the right hand side of the screen, and confirmation or next buttons should be on the left hand side of the screen. Care should be taken so as to avoid placing “confirm” and “decline” buttons above or below each other.



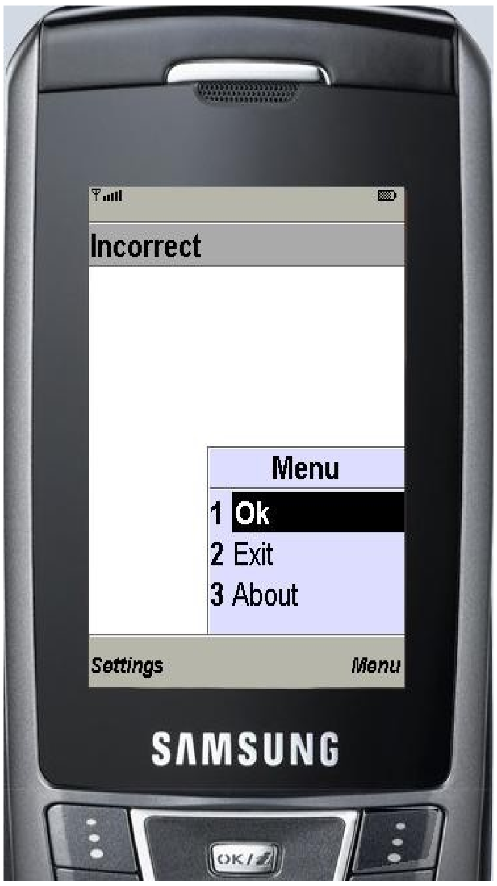


Figure 5: Correct button organization **Figure 6: Incorrect button organization**

Figure 5 is clearly the most logical and safe user interface, since it is natural in western cultures to read from left to right, and hence the "Ok" being the affirming command being positioned on the left, and the "Cancel" being the declining command being positioned on the right. In non-western cultures this interface might not be the most logical, since they read from right to left [Hays, 2008].

Figure 6 on the other hand is completely illogical regardless of cultural differences, since it would be quite easy for the user to press the "Exit" button when they intended to press the "Ok" button. This scenario might not be disastrous in a mobile chat application, but in a banking application this result in potentially catastrophic errors [Hays, 2008]. Apart from the positioning of buttons, device user interface implementations and limitations need to be considered. The source code that produced the buttons in Figure 6 above is identical to the source code that produced the buttons in Figure 7 below, but yet varying results occur on different devices [Hays, 2008].

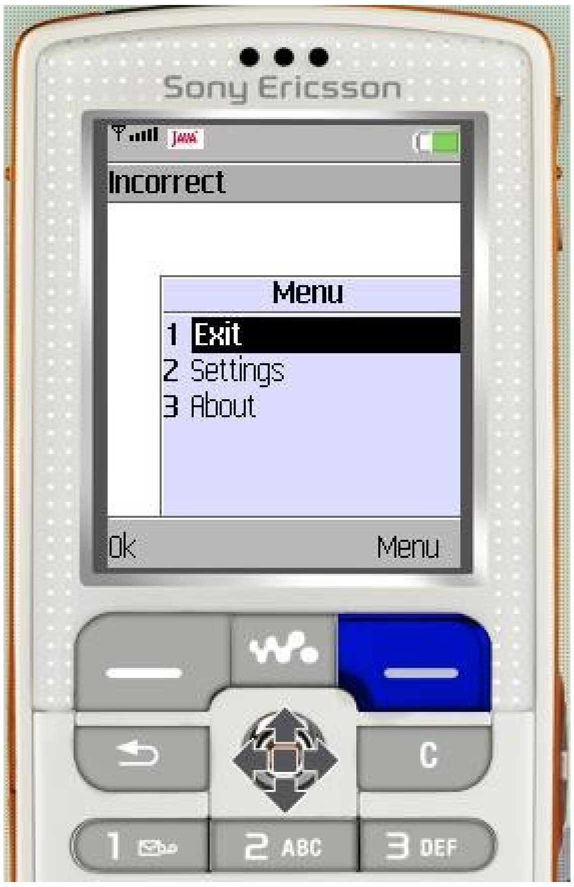


Figure 7: Re-ordering of buttons on based on cell phone model

Apart from the varying results across different devices, the position of the "Exit" button should be noted. Termination buttons should usually be placed as the last item on a list, which doesn't pose too much of a risk for users pressing the wrong button, since the most common commands will be placed at the beginning of the list [Hays, 2008].

In the case of data input informative titles such as dd/mm/yy should be used and format and type fields should also be enforced, which prevent users from entering invalid data. Numeric and alpha fields should also be distinguished which then speeds up data entry [Hays, 2008]. When a user has to enter data in a specific format it greatly aids the user when the format of the input data is conveyed to that user, and similarly, when users only need to input numbers into a text field or letters into a text field, then the input mode of the phone should be changed accordingly to simplify data entry without the user having to manually shift between numeric and text only entry states.

2.7 Assessment Methodologies

An application can be well designed and have a plethora of features, but without a decent user interface and ease of use, most users wouldn't want to use that application. It is therefore vitally important that the user interface is designed is simple and easy to use and that some model is used to evaluate the application. The sections below will outline heuristic and predictive evaluation as well as the Keystroke Level Model (KLM) for mobile devices.

2.7.1 Heuristic vs Predictive evaluation

In order to qualitatively and quantitatively assess user interface designs, a number of methodologies are available, two of which are heuristic evaluation and predictive evaluation.

Heuristic evaluation – is a user interface critiquing process carried out by experts with reference to a shared set of usability guidelines [Rosson, 1998]. Nielsen [1994] describes 10 heuristics for user interface evaluation, namely:

1. Match between the system and the real world
2. Visibility of system status – keep users informed by providing system feedback
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall – users shouldn't have to remember information from other parts of system
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design

9. Error reporting, diagnosis, and recovery
10. Help and documentation

Predictive evaluation on the other hand requires a model for how a user interacts with an interface. For the purposes of this project, predictive evaluation is most suited to the qualitative and quantitative evaluation of user interfaces. GOMS is a typical model used for predictive evaluation, and will be discussed in the next section.

2.7.2 GOMS and the Keystroke-level model

Modelling user tasks and processes has increasingly become an important factor in user interface design. The GOMS (Goals, Operators, Methods, Selection rules) model allows for the modelling of user behaviour during given tasks and also analyses use complexity for interactive systems. The KLM is a customized instance of GOMS and allows tasks to be described using operators that model component tasks such as key presses, mouse movements, decision timing and system response times.

Since the KLM is aimed at modelling user behaviour on desktop computers, it cannot be directly applied the modelling of user behaviour on mobile devices. A variation of the KLM tailored for mobile devices has been produced by Holleis, Otto, Hußmann and Schmidt [2007] which includes slight modifications of the original KLM.

The original KLM as seen in Card **et al.** [1980] defines 6 operators, namely:

- Keystroke (K) – Key and button presses
- Pointing (P) – Mouse movements
- Drawing (D) – Straight line drawings with mouse
- Homing (H) – Hand movement between keyboard and mouse
- Mental Act (M) – Time required to make decision for steps to follow

Since cell phones don't have a mouse, operators such as P, D, and H are not needed for the modelling of user behaviour. Some new operators for the modified KLM are needed, namely: Macro attention shifts ($S_{macro.}$); micro attention shifts (S_{Micro}); and finger movements (F). Macro attention shifts take place when users use their cell phones in public places where there are constant interruptions, and time need to be taken between when users focus on the mobile device and the real world. Micro attention shifts model the time needed to look from the display to the keypad and hotkey regions and visa versa. Figure 8 shows the the various regions of a cell phone (Keypad, hotkeys and display). Finger movement models the time needed to move a finger from one key to another and is usually followed by a Keystroke (K). This operator could be compared with the Homing operator of the original KLM [Holleis **et al.**, 2007].

The Keystroke (K) operator and the Mental Act (M) operator from the orginal KLM remain unchanged and are utilized in the keystroke-level model for advanced mobile phone interaction. For purposes of brevity, only the applicable operators that are used for evaluation of the user interfaces in Section 4.3 are included above and in Table 2 below. Figure 8 shows the categorization of the user interface components as display, hotkeys and keypad respectively. These components of the user interface are used in the KLM and allow for quantitative and qualitative analysis of the user interface.

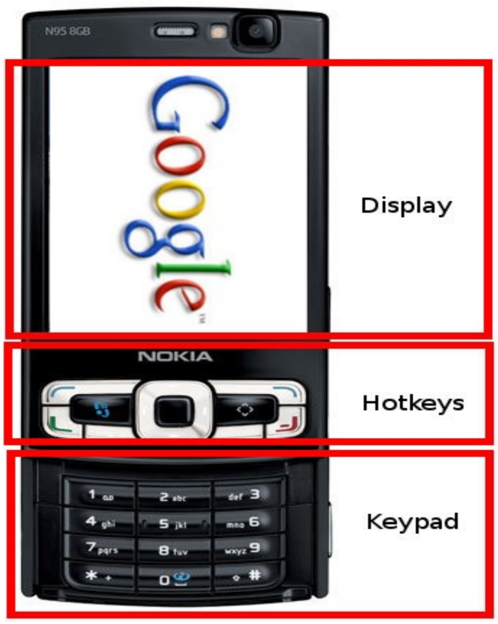


Figure 8: Regions of a cell phone

Table 2 below shows the average time needed to perform the various operators. By listing the operators needed to perform a certain task on the cell phone, applying the KLM rules and adding the times together, the total time to perform that task on the cell phone is calculated and the efficiency of the user interface can be evaluated.

Table 2: Times needed to execute operators

Operator	Letter representation	Time (seconds)
Macro attention shift	S_{Macro}	0.36
Micro attention shift	S_{Micro}	Keypad ↔ Display = 0.14 Hotkey ↔ Display = 0.12 Keypad ↔ Hotkey = 0.04
Finger Movement	F	0.23
Keystroke	K	0.39
Mental Act	M	1.35

2.8 Summary

Java Mobile (J2ME) provides opportunities for application development on lower end mobile devices (CLDC) as well as high end devices (CDC). Since J2ME is a subset of Java, and it is intended for lower powered mobile devices, the set of functions available in J2SE have to be scaled down to accommodate for bandwidth limitations, intermittent network conenctions, limited memory and user interface, as well as underpowered CPU's.

Even though most cell phones being manufactured nowadays are J2ME compatible and are considerably more powerful than during their inception, there are still cell phones in existence with native user interface components such as non-colour screens and sequential menu structures. Java Card technology allows the development of applications for these earlier cell phone models by targeting development at the SIM card level and thus it is non-crucial to have Java Mobile support at the application level. J2ME and Java Card both make use of Kilobyte Virtual Machines (KVM's), and have very similar functionality, the only difference being the level of development, installation and execution.

J2ME allows for user data to be stored in a simple, secure manner by making use of persistent storage such record stores. The deployment and installation of Java Mobile applications is is a straightforward process, since it simply makes use of a JAR and JAD files which prompt the user to confirm the installation. Because of the interoperability and simplicity of J2ME it proves to be a brilliant platform for development on mobile devices.

