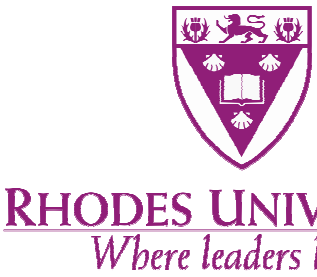
A qualitative and quantitative comparison of Google's Android Toolkit against Sun's Wireless Toolkit for location-based services



RHODES UNIVERSITY
Where leaders learn

Written By: Takayedzwa Gavaza

Supervisor: Mrs. Madeleine Wright

Submitted in partial fulfilment for the requirements for Bachelor of Science (Honours) degree at Rhodes University

3 November 2008

Acknowledgements

The biggest challenge for doing a sound project on a fast-evolving topic is to keep learning new things. This learning and writing is not possible without the help and guidance of many individuals.

I would like to thank my supervisor, Mrs. Madeleine Wright for all the patience and guidance throughout the year.

I would like to thank Mr. John Ebden. You have been a good father-figure to me.

I would like to thank my friends Pamela Gracious Hlahla, Ray Musvibe, Shange-Ishwa Ndakunda, Curtis Sahd, Sinini Ncube, Bwini Mudimba, Zvikomborero Hommannie Nyamazunzu, Kizito Matemera and Rumbidzai Nachangwama for the support they gave me.

I acknowledge the financial and technical support of this project of Telkom SA, Business Connexion, Comverse SA, Verso Technologies, Stortech, Tellabs, Amatole, Mars Technologies, Bright Ideas Projects 39 and THRIP through the Telkom Centre of Excellence at Rhodes University.

Lastly I would like to thank my family.

Abstract

Since ancient times, man has been curious to find out his location. Ancient people used landmarks, stars and verbal queries to locate themselves. In the middle ages, man used compasses and maps to find out his locations. The mobility, portability and the ever increasing services that are being offered by mobile devices has led people living in the current world into focusing on embedding Location Based Services (LBS) into mobile devices. This is a result of the curiosity for knowing each other's present position and the mobile user's current position. This is a feature that can be used to locate a mobile device or the user carrying the mobile device. To accurately locate a mobile device, developers must have the right tools to make it easier and more efficient to develop, deploy and manage location-based applications on mobile devices. As a result of the hunt for better tools, new mobile platforms like Android are now emerging and joining the competition with well established platforms such as JME.

This project focuses on the study, analysis and comparison of the emerging Google's Android Toolkit with the well established Sun's Wireless Toolkit. A feature-by-feature comparison was done to find out each platform's strength and weaknesses. The results from this project can be used by developers to make trade-off decisions on which tools to best use.

In general, the Android location API has more functionality compared to the JME API because of its numerous classes. Android also requires less lines of code because most of the methods are pre-defined for developers

Table of Contents

Table of Figures

Table of Figures

Chapter 1: Introduction

In this chapter, we are going to take a look at the market background of Location-based services (LBS) and how they are being widely adopted. On analysing their wide adoption, we will be looking on the statistics and surveys carried out. We are also going to discuss the reasons why we chose to do LBS and how these services are affecting our daily life. We will also look at the motivation, objectives of this project and the approach that was used to solve the problem. Another section of this chapter gives us the light on how the whole thesis is structured.

1.1 The Market Background

Location-based services (LBS) are after the integration of a user's current geographic location with the general notion of services. An example would be using a navigation system in a car, which retrieves information about a certain location.

Mobile LBS are now becoming popular and more useful to many people. Most of the smart-phones are now coming with an embedded Global Positioning System (GPS) receiver to provide the current location. The supply of smart-phones with GPS receiver has gradually grown from the year 2001 up to now.

Basing on the information provided by various international marketing survey groups, the market size for LBS in the year 2002 ranged from US$2billion to US$2.5 billion. It was projected in 2002 by Allied Business Intelligence that worldwide revenues in location based services would exceed US$40 billion by 2006 [Kin, 2003].

Different analysts have independently projected the figures to rise from US$2 billion in 2001 to US$18.5 billion in 2006 [The Asian GIS Portal, 2006].

1. According to the forecast done by the Agricultural Research Council (ARC), the LBS market was expected to reach US$33 billion by 2005 [Annex-A, 2002].
2. Cahners In-Stat predicted a growth in revenues from US$3.7 million in 2001 to more than US$13 billion in 2005 [Annex-A, 2002].
3. Ovum predicted a US$4.7 billion market for LBS by 2004, growing to US$19.5 billion by 2006 [Annex-A, 2002].
4. The Strategies Group combined this information and estimated that there would be more than 60 million users and over US$16 billion in annual revenues worldwide by 2005 [Annex-A, 2002].

5. Nokia expects to ship 35 million GPS-phones in 2008 [Zou, 2008].
6. An ABI report blueprinted a $ 3.3 billion market value for LBS by the year 2013 [Belic, 2008].

It is believed that LBS will create new markets and new revenue opportunities for device manufacturers, wireless providers and application developers.

1.2 Why Focusing on LBS

With the number of mobile users above 3 billion and many of mobile internet users, plus the increase in the number of people in transit on a daily basis, there has been a greater demand for services offered by mobile phones. So many daily transactions nowadays are done over the internet, but because people are always on the move, it's no longer so easy for them to use desktop computers for their services. Mobile devices are no longer just voice communication devices. Many mobile data services are now incorporated into mobile phones.

One of the increasing popular data service LBS emerged a response to the curiosity of humans about their current location, the things surrounding them and how they might move from one place to another. LBS provide users of mobile devices personalized services specifically tailored to their current geographical location. Commercially LBS are important because they open a new market for developers, cellular network operators, and service providers to develop and deploy value-added services. Smart phones offer the best functionality for incorporating these services. LBS are a promising growth area for the mobile wireless industry. LBS will greatly impact the way we live, conduct business and acquire information about a specific geographical area.

We expect an improvement in terms of convenience given that LBS information is forwarded to right people without spamming. Forwarding information to people who do not want it might inconvenience them. Cost savings is also an advantage, as people will be receiving information about products according to their current location due to the use of Mobile Advertisement Services. All products information in the area will be forwarded to users, thereby making an appropriate choice without wasting money. Another important aspect is security and customer care since LBS can locate the accurate current position of a subscriber. Information about the current position of a child and whether he/she has reached home might be forwarded to parents. Time conservation and best route to use are likely going to be part of

this service. This can be made possible by Traffic Alerts from Real-Time traffic services that will notify users of the status of their predefined route.

1.3 Motivation

Programming should aim to improve the speed and quality of data services on our phones. The strengths and weaknesses of each platform must be calculated so that they can be used to improve mobile data services.

Due to the increase in competition and the number of services being offered by mobile phones, new platforms for developing mobile data services are now emerging. This triggered our interest in comparing the Location-Based mobile data services developed by the emerging Android API against the well established Sun's Wireless Toolkit. As a result of stiff competition in the mobile phones industry, mobile operators are now forced to produce quality data services to differentiate themselves from their competitors.

JME stands for Java Mobile Edition which is mainly for resource-constrained devices such as mobile phones, PDAs. It is Java for devices with limited user interaction (no keyboard/mouse), limited display (size, colour), limited memory, communication, limited power (small battery), size constraints (small, light) and devices that are less shock resistant. JME is a product of the Sun Microsystems Laboratories and was first introduced in June 1999 then was standardised in October 1999. JME is backed by a number of promoters and users which include Mobile Device Manufacturers like Ericsson, NEC, Nokia, Palm Computing, and Research in Motion (RIM), LG TeleCom, Samsung, and Motorola which have all shipped Java-ready Mobile Information Device Profile (MIDP) devices. On top of these companies, Standards Organizations like the Third Generation Partnership Project (3GPP), which is responsible for defining the specification for the next-generation handset application environments identified, JME as the industry standard for mobile devices and 3G wireless applications. This clearly show that JME has been here for quite a long period and also shows how well established it is.

