Bluetooth Vibration API for mobile phones

By

Muyowa Mutemwa

2550606

Supervisor	Co-supervised
W. D. Tucker	Prof. I. Venter & M Norman

A thesis submitted in partial fulfillment of
the requirements for the degree of

B.Sc. Honours

University of the Western Cape

Date 3rd November 2008

ABSTRACT

The main idea here is to develop an API that makes a phone vibrate. Once the API is developed, an sms function will be added to make the interface of the application more user friendly to the intended user. The API should be able to connect to the Kiara account when a user name and password are provided. The application would then be paired with the computer using Bluetooth. The choice to use Bluetooth is because it does not cost the intended user anything (it is free) but it has the limitation of distance. In addition a message would be sent to the mobile phone, to show that a new email has arrived.

TABLE OF CONTENTS

Contents

Abstract ..... *ii*

TABLE OF CONTENTS III

ACKNOWLEDGMENTS VI

GLOSSARY VII

CHAPTER 1 1
Introduction *1*
Introduction 1
Background 1
The project 2
Motivation 2
Roadmap (Thesis Layout) 2
CHAPTER 2 3
Users Requirement Document *3*
Introduction 3
Interview Impressions 3
User's view of the problem 3
The problem at hand is: 4
The following are not expected: 5
Summary 5
CHAPTER 3 6
Requirements Analysis Document *6*
Introduction 6
Problem Analysis Preparation 6
The Problem 6
Current existing solution 6
Breaking down the solution 8
Receiving the e-mail 8
Signal send to win32 application 8
Win32 application runs code 8
Code sends a vibrating notification, an SMS 9
SMS received on the mobile phone 9
Testing Methods 9
Summary 9
CHAPTER 4 10
User Interface Specifications *10*
Introduction 10
The Complete User Interface 10
Kiara (instant message program running on the computer). 10
An agent running on the phone. 12
How the user interface behaves 13
How the user interacts with the system 14
Summary 15

CHAPTER 5 16
High level Design 16
Data Dictionary 16
Object 16
Description 16
Class diagrams showing the name, attributes, and methods of each class. 17
The relationship between objects 19
Diagrams representing the solution 20
CHAPTER 6 22
Low Level Design 22
Introduction 22
Inner details of class attributes (data types) 22
Class 22
Attributes (data types) 22
Inner details of class methods (functions) 23
Class Name 23
Method 23
Detailed object (as opposed to class) diagrams for OOD 29
State diagrams 29
Pseudo-code 29
Win32_Application 29
BLuetooth_code 30
BluetoothPMPExampleApp 31
BluetoothPMPExampleDocument 32
BluetoothPMPExampleAppUi 33
DeviceDiscoverer 33
Listener 33
ServiceAdvertiser 33
ServiceDiscoverer 34
Connector 34
BluetoothPMPExampleEngine 34
BluetoothPMPExampleRichTextEditorRTE 34
BluetoothPMPExampleRTEContainer 34
Summary 34
CHAPTER 7 35
User Manual 35
Introduction 35
User Manual 35
1). Start: 35
2). Help 35
3). Exit 35
CHAPTER 8 36
Testing 36
Introduction 36
Reason why test the system 36
Preparation for testing: 36
Planning the Test 36
What are we going to Test? 39
How do we evaluate how exactly the test subjects feel about the software? 40
How I tested the program to prove it works: 43
Testing the product at the DCC in Cape Town. 44

The results of the test at the DCCT in Cape Town. 44
User installation guide 46
Completing the setup: 47
Final tests: 49
Portability: 49
Signal obstruction: 50
CHAPTER 9 51
Problems Encountered During the Project Development *51*
Introduction 51
Problems with obtaining help and Information from an abstract level. 51
The two major problems encountered 52
Sending a message to the cell phone. 52
Conclusion 54
CHAPTER 10 55
Source code *55*
BIBLIOGRAPHY 56
INDEX 53

ACKNOWLEDGMENTS

The author wishes to thank for the following people:

- W D Tucker
- N Norman
- Prof I Venter
- A Henny
- Long Yee

GLOSSARY

DCCT, a Community of Deaf people who help each other through self help. It was founded in 1987 –a non government welfare organization whose aim is to address the needs of Deaf people in the Western Cape.

Mxit, (Instant messaging software**)** is a free instant messaging software application developed in South Africa that runs on GPRS/3G mobile phones with java support and on PCs, using Adobe Flash Player. It allows the user to send and receive text messages to and from PCs that are connected to the Internet and other phones running MXit. These messages are sent and received via the Internet, rather than with standard SMS technology. The user can also exchange messages with online chat communities like MSN Messenger, ICQ and Jabber. Messages were limited to 2 000 characters, but this has been reduced to 1000 to alleviate RAM issues. MXit does not charge for sending and receiving person to person messages but some mobile operators do charge for GPRS/3G data cost although these costs are much cheaper than traditional SMS messages.

Operamini, enables you to take your full web experience and digital lifestyle with you — everywhere you go. Whether you want to access your mail, RSS feeds or bank information, Opera Mini is a quick and secure way to get your data on the go.

API, is a source code interface that an operating system or library provides to support requests for services to be made of it by computer programs.

J2ME, (Java 2 Platform, Micro Edition) is a specification of a subset of the Java platform aimed at providing a certified collection of Java APIs for the development of software for small, resource-constrained devices such as cell phones, PDAs and set-top boxes

Python, is a general-purpose, high-level programming language. Its design philosophy emphasizes programmer productivity and code readability.[2] Python's core syntax and semantics are minimalist, while the standard library is large and comprehensive.

IM, Instant Message.

Chapter 1

INTRODUCTION

Introduction

A Deaf person is someone with a severe auditory impairment; in simple words a deaf person has a disability that impacts their ability to access or process information with their ears. The idea that a person who was deaf could achieve a notable or distinguished status was not common until the latter half of the 18th century, when Abbé Charles-Michel de l'Épée founded the world's first public school for deaf students in Paris.

In Cape Town there is "a community of Deaf people who help each other through self help", the DCCT. It is a non-profitable, non-governmental welfare organization, founded in 1987 whose aim is to address the needs of Deaf people in the Western Cape. Many deaf people do not have a basic education, mainly because many of them where born deaf.

Background

Like any other human being, Deaf people wish to communicate with their loved ones, friends and also do business. One of the most developed communication tool of the 21 century is the cell phone. Deaf people wish to make full use of it as much as possible, but for them there are some functional "Limitations" on a cell phone such as ring tones or audio music functions that they cannot use.. The two main features on the modern cell phone that Deaf people use are the SMS and Vibration functions. So when technology is made that realizes heavily on sounds, Deaf people cannot make use of such technology.

The project

The main idea here is to develop an API that makes a phone vibrate. Once the API is developed, an sms function will be added to make the interface of the application more user friendly to the intended user. The API should be able to connect to the Kiara account when a user name and password are provided. The application would then be paired with the computer using Bluetooth.

Motivation

The motivation comes from the fact that, this technology will indeed improve communication between a hearing person and a Deaf person.

Roadmap (Thesis Layout)

The first chapter of this thesis introduces a brief history of the deaf society, and how it has affected them. Chapter 2 then takes a look at the user requirements require, chapter 3 analyses the requirement obtained in chapter 1. Chapter 4 will look at the User Interface Specification and how the intended user will interact with the solution. Chapter5 will be a High level design, which will be implemented and before any coding is done, chapter 6 will get as close to the code as possible before actual coding is done. Chapter 7 is the User Manual, while chapter 8 is the testing procedures used in the development of this project. In between chapter 6 & 7, the code was implemented. In chapter 8 I take a look at the various problems encountered during the construction of the software, some of which could not be resolved in time for the project completion.

Chapter 2

USERS REQUIREMENT DOCUMENT

Introduction

In this section the "User Requirement Document" (URD) is presented; this is simple known as "The user's view of the problem". The user requirements were obtained here by interviewing the stakeholders. A few questions where asked on 19 February 2008 at the Deaf Community of Cape Town (DCCT). Interview questions were prepared in order to get the users' involved in the project as far as possible. Only one person was interviewed at the DCCT on behalf of the intended users. The deaf community users make use of their phones mainly for SMSs. This indication needs to be included inside forum applications. In this chapter the URD is discussed.

Interview Impressions

From the interview performed on the 19th February 2008, the following were some problems encountered during the interview; the person interviewed is deaf so an interpreter had to be brought in to ease the communication, my interpreter was partially deaf. According to him most of the deaf are born that way or where deaf at a young age, hence a majority of them do not understand the basic common things and are unable to read & write hence communication was a problem and deaf people are just like everyone else they are pretty much interested in technology.

User's view of the problem

On February the 19th, 2008; an interview with Gavin from the DCCT was carried out. The DCCT's main concern is to educate and introduce the Deaf people of Cape Town to technology and to help them make use of computers. In this way they can improve their lifestyles as individuals and go on about their daily activities like normal people. For this project they would like to make use of instant messengers and phones more user friendly to the deaf. Because most deaf people not properly educated, the common or rather basic day to day activities performed on the internet like checking one's e-mail are not well understood and preformed to the technology's full potential, Figure 1.

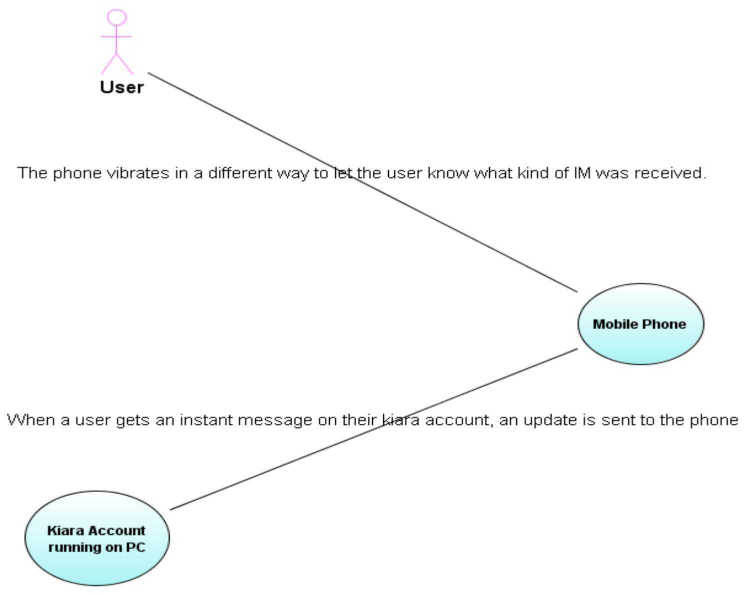

User
The phone vibrates in a different way to let the user know what kind of IM was received.
Mobile Phone
When a user gets an instant message on their kiara account, an update is sent to the phone
Kiara Account running on PC

Figure 1: User's view of the problem

The figure above shows how the user will interact with the system. Suppose Kiara is running on the PC, now whenever a new instant message arrives in the user's account on Kiara, an updates should be sent to the phone.

The problem at hand is:

To help increase the use of instant messengers or communication in general among the deaf; to connect a cell phone user's instant messenger account to their mobile phone using Bluetooth because the cost of using it is free and from the above we get a system, that should give a vibration notification when a new instant message arrives or anything is received in a mail box. See figure1

The following are not expected:

From the final product, the user should not expect to read their instant messages from their mobile phone; or be able to login into the instant messenger account, or send instant messages to another phone and last but not least expect to make use of the internet.