Text messages have provided a great alternative to making voice calls on mobile phones, but since they are limited to 160 alphanumeric characters, the price of the information conveyed in the text message is still expensive and inconvenient for network subscribers. Instant messaging made its mark through usage on the computer, and since cell phones has improved since its inception, and with the combination of the Java Mobile and Java Card platforms, an opportunity has arisen in the field of mobile instant messaging. The Jabber protocol is especially suited to making use of this opportunity, due to its open standards and security benefits. Since instant messages arent limited to the number of characters which can be transmitted, they prove to be more versatile than SMS, and considerably more cost effective. Instant messages however, can only be exchanged between contacts who are online using a certain instant messaging protocol, which is clearly less desireable than being able to receive text messages without having to be online.

E-mail has been the most widely used Internet application since its inception in the late 1970's. The traditional model of e-mail has always been intended for use on the PC, but this is now changing with the introduction of e-mail capable cell phones. Most cell phones have built in e-mail clients, which prompt users to enter the relevant e-mail server settings, and enable them to send and and receive e-mail's to and from their commercial or personal e-mail accounts. It was mentioned above that not all cell phones are J2ME compatible, and in the same way, not all cell phones are advanced enough to have built in e-mail clients. With the availability of e-mail API's for J2ME, e-mail client development has been greatly simplified. The problem however is not normally with the implementation of e-mail clients on cell phones, it lies with the firewall configuration for external access to company mail boxes. To compound this problem, average cell phone and e-mail users do not know their e-mail server settings.

Since J2ME was intended for low powered mobile devices (when compared with high powered personal computers), the user interface components are very basic and aesthetically plain. There are a number of API's which make use of Cascading Style Sheets (CSS) to improve the native user interface components of J2ME. Even with these basic components, the general rules of usability still apply, such as never placing two buttons of completely opposite functions adjacent to one another. The efficiency of these user interfaces can also be evaluated through the use of the Keystroke-Level Model for Advanced Mobile Phone Interaction. The time taken to perform certain basic functions on a mobile device such as the key presses and finger movments are summed to give and indication of the efficiency of the user interface.

Chapter 3. Design and Implementation

This chapter describes the design consideration and implementation details for ThEm. The first section outlines the overall design for ThEm and provides an introduction to each application. This section is followed by a detailed description of the design and implementation details of each of the respective applications: JabberEm; SendEm; and MailEm.

3.1 Design of ThEm

ThEm was implemented as a proof of concept to show that inexpensive and equivalent communication can take place on mobile devices. To investigate mobile devices as a platform for convergent technologies, it was decided to implement an instant messaging application, text messaging application and e-mail application. Even though the functionality of applications designed for mobile devices is somewhat limited when compared to that of the personal computer, the basic tools which are necessary for various implementations are provided in J2ME.

There are obviously tradeoffs when using one form of communication as opposed to another. This proves to be the case whether sending a text message, an instant message, or an e-mail. Text messages allow for the direct delivery of information from one cell phone to another without the need for the recipient to be logged into some service. Instant messaging requires both parties to be logged into some service, and is superior to sending text messages when compared on the grounds of maximum data transmission sizes and cost. Instant messages usually have no limit on the maximum data transmission size, but if the same amount of information is to be sent in an instant message as what should be sent in an e-mail, the format and display of such a large amount of information could be undersirable. E-mail is thus convenient for sending large amounts of information cheaply and easily. Disadvantages of sending large amounts of information, and hence large packets via a cell phones intermittent data connection mean that if that packet is lost, then it will be costly and cumbersome to retransmit that packet.

ThEm consists of a number of applications, namely: JabberEm, SendEm and MailEm. Each of these applications was designed as a proof of concept of mobile communication for mobile devices. Instant messaging, text messaging, and e-mail are fast becoming part of everyday communication. JabberEm is an implementation of the Jabber protocol on cell phones and enables users to communicate with Google Talk contacts, Mxit contacts, and 2Go contacts. Essentially JabberEm is able to send and receive messages between any client which makes use of the Jabber protocol. SendEm is a proof of concept application which enables users to send text messages by making use of the free text messaging service provided by the Vodacom4me website. Users are limited to twenty text messages per day. MailEm is an e-mail client implementation which allows users to send and receive e-mails using their Google e-mail address. MailEm too is a proof of concept application implementation, and thus HTML and attachment support for e-mail has not been included.

JabberEm, SendEm and MailEm all make use of one of J2ME's storage functions, namely: Record Stores. Record Stores allow information such as usernames and passwords and various other non-persistent information to be stored persistently and securely. Record stores reduce the need for the repetitive entry of information in J2ME applications and also allow for consistent management of application settings and defaults.

The implementation of all the applications developed in this project was done in J2ME, with the help of varous toolkits. Due to the mobile device design considerations mentioned in Section 2.2, all applications were developed with maximum efficiency, compact size, and a simple user interface which caters for the regular and advanced users needs. The user interface was constructed using the standard user interface components of J2ME. Many applications make use of user interface toolkits which dramatically improve the look and feel of the mobile application, but at the same time, the efficiency of the application is diminished considerably. For this reason it was decided to use the standard user interface components of J2ME in such a way so as to provide maximum aesthetic appeal to users without compromising the efficiency of the application. Since ThEm makes use of the native user interface components in J2ME, the aesthetic quality of the applications is diminished. There is however a J2ME component which forms part of the Screen class, namely the List, which allows for the appending of icons and the general improvement of an otherwise dull user interface. Because of the vast amount of varying cell phone models and a slight variation in the interpretation of J2ME on each model, the user interface sometimes differs considerably from the intended look and feel. Sometimes buttons on the cards and decks in the applications are displayed in reverse order. Refer to Section 2.5.1 for an explanation of cards and decks.

J2ME was selected as the application platform of choice for a number of reasons. The main reason being the simplistic publication and installation of JAR files in the J2ME environment. J2ME has integrated security mechanisms which protect the integrity of the data contained in Java Mobile applications. One of these security measures implemented in J2ME is application level security, where access to libraries and resources is only granted where the user has explicitly authorized such actions. Various other levels of security were explained in Section 2.1.3

Since all the applications make use of network connectivity, the need for threads is imperative, if the user is able to navigate around the user interface, and simulaneously allow the application to perform the core functionality, which is the exchange of data over the internet. All of the applications provide error prevention, thus minimizing application exceptions. Error prevention is implemented by ensuring that users have entered expected data types and that required fields have been filled in. ThEm has been designed for MIDP 2.0 and CLDC 1.1 compliant devices. All applications were tested on Nokia devices, ranging from lower end Nokia handsets to Symbian Series 40 and Symbian Series 60 devices. Emulators for Motorolla, Sony Ericsson, Nokia, and Samsung devices were used, but the majority of the testing was performed on the physical Nokia Series 40 and Series 60 platforms.

The average size of the applications in ThEm is 84.3 kb.

- JabberEm (Instant Messaging) – 132 kb
- SendEm (Text Messaging [HTTP]) – 52 kb
- MailEm (E-mail) – 69 kb

Clearly any of these applications can be installed on cell phones with the lowest processing and memory capabilities.

One of the major benefits of messaging, both instant messaging and text messaging, is that messages are able to be sent asynchronously. Asynchronous messaging is non-blocking, meaning that users can send messages and continue to do other operations without having to wait for the sending of the messages. Synchronous messaging blocks the process until the sending and receiving of messages is complete, meaning that users cannot continue working while messages are being sent and received. It is usually said that asynchronous messaging is superior to synchronous messaging, but in certain circumstances this is not the case, as will be explained next. For seamless user interaction with JabberEm it is necessary for an implementation of asynchronous messaging, since the user often wants to send a message to one contact and then continue chatting to another contact while the message is in transit to the first contact. This however differs somewhat in the implementation of SendEm, where the number of messages sent simultaneously as well as sequentially is governed by the Vodacom4me server. SendEm is thus more of a synchronous messaging implementation, apart from the fact that certain buttons and menus can be accessed [Mallick, 2003].

3.2 JabberEm

As this project was created to investigate issues in mobile development, it was decided to interface to an existing Jabber server, namely: The Google Talk server. The Google Talk server would be a much more efficient and effective implementation than a custom server. JabberEm makes uses of the iLabs Mobile Toolkit [iLabs, 2008], which provides an abstraction of the Jabber protocol. The iLabs Mobile Toolkit is ideal for the sending and receiving of messages and statuses and provides a simple API for Jabber client development.

Whenever a chat with a contact is initiated, a new form object is instantiated, and stored in a vector. The information of the contacts that we are in a chat with, i.e their email address, nickname, and status, are stored in another vector, which is a logical representation of the vector that stores the chats, and the messages contained in each chat. Here is a pseudo code example of the code that takes care of where to display the messages sent and received:

```
A message is received from contact A
    iterate through the vector of contact chats
        if contact A is not in our vector of contact chats
            create a new chat form
            append the message received to that chat form
            store the chat form inside the vector of contact chats
        else if contact A is in our vector of contact chats
            grab the current contact chat form
            append the message received to the contact chat form
```

From the pseudocode it can be seen that when a message is received, the vector of contact chats is checked to see if the current chat has already been added to the vector, and if so then that specific contact chat form is retrieved from the vector and the message received is appended to the chat form. If the current chat is not contained in the vector, then a new chat form is created, the message received is appended to that chat form, and the newly created chat form is then stored inside the vector of contact chats.

Checks also need to be done to determine whether we have read the message received or not. The following pseudo code shows how this is done:

```
When a message is received
    If contact is in view
        display their status next to contact name
    Else if contact not in view
        display the new message icon to contact name
```

When the user is not focused on the contact that a message was received from, then the new message icon should be displayed next to that contact's nickname, otherwise if the user is focused on the contact that a message was received from then the contact's status should be displayed next to their nickname.

When a contact chat form contains more than thirty messages, the form is slow to load. The reason for this, is because all the message information is stored in the contact chat form, and when the user clicks on the chat button to view the contact chat form, then all this message information has to be loaded from the vector of contact chat forms. Clearly, if the number of messages sent and received increases, so does the size of the contact chat form, and thus the time to load and display the contact chat form. The delay is not significant, but can be noticed. Depending on the size of the contact chat form, the delay is usually between 0.1 and 1 second.

3.2.1 Contact Presence

Section 2.3 outlines the various types of presence updates for instant messaging, namely: presence query, presence notification, and presence update. When a contact changes their status between online, away, busy or offline, then one of the presence update methods is used to inform other contacts of the changed presence.

iLabs Mobile Toolkit provides an abstraction for contact presences through the use of a method that receives all status changes. When a contact changes their status from online to away for example, a method receives the change and in turn calls a method in JabberEm, that determines whether the contact is online, away, busy, or offline. Before changing the image next to the contact to their current presence, a check needs to made to determine whether that contact contains new messages, and if so, then the contact presence will only be updated once the message has been read, thus removing the new message image next to the contact nickname.

When JabberEm is installed, the Jabber Bot (jabberem@gmail.com) is automatically added to the user's contact list, and may not be removed or renamed. The purpose of the Jabber Bot is to provide a means to communicate to all users using JabberEm, thus informing them of product updates, and also providing a means for advertising. Users are able to chat with the Jabber Bot, and post feedback.