Android is an emerging open source Linux-kernel-based software platform and operating system for mobile devices initially developed by Google and later on joined by the Open Handset Alliance which is a group of hardware, software, and telecom companies that support the advancement of open standards for mobile devices. Android was released in November 2007. Although one is in production, there is no phone yet that currently run Android [Taves, 2008]. Although underlying components of android are built in C/C++, user

applications are built in Java and then are translated to a different representation called dex files which permits Android to run its applications in the Dalvik Virtual Machine [Ableson, 2008: 43].

Mobile data services require the best development tools to ensure their rapid and efficient creation, deployment and management. The solution to these challenges is the ability to quickly and easily create, deploy and manage content and applications on mobile devices. It is critical that mobile phone content developers have the right tools to respond to the challenges of the growing mobile phone market.

1.4 Objectives

The main aim of this project has been to make a qualitative and quantitative comparison of Android against Sun's Wireless toolkit and work out how best to use each platform's strength and avoid its weaknesses for the development of the mobile data services mainly focusing on LBS.

The idea is to find a platform with which a developer can quickly and easily create, deploy and manage content and applications on mobile services.

1.5 Approach

Two nearly identical Location-Based mobile applications were developed in both JME and Android. The LBS applications were made to offer the same interface and service in both platforms. From these two different systems a qualitative and quantitative comparison study was made. The qualitative comparison was based mainly on the mobile development environment, deployment environment, application environment (the tools available) and application maintenance. The quantitative comparison included memory allocation, the number of files, the number of functions, the speed of execution and number of lines in the files.

1.6 Thesis Structure

Chapter 2 takes us back to the history of mobile LBS. This covers some of the relevant work and how LBS emerged. We also follow and analyse past LBS trends. It reveals current state-of-art, and other use of LBS. Chapter 3 reveals the design and implementation of the two systems developed in these two different platforms, JME and Android. Chapter 5 gives us the

comparison of the two applications and what was concluded from the development, use and maintenance of the two applications. Chapter 5 then concludes the whole project.

1.7 Chapter Review

In this chapter, we were introduced to the whole structure of the thesis and the background of LBS. The chapter explained the objectives of this project. From there, it also gave us the approach that was used to do this project and lastly the whole structure of the thesis. The structure gives us a short introduction to every chapter in this thesis

Chapter 2: Related Work

In this chapter we discuss a brief history and the trends in LBS. We discuss the wide adoption and use of LBS in our daily lives. We also look at how LBS are affecting our daily living and improving our businesses and lifestyle.

2.1 Introduction

The revolution of desktop computers has come and gone. The new revolution involves wireless devices, which provide a cheap, lightweight, and often stylish portal to the full power of the Internet.

Mobile phones already outnumber desktop computers as a method of connecting to the Internet, and the trend will only continue. With more and more applications moving from the desktop to the Internet, a mobile phone is a much simpler and more convenient alternative to a desktop computer or even a laptop.

According to [Knudsen, 2005:4] the wireless networks of today are still slow compared to cable modem and DSL technologies that are available in many homes. Nevertheless, much useful work can be done. Faster networks, which will open up new worlds of applications, are deployed in some parts of the world and will become widespread in the next few years.

Since ancient times, man has been curious to find out his location. Ancient people used landmarks, stars and verbal queries to locate themselves. In the middle ages, man used compasses and maps to find out his locations. The mobility, portability and the ever increasing services that are being offered by mobile devices has led people living in the current world into focusing on embedding Location Based Services (LBS) into mobile devices. This is a result of the curiosity for knowing each other's present position and the mobile user's current position. This is a feature that can be used to locate a mobile device or the user carrying the mobile device.

2.2 Location Based-services (LBS) and Global Positioning System (GPS)

Today's real-world applications are often focused on location parameters for mobile applications. Convergence of Location and context are one of the major research focuses on LBS and Geographical Information System (GIS). The ever increasing demand for location-

aware mobile devices triggered our interest in researching towards LBS and GIS-related work.

The LBS market is believed to have first emerged in South Korea and Japan, driven by personal navigation and some family- and people-finder services. In the United States, Nextel and Sprint initially drove LBS adoption with a focus on fleet applications. Verizon Wireless also entered the market in 2006 [Fabris, 2006].

LBS are services accessible with mobile devices through the mobile network and make use of the geographical position of the mobile device. LBS include all information and entertainment applications that make use of location and geographic data. LBS enable brand new applications not possible on the desktop world [Yuan, 2004:346]. These applications including receiving information about your current location as you move from place to place, locating users in emergency situations and other scenarios.

There are a large number of location-related standards like OpenGIS which use a protocol for requesting location information from a database [Open Geospatial Consortium, Inc., 1994-2007]. Most of the mobile location-aware applications are driven by the ability to retrieve data from application servers.

Past location-aware applications were created by connecting a non-mobile GPS receiver that streamed strings over a serial connection, which could be implemented over a cable or wirelessly via Bluetooth. Modern LBS applications require more interaction with the device [Elsevier, 2008]. According to Elsevier [Elsevier, 2008], handset–initiated APIs give a software developer great control over the properties of the data returned.

Mobile devices and cell phones with GPS receivers have their location information available on the device, but their browsers send HTTP requests which do not contain this information, so it is impossible to use location-based services with these devices outside of the proprietary infrastructures provided by cell phone carriers [Djuknic & Richton, 2001:123].

For most of their history mobile devices have not been location-aware. A location management framework was implemented as a web proxy. Mobile devices had to visit a

Web-based UI which lets them configure their location. The information was stored in a database and users had to update their browsers.

The new generation of open mobile devices such as Android and OpenMoko will make it easier to embed location-based service on the client side, without the need to re-implement the service for every single device [Ableson, 2008] because they have more functionality. Android's location API looks promising, but on the other hand has no obvious way to hook into the Android browser for accessing information from the internet. So while Android implements a LocationManager, there probably is no way that this manager could be made available in the browser, which means that in order to deploy a location manager and a location-enabled browser on an Android platform, a complete browser must be deployed as a new application component [Ableson, 2008].

The core technology for any LBS solution is the Geographical Information System (GIS), which performs important functions such as determining street addresses from coordinates and vice versa [Yuan, 2004:346]. LBS on mobile devices use many techniques to obtain the device's current location as indicated in Table 1 below: where details are extracted from Yuan's text [Yuan, 2004:358]

LBS technique	Description
Terminal-based	A GPS-equipped device that calculates its coordinates using GPS satellite signals
Network-based	Cellular network operators can determine the location of any phone in the network using the phone's signal strength received by three nearby access stations (triangulation).
Network-assisted GPS	Mobile phones can use network data to determine an approximate position and then use the GPS module to get accurate corrections.
Local wireless network-based	This is location determination done in local wireless networks like WI-FI and Bluetooth networks
User-Assisted	This works in controlled environments. The user identifies the nearest landmark and from that he/she estimates her current coordinates

Table 1: shows location-based techniques for mobile phone devices [Yuan, 2004:358]

It is believed that GPS-Enabled LBS subscribers will total 315 Million by the end of 2011 [Elsevier, 2008] [3G, 2006]. This shows an estimated increase of more than 18 times from the 12 million subscribers in 2006. There will be a rise from less than 0.5% of total wireless subscribers today (2008) to more than 9% worldwide by the end of 2011 [Elsevier, 2008].