Summary

This concludes the user requirements that is a notification system that will increase the use of their mobile phones and communication between the deaf and non-deaf people. In the next chapter the requirements obtained from the user are analyzed, in the form of a Requirement Analysis Document.

Chapter 3

REQUIREMENTS ANALYSIS DOCUMENT

Introduction

In the previous chapter; the user requirement were listed and discussed. That is the user's point of view; of the problem was discussed. This chapter analyzes the requirements. In this chapter the solution and the possible language to implement in will be discussed, the analysis is from the designer's point of view.

Problem Analysis Preparation

From chapter 1, our targeted users are people from the DCCT, and they normal use of the mobile phone is through text messaging. The solution in this chapter will have to be a system that can be user friendly and usable to the deaf community.

The Problem

While working in the office the user wants to able to receive updates on their instant messenger accounts in this case we shall make use of Kiara (it is a locally available instant messenger), due to the fact that the user is deaf that means that the user can't make use of the notifications that normally come with instant messengers, these can be updates which require the user to be able to hear the sound. Another factor to note is that deaf people are always walking around with their cell phone; that means they can feel their phones vibrate when an incoming message arrives on the phone. So a good idea here would be to make vibration a main part of the solution.

Current existing solution

There are current solutions that exist but are made specifically for a specific Instant Messenger. For example Google has an instant messenger called Google talk which can communicate with cell phones but instead of using Bluetooth it uses a billing system through the cell phone network services provider. This means that when a user would like to receive updates the user has to pay for updates to be send to the phone. The important part to note here is that this solution exists for Google talk, not Kiara. Kiara currently does not have any solutions. The Google talk solution does have its advantages such as the coverage area, where as the solution implemented in this thesis has a radius of 1, 10 & 100 meters; this solution can update the user as long as they have network coverage from their service provider.

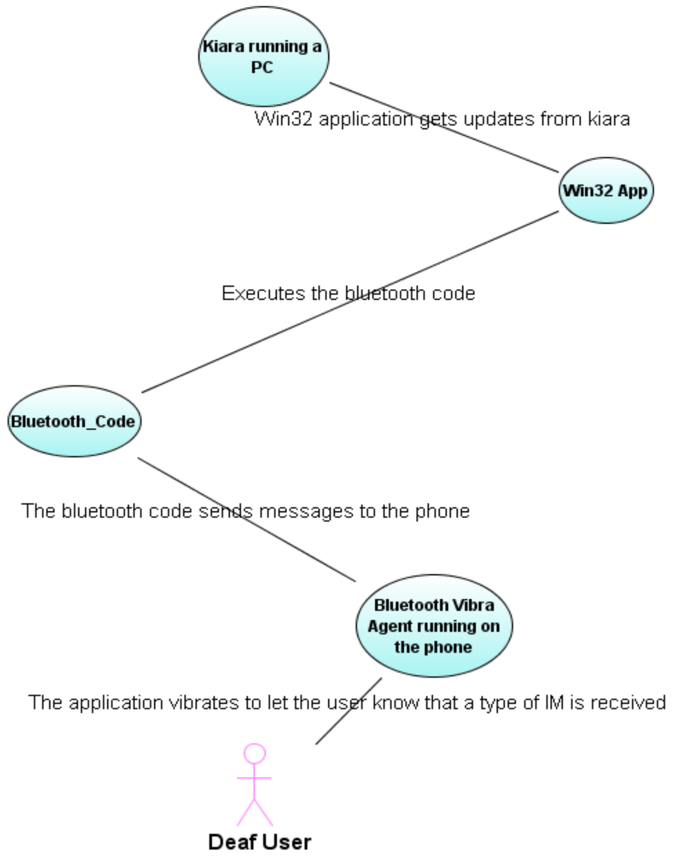

Kiara running a PC
Win32 application gets updates from kiara
Win32 App
Executes the bluetooth code
Bluetooth_Code
The bluetooth code sends messages to the phone
Bluetooth Vibra Agent running on the phone
The application vibrates to let the user know that a type of IM is received
Deaf User

Figure 2: the developer's view of the problem

Suppose the target user has got Kiara running on their PC, and that someone sends our target user an instant message whether it is a voice, text or even video IM. What happens is this would then arrive on the Kiara account on the computer. Now the win32 application would then get an update that a specific IM was received. The win32 application would then execute the Bluetooth code which would then connect sends a message through to the application running on the phone. The application then causes the phone to vibrates to let the Deaf user know that an IM was received.

Breaking down the solution

RECEIVING THE E-MAIL

When one user sends an email to another use, Kiara makes use of its APIs for sending the messages from one account to another. While it calls its specific method that handles this request, the idea is to also include a few lines of code in the same method that will send something to a win32 application. See figure 2.

SIGNAL SEND TO WIN32 APPLICATION

The win32 application will be developed in such a way that it is able to connect to the Google so that when something triggers it; it runs Bluetooth code within itself. See figure 2.

WIN32 APPLICATION RUNS CODE

There will be Bluetooth code within the win32 application. The Bluetooth code will always have the port open for send and receiving information. This Bluetooth code will programmed in C++ or Python depending on which one is able to produce the required results. Python is simpler to understand and code in, its examples have all worked so far, and the only problem is that it is not as popular as C++ which could means it not reliable. While C++ is commonly used, the only problem is it difficult to find what you are looking for and how it can be done. See figure 2.

CODE SENDS A VIBRATING NOTIFICATION, AN SMS

As mentioned above it could be Python or C++ code, either one of them will be able to open an existing virtual port, connect to the phone and the phone vibrates. See figure 2.

SMS RECEIVED ON THE MOBILE PHONE

The sms received on the phone should be like any normal sms; the only difference is that this one is received via Bluetooth. When the sms is received it should cause the phone to vibrate. See figure 2.

Testing Methods

The one way to test this project would be to install the applications on different machines and setup all the necessary connects. Switch the phone's Bluetooth on and log into two Kiara accounts, one being the send and the other the receiver. The send/server Kiara account would then send any message to the receiver/client Kiara account. Once the message is received on the receiver/client, an sms should be send to the phone showing the new status of the Kiara account, in the process causing the phone to vibrate. The sms is received then this means the system works.

Summary

In this chapter the requirement where analyzed from the developer's point of view. We began the analysis by looking at current solutions that are similar to the problem at hand then, alternative solution where considered. Then the solution was broken into small parts and finally testing methods where discussed. In the next chapter the interface is discussed and analyzed.

C h a p t e r   4

USER INTERFACE SPECIFICATIONS

Introduction

In the previous chapter; the user requirement were analyzed and discussed. The solution and the possible language of implement was discussed, the analysis was from the designer's point of view. In this chapter the User interface specification is described. That is a complete description of the API and what the interface looks like to the user.

The Complete User Interface

There are only two interfaces required for this project; one is the Kiara account. See figure 3 below.

KIARA (INSTANT MESSAGE PROGRAM RUNNING ON THE COMPUTER).

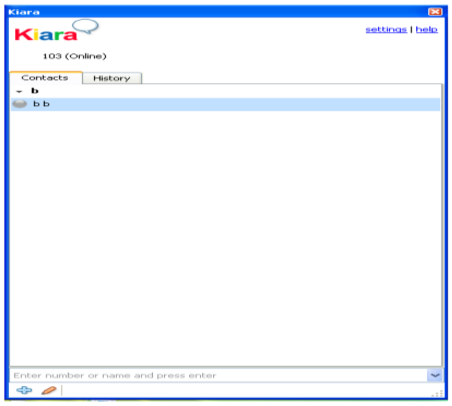
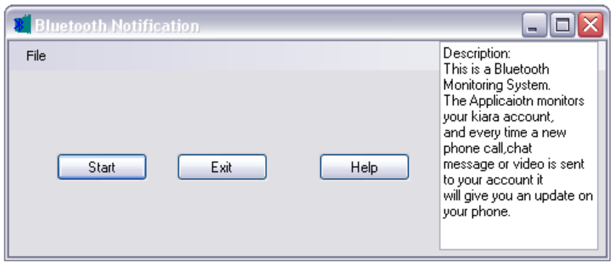


Figure 3: Kiara IM, running on the PC

An example of KIARA account running on a PC. The important thing to note here is that this is not part of the implementation but it is merely integrated into the final solution. Once a new message appears in the inbox as in the above example the update is sent to the mobile *phone. Even though the Kiara is included here, it is not part of the implementation. It is only included because the solution begins from the moment an instant message is received on the Kiara account.*



Description:
This is a Bluetooth
Monitoring System.
The Applicaiotn monitors
your kiara account,
and every time a new
phone call,chat
message or video is sent
to your account it
will give you an update on
your phone.

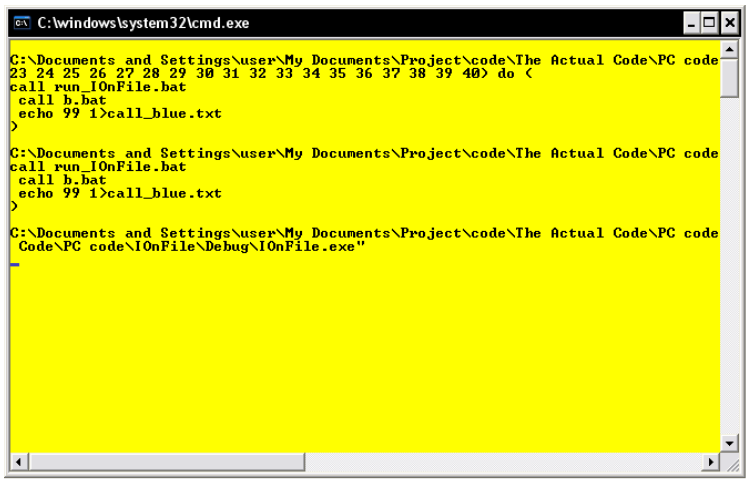
Figure 4: The interface of the software that runs on the PC to monitor an IM.

This is the actual interface used to start the application for monitoring. It is the final interface for software on the computer.

Start: used to start the software to monitor an IM account.

Exit: quit the GUI of the software monitor.

Help: get help on how to use the program or if the user is experiencing difficulties.



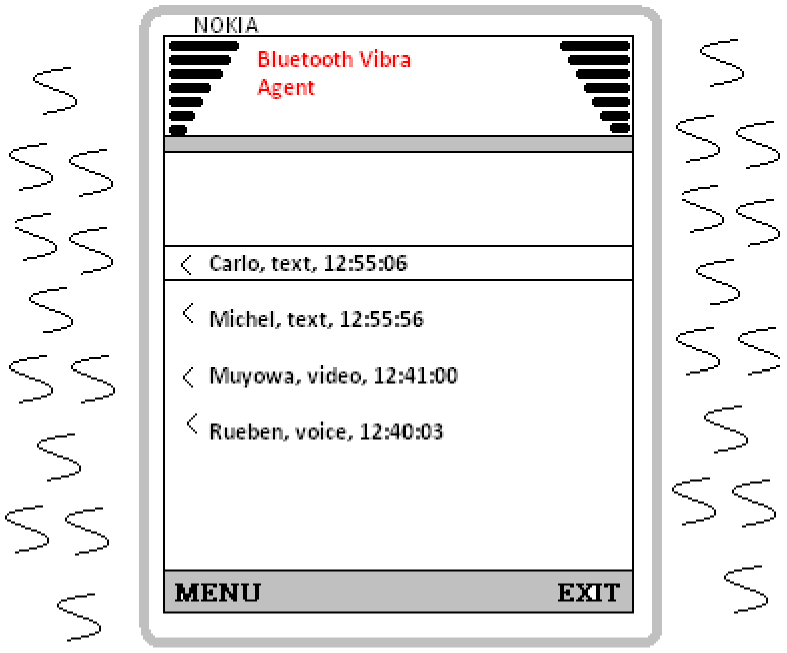
```
C:\Documents and Settings\user\My Documents\Project\code\The Actual Code\PC code
23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40) do (
call run_IOnFile.bat
 call b.bat
 echo 99 1>call_blue.txt
)

C:\Documents and Settings\user\My Documents\Project\code\The Actual Code\PC code
call run_IOnFile.bat
 call b.bat
 echo 99 1>call_blue.txt
)

C:\Documents and Settings\user\My Documents\Project\code\The Actual Code\PC code
 Code\PC code\IOnFile\Debug\IOnFile.exe"
_
```

Figure 5: The background monitoring command line interface.