3.2.1 Contacts and friend requests

The adding, editing and removal of contacts from an instant messaging client is vital, since contacts are at the very core of such an application. JabberEm is therefore capable of adding, editing and removing of contacts, as well as dealing with friend requests. There are three types of contacts that can be added to JabberEm, namely: Google Talk, Mxit, and 2Go contacts. The types of contacts that can be added, edited or removed can be seen in Figure 9 and Figure 10 below:

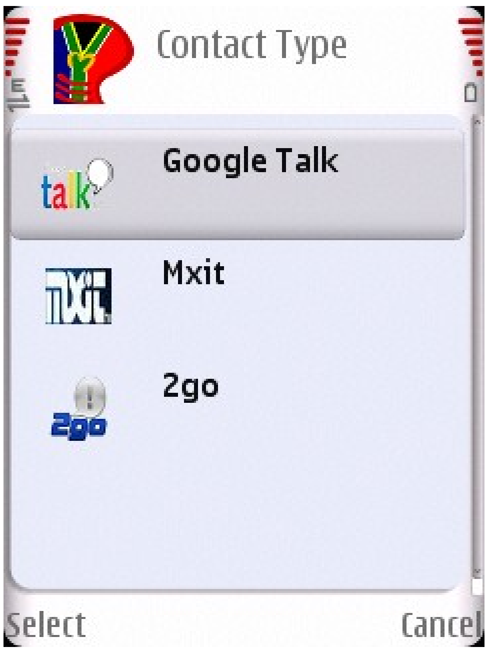


Figure 9: Types of contacts in JabberEm

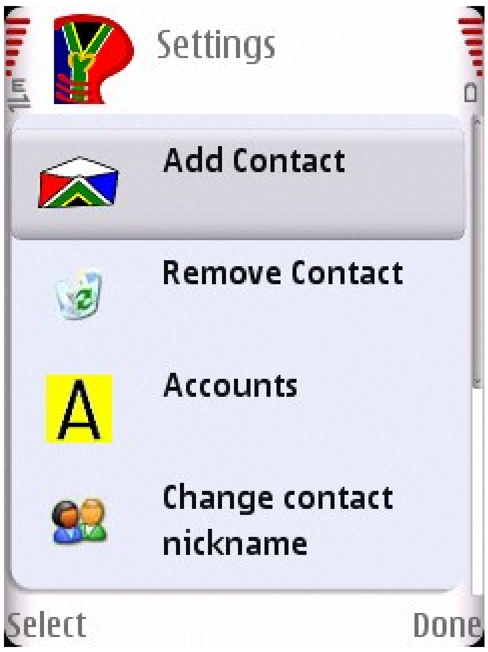


Figure 10: Settings list in JabberEm

The reason why these contacts can be added in JabberEm is because they all make use of the Jabber protocol, which is a great proof of concept to demonstrate the seamless integration of JabberEm with multiple Jabber servers. Three different functions must be supported with contacts, namely: add, edit, and remove. When a contact is added, a list of the type of contact to be added is displayed, thus allowing the user to choose whether they would like to add a Google Talk, Mxit, or 2Go contact. A contact can be edited by simply altering their current nickname. When contacts not currently on the user's contact list add the user as a contact, then a Alert pops up providing the user with the option of accepting or declining *"user A"* as a contact, and also provides information such as whether *"user A"* is a Google Talk, Mxit, or 2Go contact.

3.2.2 Storage in J2ME

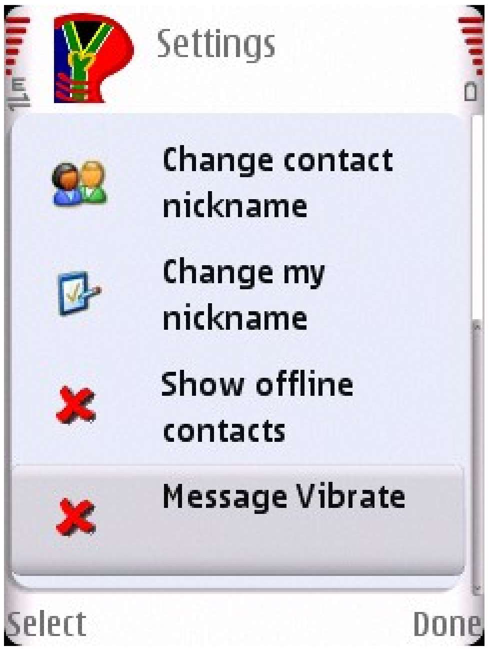
All user data such as email addresses, nicknames and passwords are stored to the persistent storage. This storage is know as RMS, and is in fact a minute database for mobile devices [Knudsen, 2007]. JabberEm also stores preferences, such as whether the user wants the phone to vibrate upon message receival, and if the user wants to view offline contacts. All data that is stored in the record store, is also stored in a vector, which represents the contents of the record store in a logical manner.

As described in Section 2.2.2 the problem with record stores arises when you remove a record. Unlike a vector, when you remove a record, all records after the remaining record do not "shift up" to take the place of the removed record. So the way that this was dealt with is: Whenever a record is removed from the record store, then you also removed the record from the vector. The record store is then closed and removed. Of course, as was mentioned above, a vector shifts elements after the deleted element up in the vector, so we have the desired representation of the data that is to be stored in the record store. We then open/create the record store, and iterate through all our elements in a vector, and at the same time, we add those elements to the record store.

The advantage of using a record store is that it is small, and it is very secure. The reason for its security is that midlets, cannot access and open record stores created and contained in other midlet suites [Topley, 2002]. Since mobile devices do not have the same computing power and space as a personal computer, space, performance and security are vitally important to developing a successful mobile application.

3.2.3 Alert profiles and contact list options

The user has the option of enabling or disabling the vibrate function upon message receival. Regardless of whether this function is enabled or disabled, when a new message is received, the new message icon is displayed alongside the contacts name on the contact list. The message vibrate function can be seen in Figure 11 below:



Change contact nickname

Change my nickname

✗ Show offline contacts

✗ Message Vibrate

Figure 11: Message vibrate option in JabberEm

If a contact sends a message and then logs off before the user has a chance to read the message, then the contact will remain in his/her original position on the contact list, with the new message icon alongside his/her name until the user reads the new message.

The user is further able to customize JabberEm by showing or hiding offline contacts. This function could prove useful in the scenario where the user has the desire to send a message to an offline contact. As was mentioned above, both the message vibrate and show/hide offline contact preferences are saved to persistent storage, which then prevents the user from having to continuously change these settings upon program startup. The option for showing/hiding offline contacts can be seen in Figure 11 above.

3.2.4 Error prevention

There are a number of areas where a user could make errors in JabberEm, some of which include: the creation of accounts (Where the user enters his/her Google Talk account details); the adding of accounts; the removal of accounts.

It seems logical to expect users to create an account before trying to login, but through user testing it was found that a number of users tried to login before creating an account. Checks were implemented to prevent such errors. When the user creates an account, their Google Talk e-mail address and password are needed. To make the entering of email addresses throughout JabberEm more user friendly, a default value is always present in the e-mail address field. E.g user@gmail.com. This allows for the user to simply remove "user" and enter in the appropriate information, thus minimizing potential errors. Checks to make sure that the "@gmail.com" is present in the submitted field are also performed.

When the user adds a contact, they are provided with the option of adding Google Talk, Mxit or 2Go contacts. When a Google Talk contact is chosen to be added, then the aforementioned procedure is performed to ensure validity of submitted fields. On the other hand, if a Mxit contact is chosen to be added, the user is then able to either enter the Mxit contact's information manually or search through their contact list. The procedure for checking the format of cell phone numbers is described in Section 3.2.2 below. Based on the assumption that the number is in the correct format, the following string: "27NUMBER@mxit.co.za" is then used to query the contact for friend request acceptance.

When the user chooses to remove a contact, a confirmation dialog pops up thus preventing a user from erroneously removing a contact.

3.2.5 Interface Design

The user interface of JabberEm was implemented in accordance with the guidelines mentioned in Section 2.6.2 and Section 2.7. Buttons with opposing functionality in JabberEm were placed at opposite ends of the list to which they are appended, which reduces user errors of pressing the wrong button. With the correct placement of buttons and a simple menu for editing the settings in JabberEm, the ease of use of the application is greatly improved. The keystoke level model was also used to improve the efficiency of the user interface and the resutls thereof can be seen in Section 4.

3.3 SendEm: Text messaging over HTTP

All MIDP devices are required to provide an HTTP connection [Feng, 2008]. HTTP makes use of sockets that are utilized in personal computers. The content that is transferred over these sockets is displayed in Hyper Text Markup Language (HTML). Lower end MIDP devices cannot deal with the same sockets as personal computers do, nor can they display HTML (Higher end cell phones are however able to make use of these sockets, and display the content communicated in HTML). The device needs to connect to a gateway which transforms HTTP messages into a protocol that is used to connect to the internet gateway. Devices often use the Wireless Session Protocol (WSP) to connect to a Wireless Application Protocol (WAP) gateway, which in turn connects the wireless network to the Internet [Topley, 2002]. One way to send messages in J2ME is using the Wireless Messaging Application Programming Interface (WMA) [Wireless Messaging API, 2008], and this simply makes use of the conventional GSM SMS services provided by your service provider. The WMA combined with the Personal Information Management (PIM) API [Mahmoud, 2005], allow the sending and receiving of messages, and at the same time, provide access to the user's received messages, sent messages, drafts, contact information and calendar information. As can be seen, the combination of these API's could provide a very powerful messaging application [Knudsen, 2007].

The SMS structure in Figure 2. above will deduct approximately eighty cents (depending on the time at which the message is sent) from the user's account everytime a conventional SMS is sent over the GSM network. Instead of utilizing the GSM text messaging services provided by the service providers, SendEm routes all text messaging data through the Vodacom4me website, which then bypasses the costs involved in sending a text message over the GSM network. Thereafter, the text message is passed onto the Vodacom SMSC, which in turn passes the text message onto the recipient's service provider's SMSC. This process is illustrated in Figure 12 below:

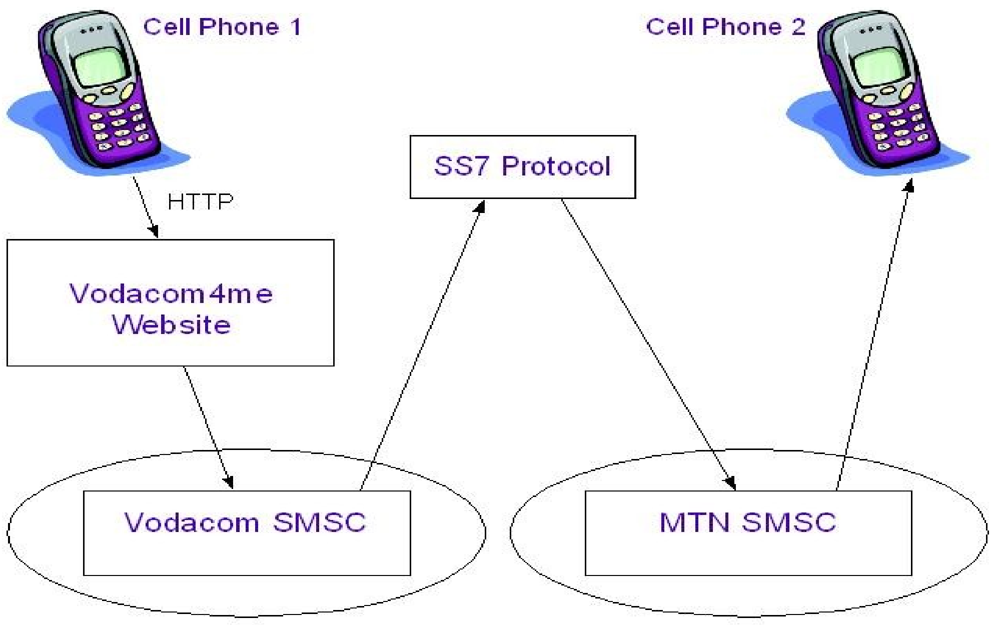


Figure 12: Text messaging over HTTP

The cost of 1Mb of data sent on the Vodacom network is R1.20 for customers who do not have a data bundle, and R0.50 for customers who have a data bundle [Vodacom, 2008]. Message data from SendEm to the Vodacom4me website is typically in the range of one to two kilobytes. A text message sent using SendEm will cost one cent for users who do not have a data bundle, and 0.5 cents for users who have a data bundle. This is a significant saving when compared to the approximate eighty cents per message during peak times on Vodacom.