The increase the use of GPS services will drive the adoption of Universal Mobile Telecommunications System (UMTS) 3G handsets [Ahonen & Barrett, 2002]. In the past the growth of 3G has been limited by customers' low uptake of many 3G services, making it uneconomical for operators to subsidize these handsets heavily. GPS-enabled LBS is expected to lead subscribers to use more 3G data services, and thereby to drive overall 3G handset sales.

North America and Western Europe are expected to be Regions of greatest growth, followed by the Asian-Pacific which is greatly affected by the varying market. South Korea and Japan will continue to be engines of LBS growth. According to 3G [3G, 2006], market growth in Western Europe has been limited by the fact that very few Global System for Mobile communications (GSM) or Wideband Code Division Multiple Access (WCDMA) handsets

have GPS. More WCDMA 3G phones will contain GPS chipsets, allowing operators to offer LBS

Ridley said that “Global mobile phone use was expected to top 3.25 billion which is equivalent to around half the world's population by the end of 2007 as cell phone demand booms in China, India and Africa. From African farmers to Chinese factory workers, mobile phone subscriptions were expected to pass the 3 billion mark in July and exceed 3.25 billion by the end of the year”, This was according to a report by UK-based telecoms analysis company called The Mobile World [Ridley, 2007].

The mobile phone has revolutionized communication. It has spread from city whiz kids to Brazilian slum dwellers. More than 1,000 new customers are signing up for mobile phones every minute around the world; a survey done by Reuters revealed this information [TelecomsEurope, 2007].

Although it took over 20 years to connect the first billion subscribers, it only took 40 months to connect the second billion, showing that the use of mobile devices is growing exponentially.

With handsets and services becoming more affordable, the prospect of a fully connected mobile world is becoming ever more real. A record 240 million handsets were sold and 135 million new customers signed up to mobile phone networks in the first quarter to the end of March as compared to 142 million and 163 million signed up respectively to the fourth quarters of 2004 and 2006. [Ridley, 2007]

The shipment of smart-phones into Europe, the Middle East and Africa reached 12.6 million by 15 August 2008. 38% of the 12.6 million had built-in GPS and 58% had integrated WI-FI [Canalys, 2008: 1].

According to the estimates done by eMarketer, there will be over 63 million location based service users by the end of this year (2008) and 486 million in 2012. Figure 1 below shows the estimated number of users of location-based services up to 2012 [eMarketer, 2008].

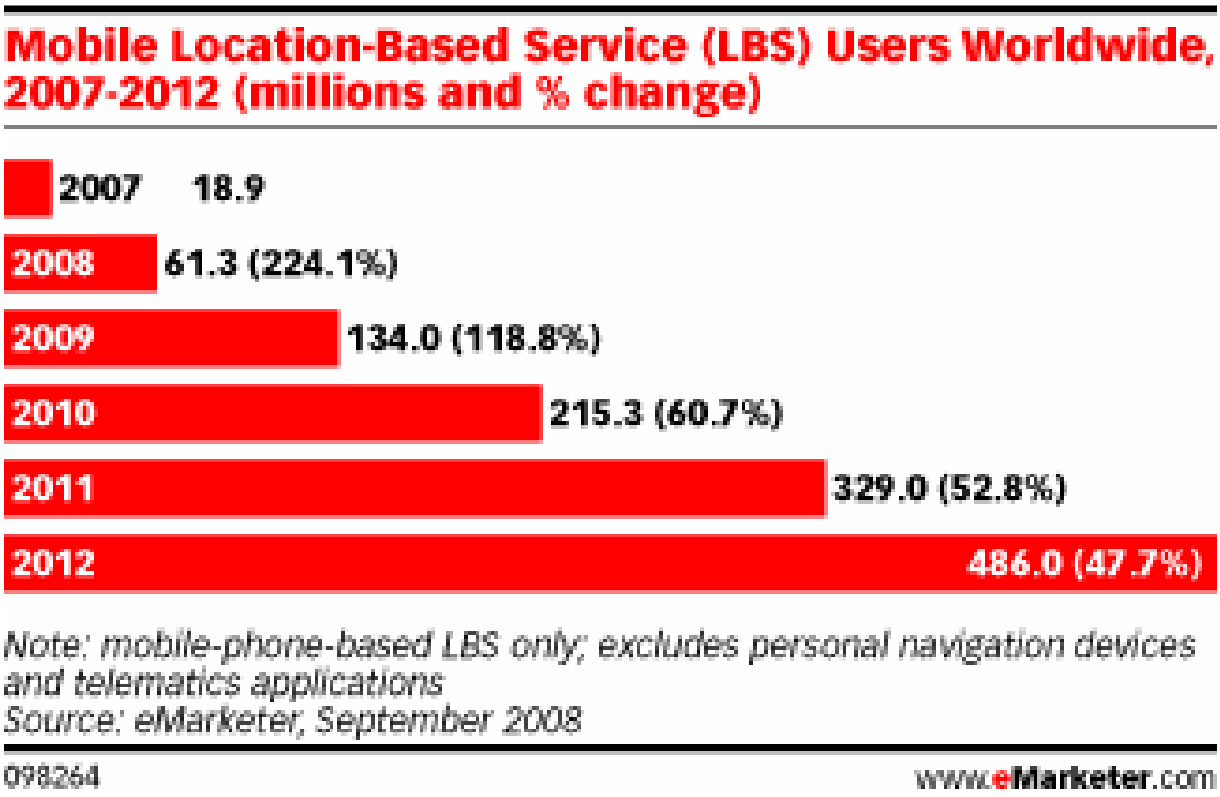


Figure 1: shows estimated number of Location based service users by the year 2012

Figure 1 above shows the wide adoption of Location-based services world wide. This shows that Location based services are becoming popular on daily basis.

2.3 Location technologies for location devices

LBS operate in two parts. The first part is taking signal measurements, followed by computing the location using the signal measurements. There are three main positioning techniques which are cell-location, advanced network-based and satellite-based positioning [Mitchell & Whitmore, 2003]. The three main categories of positioning methods are shown and explained in Table 2 below in the order of increasing accuracy.

Location Service Category	Explanation	Typical methods	Accuracy	Response time	Key limitations
Category LS1: Basic Level Service	Location of all hand sets with at least cell accuracy	Cell of Origin (COO), Cell-id, including service area identity (SAI) locWAP and enhanced CELL-ID. May also include enhancements with propagation time measurements	Low accuracy. Depends on cell size and enhancements ; typically 150m to 10000m	Very fast. typically around three seconds	Very limited accuracy in areas with low cell radius
Category LS2 : Enhanced Service level	Location of all new handsets with reasonable cost and improved accuracy	Estimated Time of Arrival (EOTD) of GSM, and variations such as Advanced Forward Link Triangulation (AF-LT) and Idle Period Downlink (IP-DL) for CDMA and WCDMA respectively	Medium accuracy. Typically around 50m to 125m	Fast EOTD takes around five seconds	Depends on visibility of base stations for signal measurements and number of location measuring units
Category LS3 : Extended service level	Location of new handsets with high accuracy and cost than LS2	Global Positioning System and Advanced Global positioning System	High accuracy. Approximately 10-20m outside buildings and approximately 50m inside	Variable. GPS around 10-50seconds but only 5 seconds with AGPS	Signal degradation and reduced accuracy in certain environments eg, inside buildings

			buildings		canyons”

Table 2 above shows the three main categories for positioning methods [Mitchell & Whitmore, 2003]

From Table 2 above we see that there are three main categories of positioning methods. The first category includes COO and this category has low accuracy but high response time. We also have a category that uses cell-tower so find the location of the mobile device by using methods like triangulation. This category is has medium accuracy and response time, Last will have handsets that access the location from GPS. In this category, there is high accuracy and low response time. The above table clearly shows that there is a trade-off between accuracy and response time.