The command line GUI which shows debug information on what is going on in the background, it is very important because once this window is closed the monitoring stops.

AN AGENT RUNNING ON THE PHONE.

This is considered the main part of the solution that is an agent running on the phone. This agent is simply just a Symbian S60 application running on the phone, see the figure 4; it is a generic notification agent. It receives messages and displays the type of message received and its content; once the message is received the agent will vibrate, see the figure below.



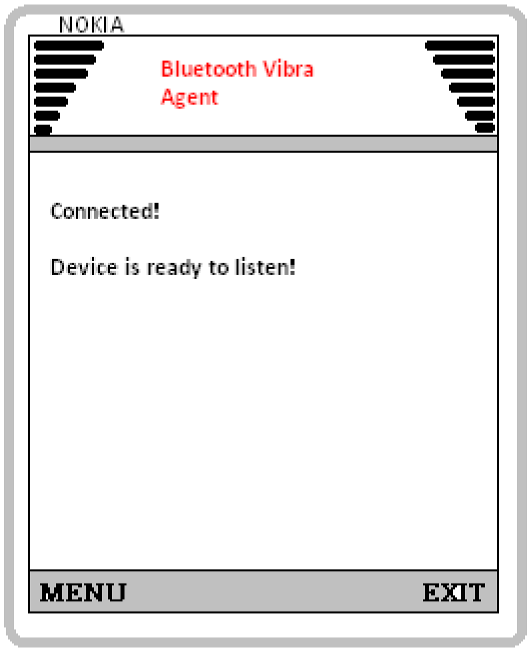
Carlo, text, 12:55:06
Michel, text, 12:55:56
Muyowa, video, 12:41:00
Rueben, voice, 12:40:03

Figure 6: Kiara agent running on the phone. Note that the phone vibrates when a new message is received.

This is known as the main part of the solution, the Kiara agent running on the phone as a Symbian s60 application. This agent only receives updates, from Kiara and cannot send any information itself. Each message that arrives causes the phone to vibrate.

How the user interface behaves

Suppose someone sends the target user an instant message from one Kiara account to our target user's Kiara account and the target user is not sitting down at the machine, but doing something else in the room, the instant message would arrive in the Kiara account that might look like the one in figure 3. Once the instant message is received on the computer which has the target user's account, an update would then be sent to the cell phone. According to figure 4, the phone application is running and it is always in standby mode that is the application is doing nothing. Figure 4 shows that the phone has received some types of messages; Figure 5 shows the contents of the Bluetooth message, which are sender's name, time the Instant Message arrived on the application box.



Connected!

Device is ready to listen!

Figure 7: Contents of the Bluetooth message from PC

If the message is opened, this is an example of the contents of a single message which the Kiara agent received. The contexts are who send the message, time,

How the user interacts with the system

The application will be able to store a certain number of messages because the user might need to see previous messages and clear all the messages.

The application's main menu has the following: clear screen, select and exit; see the figure 6 below.

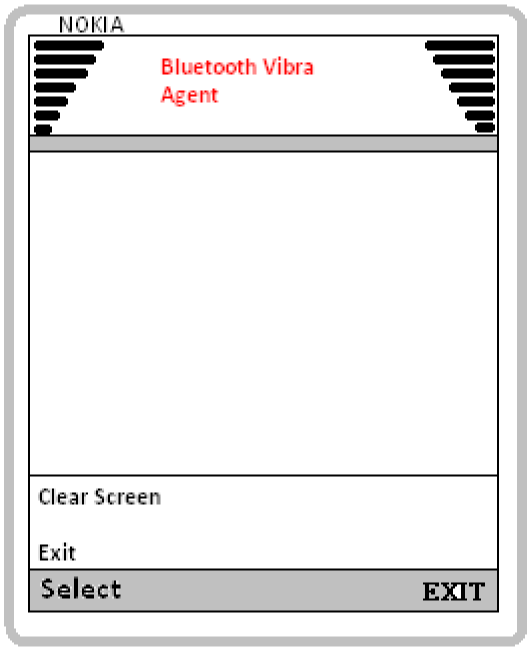


Figure 8: Various avaliable for the Kira agent

Figure 6 show the various options that the users can choice from, when the menu button is press while running the application.

I. Clear Screen – this option is used to delete all messages in the interface inbox, of the application.

II. Exit – this option is used to exit the program.

Summary

In this chapter a closer look at the User Interface was observed. The various screens involved on the cell phone where analyzed. The various options or menus that the user will need to interact with the application where analyzed. In the next chapter, the High Level Design is analyzed. We begin designing the software solution.

Chapter 5

HIGH LEVEL DESIGN

Introduction

In the previous chapter, the user interface specifications were looked at, together with the various functions or options the application had to offer. In this chapter, the high level design (which is also known as the Object Oriented analysis) is analyzed. This is an object-oriented view of the problem that is the various objects required for the solution are defined and analyzed. [http://www.cs.uwc.ac.za/index.php?option=com_content&task=view&id=50&Itemid=37#71].

Data Dictionary

Object	**Description**
Kiara on PC	This is an instant messaging tool which can receive text, video or voice messages. These messages are received in synchronous time, and need to be dwelt with in synchronous time as well. Note this not part of the solution. Even though this is the case it is still a crucial part of the solution, see figures 9 & 12.
Win 32 Application	This is the first part of the solution; this part is responsible for detecting when an instant message is receive on Kiara, and then running code. This Win32 Application is responsible for detecting if an instant message was received, for example it might have been a voice instant message. It is also responsible for running the Bluetooth code. See figures 9 & 12.
Bluetooth Code	This next part of the solution is responsible for the interconnection between the PC and the mobile phone. See figures 10 & 11.
Kiara Agent on mobile phone	This is the last part of the solution; it is responsible for receiving the message, display the message and storing messages. The main function of this part is making the phone vibrate in different ways when different types of

		messages are received that is text, voice or video. See figures 9 & 10.
	Clear Screen	This option is used when the user wants to delete all the messages in the application; these are the messages showing in the application.
	Exit	This option is used to exit the application.

Table 1: This table describes the various objects in the solution.

The table above summarizes the various objects required to make the solution possible.

Class diagrams showing the name, attributes, and methods of each class.

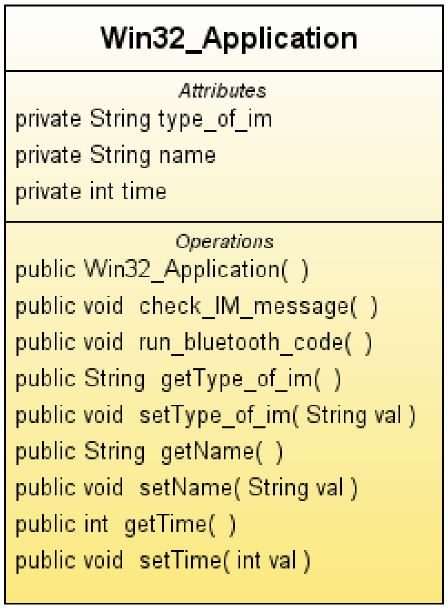

Win32_Application
Attributes
private String type_of_im
private String name
private int time
Operations
public Win32_Application( )
public void check_IM_message( )
public void run_bluetooth_code( )
public String getType_of_im( )
public void setType_of_im(String val)
public String getName( )
public void setName(String val)
public int getTime( )
public void setTime(int val)

Figure 9: the Win32 application object and its attributes.

The win32 application is responsible for connecting Kiara with the rest of the solution; it is the first step in the solution. Its functions include extract the important details from Kiara, determining if an IM was received (for example text, etc) and executing the Bluetooth code.

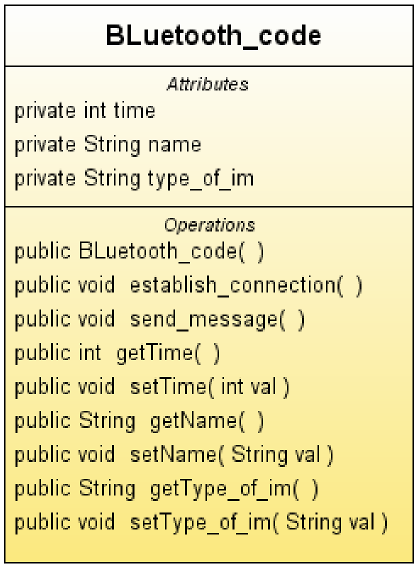

BLuetooth_code
Attributes
private int time
private String name
private String type_of_im
Operations
public BLuetooth_code( )
public void establish_connection( )
public void send_message( )
public int getTime( )
public void setTime(int val)
public String getName( )
public void setName(String val)
public String getType_of_im( )
public void setType_of_im(String val)

Figure 10: Bluetooth code and its attributes.

The Bluetooth code which is set in motion by the win32 application is responsible for connecting the PC and the phone via Bluetooth, and sending the message through.