Since this method of text messaging is only a proof of concept, messages are only able to be routed to Vodacom subscribers. This prototype demonstrates the concept of sending text messages using the HTTP protocol, and also shows that services that were intended to be used with personal computers can be adapted to a mobile context. The next section describes how it is possible for messages to be sent using the HTTP protocol by making use of HTTP cookies.

3.3.1 Session tracking via HTTP Cookies

Cookies are embedded in HTTP headers, and consist of NAME=VALUE pairs. Cookies are contained in the HTTP headers, and are thus transparent to the users. Servers assign cookies to clients through the HTTP header set-cookie, which the client then uses to continue exchanging data with the server [Yuan, 2004].

The way in which SendEm works is through the use of Cookies embedded in HTTP headers. When a login packet is sent to the Vodacom4me website, the Cookie, containing the session information is sent back to SendEm in the HTTP header, which then allows the user to send text messages provided that the session information is attached to each message. The advantage of this is that messages can only be sent by the user who knows the sessionID, thus providing a level of security. Disadvantages of placing session information in HTTP headers is that Symbian Series 60 devices are unable to extract session information from Cookies contained in HTTP headers.

3.3.2 SendEm program structure and considerations

As was mentioned in Section 3.1, SendEm makes use of threads, and a simple user interface. In order for successful user authentication and hence the sending of messages, users need to create an account, in which they enter their Vodacom4me login details, which are then saved to persistent storage. Users are able to create multiple accounts and choose which account they want to use upon program startup.

In order for messages to be sent using the Vodacom4me server, the number of messages that user intends sending, needs to be communicated to the server. The number of messages to be sent is determined by obtaining a count of the characters contained in the data the user has typed. Since spaces are represented as "*%20*", each space needs to be replaced by "*%20*", and a new data string is to be constructed. Once the new data string is constructed, an accurate count of the number of characters needs to be obtained by simply making use of the character count calculated before the "*%20*'s" were incorporated into the data string.

SendEm allows for users to enter recipient's numbers in manually, or for users to search through their list of contacts. SendEm also allows users to send a message to multiple contacts by repeatedly making use of the "Add Recipient" button, which then allows users to scroll through their list of contacts as mentioned above.

Through user testing it was discovered that when a user sends a long or urgent message SendEm would fail to deliver the message to the recipient(s) due to intermittent network connections. The successful delivery of messages from SendEm is dependent on the reliability of the network connection as well as the availability and promptness of the Vodacom4me server that handles incoming requests to send messages. Due to server downtime and intermittent network connectivity it was decided to provide the user with the ability to save and load messages in order to reduce the frustration non delivered messages. A "Remove Messages" button was also provided to allow the user to clear saved messages that are non longer needed.

Apart from incorrect account details entered, careful attention needs to paid to the format of recipient numbers entered manually or from the contact list. It is highly unlikely that a user would enter in a recipient's cell phone number in the international number format: 27827882673, but nevertheless various methods have been implemented to check for such scenarios. All South African cell phone numbers consist of 10 digits, and this serves as a solid base for format checking. When a recipient is added from the contact list, the various formats that the number could take are as follows:

- 27082788267 (number with leading 27)
- +2782788267 (international number format)
- 082788 (incorrect number of digits)
- 0838276352 (non Vodacom subscriber)

All of the above number formats are easily are dealt with accordingly, and appended to the recipient list on the main form.

3.4 MailEm: E-mail for Java 2 Mobile

From Section 2.5 it can be seen that e-mail is the most widely used Internet application. With the explosion of mobile technologies and the constant need for being in communication while on the move, e-mail applications for cell phones and other mobile devices have played a large role in mobile communication.

MailEm is the e-mail client that was developed as part of this research project. MailEm makes use of the Mail4me API [Mail4me, 2008] which allowed emphasis to be placed on the business logic of the application instead of low level client server communications..

In order to be able to receive Gmail e-mails the Mail4Me API had to be modified to be able to deal with Secure Sockets Layer (SSL) connections in POP 3 client implementations. A connection to pop.gmail.com is established on port 995 using SSL by transmitting the user's username and password. Once the connection is established then the POP 3 client can begin to check for new e-mails. In order to send e-mails using Gmail's SMTP server, an SSL connection needs to be established with smtp.gmail.com on port 465.

The Mail4me API proves to make the sending and receiving of e-mail trivial, whether the POP3 or IMAP 4 protocols are used. There is however a flaw when using the mail4me API for e-mail application development, and this is the e-mail server notification of opened e-mails. Under normal circumstances when using MailEm, the user would download “unread” e-mails and then proceed as desired, but because the mail4me API has no support for informing the e-mail POP3 or IMAP 4 server of whether an e-mail has been read, the normal set of procedures for e-mail client operation is rendered redundant. In order for MailEm to function as a successful e-mail client, a technique needed to be devised which enabled e-mail server notification of opened e-mails.

The technique in which this is performed is by making use of the following pseudo code:

```
Check for new e-mail
if new e-mails exist then
        if e-mail is stored in record store then
                if flag is marked as e-mail read and e-mail date and time is different then
                        do nothing
                else
                        retrieve e-mail from record store and append to list
        else
                Add e-mail to record store and append to list
```

As can be seen in the pseudocode, new e-mails will always exist if only using MailEm to check e-mail, due to the flaw in the mail4me API mentioned above

By making use of this custom algorithm it can clearly be seen that new e-mails are dealt with accordingly in MailEm, and only unread e-mails will appear in the inbox list. Since MailEm is a proof of concept, it was decided that only new e-mails should be displayed in the inbox list. MailEm can however be modified relatively easily to accommodate for the display of all e-mails in the inbox list. An advantage of MailEm is that valuable bandwidth is saved by only downloading the header of the message, checking if the record store has any trace of that message, and only if this is not the case then will the body of the e-mail be downloaded. The major disadvantages is that over time, the number of e-mails stored in MailEm's record store will increase and could possible compromise the efficiency of the application.

3.4.1 Application functionality

MailEm provides the same basic functions that any e-mail client would provide, namely:

- Compose an e-mail
- Receive e-mail
- Forward e-mail
- Reply to e-mail
- Delete e-mail

Since the POP3 protocol was implemented in MailEm it was decided to separate the sending and receiving of e-mails. So instead of have one “Send And Receive” button as is found on typical desktop clients, there is a “Send” button and a “Receive” button. The reason for this being that the POP3 protocol downloads new e-mails everytime the user instructs the e-mail client to receive e-mails, and this would then result in an unnecessary bandwidth consumption.

The major problem associated with implementing an e-mail client for mobile devices are outlined in section 2.3. Unsupported formats proved to be a major issue in the implementation of MailEm since most cell phones are unable to deal with e-mail attachments. The sending and receiving of text based e-mails is performed flawlessly in MailEm, but a new set of errors become apparent when e-mail's containing HTML content are received. For the purposes of this project it was decided to provide the basic functionality of an e-mail client by sending and receiving text based e-mails as a proof of concept. There are however toolkits available for dealing with information contained in HTML pages, but this was considered beyond the scope of the project.

3.4.2 Error Prevention

Since user's cannot make any errors when receiving e-mails, validation only needs to take place when user's send an e-mail. The following parts of MailEm require validation: the recipient e-mail address; the e-mail subject and the e-mail body. The receipent's e-mail address needs to be in the correct format I.e user@email.co.za. So the “@” character needs to be present, as well as “.something”. E-mail's with blank subject and body field's may be sent, but before the e-mail is delivered, a confirmation dialog is displayed when a blank subject or body field exists.

3.5 Summary

From mobile network connectivity to local data security and aesthetic appeal, the design and implementation of applications on mobile devices is fast becoming similar exercise to the development of applications for computers.

SendEm makes use of an alternative method to sending conventional text messages via GSM, instead the same text message is transmitted via HTTP to the appropriate SMSC and then delivered accordingly. This alternative approach proves to reduce expenditure when compared to conventional text messaging, and is equally as efficient and secure. Connections to the Vodacom4me server are secured through the use of the HTTPS protocol. As is was mentioned above, cell phones have intermittent data connections, and packets are lost from time to time, so in order to minimize user frustration, the aforementioned record stores were utilized to load and save unsent messages for future retransmission.

During the Internet's inception, there were a limited number of people who had access to the Internet, but since then there has been an explosion of Internet users, and with this explosion so too is there an increasing security threat. Apart from the simplistic design and trusting approach of an e-mail protocol such as SMTP, the sending and receiving of e-mails in MailEm performed in a secure manner by making use of SSL connections. Since mobile applications have limited functionality when compared to applications on the computer, it cannot be expected to have all functions normally seen in desktop applications. Most of the basic functions of a desktop e-mail client are included in MailEm such as: sending and receiving e-mails; forwarding e-mails; replying to e-mails; and deleting e-mails.

The handling of attachments and rendering of HTML based e-mails is unfortunately not supported by MailEm. Due to limitations of the Mail4me API, deleting of e-mails is handled somewhat differently.

JabberEm allows for seamless communication with Google Talk, Mxit and 2Go contacts. All of the basic functions of a desktop instant messaging client are supported, and the user interface was designed in such a way so as to provide maximum aesthetic appeal as well as usability.

ThEm makes use of extensive user validation and error checking so as to minimize user errors and frustration. From the checking of valid recipients in SendEm to the removal of Jabber bots and the sending of blank e-mails in JabberEm and MailEm respectively, error checking and validation ensures that users don't make common errors and prevents the throwing of exceptions. The evaluation of the core functionality of JabberEm as well as the efficiency of the various user interfaces described in this Chapter will now be presented in Chapter 4.

Chapter 4. Evaluation

4.1 Introduction

This Chapter attempts to compare and contrast the various functionality and user interfaces of JabberEm and Mxit. It was decided that since JabberEm is based on the same user interface design as SendEm and MailEm, and JabberEm can be compared to a similar application such as Mxit, that it should be used as the basis for evaluation in ThEm. Four experiments are conducted under three main categories, namely: Core functionality timing, user interface evaluation and timing, and user testing and evaluation. Core functionality timing is comprised of Experiment's 1 through 3. Experiment 1 deals with timing between chat clients and Google Talk. Experiment 2 handles the timing between JabberEm clients and timing between Mxit clients during off-peak hours and Experiment 3 handles the timing between JabberEm clients and timing between Mxit clients during peak hours. User interface evaluation and timing is comprised of Experiment 4. In order to effectively evaluate the user interfaces of Jabber and Mxit, Experiment 4 was conducted in order to evaluate the user interfaces using the Keystroke-Level Model for Advanced Mobile Phone Interaction.The third category of evaluation is user evaluation which takes into account user satisfaction while using JabberEm and Mxit. A combination of these three categories will allow for an overall and consistent comparison between the two applications, and will allow us to draw rough conclusions so as to determine the application with the more efficient user interface and core functionality performance.