According to [Yuan, 2004], there are two main types of location applications. These are pull-based and push-based applications. In pull-based applications, the user send out her location to a server and pulls in location information whilst in push-based applications, the service providers detect users’ locations and send them or push out services. Push-based applications need approval from the client to prevent invading peoples’ privacy. With all these applications, Geographical Information System (GIS) is the core technology for any LBS solution. GIS plays a big role in resolving coordinates to their respective physical addresses. Although GIS servers are capable of geocoding and map rendering, they are expensive to run for an individual as the cost includes hiring GIS experts to set up, maintain and update geographical information. An alternative is to use the MapPoint API [Yuan, 2004:347] which is a Web Service that uses HTTP Digest Authentication. Using the MapPoint Web Services API to access information, you have to use a an Aggregated API which does the conversion of addresses to latitude and longitude coordinates, calculates the route according to options, renders the overview map with the route start and end locations highlighted, retrieves turn-by-turn instructions for each route segment and finally renders highlighted turn-by-turn maps for each route segment. This process is shown in Figure 1 below.

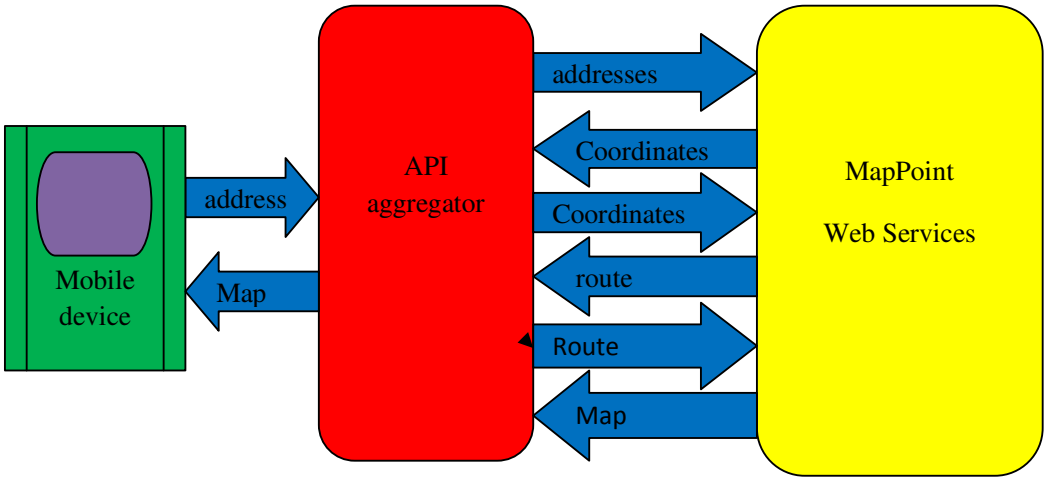


Figure 2 shows the current architecture [Yuan, 2004:350], words taken from [Yuan, 2004:350]

The problem with this system is that it is slow and the user has to input the physical addresses of his current location so that he will be able to be routed to his final destination. The main problem with entering the person's current location comes when the person does not know the physical address of his current location. To save the person's time and to help in cases where you don't know the physical address, the coordinates of the current position are input using a GPS. This eliminates the process of entering the current physical location and searching for coordinates from the web service. The direct coordinates will then be used to retrieve the route and direction of the desired destination point as shown in Figure 3 below

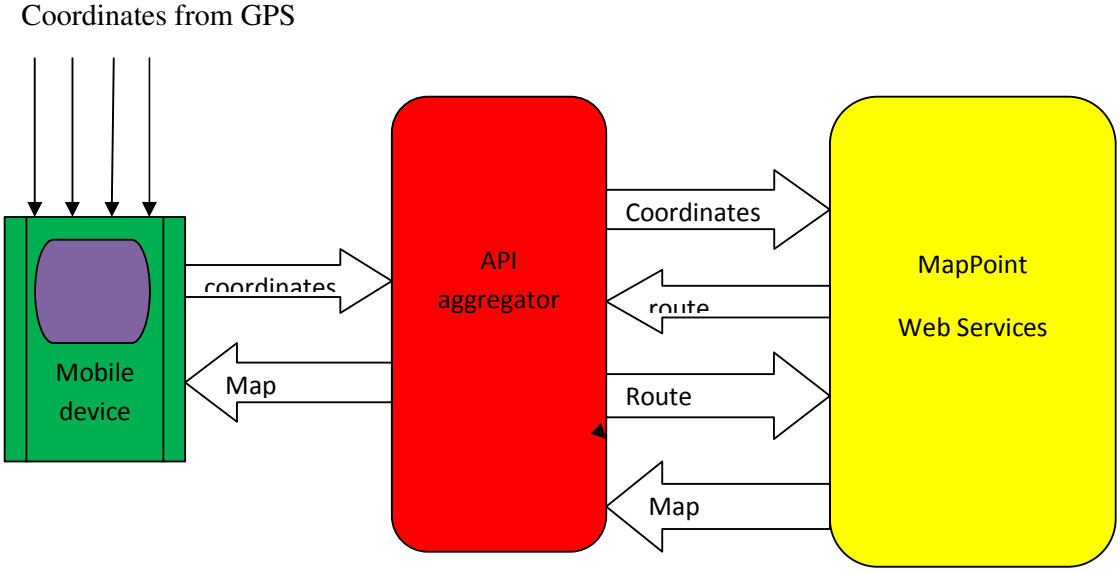


Figure 3 above show the process after eliminating input of physical address

2.4 Emerging Location-based services

Location-based technologies allow many advanced forms of consumer data services based on the position of the user, as discussed below.

2.4.1 Safety

The main push for location-aware mobile device in the United States of America is the need for applications concerning the users' safety. There is a great need for knowing the exact location of a person who dials the emergency number or sending an emergency message. This led to a rule stating that handset-based solutions must locate an emergency caller to within 50 meters for 67% of calls and within 150 meters for 95% of calls [Rockwell, 2003]. In South Africa, the same technology is of great value for aspects of personal safety particularly roadside assistance. The consumer's mobile device can be used to assist in getting roadside assistance to the right location.

Mobile phones are also being used in disaster response. The mobile phone signal of an injured individual can be used to locate the person. In other parts, alert messages are sent to subscribers with information about an upcoming natural disaster or event. LBS can also be used in emergency caller location, asset tracking, navigation, location-based information or geographically sensitive billing.

2.4.2 Navigation and Tracking

Location-based services can be of great value to the South African tourism sector. It can be used to keep track of tourist since they are often found in an unfamiliar geographical environment. Tourists can use services like Bluesigns where they call the tourist information centre and then their location is determined during the call using GPS and some location-sensitive information about their current position is send to them. This information can be used to guide him to the nearest resources he wants or just make him be aware of his surroundings.

Tracking is also in current use in cab companies. This allows them to find the nearest cab to a customer. As a result they save time and avoid keeping customers waiting [Varshney and Vetter, 2002].

2.4.3 Information

Similar to the tourism situation, information can be pushed to a mobile user according to his present location. Advertisements may need to be sent to people in a certain area only [Tseng, Wu, Liao & Chao, 2001]. These include locational advertising for targeted advertisements, public info-stations for distributing public information, geographic messaging for localized information and alerts. Lastly, we have Yellow Pages for finding the proximity of a specific business.