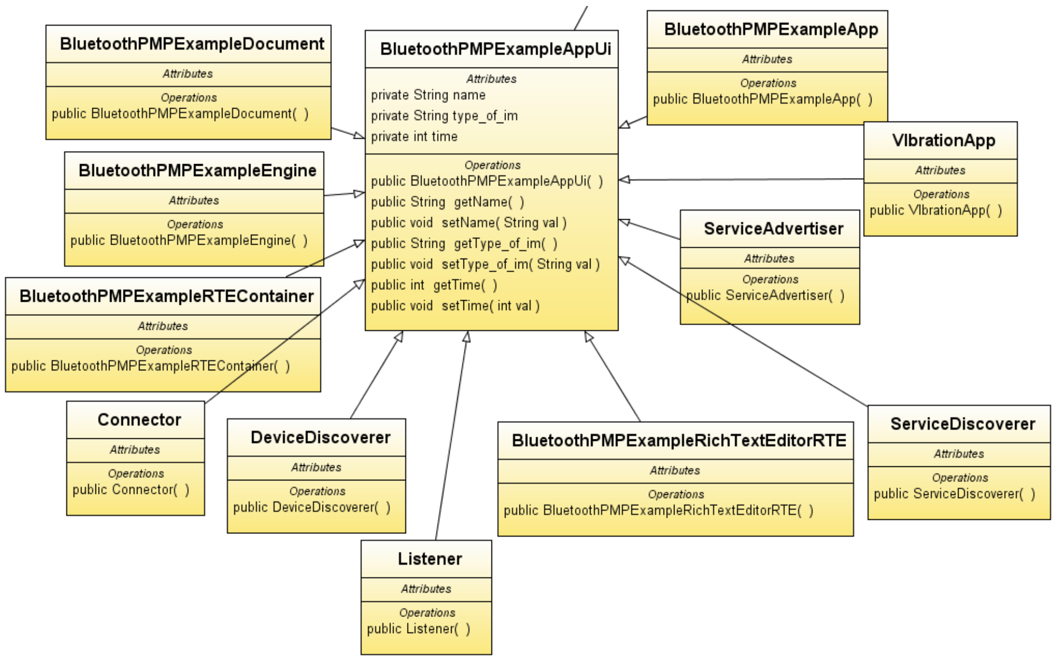

BluetoothPMPExampleDocument
Attributes
Operations
public BluetoothPMPExampleDocument( )
BluetoothPMPExampleAppUi
Attributes
private String name
private String type_of_im
private int time
Operations
public BluetoothPMPExampleAppUi( )
public String getName( )
public void setName(String val)
public String getType_of_im( )
public void setType_of_im(String val)
public int getTime( )
public void setTime(int val)
BluetoothPMPExampleApp
Attributes
Operations
public BluetoothPMPExampleApp( )
VIbrationApp
Attributes
Operations
public VIbrationApp( )
BluetoothPMPExampleEngine
Attributes
Operations
public BluetoothPMPExampleEngine( )
ServiceAdvertiser
Attributes
Operations
public ServiceAdvertiser( )
BluetoothPMPExampleRTEContainer
Attributes
Operations
public BluetoothPMPExampleRTEContainer( )
Connector
Attributes
Operations
public Connector( )
DeviceDiscoverer
Attributes
Operations
public DeviceDiscoverer( )
BluetoothPMPExampleRichTextEditorRTE
Attributes
Operations
public BluetoothPMPExampleRichTextEditorRTE( )
ServiceDiscoverer
Attributes
Operations
public ServiceDiscoverer( )
Listener
Attributes
Operations
public Listener( )

Figure 11: Kiara agent and its attributes.

The Kiara agent or simply Symbian application is responsible also for establishing the connection between the phone and the PC, vibrate the phone is three different ways, store messages and provide an interface for viewing the messages sent from the PC to the phone.

The relationship between objects

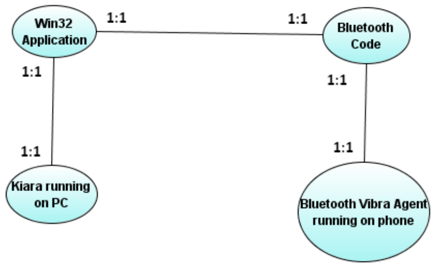


Figure 12: the relationship between the objects.

The diagram shows the various relationships between the objects. The all have a one-to-one relationship between the objects and the maximum number of the objects that each can interact with.

Diagrams representing the solution

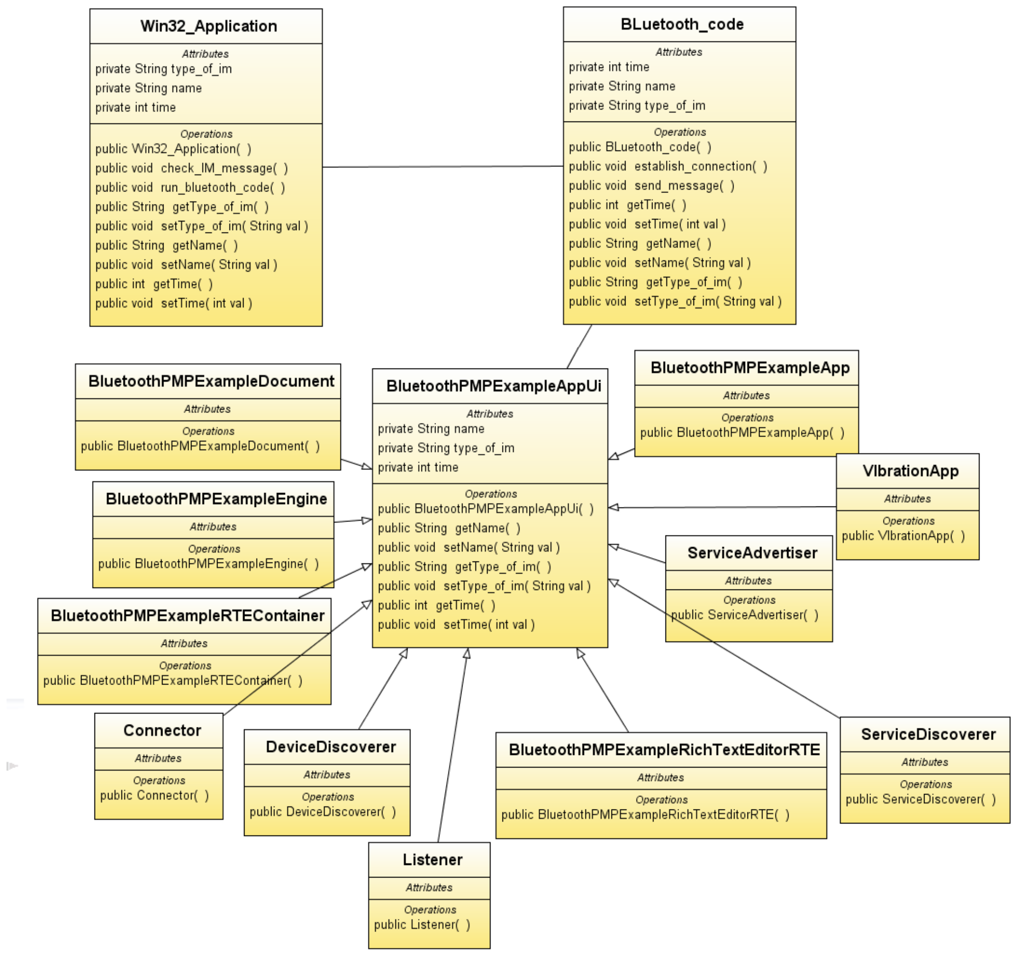

Win32_Application
Attributes
private String type_of_im
private String name
private int time
Operations
public Win32_Application( )
public void check_IM_message( )
public void run_bluetooth_code( )
public String getType_of_im( )
public void setType_of_im(String val)
public String getName( )
public void setName(String val)
public int getTime( )
public void setTime(int val)
BLuetooth_code
Attributes
private int time
private String name
private String type_of_im
Operations
public BLuetooth_code( )
public void establish_connection( )
public void send_message( )
public int getTime( )
public void setTime(int val)
public String getName( )
public void setName(String val)
public String getType_of_im( )
public void setType_of_im(String val)
BluetoothPMPExampleDocument
Attributes
Operations
public BluetoothPMPExampleDocument( )
BluetoothPMPExampleAppUi
Attributes
private String name
private String type_of_im
private int time
Operations
public BluetoothPMPExampleAppUi( )
public String getName( )
public void setName(String val)
public String getType_of_im( )
public void setType_of_im(String val)
public int getTime( )
public void setTime(int val)
BluetoothPMPExampleApp
Attributes
Operations
public BluetoothPMPExampleApp( )
VIbrationApp
Attributes
Operations
public VIbrationApp( )
BluetoothPMPExampleEngine
Attributes
Operations
public BluetoothPMPExampleEngine( )
ServiceAdvertiser
Attributes
Operations
public ServiceAdvertiser( )
BluetoothPMPExampleRTEContainer
Attributes
Operations
public BluetoothPMPExampleRTEContainer( )
Connector
Attributes
Operations
public Connector( )
DeviceDiscoverer
Attributes
Operations
public DeviceDiscoverer( )
BluetoothPMPExampleRichTextEditorRTE
Attributes
Operations
public BluetoothPMPExampleRichTextEditorRTE( )
ServiceDiscoverer
Attributes
Operations
public ServiceDiscoverer( )
Listener
Attributes
Operations
public Listener( )

Figure 13: the various interaction between the classes together with their main functions

The diagram shows show how the classes interact and their main functionality is described. For example the class BluetoothPMPExampleAppUi, has eleven subclasses that inherit from it.

Summary

In this chapter an analysis of the High Level Design was obtained. The various classes involved and how they are expected to interact with each other. The problem was analyzed from an object oriented approach. In the next chapter, the classes discussed in this chapter are analyzed further; we get as close to the actual code as much as possible by making pseudo code.

C h a p t e r   6

LOW LEVEL DESIGN

Introduction

In the last chapter, the classes used to solve the problem where looked at together with their attributes and the relationship between them. In this chapter we take an even closer look at those classes, so much as almost touching the code. This will develop into pseudo-code, which is the closest thing to the code.

Inner details of class attributes (data types)

Class	**Attributes (data types)**
Win32_Application	**String type_of_im**: stores the type of instant message received from Kiara whether it is text, voice or video.
	String name: stores the instant message name of the person or user who sent the message to our target user.
	int time: stores the time the IM was received in the target user's Kiara.
Bluetooth_code	**String type_of_im**: stores the type of instant message received from Kiara whether it is text, voice or video. These are passed on from the Win32_application program.
	String name: stores the instant message name of the person or user who sent the message to our target user. These are passed on from the Win32_application program.

	int time: stores the time the IM was received in the target user's Kiara. These are passed on from the Win32_application program.
BluetoothPMPExampleAppUi	**String type_of_im**: stores the type of instant message received from Kiara whether it is text, voice or video. These are passed on from the Bluetooth_code program.
	String name: stores the instant message name of the person or user who sent the message to our target user. These are passed on from the Bluetooth_code program.
	int time: stores the time the IM was received in the target user's Kiara. These are passed on from the Bluetooth_code program.

Table 2: A description of the attributes of each of the main domain classes.

The table takes a closer look at the classes' attributes involved in the solution. The data types of each attributes are analyzed in more detail to remove and future problems thus defining the solution more clearing.