4.2 Design of experiments

In order to compare the efficiency of sending and receiving of messages, it is necessary to compile timing information taken during peak and off-peak hours so as to get an accurate representation of the results. The most important timing data that should be noted is that which is taken when exchanging messages between two JabberEm clients and two Mxit clients. I.e both cell phone A and cell phone B would be running JabberEm and time taken to send messages between the two would be noted. The same applies for Mxit.

In Section 3.1.2 it was mentioned that JabberEm is a proof of concept implementation which allows for the sending and receiving of messages to and from multiple Jabber server implementations. Since Mxit also allows messages to be exchanged between multiple Jabber servers, this enables us to evaluate the time taken when exchanging messages between JabberEm and Google Talk, as well as between Mxit and Google Talk.

In order to evaluate the efficiency of the user interfaces of JabberEm and Mxit, we need a way of qualitatively and quantitatively representing the results. The way in which this is done is by making use of Keystroke-Level Model for Advanced Mobile Phone Interaction, which effectively calculates the time taken to complete common user operations, such as key presses and finger movements. The addition of these times gives a quantitative indication of the efficiency of the user interface.

Without user evaluation and approval, the time to exchange messages and the efficiency of the user interface are rendered useless. User evaluation will consist of categories such as aesthetic appeal, ease of use, and functionality of the application.

4.3 Methodology

The first experiment collects and analyses the times of messages sent and received between JabberEm and a Google Talk client, and Mxit and a Google Talk client.

Experiment one consists of a sample period of five messages sent from a Google Talk client to both JabberEm and Mxit. five messages are also sent from JabberEm and Mxit to a Google Talk client. By combining these results a total of ten messages for JabberEm and Mxit are analysed, and enables relatively consistent conclusions to be drawn so as to which which client possesses faster communication with a Google Talk client.

The sending and receiving of messages in Experiments 1 and 2 was performed between a Nokia N95 8GB and Google Talk running on Windows XP.

Experiments 2 and 3, however, provide more conclusive results in determining the faster client with regards to message exchange. Experiment 3 monitors and analyses the time it takes for messages to be delivered between two JabberEm clients and the time taken for message delivery between two Mxit clients during off-peak hours. Experiment 3 is similar to Experiment 2, only that it is performed during peak hours. Since most people make use of Mxit during the early evening, it was decided that timings of message exchange should be conducted both during peak and off-peak hours. Peak sampling was conducted between 7:26 pm and 8:35 pm, and off-peak sampling was conducted between 10:00 pm and 11:30 pm. Clearly for Experiments 2 and 3, two cell phones are needed, with both phones having JabberEm and Mxit installed. The two phones used were the Nokia N95 8GB and the Sony Ericsson W850i. A sample period of 20 messages was used for both Experiments 2 and 3.

Experiment 4 makes use of the Keystroke-Level Model for Advanced Mobile Phone Interaction, and evaluates the user interfaces of both JabberEm and Mxit when performing certain key functions.

All timing was performed using a stopwatch in Ubuntu Linux. The stopwatch began timing when the send button in JabberEm and Mxit was pressed. When the messages arrived at the Google Talk and/or the other clients, the stopwatch was stopped and the time recorded.

The following experiments were conducted in order to obtain results for user evaluation of the applications:

In order to get an accurate set of results for learnability, efficiency and memorability it was decided to make use of timing methods. Since learnability is defined as the ease of which applications can be used for the first time, a sample of six users were asked to perform each of the following core functions in JabberEm:

- Send message (The message “hi” was sent to a contact)
- Add contact (A Google Talk contact “test” was added)
- Remove contact (Contact at the top of contact list was removed)
- Show offline contacts

- Change contact nickname (Contact at top of contact list changed to “hi”)
- Change current user nickname (Currently logged in user nickname changed to “hi”)
- Message Vibrate

The times taken to perform each function for the first time were recorded, and then averaged for both Mxit and JabberEm in order to determine the learnability of the applications. Since the timing between the learnability and efficiency test results is irrelevant, the efficiency test was performed immediately after the learnability test.

Efficiency can be defined as the ease at which a user can perform certain functions once the design has been mastered. The way in which the efficiency of both applications was quantified was by asking each of the 6 test subjects to cycle through the above core functions twice over with a 2 minute break in-between each cycle. The times for each round were added and then averaged which then gave an indication of the efficiency of both applications.

In order to determine the memorability of the applications, the 6 test subjects were asked to perform each of the core functions exactly 2 days after the tests for learnability and efficiency were conducted. All usability tests were conducted on a Nokia N95 8GB.

Questions 1 to 12 below, obtained from a paper written by Yeng [2005] were used in combination with the results from the above experiments to determine user satisfaction with the applications. The questions below cover some of the important areas of user evaluation, such as question 1 investigating ease of use and question 3 refers to aesthetic appeal. Question 7 takes care of error recovery and questions 9 through 11 cover areas such as efficiency and learnability. The questions in Table 3 were subsequently used in the user evaluation after the timings for the core functions were performed.

Table 3: Questions for user evaluation of applications

Question	Rating
1. Please rate the ease of use of the application.	1 (Easy) – 5 (Difficult)
2. What do you think about the menu structures in the application, are they clearly labeled?	1 (Clear) – 5 (Unclear)
3. Is the application aesthetically appealing?	1 (Attractive) – 5 (Unattractive)
4. What is/are the BEST feature(s) of the application?	Text
5. What is/are the WORST feature(s) of the application?	Text
6. What new content or features would you like to see in the application?	Text
7. Can you recover from mistakes easily?	1 (Easy) – 5 (Difficult)
8. Your overall reaction to the application:	1 (Satisfied) – 5 (Unsatisfied)
9. Do you feel lost when using the application?	Yes / No
10. Is the application easy to navigate?	Yes / No
11. When you press a button in the application, do you expect the button press to lead to the correct answer?	Yes / No
12. Are there any comments about the application?	Text

4.4 Results and Analysis

This section presents the results of each of the four experiments mentioned in Section 4.1 above, and attempts to compare JabberEm and Mxit in terms of user interface efficiency and core functionality performance.

4.4.1 Experiment 1: Timing between chat clients and Google Talk

The time taken for JabberEm and Mxit clients to receive messages from Google Talk is presented in Table 4 below:

Table 4: Message receival time from Google Talk in seconds

	JabberEm	**Mxit**
Message 1	3.03	3.62
Message 2	3.36	2.85
Message 3	2.41	3.16
Message 4	4.64	3.86
Message 5	3.13	5.17
Total Time	**16.56**	**18.66**

The time taken for JabberEm and Mxit clients to send messages to a Google Talk client is presented in Table 5 below:

Table 5: Time in seconds to send messages to Google Talk client

	JabberEm	**Mxit**
Message 1	1.53	1.83
Message 2	3.14	2.06
Message 3	1.57	1.86
Message 4	1.57	1.67
Message 5	1.46	1.75
Total Time	**9.26**	**9.17**

It can clearly be seen from Table's 4 and 5 above that the time taken to receive a message in JabberEm from a Google Talk client is 2.1 seconds faster than receiving the same message using Mxit.

Mxit is however 0.09 seconds faster than JabberEm when sending a message to a Google Talk client. Overall it can be concluded that over a sample period 10 messages for each client, JabberEm is 2.01 seconds faster than Mxit at sending and receiving message to and from a Google Talk client. Even though these timings are not significant, this experiment is needed in order to determine the application with then faster sending and receiving times of messages.

4.4.2 Experiment 2: Timing between JabberEm clients and timing between Mxit clients during off-peak hours

A sample period of twenty messages was taken to determine the timing of the sending and receiving of messages between clients. All timing was done between 10:00 pm and 11:47 pm, which is the optimal time since the number of users logged into Mxit is less than if the timings were done at 8:00 pm.

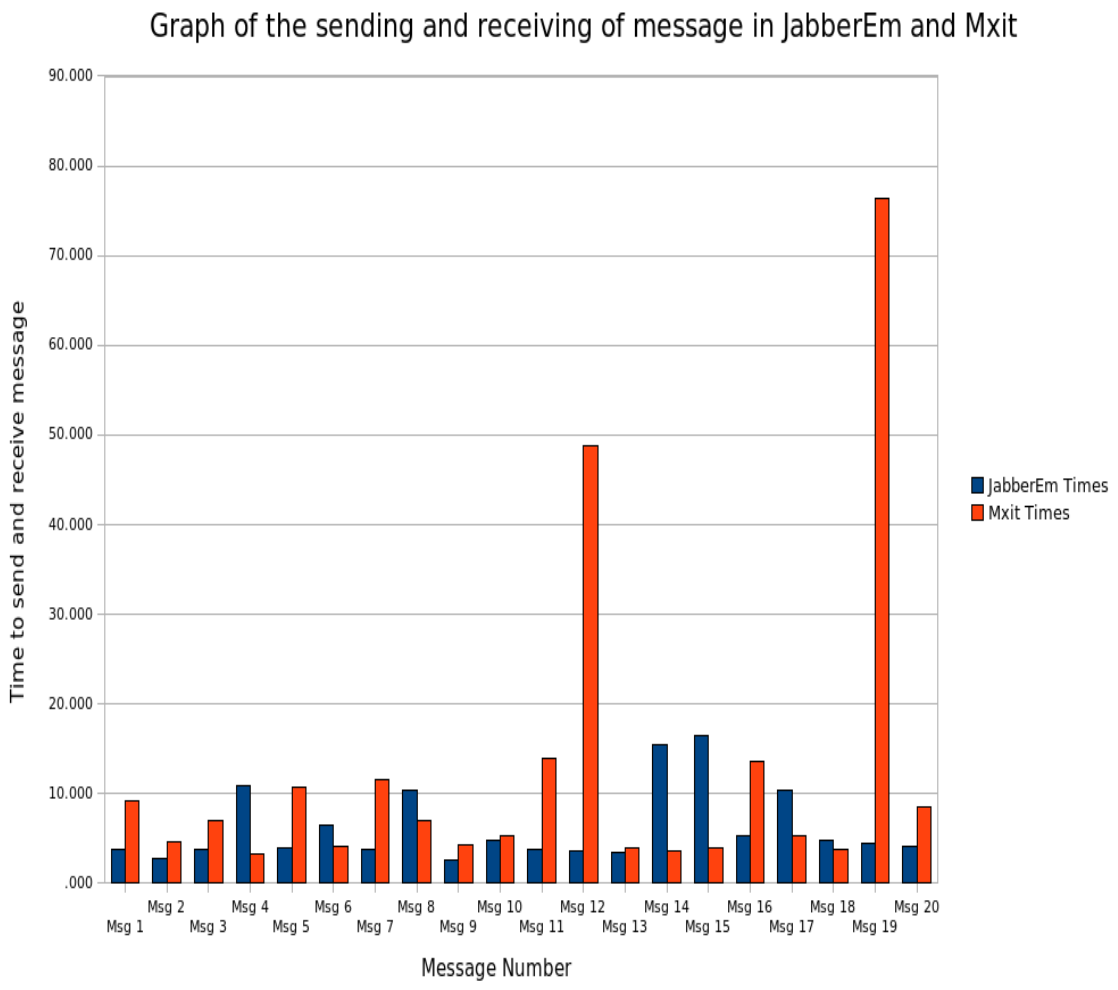


Figure 13: Graph of the sending and receiving message times in JabberEm and Mxit during off-peak hours.