2.5 JME Literature Review

The main focus of Mobile Network Operators (MNO) is to bring in new technology so that they can create values and profits from ultimate customer satisfaction. The main concerns are the new values or cost savings that mobile technologies can create.

New generation mobile devices empower us to access information anywhere at any time. This includes the mobile communication devices and mobile internet. For the first time in history, a person's information access can be disassociated from his environment. A traveler no longer needs to be in front of a PC to get ticket information. Accessing information

without the constraints of landlines and bulky desktop PCs will create great business opportunities and improve our quality of life for years to come.

People are as likely to use their mobile phones for mobile data services at home as they are while they are away from home. Usability of handsets is becoming more important to consumers as most of them are opting for a handset with a large screen [Pell, 2006].

The fact that people are able to go everywhere with their mobile phones has increase the demand for more services. Mobile phones are now playing roles in many law-enforcement acts. In most cases, tower triangulation is used to determine the position of an individual's cell-phone. Mobile telephone forensic specialist use mobile devices location as evidence of criminal acts. Apart from triangulation, recordings of phone conversations can give certain clues. Triangulation helps mobile phones to be located accurately. These was of great help to us since we were aiming to locate the current position of a mobile phone and plot it geographically on a map that will be shown on the mobile device's screen.

2.5.1 Mobile Operating System Providers

There are many mobile device operating systems. This includes PalmOS, Symbian OS, Windows CE, and Embedded Linux. The OS SDKs often lack advanced programming language support and important libraries for business functions.

2.5.2 The Mobile Development Platform

The Sun Java Wireless Toolkit for CLDC [Knudsen, 2005:15-27] includes three main components:

i) A Ktoolbar that allows you to manage and build projects.

ii) A Device emulator to test applications on the desktop computer before it is deployed. The emulator is a simulated mobile phone used to test applications without even using a real device. Emulators enable us to write source code, build an application and run it, all on one desktop computer.

iii) A collection of utilities and tools providing support for many MIDlet features and optional packages.

Sun's Java Wireless Toolkit has a simple drag and drop user interface builder to generate user interface classes for the phone application from a visual designer. The toolkit and its emulators support the Mobile Service Architecture (MSA) specification, providing a highly

capable development environment believed to be well ahead of actual devices. The toolkit provides a simulation environment for Short Message Service (SMS) and Multimedia Message Service (MMS) such that different instances of the emulator can exchange messages. A messaging console utility can also exchange messages with running emulators. The toolkit provides tools to monitor running applications. These include a memory monitor that shows every object and its size, a network monitor that displays all the network traffic in or out of the emulators and a profiler that shows how much time your application spends in each of its methods.

2.5.3 The JME Location API

The location API provides applications with access to a device's physical location using the Global Positioning System. JSR 179 and JSR 293 are the JME location API. These two APIs are designed to be compact and generic APIs that provide information about the present geographic location of the terminal Java applications. These APIs covers obtaining the information about the present geographical location and orientation of the terminal and accessing a database of known landmarks stored in the terminal. The JSR 293 LBS API is important in my project because of its new features, It provides `Geocoding` services which map Geographical Positioning System coordinates into the physical location understood by human beings (addresses and city Names). The physical location can be displayed in the form of a map using the `MapServiceProvider`. Some map features can also be added or removed using the `MapOverlays` application. JSR 293 gives the ability to add directions and real-time guidance to mobile location-aware-applications. This is made possible by the `NavigationServiceProvider` and the route Object which shows the best route. Smooth coordination with geographic data providers must take place since Geographic Services are of little value without up-to-date geographic information.

The above section (2.5) gave us a brief insight into the JME toolkit that is used for developing mobile data services. This section gave us an outline of what is included in the JME software for CLDC and how the tools can be best utilized. We also looked at the classes that are found in the latest version of the Location API v1.0 (JSR179) and the yet to be released version 2.0 (JSR293).

2.6 Android Toolkit

2.6.1 What is Android Toolkit?

Android is a software platform [Ableson, 2008] from Google and The Open Handset Alliance (an organization of approximately 30 organizations). It is a software stack for mobile devices that includes an operating system, middleware and key applications. Android includes a Linux-Kernel-based operating system, a rich user interface, end-user applications, code libraries, application frameworks and multimedia support.

According to [Ableson, 2008:3] components of Android's underlying operating system are written in C or C++, but user applications are built for Android using the Java programming language. Android is an open-source platform, therefore missing elements can be provided by the global developer community. Figure 4 below shows the relationship between the Environment, Linux Kernel and built-in applications.

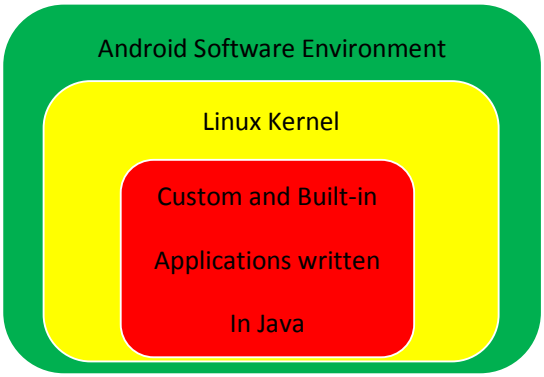


Figure 4 shows the relationship between the Environment, Linux Kernel and built-in applications as illustrated in [Ableson, 2008:5]

2.6.2 Location-based Service (LBS) API in Android

The Android LBS API allows software to obtain the phone's current location based on GPS satellite constellations. `Android.location` and `com.google.android.maps` provide an initial look at the support for building location-based services in the Android platform. The `MapView` API is another important API in our project. It is an Android view

that allows third party code to be displayed and to control a Google map. `MapView` provides a view which displays a map. Another alternative to the `MapView` is the `MapActivity`. The `MapView` gives a tight view with your own layout but requires more code to display a Google map. The most important feature of `MapView` to our project is that, it can be controlled programmatically and can draw a number of `Overlays` on top of the map. An `Overlay` is a layer or drawing that is done on top of a map. `Overlays` are important for drawing the user's current position or location on the map [Ableson, 2008]. An `Overlay` needs constant repaints as long as it is active.

2.6.3 Android Features

i) An application framework that enables the re-use and replacement of components.

ii) Dalvik Virtual Machine optimized for mobile phones.

iii) An Integrated Browser based on the open source Webkit engine.

iv) Optimized Graphics powered by a custom 2D graphics libraries and 3D graphics based on the OpenGL Es 1.0 specification(hardware accelerated

v) SQlite for structured data storage.

vi) Media support for common audio, video and still image formats(MPEG4, H264, MP3, ACC, AMR, JPG,PNG,GIF)

vii) GSM Telephony,Bluetooth,Edge,3G,WI-FI, Camera, GPS, Compass and accelerometer

viii) On top of all, android provides a rich development environment including a device emulator, tools for debugging, memory and performance profiling and a plug-in for the Eclipse IDE.

In the project we are using the Eclipse IDE for the development of Location-based data services. Eclipse can be used to develop both JME and Android applications and Google has provided an Eclipse plug-in for Android but did not give direct support to Netbeans and others. From our own point of view, Eclipse is one of the best IDEs to use contrary to what Ganesan [Ganesan, 2008] argues in her article on the Android tutorial. In her article, she encourages people to use the command-line to develop Android applications. We found that developing applications using the command-line is time-consuming and needs a lot of commands to be carried out before the development process can be started. The other

advantage that we have found compared to the use of the command-line is that, using the command line you have to create and compile the java files and the main.xml files in other development environments and then you have to import or copy the compiled files using the command line. In Eclipse, all you have to do is create and compile your applications in the IDE. Eclipse has services that include Smart Code Completion and Refactoring and gives hints when you are working with both Android and the JME. These save time and developers don't have to write a lot of commands to make the program run. On the other hand, the article gives us more information on how to develop Android applications using the command line and gives us more information on how to go on, since the Android site lacks enough clarity on how to develop applications in both the command-line and Eclipse.