Inner details of class methods (functions)

Class Name	**Method**
Win32_Application	**public Win32_Application()**: this method initializes all the necessary variables and put everything in place.
	public void check_IM_message(): this method waits for input from Kiara, when something happens in Kiara a return is

	invoked. It returns the message details of a new IM.
	public void run_bluetooth_code(): this method just initiates the command line to run Bluetooth code.
	public String getType_of_im(): returns the types of IM.
	public setType_of_im(String val): changes the types of IM from an old value to a new one or from null to an existing one.
	public String getName(): returns the name of the person who sent the IM.
	public setName(String val): changes the name of the person who sent the IM from an old value to a new one or from null to an existing one.
	public String getTime(): return the time at which the IM was received.
	public setTime(String val): changes the time the IM was sent from an old value to a new one or from null to an existing one.
Bluetooth_code	**public Bluetooth_code()**: initializes all

	the variables and put everything in place.
	public void establish_connection(): this method is responsible for establishing the connection between the PC and the mobile phone.
	public void send_message(): this methods then send the mobile phone.
	public String getType_of_im(): returns the types of IM.
	public setType_of_im(String val): changes the types of IM from an old value to a new one or from null to an existing one.
	public String getName(): returns the name of the person who sent the IM.
	public setName(String val): changes the name of the person who sent the IM from an old value to a new one or from null to an existing one.
	public String getTime(): return the time at which the IM was received.
	public setTime(String val): changes the time the IM was sent from an old value to

	a new one or from null to an existing one.
BluetoothPMPExampleAppUi	**public Kiara_Agent()**: this method initializes all the attributes and variables need locally.
	public String getType_of_im(): returns the types of IM.
	public setType_of_im(String val): changes the types of IM from an old value to a new one or from null to an existing one.
	public String getName(): returns the name of the person who sent the IM.
	public setName(String val): changes the name of the person who sent the IM from an old value to a new one or from null to an existing one.
	public String getTime(): return the time at which the IM was received.
	public setTime(String val): changes the time the IM was sent from an old value to a new one or from null to an existing one.
BluetoothPMPExampleApp	**Public BluetoothPMPExampleApp ()**: initializes all the variables and sets up the

	application. Draws the canvas.
BluetoothPMPExampleDocument	**public BluetoothPMPExampleDocument ()**: contains the include files, required for the application to run successfully. Documents errors in a log file.
BluetoothPMPExampleEngine	**public BluetoothPMPExampleEngine ()**: when an input is received this method searches for the relevant class or method with a class to call.
VibrationApp	**public VibrationApp()**: this is the crucial part of the solution, the class that make the phone vibrate. This where the important code and most of the time hidden from users, due to sensitivity. It can ruin a device if not implemented properly.
ServiceAdvertiser	**Public ServiceAdvertiser ()**: This class advertises the availability of the remote device together with the services it offers.
BluetoothPMPExampleRTEContainer	**Public BluetoothPMPExampleRTEContainer ()**: this class draws the actual area where the text is displayed.
Connector	**Public Connector ()**: connection between

	a device with Bluetooth and this application is determined by this class.
DeviceDiscoverer	**Public DeviceDiscoverer ()**: this class is used to discover other remote devices, before connection occurs.
BluetoothPMPExamplePMPExampleRichTextEditorRTE	**Public ~RichTextEditorRTE ()**: this class is used to draw text on the container.
Listener	**Public Listener ()**: this class listens for any remote device that wants to communicate using Bluetooth and checks if the services are in line with its own so that they can connect using the Bluetooth stack protocol.
ServiceDiscoverer	**Public ServiceDiscoverer ()**: this class is used to discover the service on a device that wants to connect to it. if they do not have similar services then no connection occurs.

Table 3: A description of the methods of each of the classes.

The above table, table 3 describes the method used or required for the solution to work.

Detailed object (as opposed to class) diagrams for OOD

State diagrams

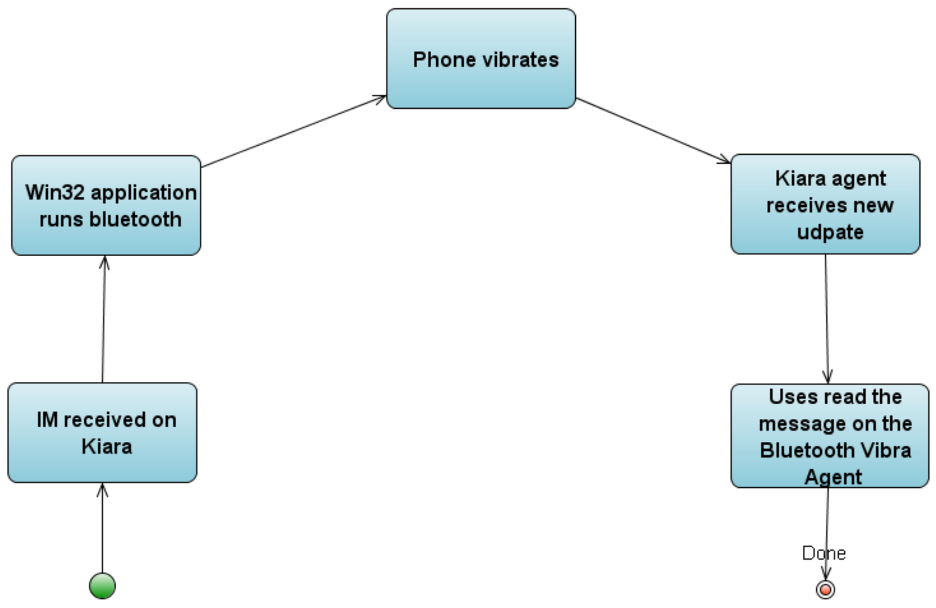

Phone vibrates
Win32 application runs bluetooth
Kiara agent receives new udpate
IM received on Kiara
Uses read the message on the Bluetooth Vibra Agent
Done

Figure 14: shows the various states of the solution.

The above figure shows how the system move from the first state which is the one that starts with a green dot and ends up with the red double circular dot.

Pseudo-code

WIN32_APPLICATION

```
Class Win32_Application {
      String type_of_im

      String name

      int time

    method Win32_Application() {
        set name, type_of_im, time = 0
        while(true)
           get message from Kiara application
```

```
            sleep(1)
    }

    method void check_IM_message() {
        if(message != null)
            run Bluetooth code()
    }

    method void run_bluetooth_code() {
        use a command line to run the Bluetooth code
    }

    method String getType_of_im() {
        return type_of_im
    }

    method void setType_of_im(String val) {
        this.type_of_im <- val
    }

    method String getName() {
        return name
    }

    method void setName(String val) {
        this.name <- val
    }

    method int getTime() {
        return time
    }

    method void setTime(int val) {
        this.time <- val
    }
}
```

BLUETOOTH_CODE

```
Class BLuetooth_code {
      int time

      String name

      String type_of_im

    method BLuetooth_code() {
       initialize time, name, type_of_im <- null
       while(true)
          get message <- args[1]            //this is done by
running this
```

```
                                              //code with
parameter
          Sleep(1)
    }

    method void establish_connection() {
        if(tray == null)   //no devices stored by the computer
            scan for devices
    }

    method void send_message() {
         invoke the send(message) API
    }

    method int getTime() {
        return time
    }

    method void setTime(int val) {
        this.time = val
    }

    method String getName() {
        return name
    }

    method void setName(String val) {
        this.name = val
    }

    method String getType_of_im() {
        return type_of_im
    }

    method void setType_of_im(String val) {
        this.type_of_im = val
    }
}
```

BLUETOOTHPMPEXAMPLEAPP

```
Class Kiara_Agent {
      String name

      String type_of_im

      int time

    method Kiara_Agent() {
        while(true)
        get message from PC
        if(message != null)
           set values to a message structure
        **sleep(1)**
        **pass on control to the next class**
  -…

    }

    method String getName() {
        return name
    }

    method void setName(String val) {
        this.name = val
    }

    method String getType_of_im() {
        return type_of_im
    }

    method void setType_of_im(String val) {
        this.type_of_im = val
    }

    method int getTime() {
        return time
    }

    method void setTime(int val) {
        this.time = val
    }
}
```

BLUETOOTHPMPEXAMPLEDOCUMENT

```
Class Kiara_AgentDocument extends Kiara_Agent {
    method Kiara_AgentDocument() {
        check for error while application is running
        store errors in a log file
    }
}
Class VIbrationApp extends Kiara_Agent {
    method VIbrationApp() {
        If(message.type == text)
            start.vibrate(time1, intensity1);
            release.vibrate()
            pop.vibrate()
        else(message.type == voice)
            start.vibrate(time2, intensity2);
           release.vibrate()
           pop.vibrate()
        else(message.type == video)
           start.vibrate(time3 intensity3);
           release.vibrate()
           pop.vibrate()
        else
           default

    }
}
```

BLUETOOTHPMPEXAMPLEAPPUI

```
 Class Kiara_AgentAppUi extends Kiara_Agent {
    method Kiara_AgentAppUi() {
        handle the various commands entered by the user
        setup the menu option handle    //case statement for
                                        //the menu
        Case connect:
            Browse for device and connect
        Case refresh
            Refresh the canvas
        Case delete all message
            delete the file containing all the messages
            recreate the file
        Case About:
            Print the support details
        Case Help:
            Suspend the application
            Call the nokia library
        Case exit:
            Call exit method
            Pop all the stacks
            Exit the application
        Case default:
           Print error 21               //nokia's default
                                        //error message for
                                        //access denied.
        calls the necessary classes
    }
}
```

DEVICEDISCOVERER

```
public class DeviceDiscoverer extends BluetoothPMPExampleAppUi
{
    public DeviceDiscoverer() {
    }
}
```

LISTENER

```
public class Listener extends BluetoothPMPExampleAppUi {
    public Listener() {
    }
}
```

SERVICEADVERTISER

```
public class ServiceAdvertiser extends
BluetoothPMPExampleAppUi {
    public ServiceAdvertiser() {
    }
}
```
SERVICEDISCOVERER

```
public class ServiceDiscoverer extends
BluetoothPMPExampleAppUi {
    public ServiceDiscoverer() {
    }
}
```
CONNECTOR

```
public class Connector extends BluetoothPMPExampleAppUi {
    public Connector() {
    }
}
```
BLUETOOTHPMPEXAMPLEENGINE

```
public class BluetoothPMPExampleEngine extends
BluetoothPMPExampleAppUi {
    public BluetoothPMPExampleEngine() {
    }
}
```
BLUETOOTHPMPEXAMPLERICHTEXTEDITORRTE

```
public class BluetoothPMPExampleRichTextEditorRTE extends
BluetoothPMPExampleAppUi {
    public BluetoothPMPExampleRichTextEditorRTE() {
    }
}
```
BLUETOOTHPMPEXAMPLERTECONTAINER

```
public class BluetoothPMPExampleRTEContainer extends
BluetoothPMPExampleAppUi {
    public BluetoothPMPExampleRTEContainer() {
    }
}
```

Summary

In this chapter the pseudo code required to understand the solution was developed. The various class attributes and method were explained in detail. In the next chapter, we implement the solution based on the pseudo code obtained here. Document code will be produced in the next chapter.

Chapter 7

USER MANUAL

Introduction

In the previous chapter we took a look at the low level of the design, and the software was implemented between chapter 6 and this chapter. In this chapter we take a look at the User Manual. The User Manual can be used to give guidance to a user who wishes to make use of the Bluetooth Notification System.

User Manual

1). Start
2). Help
3). Exit

1). START:

To begin the monitoring process of an IM such as Kiara; the user has to press the Start Button. The windows command prompt appears which the user should minimize and leave it running on the task bar. If the user closes the windows command prompt the program will stop and hence the monitoring will stop as well. The user can then choose to minimize the Bluetooth Notification main window as well.

2). HELP

If the user is experiencing problems or has some FAQs then the user can first check this User Manual and the installation guide for any steps that the user might have skipped or caveats that the program has reached.

3). EXIT

When the user clicks this option the program exits and the monitoring stops.

Chapter 8

TESTING

Introduction

In the previous chapter we took a look at the User Manual, and the various options the user can click on the computer. In this chapter we take a look at the various ways of testing the system to ensure that meets its primary requirement and that it is robust.

Reason why test the system

To evaluate whether the software works or not, I carried out three types of tests: The first being testing with the actual intended users of the software (at the DCCT); the other with a local students who use Instant messaging services everyday; and lastly I performed stress testing to see how the software behaviors overtime and in stressful conditions together with portability testing.

PREPARATION FOR TESTING:

Things to note:

1). Testing should be interactive.

2). User centered testing breaks down the wall between designer and user. 3). Allows the designer to see how the users perform tasks in the real world.

4). Testing brings about an improved product.

5). The aim of the testing is to uncover pit falls, before redesigning and evaluating the system.