Over a sample period of twenty messages, the sending and receiving of messages between JabberEm clients was 8.754 seconds faster than the sending and receiving of messages between Mxit clients.

From the Figure 13 it can be seen that the sending and receiving of messages in JabberEm follows a more gradual increase and decrease in time than that of Mxit. A possible reason for this is that the Mxit Jabber servers aren't equipt to handle volume as efficiently as those of Google. Appendix 1 contains the timings for each of the twenty messages sent in Mxit and JabberEm during off-peak hours.

4.4.3 Experiment 3: Timing between JabberEm clients and timing between Mxit clients during peak hours

The following timings were done between 7:26 pm and 8:35 pm, which is when most users are logged onto Mxit during the week.

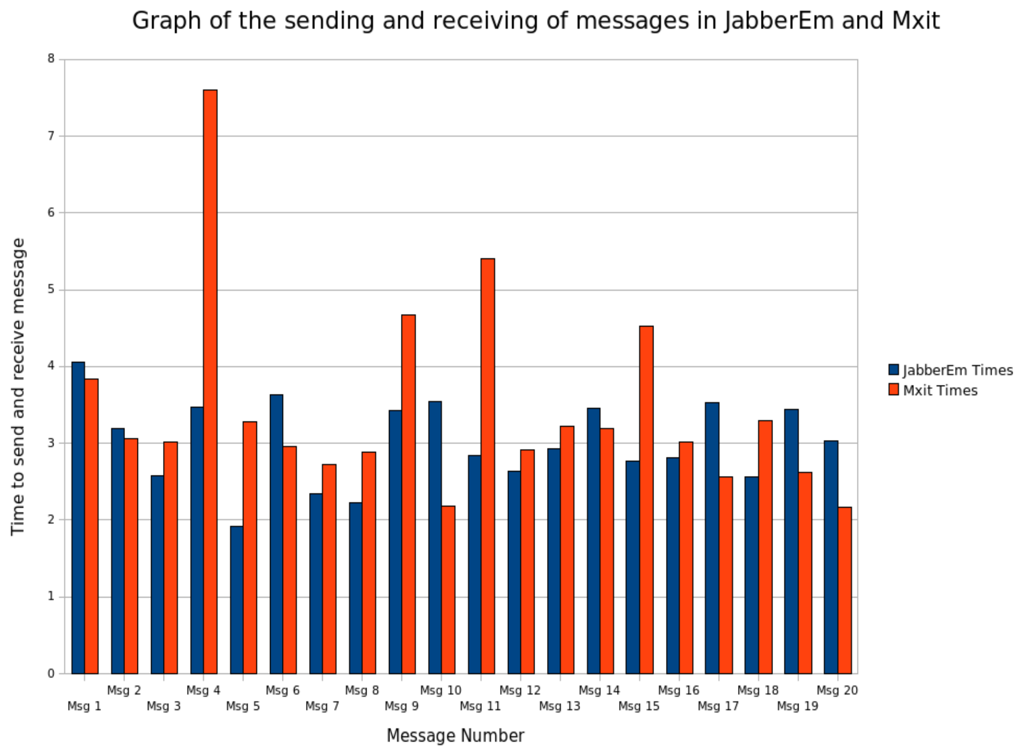


Figure 14: Graph of the sending and receiving message times in JabberEm and Mxit during peak hours.

During off-peak hours, the amount of time required to send twenty messages on Mxit equated to 69.171 seconds. During peak hours the time required to send twenty messages equated to 248.770 seconds which is 359.64% slower than sending the messages during off-peak hours. During off-peak hours the time needed to send twenty messages on JabberEm equated to 60.417 seconds. The time required during peak hours equated to 125.155 seconds, which is 207.15% slower than sending the messages during off-peak hours. During peak hours Mxit is 123.615 seconds slower than JabberEm at sending twenty messages. Clearly during peak and off-peak hours JabberEm is able to deliver messages more quickly, which in turn results in less user frustration and confusion. Appendix 2 contains the timings for each of the twenty messages sent in Mxit and JabberEm during peak hours.

4.4.4 Experiment 4: Evaluation of user interfaces using the Keystroke-Level Model for Advanced Mobile Phone Interaction

In order to evaluate the efficiency of the JabberEm and Mxit's user interfaces, it was decided to use the Keystroke-Level Model (KLM) to determine the time required to perform the following core components of an instant messaging client: send message; add contact; remove contact; show offline contacts; change contact nickname; change current user nickname; and message vibrate.

For purposes of brevity it was decided to show the workings of the KLM for sending a message. The results of the evaluation are based on the assumption that the user has the cell phone in hand, and that predictive text is activated. Since contacts may be listed in different positions on the contact list, the number of key presses required to select a contact was considered to be one. All evaluation was performed on a Nokia N95 8GB handset.

Appendix 3 and Appendix 4 show the details of the KLM that was used to calculate the theoretical time to send a message in JabberEm and in Mxit respectively. It should be noted that the time required to send the message was not taken into account. All test results are independent of the time needed for data transmission. This is purely an indication of the time required to navigate and execute various functions embedded in the user interfaces of

JabberEm and Mxit.

Table 6: User interface navigation time of JabberEm and Mxit for specific function

	JabberEm (seconds)	**Mxit (seconds)**
Send message	5.66	5.88
Add Mxit contact	6.43	9.56
Add Google Talk contact	14.26	28.15
Remove contact	2.04	4.09
Show offline contacts	2.08	4.77
Change contact nickname	3.31	6.46
Change current user nickname	3.82	9.03
Message Vibrate	1.52	4.16
Total	**39.12**	**72.10**

From Table 6 it can be deduced that JabberEm has a considerably more efficient user interface than Mxit. Section 2.4.1 describes the use of cards and decks in user interfaces, which reveal reasons why JabberEm has a more efficient user interface than Mxit. JabberEm only has one deck, namely the "Settings" deck, whereas Mxit has multiple decks as well as sub-decks. Since users only ever need to navigate one deck in JabberEm in order to carry out all the functions in Table 6, this results in a more efficient user interface than Mxit where users often need to navigate multiple decks and sub-decks.

4.4.5 User evaluation of applications

Progress of the information society has resulted in a huge increase in the amount of software being developed and used, and with this increase, so too have the effects of software defects. From the engineering point of view, the improvement of software standards if based on the utilization of design and verification tools and techniques. In addition to the engineering approach, the management approach also makes major contributions to the testing and control of software by defining quality goals and a technology for measuring, evaluating, and controlling these goals.

Appendices 5 to 7 contain the raw user evaluation data, showing the time taken for each user to perform the seven core functions of instant messaging mentioned in table 6 above.

From the Figure 15 below it can be seen that JabberEm has a much lower learnability time than Mxit, meaning that JabberEm is easier to use for the first time when compared to Mxit. Figure 15 was obtained by taking the total time for each user to perform each of the seven core functions in both JabberEm and Mxit. The times taken by each user to perform each of the core functions can be seen in Appendix 5. JabberEm had a total time of 376.33 seconds and Mxit had a total time of 602.79 seconds.

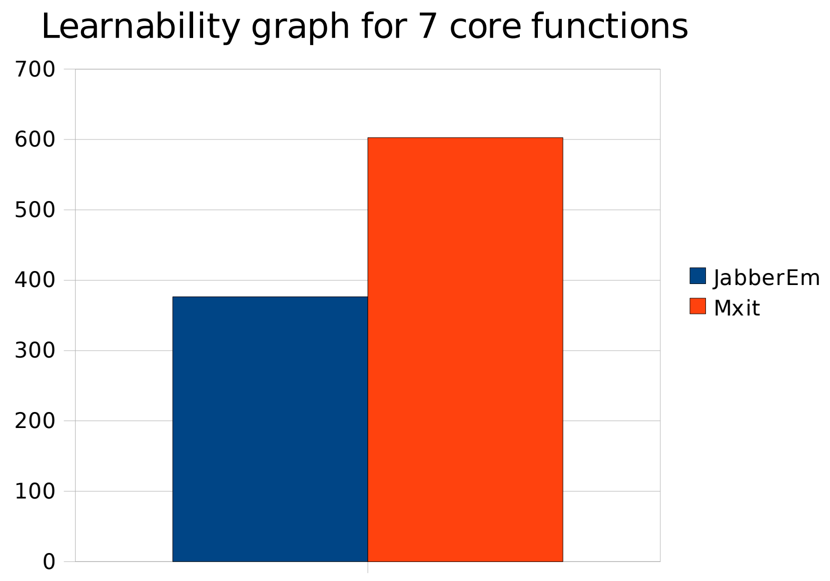


Figure 15: Total time taken to perform 7 core functions by each user

To calculate the efficiency of the applications, the users were asked to cycle through each of the core functions twice. The average of the two core function cycle times for each user was then obtained (highlighted in blue in Tables 9 and 10) and summed. JabberEm obtained a total time of 268.13 seconds and Mxit obtained a time of 337.7 seconds. JabberEm proves to be a lot faster at establishing proficiency than Mxit. Figure 15 shows that JabberEm has a much better learnability than Mxit, and since it is initially easier to use than Mxit, it is logical to expect that users would be able to establish proficiency more easily in JabberEm than Mxit.

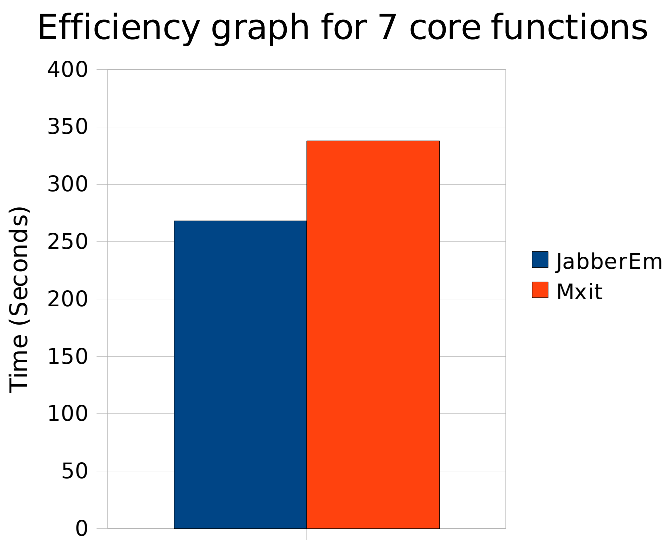


Figure 16: Total time taken to perform core functions twice over

The six users that performed the user evaluation are represented by three males and three females. User 5 (Male) had used Mxit extensively before the user evaluation for this thesis. User 1 (female) also used Mxit extensively before the user evaluation. Even with two out of the six users being regular users of Mxit, their timings from JabberEm still proved to be better than those from Mxit.

To prove the accuracy of the Keystroke-Level Model for Advanced Mobile Phone Interaction (KLMAMPI), the times taken for the six users to send a message in JabberEm were totalled an averaged. The average time of the users, excluding the transmission time of the packet over the network equals ***5.5609*** seconds which is remarkably close to the calculated KLMAMPI value of ***5.66*** seconds. The average time calculation can be seen in Appendix 6.