2.7 Chapter Review

In this chapter we gave an introduction to what Location-based services are, how they emerged and why we chose to do a research towards LBS. We also looked at the core technology that supports LBS which is GIS. GIS provides all the information that is used in LBS. We also looked at how LBS evolved, from ancient times, during the time in the middle ages up to the current state. We also discussed how LBS are of use at the present moment and then concluded by taking a short overview of the two mobile development platforms that were used in this project.

Chapter 3: Design and Implementation

For us to be able to compare the development of Mobile LBS under the JME and Android environments, we built two nearly-identical route-finding systems in both platforms that incorporated all the relevant features of the JME and Android LBS APIs. These features contribute to the wide adoption of LBS and they include user interaction and the interaction of the applications with external components such as databases, GPS and web servers. Other important user-interaction features like graphics were also included. It was very interesting to take a deep look into both the JME and Android platforms, and compare them. This chapter mainly looks at the design and implementation of two nearly-identical LBS routing systems developed in both JME and Android.

3.1 The route-finding System

Both the route-finding systems carry out almost the same services. They both allow people to query their current location and, in return, their current location including all the surroundings is displayed on a map on the screen of their mobile phones. The physical address and geographical coordinates of their current position are displayed on the screen as well. People can also query for a certain address and then choose to display the location, show the directions to the place or get route information to the location. In this case the route is displayed on a map on the screen and the distance to destination is also displayed. As the user moves, his current position is displayed on the map and all the information is updated relative to his current position. The user can also query for a certain category of business in a certain radius relative to his current position. For example, someone can query for hotels, accommodation, and restaurants that are in a radius of 1000m from his current position. The system also allows people to enter certain landmarks so that when they come within a certain radius of them, a warning or alert is send to them. Alerts are also sent to them when they arrive at their destination. When people enter a landmark, it is persistently stored in a local database. People are also allowed to define their own landmarks. Most popular landmarks that we use are already defined for us and stored in GIS servers. The systems use `HTTP` connections to query this information from the servers. All this uses airtime or network connections.

There are a number of interactions between the user and the routing systems. These interactions include querying for a certain address or business category, for a landmark, current location, directions or distance to another location. It's also the user's responsibility

to permit the mobile system to use GPS data and the network. The above information truly reflects that there is strong interaction between the user and the route-finding systems. Figure 5 below shows the Use Case Diagram for the route-finding System

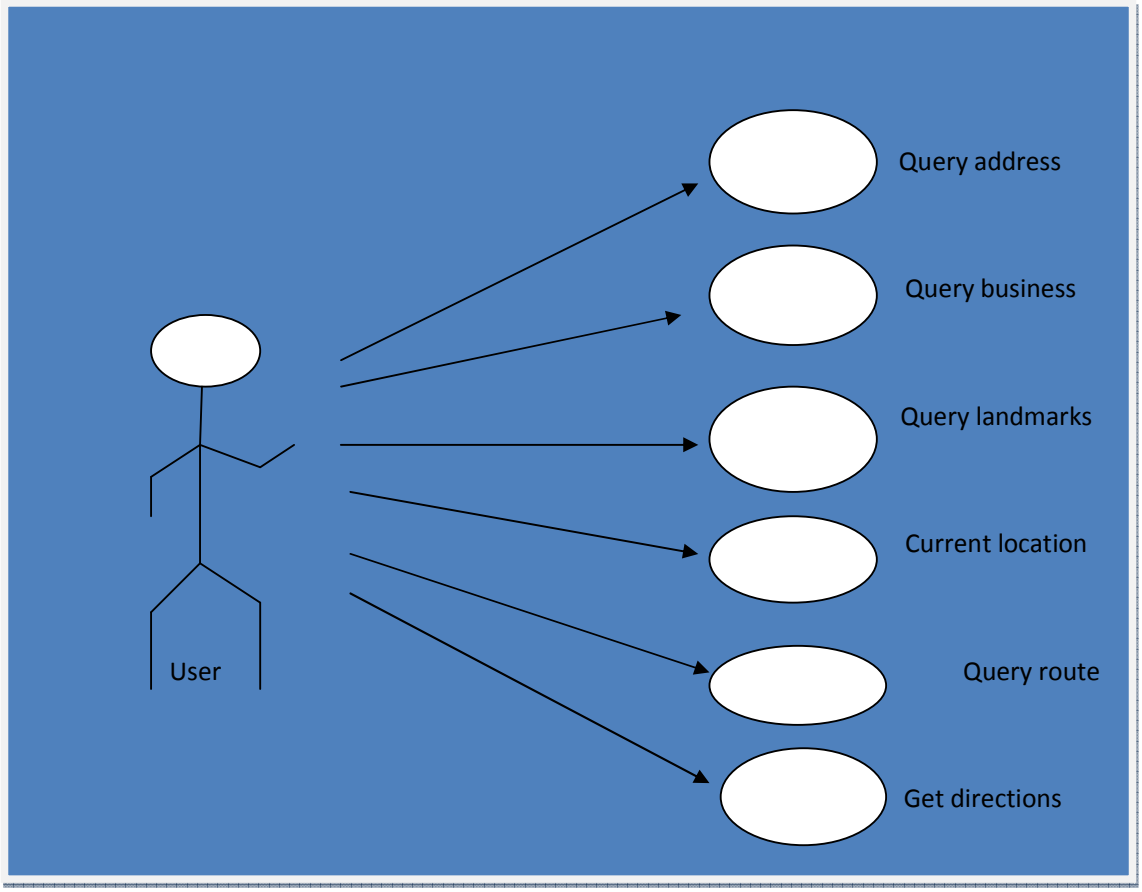


Figure 5: Use Case Diagram for the route-finding system

The systems also retrieve maps from servers and render overlays on top of the maps. This allows the comparison of graphical interfaces and dynamic multimedia capabilities between JME and Android.

3.2 Hardware, Software and Network Environment

3.2.1 Hardware and Network Environment

Because JME and Android are different platforms, they run on different mobile phones. We had to use an emulator in place of an Android phone because there is currently no phone that runs Android. A phone that runs Android is still being manufactured and it is going to be released in November this year (2008). Most smart-phones run JME applications. We used the Nokia N82 for testing the JME applications because its features are suitable for the route-finding system developed. The features include a built-in GPS receiver, a GPS function for getting the current location, the ability to access the internet so that we can retrieve GIS information, an installed Maps application covering over a hundred countries, a WAP 2.0/xHTML, HTML browser and 3G network capabilities.