Planning the Test

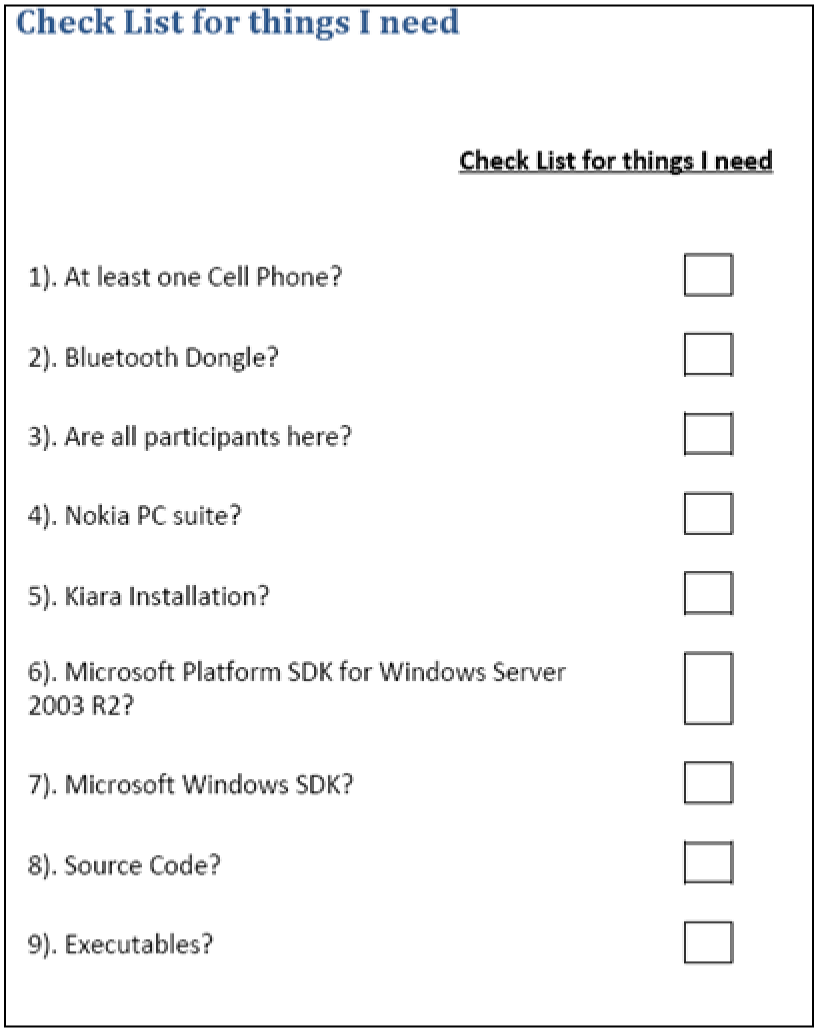
1). 3 - 4 people to test the product. No incentives.

2). The tests should last around five minutes.

3). the location should be the honors lab or the net lab and the DCCT in Cape Town.

4). There will be a Test Script, which will be followed during the testing.

5). Pre-Test Check List, to make sure that we have everything we need before testing commences; see figure 15.


Check List for things I need

Check List for things I need

1). At least one Cell Phone? ☐

2). Bluetooth Dongle? ☐

3). Are all participants here? ☐

4). Nokia PC suite? ☐

5). Kiara Installation? ☐

6). Microsoft Platform SDK for Windows Server 2003 R2? ☐

7). Microsoft Windows SDK? ☐

8). Source Code? ☐

9). Executables? ☐

Figure 15: check list to make sure we have everything.

6). Do not tell the people evaluating the system what to do...

7). The user interface will not be explained to the users this is to allow them to figure it out.

What are we going to Test?

1). we are going to test the notification feature when the user receives some an IM in Kiara.

12). things to note is that the phone will not vibrate!

3). Three Scenarios:

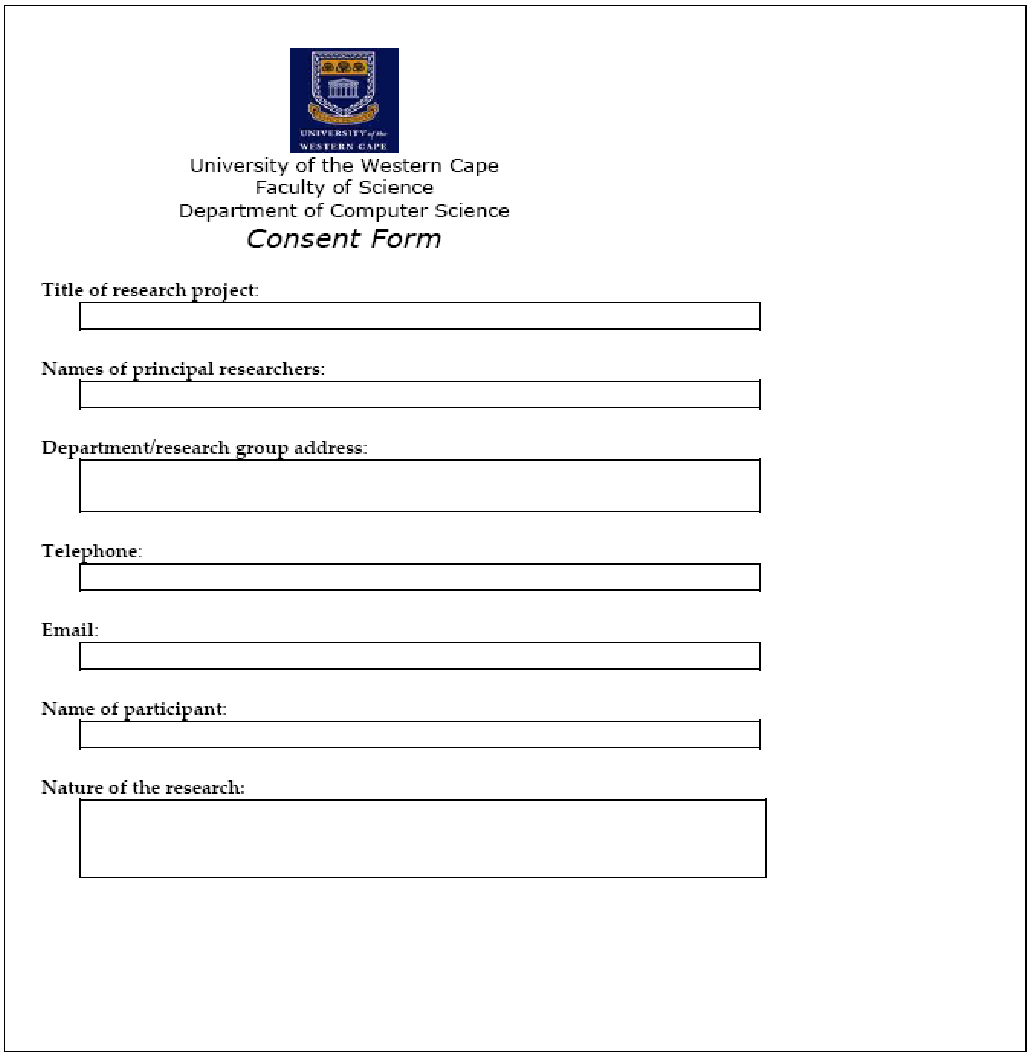
-Suppose a user is at home chatting to a friend of theirs using an IM (Kiara), now then something comes up like the user gets thirsty and goes to kitchen to get a glass of water... A lot of things happen in the kitchen that might cause the user to forget that they were chatting to a friend. Suddenly the phone lights up and the user sees that He has received a new IM; and returns in time to carry on the conversation.

-Suppose our user is working in an office and happens to be working with someone in another place; not in the same location. Suppose that the two are using an IM, to communicate with each other. And suppose that they encounter a certain difficulty and the other employee in the other location wants to try a different approach, will let the user with the Bluetooth Notification System known when she is done. The user with the Bluetooth Notification System, decides the other collogue might take long so she decides to go over to her friend in the same office to chat to the person, and while she is there the other employee responds, the Bluetooth Notification System lights this user's phone and the user can return to their desk to continue with their work in time.

-Two friends are chatting using IM, when the user's friends internet connection goes down for reasons unknown... the user decides to do watch TV for while... While watching TV the user realizes that the phone's light goes on and check to see that new IM has arrived, the user returns to the computer to continue chatting to the friend.

HOW DO WE EVALUATE HOW EXACTLY THE TEST SUBJECTS FEEL ABOUT THE SOFTWARE?

1). Legalities and Consent form; see figure 16 and 17.



University of the Western Cape
Faculty of Science
Department of Computer Science

Consent Form

Title of research project:

Names of principal researchers:

Department/research group address:

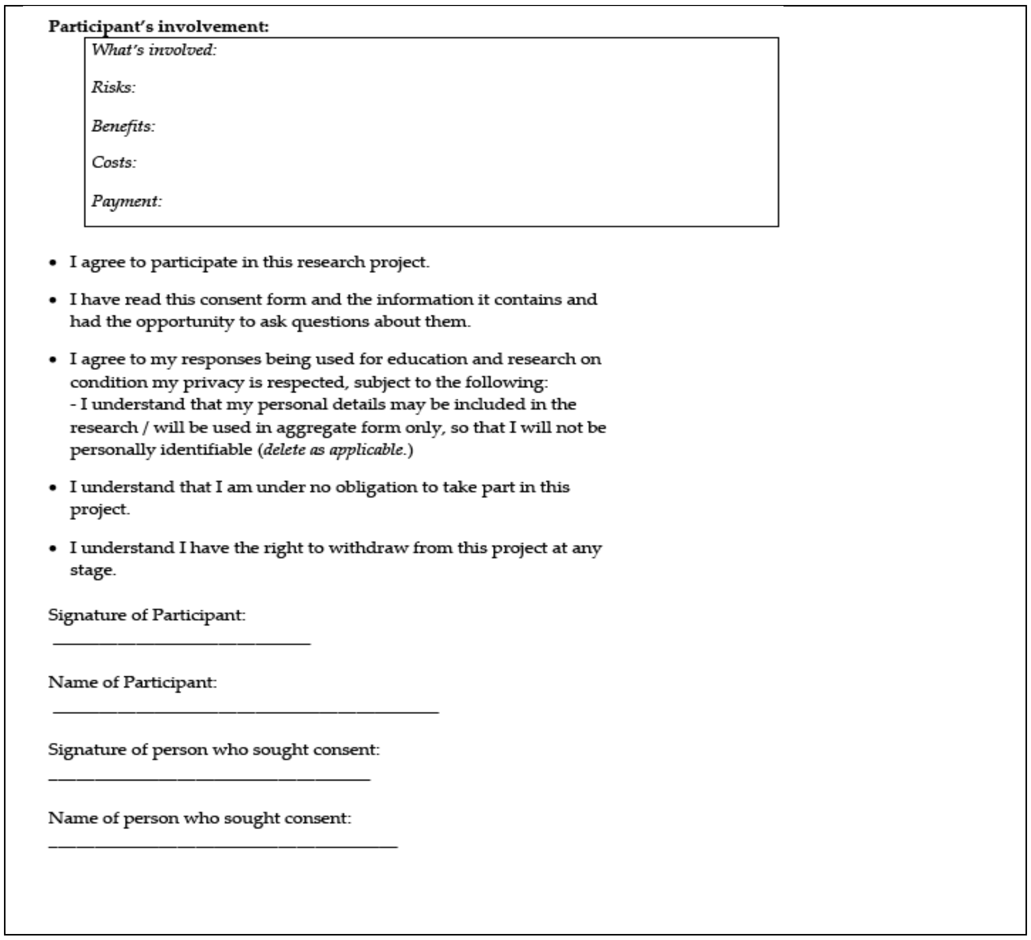
Telephone:

Email:

Name of participant:

Nature of the research:

Figure 16: part 1 of the consent form.