JabberEm has an efficiency average of 44.68 seconds (Appendix 6) and a memorability average of 51.08 seconds. Mxit has an efficiency average of 56.28 seconds and a memorability average of 73.63 seconds. In order to calculate the memorability, the times obtained in the memorability tests of both applications cannot simply be compared, but the difference in time between efficiency results and the results obtained two days after this (I.e the memorability results) are needed. The difference between the average efficiency time and the average memorability time of JabberEm is 6.4 seconds, and the difference between the two times for Mxit is 17.35 seconds. Clearly JabberEm has a much more memorable interface than Mxit.

JabberEm thus has faster learnability, efficiency and memorability times, and it can be concluded, with a high degree of certainty that JabberEm is a much more usable and efficient application than Mxit. JabberEm not only has a more efficient user interface than Mxit, but it is also outshines Mxit in message delivery time, as can be seen in section 4.4.3.

4.5 Summary

Without formal and user evaluation of applications developers have little idea of the quality of the developed applications. Formal evaluation makes use of quantitative methods and scientifically proven principles in order generate a set of results which can be used to analyse the efficiency of certain aspects of an application. Since these applications are of no use without user interaction, it is very important to take into account user opinions and evaluations.

Mxit is a large scale commercial instant messaging client which is an implementation of the Jabber protocol. Since JabberEm is also an instant messaging client which makes use of the Jabber protocol, the comparison of the core functions of both applications would not only prove to be interesting, but also allow for possible commercialisation of JabberEm. Sample periods of various sizes and between different applications were taken in order to analyse the message delivery times and hence the core the functionality of both applications.

The efficiency of the user interfaces of both JabberEm and Mxit were evaluated using the Keystroke-Level Model for Advanced Mobile Phone Interaction which enables us to quantify the efficiency of the user interfaces and in combination with user evaluation, determine which user interface is more efficient.

Chapter 5. Conclusion

During the inception of cell phones, they were intended to enable people to make voice calls without being confined to one place. With the rapid advances in technology and widespread use of cell phones, making simple voice calls was not enough. Java Mobile has simplified the development of applications for J2ME enabled devices, and has almost single handedly bridged the gap between communication on computers and cell phones. Text messaging allowed for a cheaper and less intrusive means of communication and greatly revolutionized the way in which we communicate on cell phones. When J2ME was introduced, there was an explosion of communication possibilities over the data networks such as instant messaging and e-mail. The price to send information using conventional methods such as text messaging over GSM as well as voice calls turn out to be considerably more expensive than sending the same information over a data network. Not all cell phones are J2ME compatible, and this makes Java Card an integral tool in developing applications which target a large number of cell phone makes and models.

With the introduction of colour screens on cell phones, combined with the native user interface components embedded in J2ME, the aesthetic appeal and usability of applications on cell phones has been improved dramatically. Since the screen area of cell phones is dwarfed when compared to that of a computer, it was thought that user interface design methodologies for computers were rendered useless on cell phones. This is however, not the case as can be seen with the application of the Keystroke-Level Model for Advanced Mobile Phone Interaction within this thesis. A well designed application with powerful functions is of no use without user evaluation. It is not only important to determine the ease at which a user is able to recover from an error, but error prevention functionality should be built into the application to try to minimize user mistakes. This is a classic case of prevent or eradicate the problem before treating the symptoms.

This research project has successfully produced three prototype applications which implement three text based communication technologies, namely text messaging (SendEm), instant messaging (JabberEm) and e-mail (MailEm). By making use of the the Advanced Keystoke Level Model for Mobile Phone Interaction, the times taken to perform various functions in the user interfaces of JabberEm and Mxit were obtained and it was concluded that JabberEm has a more efficient user interface than Mxit. Actual timings of performing various functions in the user interfaces of JabberEm and Mxit were also obtained and these results also showed that JabberEm has a more efficient user interface than Mxit. Since the times taken to send and receive messages in JabberEm and Mxit form the core of each application, experiments were conducted in order to determine which application was more efficient in the receiving and delivery of messages, and these experiments showed that JabberEm once again, emerged as the superior application in terms of core functionality performance. With the successful implementation of JabberEm, SendEm and MailEm, it can be concluded that mobile phone platforms can be used as a convergent technology for text based communication.

5.1 Future Work

All of the applications developed as part of this research project could be improved and combined to create a powerful mobile application capable of performing each of the functions of its constituent components. The functionality of MailEm could be greatly enhanced by including support for HTML based e-mails and attachments. SendEm could be improved by making it compatible with Symbian Series 60 devices. One possible way in which this could be done is by making use of a servlet to which the Symbian Series 60 devices would send requests to login, and the servlet would in turn act as a “middle man” in the communication between the device and the Vodacom4me website. By making use of the servlet, the problem with sesssion ID extraction from HTTP headers on Symbian Series 60 devices is eliminated. These applications could also be extended to the Java Card platform, which would then enable lower level mobile devices to run the applications.

Chapter 6. References

Answers.com. “Cellular Telephone”. 2008. Available At: http://www.answers.com/topic/mobile-phone?cat=technology. [Accessed 08-08-2008]

Bedlow, R. “Blurring the Lines”, Mobile World Congress Daily. 2008.

Berkeley. “UC Berkeley Email Stats” . 2000. Available At: http://www2.sims.berkeley.edu/research/projects/how-much-info/internet.html#11. [Accessed 28-08-2008]

Card, S. K., Moran, T. P., Newell, A., The Keystroke-Level Model for User Performance Time with Interactive Systems. Comm. ACM 23, 7. 396-410. 1980.

cellular-news. “Instant Messaging Via Mobile Set to Challenge SMS Traffic”. 2008. Available At: http://www.cellular-news.com/story/30803.php. [Accessed 11-07-2008]

Elkins, M. “MIME Security with Pretty Good Privacy (PGP)”, Internet RFC 2015, October 1996. Available At: http://tools.ietf.org/html/rfc2015. [Accessed 14-05-2008]

Evett, D. “Spam Statistics 2006”. 2006. Available At: http://spam-filter-review.toptenreviews.com/spam-statistics.html. [Accessed 05-09-2008]

Freed, N. and Borenstein, N. “Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies”. Internet RFC 2045, Nov. 1996. Available At: http://www.faqs.org/rfcs/rfc2045. [Accessed 18-05-2008]

Holleis, P. et al. “Keystroke-Level Model for Advanced Mobile Phone Interaction”. To appear in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems 2007 (CHI '07), San Jose, CA, USA, April/May 2007.

Jussawalla, M. (1999) “The impact of ICT convergence on development in the Asian region” in Telecommunication Policy. Published by Elsevier Science. Vol 24. p. 217-234

iLabs Mobile Toolkit. “The Source for Java Technology Collaboration”. 2008. Available At: https://ilabsmobiletoolbox.dev.java.net/. [Accessed 13-05-2008]

Knudsen, J. Kick Butt with MIDP and MSA: Create Great Mobile Applications. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.

Linn, J. “Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures”, Internet RFC 1421, February 1993. Available At: http://tools.ietf.org/html/rfc1421. [Accessed 14-05-2008]

Mail4me. 2008. Available At: http://mail4me.objectweb.org/. [Accessed 02-06-2008]

Mallick, M. *Mobile and Wireless Design Essentials.* John Wiley & Sons. 2003. 117-118.

Nielsen, J. *Heuristic evaluation*. “Usability Heuristics for User Interface Design Inspection Methods”. John Wiley & Sons, New York, NY, 1994.

Rosson, M. B. “Brief HCI Glossary”. 1998. Available At:

http://ei.cs.vt.edu/~cs5714/glossary.html. [Accessed 26-07-2008]

Schwartz, J. and Retford, B. How to Build and SMS Service. O'Reilly, Sebastopol, CA, 2007.

Sun. “Java Card Technology Overview”, Sun Developer Network (SDN). 2002. Available At: http://java.sun.com/javacard/overview.jsp. [Accessed 05-09-2008]

Sun. “Java ME at a Glance”, Sun Developer Network (SDN). 2008. Available At: http://java.sun.com/javame/index.jsp. [Accessed 16-08-2008]

Topley, K. *J2ME in a Nutshell*. O'Reilly, Sebastopol, CA, 2002.

Toshihiro, K. “Usability evaluation based on international standards for software quality evaluation”, NEC Technical Journal Vol.3 No.2. 2008.

University of Colorado. “Understanding IMAP and POP”. Information Technology Services. Available At: http://www.colorado.edu/its/docs/email/imap-pop.html. [Accessed 08-05-2008]

Valdes-Dapena, P. “How's your cell service rate?”. 2003. Available At: http://money.cnn.com/2003/07/31/technology/cellular_survey/index.htm?cnn=yes. [Accessed 13-07-2008]

Virki, T. “Global cell phone use at 50 percent”. 2007. Available At:

http://www.reuters.com/article/technologyNews/idUSL2917209520071129.[Accessed 05-08-2008]

Vodacom. 2008. Available At: http://www.vodacom.co.za. [Accessed 15-05-2008]

Wikipedia.“Short message service”. 2008. Available At:

http://en.wikipedia.org/wiki/Short_message_service. [Accessed 16-07-2008]

Yeng, J. “Usability Assessment of Academic Digital Libraries: Effectiveness, Efficiency, Satisfaction, and Learnability ”, Libri, vol. 55, 2005, pp. 96–121.

Zheng, P. and Ni, L. M. *Smart Phone & Next Generation Mobile Computing.* Morgan Kaufmann Publishers, San Francisco, 2006. 440 - 441, 237 – 239.

Appendix 1 - Time to send and receive messages using Mxit and JabberEm clients during off-peak hours.

	Mxit
Message 1	3.832
Message 2	3.063
Message 3	3.014
Message 4	7.599
Message 5	3.278
Message 6	2.966
Message 7	2.728
Message 8	2.879
Message 9	4.680
Message 10	2.183
Message 11	5.408
Message 12	2.918
Message 13	3.222
Message 14	3.191
Message 15	4.528
Message 16	3.016
Message 17	2.567
Message 18	3.302
Message 19	2.623
Message 20	2.174
Total	**69.171**

	JabberEm
Message 1	4.061
Message 2	3.191
Message 3	2.575
Message 4	3.469
Message 5	1.926
Message 6	3.632
Message 7	2.343
Message 8	2.230
Message 9	3.423
Message 10	3.550
Message 11	2.847
Message 12	2.639
Message 13	2.928
Message 14	3.455
Message 15	2.768
Message 16	2.814
Message 17	3.527
Message 18	2.568
Message 19	3.440
Message 20	3.031
Total	**60.417**

Appendix 2 – Time to send and receive messages using Mxit and JabberEm clients during peak hours.

	Mxit
1	9.270
2	4.638
3	6.919
4	3.302
5	10.663
6	4.151
7	11.607
8	6.928
9	4.247
10	5.247
11	14.008
12	48.856
13	3.880
14	3.536
15	3.935
16	13.520
17	5.311
18	3.768
19	76.458
20	8.526
Total	**248.770**

	JabberEm
1	3.866
2	2.734
3	3.743
4	10.880
5	3.935
6	6.431
7	3.759
8	10.416
9	2.591
10	4.806
11	3.831
12	3.655
13	3.416
14	15.544
15	16.496
16	5.336
17	10.320
18	4.727
19	4.487
20	4.182
Total	**125.155**

Appendix 3 – Keystroke Level Model analysis of sending a message in JabberEm

Using the select button (above the call button)

The following is the generic sequence of actions required to type and send the word “hello' to a contact:

- Move the selected region to the contact you wish to send the message to
- Press the options button
- Press the chat button
- Press the write button
- Type in the word “hello”
- Press the send button

The following is the exact sequence of actions required to type and send the word “hello' to a contact:

1. One key press on the keypad to highlight the contact
2. Finger movement from the direction button to the left select button
3. Press the options button
4. Press the chat button
5. Press the write button
6. Micro attention shift from hotkeys to keypad
7. Finger movement from hotkeys to 'h'
8. Press 'h'
9. Finger movement from 'h' to 'e'
10. Press 'e'
11. Finger movement from 'e' to 'l'
12. Press 'l'
13. Press 'l'

14. Finger movement from 'l' to 'o'
15. Press 'o'
16. Micro attention shift from keypad to hotkeys
17. Mirco attention shift from keypad to screen to check if word was typed correctly
18. Micro attention shift from screen to hotkeys to send the message
19. Finger movement from 'o' to the left select button
20. Press the send button

Thus, the total time required to send a message using the left select button in JabberEm is ***5.82*** seconds. A practical experiment was conducted in order to validate the calculated time, and the time required to send a message worked out to be 4.62 seconds provided that the user if already familiar with the user interface.