Apart from running on different phones, each system accesses the same servers and databases and utilises the same network. Different mobile applications can be developed in a common Integrated Development Environment (IDE). This allows an easier comparison between the services. In this case, the mobile data services were developed in Eclipse. A diagram of the hardware and network structure of the route-finding system is illustrated in Figure 6 below

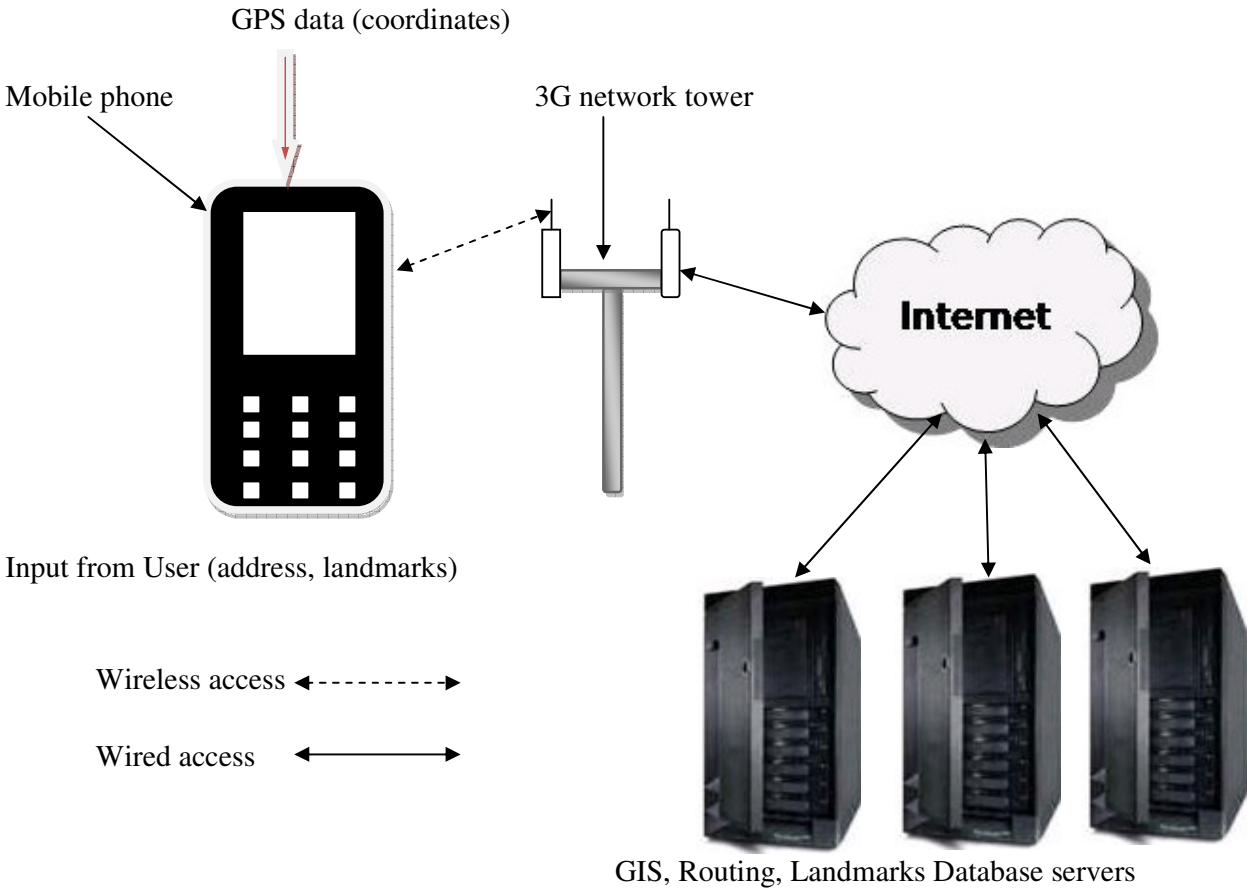


Figure 6: the hardware and network environment

Figure 6 above shows the hardware that was used in building the two systems. The phone receives GPS data from satellites in orbit. The data is received by a built-in GPS receiver. The data is in the form of the coordinates of the current location of the phone. The user inputs the location he wants to be routed to, and also chooses the service that he wants eg directions to the desired location or a route-overlay. From the information that is entered plus the coordinates from GPS, the phone can make `HTTP` connections to GIS servers through cell towers which provide the cell phone service provider's network.

GIS databases were accessed using `HTTP` connections. This was necessary for accessing information for Geocoding and retrieving information that may not be owned by a single person. Such information is kept by big organisations that regularly maintain and update the databases eg the Nebraska Geospatial Data Centre and the USGS Global GIS database owned by the U.S Geological Survey (USGS) and the American Geological Institute (AGI). Cell towers were used for internet connections through network service providers.

3.2.2 Software used

Three main programs were used in the building of the two systems. Sun's Wireless Toolkit 2.5 is a software development kit that is used to build wireless mobile applications for JME. The toolkit provides most of the features that developers need for building applications. These include a well-documented instruction book and examples on how to develop applications in most of its APIs. The Android toolkit was used for building the Android applications. The Android software development toolkit m5-rc15 for windows was used because the development took place in windows. Android lacks good documentation and as it is open source software still under development, it still has many holes eg although there are examples of how to develop applications, there is no description of how the java code links to the XML files for the view. According to Core Security, a company that specialize in software penetration-testing, Android uses outdated and vulnerable open-source image processing libraries [Naraine, 2008].

Eclipse was the IDE of choice. EclipseMe 1.7.9, a plug-in for developing JME MIDlets was used. The main reason why we used Eclipse was that mobile data services in Android are recommended to be developed under Eclipse. Eclipse also has a plug-in for developing JME MIDlets. Android recommends Eclipse and provides its own Eclipse plug-in. This makes the comparison of the mobile data services easier when it comes to qualitative and quantitative analyses involving features such as memory usage and speed of execution. Eclipse was also an impartial environment whereas NetBeans, for example, might have had a bias towards Sun, as it is a product of Sun Microsystems.

3.3 Android and JME Overview

At this stage it is important first to take a deeper look at the architecture of JME and Android.

3.3.1 Android and JME Architecture

Android phones will ship with a set of core applications. These include an email client, sms program, calendar, maps, browser, contacts and other applications written in the Java programming language.

JME phones contain several components known as configurations, profiles, and optional packages that help to balance portability, performance and feasibility. Figure 7 shows the Android and JME components