Participant's involvement:

What's involved:

Risks:

Benefits:

Costs:

Payment:

- I agree to participate in this research project.
- I have read this consent form and the information it contains and had the opportunity to ask questions about them.
- I agree to my responses being used for education and research on condition my privacy is respected, subject to the following:
  - I understand that my personal details may be included in the research / will be used in aggregate form only, so that I will not be personally identifiable (*delete as applicable.*)
- I understand that I am under no obligation to take part in this project.
- I understand I have the right to withdraw from this project at any stage.

Signature of Participant:
____

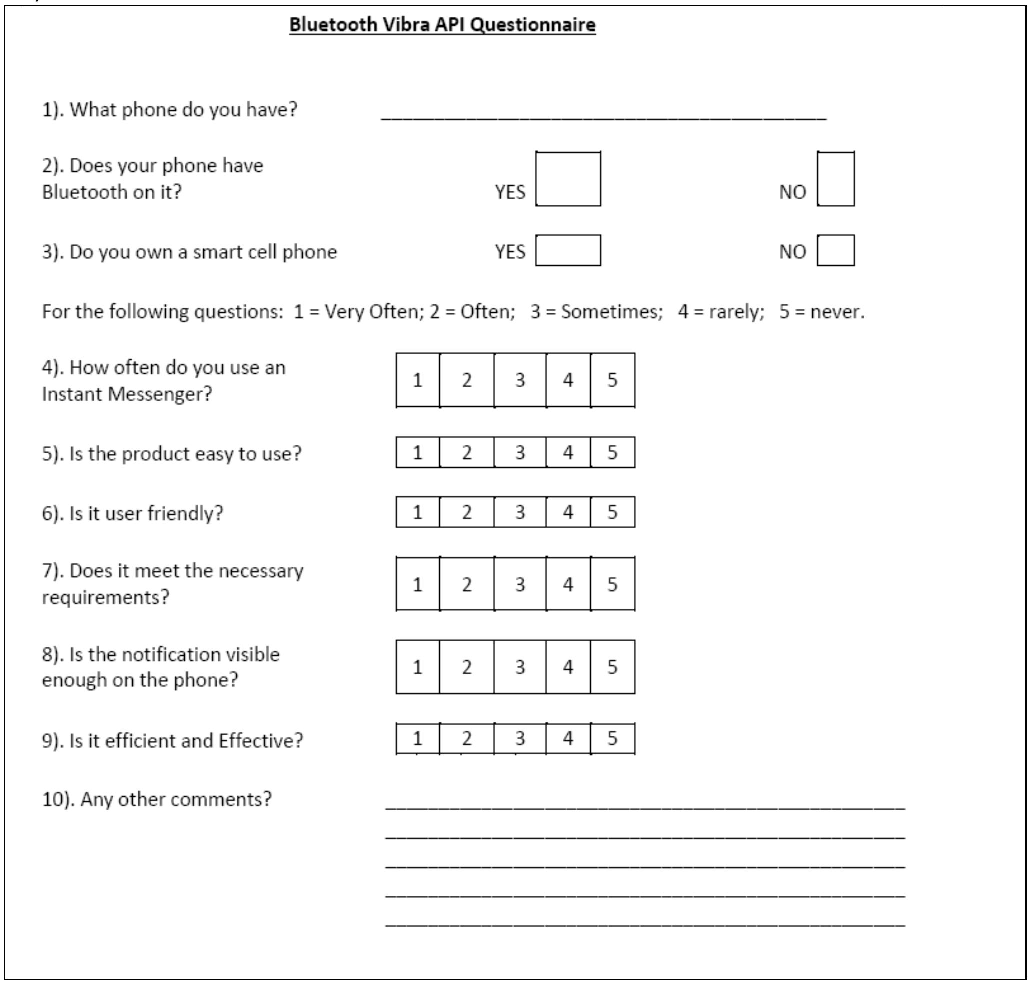
Name of Participant:
____

Signature of person who sought consent:
____

Name of person who sought consent:
____

Figure 17: part 2 of the consent form.

2). Questionnaires.



Bluetooth Vibra API Questionnaire

1). What phone do you have? ____

2). Does your phone have Bluetooth on it? YES ☐ NO ☐

3). Do you own a smart cell phone YES ☐ NO ☐

For the following questions: 1 = Very Often; 2 = Often; 3 = Sometimes; 4 = rarely; 5 = never.

4). How often do you use an Instant Messenger?	1	2	3	4	5
5). Is the product easy to use?	1	2	3	4	5
6). Is it user friendly?	1	2	3	4	5
7). Does it meet the necessary requirements?	1	2	3	4	5
8). Is the notification visible enough on the phone?	1	2	3	4	5
9). Is it efficient and Effective?	1	2	3	4	5

10). Any other comments? ____
____
____
____
____

Figure 18: the questionnaire given to users to evaluate the software after testing the product.

3). Facial expressions from the users.

How I tested the program to prove it works:

The first thing to do is to check whether the software works that is every time a new phone call arrives, the software should update the phone on proceeding. Secondly that the computer doesn't send unnecessary updates of if there are no phone calls.

Testing the product at the DCC in Cape Town.

I had three participants to help me out. First they signed the consent forms see figure 16 and 17. Then they each sat down and evaluated the software one at a time. All three participants where male, each they had the same cell phone which is the Samsung SGH - D900I. They where deaf as well, but one could speak and read lip, communication problems where encountered but we worked through it eventually.

The results of the test at the DCCT in Cape Town.

The result given here is more general, and reflects what all participants mainly went through; and their general feeling towards the software idea and the way the software works.

Since all participants had the same phone, I had to borrow my phone to one of them to use temporarily; this was to prove that the software works across different phone to the intended users. According to the questionnaire these is the general feel I got from the users.

1). What phone do you have.

Even though they all had the same phone, two had the Samsung SGH – D900I, and the other participant took mine which is a Nokia E51.

2). Does your phone have Bluetooth on it?

Yes.

3). Do you own a smart phone?

Yes.

4). How often do you use an Instant Messenger?

They use IM generally moderate – often.

5). Is the product easy to use?

They found the program definitely easy to use.

6). Is it user friendly?

The found the program moderately user friendly to use.

7). Does it meet the necessary requirements?

They found that program does meet its primary requirement which is to give notification but they expected more: they would have loved it to vibrate or give the context of the IM which is received on the computer.

8). Is the notification visible enough on the phone?

They generally found that the notification was visible enough.

9). Is it efficient and effective?

They found it generally moderate on this part.

10). Finally and comments and questions:

-They would love to hear more about Bluetooth product because they don't cost a thing.

-More Bluetooth chat related products.

-they wanted the product to vibrate because they won't be looking at the phone, the whole time.

-if Bluetooth could increase it range of communication from 10 meters to more.

User installation guide

For the product to work on your machine you would need to install the following softwares to make sure they product works without any glitches, and smoothly. If you wish to install the following software or need help on them, please visit the sites provide.

Please note that user is not presented here how to install the various softwares mentioned:
1). Kiara
http://softbridge.uwc.ac.za/trac
2). Microsoft Platform SDK for Windows Server 2003 R2 - http://www.microsoft.com/downloads/details.aspx?familyid=484269E2-3B89-47E3-8EB7-1F2BE6D7123A&displaylang=en
3). Nokia PC Suite
http://europe.nokia.com/pcsuite
4). Microsoft Visual Basic 2005 Express Edition

http://www.microsoft.com/Express/VB/ and follow the instruction on how to obtain it, or just purchase it from you r location computer dealer shop.

For physical tools you would need are:

5). Bluetooth Dongle – You can obtain one and at any computer shop, but I recommend the X-MICRO Bluetooth 2; mainly because it doesn't need drivers to work on any Microsoft Windows XP service pack 1 with usb ports, or higher.

Figure 19: Recommended Bluetooth dongle

6). Smart mobile phone – You could use any phone with Bluetooth capabilities, but I recommend a Nokia Smart phone, mainly because that is what the software was intended for (I recommend any phone that can read text files and has Bluetooth).

Completing the setup:

Once the first 3 are installed and the user has a dongle and a smart phone, the user can now copy the application into the right place.

1). Create the folder with it directories:

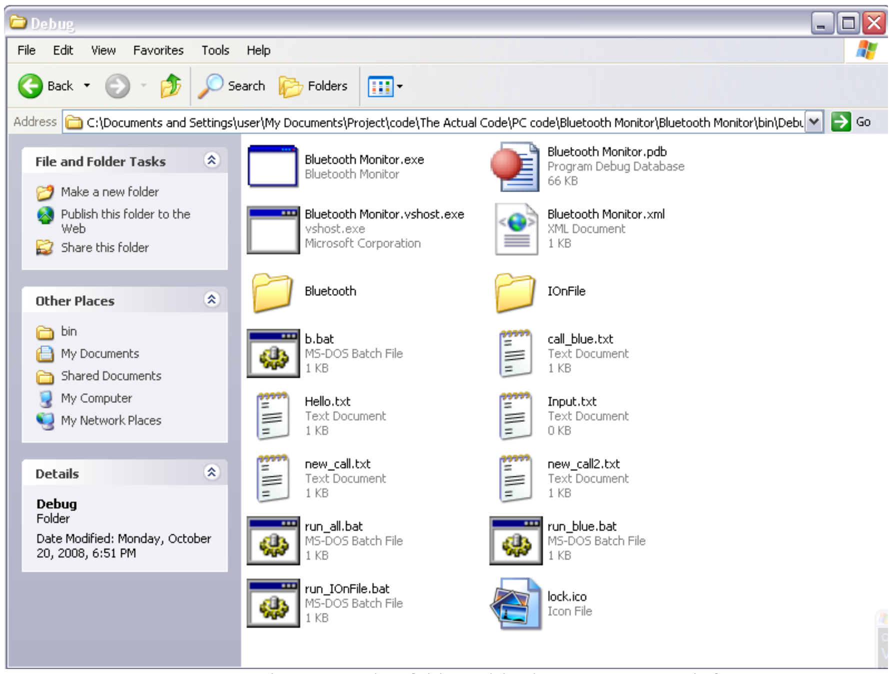
On the C drive in the "My Documents" folder create the following folders and directories.

Bluetooth Monitor \

You can then dump the source code you obtained from the CD here, in this folder.