Using the center select button

The following is the generic sequence of actions required to type and send the word “hello' to a contact:

- Move the selected region to the contact you wish to send the message to
- Press the middle button
- Press the middle button to begin writing the message
- Type in the word “hello”
- Press the middle button to send the message

The following is the exact sequence of actions required to type and send the word “hello' to a contact:

1. One key press on the keypad to highlight the contact
2. Finger movement from the direction button to the middle button
3. Press the middle button to navigate to the contact chat screen
4. Press the middle button to write the message
5. Micro attention shift from hotkeys to keypad
6. Finger movement from hotkeys to 'h'

7. Press 'h'
8. Finger movement from 'h' to 'e'
9. Press 'e'
10. Finger movement from 'e' to 'l'
11. Press 'l'
12. Press 'l'
13. Finger movement from 'l' to 'o'
14. Press 'o'
15. Micro attention shift from keypad to hotkeys
16. Mirco attention shift from keypad to screen to check if word was typed correctly
17. Micro attention shift from screen to hotkeys to send the message
18. Finger movement from 'o' to the left select button
19. Press the send button

The only difference between using the left select button and the middle button is that the latter doesnt require step 3, which is the pressing of the options button. So the time of a simple hotkey keypress can be deducted from the total time above to produce a new time of ***5.66*** seconds to send a message to a contact using the middle button as a means of execution.

Appendix 4 – Keystroke-Level Model analysis of sending a message in Mxit

<u>Using the chat button (right side of screen)</u>

The following is the generic sequence of actions required to type and send the word “hello' to a contact:

- Move the selected region to the contact you wish to send the message to
- Press the chat button
- Press the middle write button
- Type in the word “hello”
- Press the options button (left side of screen)
- Press the send button

The following is the exact sequence of actions required to type and send the word “hello' to a contact:

1. One key press on the keypad to highlight the contact
2. Finger movement to the right chat button
3. Press the chat button to go to the contact chat screen
4. Micro attention shift from hotkeys to keypad
5. Finger movement from chat button to 'h'
6. Press 'h'
7. Finger movement from 'h' to 'e'
8. Press 'e'
9. Finger movement from 'e' to 'l'
10. Press 'l'
11. Press 'l'
12. Finger movement from 'l' to 'o'
13. Press 'o'

14. Micro attention shift from keypad to screen to check if word was typed correctly
15. Micro attention shift from screen to hotkeys
16. Finger movement to options button (left button)
17. Press the options button
18. Micro attention shift from hotkeys to screen to find send button
19. Micro attention shift to hotkeys
20. Press the send button

The time required to send a message in Mxit using the right chat button and left select button equates to ***5.88*** seconds.

Using the center select button button

The sequence of actions required to send a message using the center select button is the same as the sequence of actions required if using the right chat button, and the left select button. Thus, the time required to send a message using the center select button is ***5.88*** seconds.

Appendix 5 – User evaluation data for learnability

Table 7: Learnability test results for JabberEm

Test Subject	Send Message	Add Contact	Remove Contact	Show offline contacts	Change contact nickname	Change user nickname	Vibrate upon message receival	TOTAL
User 1	5.759	16.007	3.502	4.520	10.544	4.815	3.023	48.17
User 2	9.797	36.035	3.267	7.256	13.176	11.448	8.650	89.63
User 3	9.453	21.773	6.311	5.250	9.682	4.652	4.396	61.52
User 4	6.302	29.313	3.247	5.431	6.743	5.368	2.342	58.75
User 5	5.955	17.949	3.110	4.272	8.227	15.838	7.041	62.39
User 6	7.570	25.986	4.032	3.898	6.452	5.872	2.063	55.87

Table 8: Learnability test results for Mxit

Test Subject	Send Message	Add Contact	Remove Contact	Show offline contacts	Change contact nickname	Change user nickname	Vibrate upon message receival	TOTAL
User 1	3.334	11.216	4.551	3.310	7.615	11.784	5.608	47.42
User 2	14.634	21.266	7.149	62.151	11.397	27.942	14.405	158.94
User 3	7.631	21.383	5.279	6.228	11.114	17.389	11.700	80.72
User 4	8.247	20.240	4.725	52.969	10.122	11.691	38.242	146.24
User 5	8.382	18.892	3.897	18.865	9.872	19.783	16.983	96.67
User 6	4.178	15.998	8.845	8.755	10.431	10.447	14.144	72.8

Appendix 6 – User evaluation data for efficiency

Table 9: Efficiency test results for JabberEm

Test Subject	Send Message	Add Contact	Remove Contact	Show offline contacts	Change contact nickname	Change user nickname	Vibrate upon message receival	TOTAL
User 1	3.127	10.880	2.160	2.927	5.891	4.040	2.375	31.4
User 1	2.800	11.999	2.015	2.535	5.775	3.382	1.839	30.35
AVERAGE	2.9635	11.4395	2.0875	2.731	5.833	3.711	2.107	30.87
User 2	13.413	15.734	17.387	7.141	9.743	12.248	5.927	81.59
User 2	5.488	13.844	3.155	4.315	7.891	13.182	2.650	50.53
AVERAGE	9.4505	14.789	10.271	5.728	8.817	12.715	4.2885	66.06
User 3	6.174	17.289	2.057	3.817	5.867	3.427	3.119	48.19
User 3	6.219	16.165	7.035	3.116	11.933	7.158	3.005	54.63
AVERAGE	6.2	16.73	4.55	3.47	8.9	5.29	3.06	48.19
User 4	4.623	17.631	3.151	2.823	8.271	4.583	3.199	44.28
User 4	3.159	12.127	2.982	2.543	5.391	5.286	2.191	33.68
AVERAGE	3.89	14.88	3.07	2.68	6.83	4.93	2.7	38.98
User 5	5.022	10.708	9.247	4.212	3.917	13.059	2.413	48.58
User 5	4.454	9.870	4.002	2.896	8.308	6.963	2.906	39.4
AVERAGE	4.74	10.29	6.62	3.55	6.11	10.01	2.66	43.99
User 6	7.281	18.765	1.810	1.846	5.184	4.185	1.972	41.04
User 6	5.009	14.750	2.021	2.511	5.920	6.110	2.717	39.04
AVERAGE	6.15	16.76	1.92	2.18	5.55	5.15	2.34	40.04

Average keystroke time to send message in JabberEm = (2.935 + 9.4505 + 6.2 + 3.89 + 4.74 + 6.15)/6 = 5.5609

Average time to perform all core functions (Average of averages) = (30.87 + 66.06 +48.19 + 38.98 + 43.99 + 40.04)/6 = 44.68

Table 10: Efficiency test results for Mxit

Test Subject	Send Message	Add Contact	Remove Contact	Show offline contacts	Change contact nickname	Change user nickname	Vibrate upon message receival	TOTAL
User 1	3.374	8.311	3.463	3.231	8.062	8.455	5.678	40.57
User 1	3.111	8.103	3.535	2.790	7.599	7.695	4.423	37.26
AVERAGE	3.2425	8.207	3.499	3.0105	7.8305	8.075	5.0505	38.92
User 2	22.243	11.184	7.276	14.239	7.763	15.879	10.672	89.26
User 2	9.415	22.091	3.931	6.394	10.510	18.713	9.009	80.06
AVERAGE	15.829	16.637	5.603	10.316	9.136	17.296	9.840	84.66
User 3	7.970	16.233	3.574	5.583	8.830	13.875	7.680	63.75
User 3	7.867	10.877	4.901	4.157	8.402	10.736	6.646	53.59
AVERAGE	7.9185	13.555	4.2375	4.87	8.616	12.3055	7.163	58.67
User 4	6.216	11.774	5.361	6.094	11.997	9.618	9.401	60.46
User 4	6.172	13.675	2.850	7.112	5.471	11.654	6.003	52.94
AVERAGE	6.19	12.72	4.11	6.6	8.73	10.64	7.7	56.7
User 5	6.563	11.266	3.452	3.282	10.783	9.528	9.372	54.25
User 5	6.248	10.875	4.125	3.168	8.762	8.638	10.823	52.64
AVERAGE	6.41	11.07	3.79	3.23	9.77	9.08	10.1	53.44
User 6	4.660	8.868	3.307	18.056	4.284	6.318	9.175	54.67
User 6	3.926	7.171	3.434	4.512	4.933	6.807	5.160	35.94
AVERAGE	4.29	8.02	3.37	11.28	4.61	6.56	7.17	45.31

Average keystroke time to send message in Mxit = (3.2425 + 15.829 + 7.9185 + 6.19 + 6.41 + 4.29)/6 = 7.3133

Average time to perform all core functions (Average of averages) = (38.92 + 84.66 + 58.67 + 56.7 + 53.44 + 45.31)/6 = 56.28

Appendix 7 – User evaluation data for memorability

Table 11: Memorability test results after 2 days for JabberEm

Test Subject	Send Message	Add Contact	Remove Contact	Show offline contacts	Change contact nickname	Change user nickname	Vibrate upon message receival	TOTAL
User 1	2.821	12.334	2.154	2.800	5.815	2.997	2.279	31.2
User 2	11.215	17.298	2.644	4.520	12.171	7.482	3.233	58.56
User 3	5.859	15.118	7.829	5.082	8.742	4.680	3.125	50.44
User 4	6.165	13.298	5.838	3.408	9.894	8.854	5.518	52.98
User 5	6.382	15.437	6.218	3.189	10.742	7.016	4.51	53.49
User 6	5.625	18.834	6.987	3.766	10.928	8.734	4.961	59.84
AVERAGE	6.34	15.39	5.28	3.79	9.72	6.63	3.94	51.08

Table 12: Memorability test results after 2 days for Mxit

Test Subject	Send Message	Add Contact	Remove Contact	Show offline contacts	Change contact nickname	Change user nickname	Vibrate upon message receival	TOTAL
User 1	3.726	8.461	3.400	3.204	8.933	7.468	5.586	40.78
User 2	9.462	23.733	4.437	8.629	16.682	17.280	10.452	90.68
User 3	5.230	14.542	5.172	7.482	7.785	12.025	7.679	59.92
User 4	8.836	17.178	5.324	12.816	10.370	16.911	14.892	86.33
User 5	7.372	18.824	5.637	10.463	9.836	19.389	14.767	86.29
User 6	8.863	15.283	4.264	11.501	8.489	15.540	13.862	77.8
AVERAGE	7.25	16.34	4.71	9.02	10.35	14.77	11.21	73.63