2). Upon completion, in the "PC code\" folder navigate your way to the following directory:

~\Bluetooth Monitor\Bluetooth Monitor\bin\Debug\ {see figure 20}



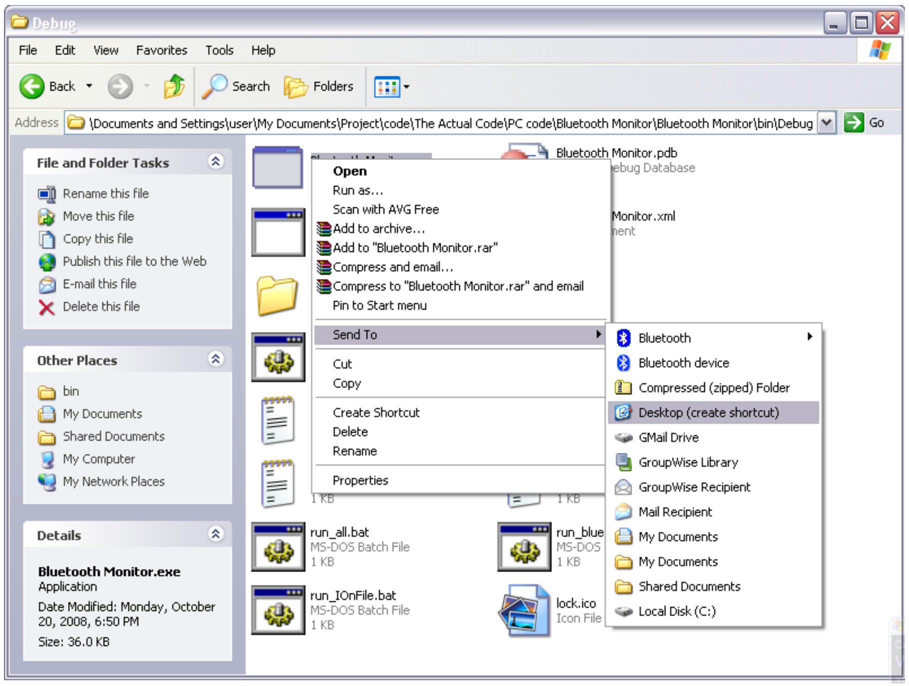
C:\Documents and Settings\user\My Documents\Project\code\The Actual Code\PC code\Bluetooth Monitor\Bluetooth Monitor\bin\Debu

Debug
Folder
Date Modified: Monday, October 20, 2008, 6:51 PM

Bluetooth Monitor.exe Bluetooth Monitor	Bluetooth Monitor.pdb Program Debug Database 66 KB
Bluetooth Monitor.vshost.exe vshost.exe Microsoft Corporation	Bluetooth Monitor.xml XML Document 1 KB
Bluetooth	IOnFile
b.bat MS-DOS Batch File 1 KB	call_blue.txt Text Document 1 KB
Hello.txt Text Document 1 KB	Input.txt Text Document 0 KB
new_call.txt Text Document 1 KB	new_call2.txt Text Document 1 KB
run_all.bat MS-DOS Batch File 1 KB	run_blue.bat MS-DOS Batch File 1 KB
run_IOnFile.bat MS-DOS Batch File 1 KB	lock.ico Icon File

Figure 20: the folder with the program used for starting the program.

Once you are here right click on the Bluetooth Monitor.exe and send to the desktop as shortcut. See figure 21.



Address \Documents and Settings\user\My Documents\Project\code\The Actual Code\PC code\Bluetooth Monitor\Bluetooth Monitor\bin\Debug

Bluetooth Monitor.exe
Application
Date Modified: Monday, October 20, 2008, 6:50 PM
Size: 36.0 KB

Figure 21: creating a shortcut.

3). Now the program can be started by simply clicking on the shortcut on the desktop, see figure 22.

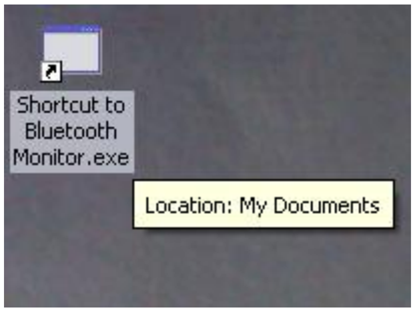


Figure 22: the shortcut used to start the program.

Final tests:

Portability:

The main aim of this test was to evaluate how the system would react if it was in a different environment, which is a different phone; because during the design the software was to work on a Nokia phone only preferably one that is Symbian capable. In this test I used three different phones.

The phones I used were the Samsung SGH - D900I, Samsung K810i and the Motorola SLVR L7, all phone responded well with the main part of the notification, just like the Nokia E51 and Nokia N73 that I used which are both Symbian enabled phones.

Signal obstruction:

Because Bluetooth is not affected by signal that much especial if the connection between the two devices with Bluetooth does last that long, the only problem is when there many devices around then the software had to scan through all of them, first looking for the name it wants then the services on the phone which took just a little long than usual. So in a place where there are many phones around the program will take a longer time to send the notification.

Chapter 9

PROBLEMS ENCOUNTERED DURING THE PROJECT DEVELOPMENT

Introduction

In the previous chapter we took a look at the various testing methods that where implemented, in this project. In this chapter we take look at the various difficulties, problems or hiccups that where encountered during the development of the project. This problems defined here are such that they could not be resolved before the final product was handed in for evaluation.

Problems with obtaining help and Information from an abstract level.

During the project development I found my help on the following WebPages:

1). http://fourm.nokia.com/

2). http://forums.microsoft.com/msdn/

3). http://msdn.microsoft.com/en-us/library/

4). http://www.codeproject.com/

5). http://www.openobex.org/

But the problem is that information on Bluetooth is difficult to find, especially debugging information. Many a time when I had gotten an idea on how to approach a certain section of the project, after coding the idea a huge bug would arise, it was almost and sometimes impossible to find out what the problem was; visiting these sites would help but in most cases it did not.

The two major problems encountered

SENDING A MESSAGE TO THE CELL PHONE.

The concept:

The idea here was that if a message could be sent to the phone, then making an application that reads the inbox for check for new messages of type Bluetooth would not be a problem. Once the application could read only the Bluetooth messages then making the application to vibrate would not be a problem it would just be a matter of call the CHWRMVibra class to make the phone vibrate in a certain way.

Reason for failure:

Windows does provide an application: fsquirt.exe to send files to a phone even text files which would have been the ideal case but, getting the fsquirt.exe into a program is impossible; coding the fsquirt.exe application is I feel out of the scope for this Honors project. Even though windows does provide a function in c++ to send string to a device on a network; the problem with Bluetooth is that even thought the approach towards Bluetooth is similar to a network approach the problem is that the net work layers are different as compared to a normal network communication between two devices. So even though the message was sent to the phone, unpacking the packet sent was a problem; because the network encoding a different and are handled in different ways.

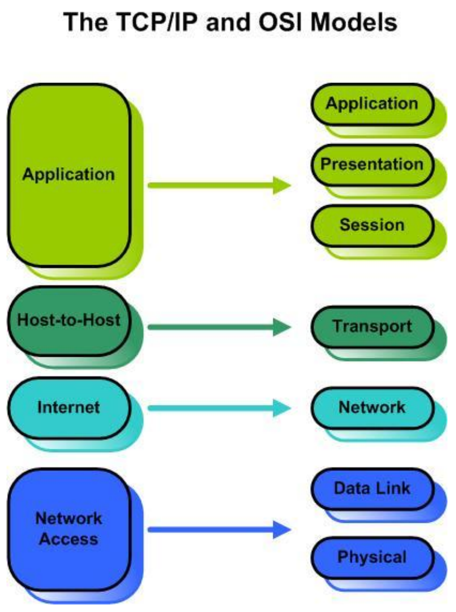


Figure 23: Network ISO model

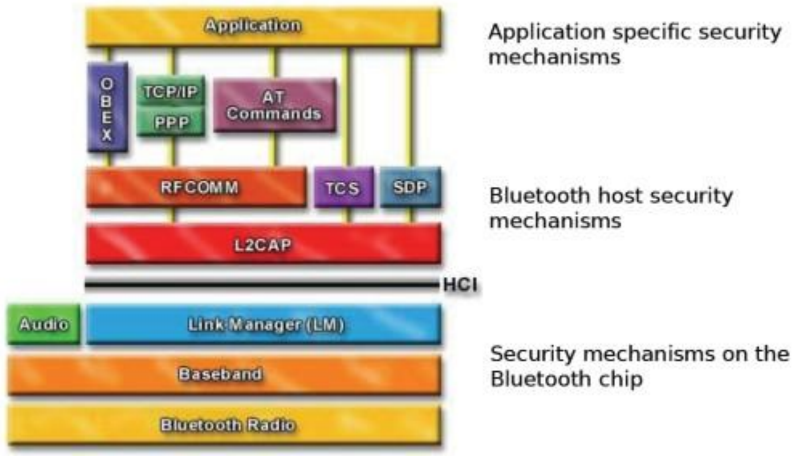

Application
OBEX
TCP/IP
PPP
AT Commands
RFCOMM
TCS
SDP
L2CAP
HCI
Audio
Link Manager (LM)
Baseband
Bluetooth Radio
Application specific security mechanisms
Bluetooth host security mechanisms
Security mechanisms on the Bluetooth chip

Figure 24: Bluetooth Protocol stack

If one considers the diagrams 21 & 22, one can obviously notice that the network stacks are different. Now communication between the two must be made in such a way that a translator is placed either on the phone's side or the computer's side.

The best idea to place on the computer side to increase portability but once again when I came to finding out how to begin or either debug the part of the development it became impossible to solve due to the lack of information.

Another approach attempted was to just make the phone vibrate when some new came on the phone; there are two stages to this:

1). For something to arrive on the phone, requires network communication which in turn requires that some kind of message is send as discussed above this was a problem.

2). How about checking for any Bluetooth connection from the computer and making the phone vibrating, the problem here is that the kind of function called to make this work are not given to non Nokia developers for some reason unknown. So these functions obviously lie within the Operating System documentation which I felt trying to reveal was difficult and I did not have enough information on.

To counter the negativity describe above, I just to leave the phone's Operating System detect if a message is supposed to be received or not. This clearly visible but it doesn't offer the Deaf user the functionality of vibrating the phone which I feel is a major hiccup. Hope documenting these problems I encountered could someday help a future developer to overcome or find a way around them.

Conclusion

In this chapter we discussed the various problems that occurred when developing the software, this is done to help a future developer in find out how to solve the problems here before attempting the development and later find the same problems encountered here.

Chapter 10

SOURCE CODE

Refer to the CD which came with this documentation.

BIBLOGRAPHY

Beau, Crawford. *Kiara: A New Breed of Email.* CS 5761 Final Project May 7,

C Bala Kumar, Paul J. Kline & Timothy J. Thompson*: Bluetooth Application Programming with the Java APIs.* Elsevier Inc 2004.

Joshua Kaufman: *Practical Usability Testing.* Digital Web Magazine, February 13, 2006.

Ranjith Antony and Bruce Hopkins. *Bluetooth for Java,* Apress 2003.

http://www.operamini.com/

http://fourm.nokia.com/

http://msdn.microsoft.com/en-us/library/

Bruce Hopkins & Ranjith Antony: *Bluetooth for Java* .Apress 20003

Smith, Chris. *Theory and the Art of Communications Design.* State of the University Press, 1997.

University of Liverpool._*Bibliography and References: MHRA System.* University of Pennsylvania Press, 2001

http://www.mxit.co.za/

http://www.codeproject.com/

http://www.openobex.org/

http://forums.microsoft.com/msdn/

INDEX

A

API, i, vii, 2, 10, 30

D

DCCT, vii, 1, 3, 6

I

IM, vii, 10, 17, 21, 22, 23, 24, 25, 29

J

J2ME, vii, 8, 9

M

Mxit, vii

O

Operamini, vii

P

Python, vii, 8, 9

S

SMS, vii, 1, 9

U

URD, 3