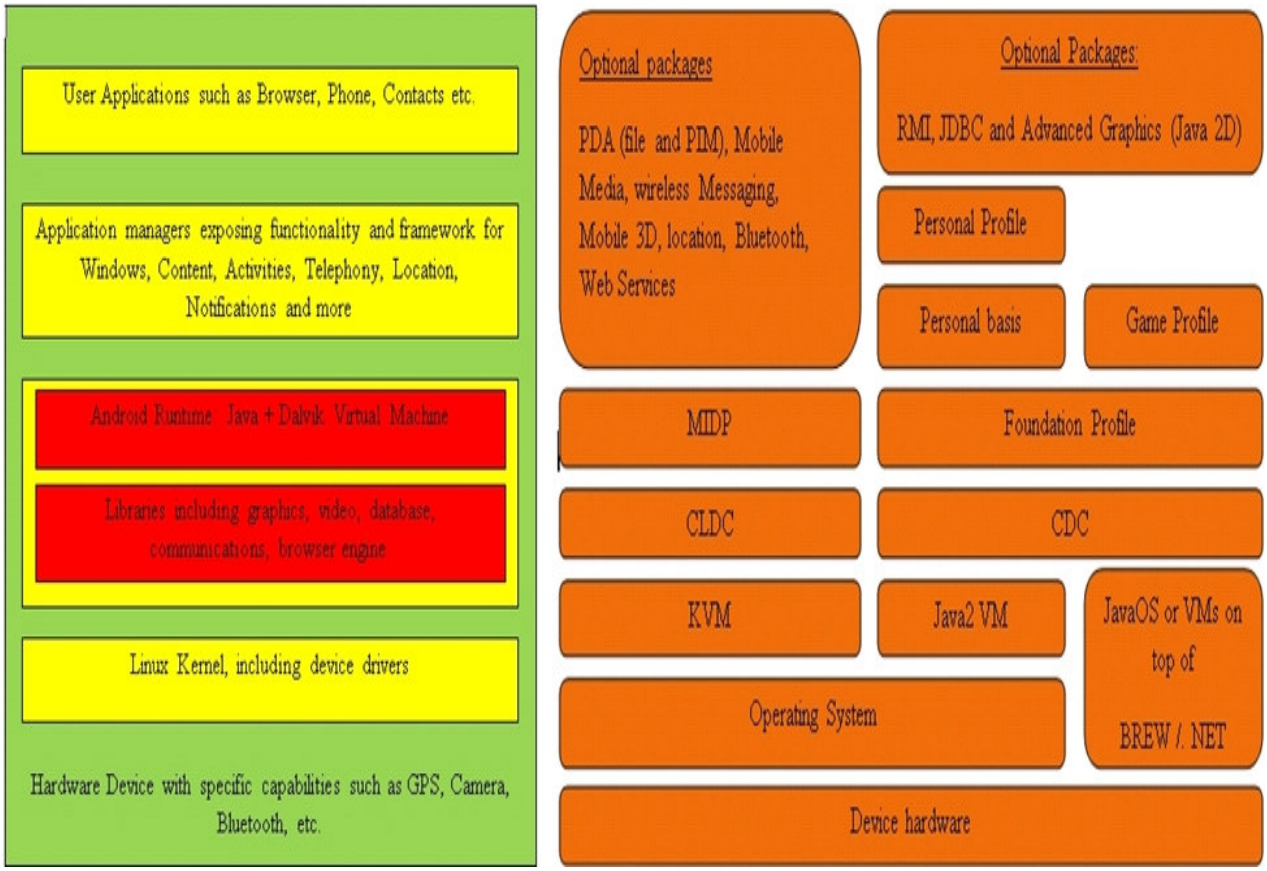


Figure 7: JME components [Yuan: 2004, 21]

In JME configurations provide the most basic and generic language functionality. Profiles sit on top of configurations and support more advanced APIs, such as a graphical user interface (GUI), persistent storage, security and network connectivity. To support specific application needs, optional packages can be bundled with standard profiles. JME has two most important configurations as described below. The JME architecture includes the Connected Limited Device Configuration (CLDC), which is specifically for small devices with limited capabilities. The diagram above clearly shows the packages that are supported by the KVM, which are discussed in the CLDC section below.

The Android architecture looks more like a general computing environment although the platform was specifically built for mobile environments because all packages are supported by same virtual machine. Because Android was specifically built for mobile devices with limited capabilities, all the packages are supported by a virtual machine. Android provides only one virtual machine which is the Dalvik Virtual Machine as compared to the JME architecture which has two Virtual machines. In JME the Kilo Virtual Machine is there to support the CLDC and the Java Virtual Machine supports the Connected Device Configuration (CDC). CDC has better capabilities than the CLDC.

3.3.1.1 Connected Limited Device Configuration (CLDC)

CLDC is a configuration for the smallest wireless devices. The CLDC has limited functionality and lacks features such as the Java Native Interface (JNI) and custom class loaders. Memory and processing power can be scarce on Mobile Information Device Profile (MIDP)/CLDC devices. The CLDC virtual machines support a small subset of JSE core libraries. MIDP is the most important and successful JME profile based on the CLDC that targets the smallest mobile devices. MIDP-compatible optional packages include the Personal Digital Assistant (PDA) Optional Package, the Mobile Media API, the Wireless Messaging API, the Location API, the JME Web Services API, the Bluetooth API, the Security and Trust API, the Mobile 3D Graphics API, the Session Initiation Protocol (SIP) API for JME and the Presence and Instant Messaging (IM) APIs.

3.3.1.2 Connected Device Configuration (CDC)

CDC is meant for a more capable wireless device which supports a fully featured Java Virtual Machine capable of taking advantage of JSE libraries and applications that run on a range of network-connected consumer and embedded devices. Most of the packages run on the CDC profiles as well. CDC profiles and optional packages include the Foundation Profile (FP), the Personal Basis Profile (PBP), the Personal Profile, the Game Profile, the Remote Method

Invocation (RMI) Optional Package, the Java Database Connection (JDBC) Optional package and the advanced Graphics and User Interface (UI)

3.3.1.3 Application Framework

In Android, any application can publish its capabilities and any other application may then make use of those capabilities. Underlying all applications is a set of services and systems, including a rich and extensive set of views that can be used to build an application. These include lists, grids and even an embeddable web browser [http://code.google.com/android/refence/view-gallery.html].The application framework also includes `ContentProviders` that enable applications to access data from other applications or to share their data. The `Content Provider` is a data layer that acts as centralised storage for all applications and contains classes for accessing and publishing data on the device [http://code.google.com/android/devel/data/contentprovider.html].

The application framework also comprises a group of managers. A resource manager provides access to any non-code resource such as localised strings, graphics and layout files. Localised strings are names that are declared in an XML file called strings in the resources folder. In JME, strings are referenced to using java code. There is a notification manager that enables all applications to display custom-alerts on their status bar in Android. An Activity-manager manages the life cycle of applications and provides a common navigation backtracks. The framework also consists of a surface-manager that controls access to the display subsystem and includes 2D and 3D graphic layers from multiple applications.

Android includes a set of C/C++ libraries that are exposed to developers through the Android application Framework, a system C library that is a Berkeley Software Distribution (BSD)-derived implementation of the standard C system library and a Media library based on the Packet Video's Open core [Ableson 2008: 11].

3.3.2 Android and JME Runtime

Every Android application runs in its own process. The Dalvik Virtual Machine (DVM) has been written so that a device can run multiple virtual machines efficiently. Each thread has its own virtual machine. Since the Android SDK employs the Dalvik Virtual Machine, the Java byte codes created by the Eclipse compiler must be converted to the Dalvik Executable (dex) file format. The DVM executes files in the dex format that is optimized for a minimal footprint. Android is written in Java, which is first compiled to Java byte codes and then

subsequently translated to dex files. The files are logically equivalent to the Java byte codes, proprietary to Android. Android is a Java environment with a runtime which is not strictly a Java Virtual Machine. The main reason for not using the JVM is because it needs licensing from Sun Systems, so Android developers decided to use their own virtual machine. The Dalvik Virtual Machine relies on the Linux Kernel for its underlying functionality such as threading and low-level memory management. Android relies on Linux for core system services such as security, memory management, process management, a network stack, and a driver model.

MIDP, CLDC and mobile device Java technology are grouped together as the Java Platform, Micro Edition (JME). Desktop Java technology is the Java Platform, Standard Edition (JSE). MIDP is based on CLDC which is both a virtual machine specification and a set of core APIs. The Virtual Machine is more compact and has fewer features than the Desktop one. The CLDC virtual machine is called the Kilo Virtual Machine (KVM). Some people believe the K refers to the kilobyte scale of CLDC devices whilst others are content that K is short for Kaui, a project name from the early days of MIDP. Regardless of naming, the KVM has some features missing compared to the Java SE platform. It has no native methods, no object finalization and no reflection. The CDLC implementation has a class-loader that is not accessible to applications. Class files in the KVM are pre-verified at build time and the second stage of verification is run when classes are loaded on the device. Pre-verifying is when the class file is checked to confirm that it behaves well and does not damage the device or write to memory it doesn't own when it's deployed. When verifying, the size of the file is checked, as is memory usage, because of scarce memory on small devices.

3.3.3 The Content Provider

A Content Provider was used to expose or access data from other applications in the Android `route-finding` system. A `Content Provider` may use any form of data storage mechanism. The `Content Provider` is a data layer providing data abstraction for its clients and centralizing storage and retrieval routines in a single place. Figure 8 below shows how the content provider works

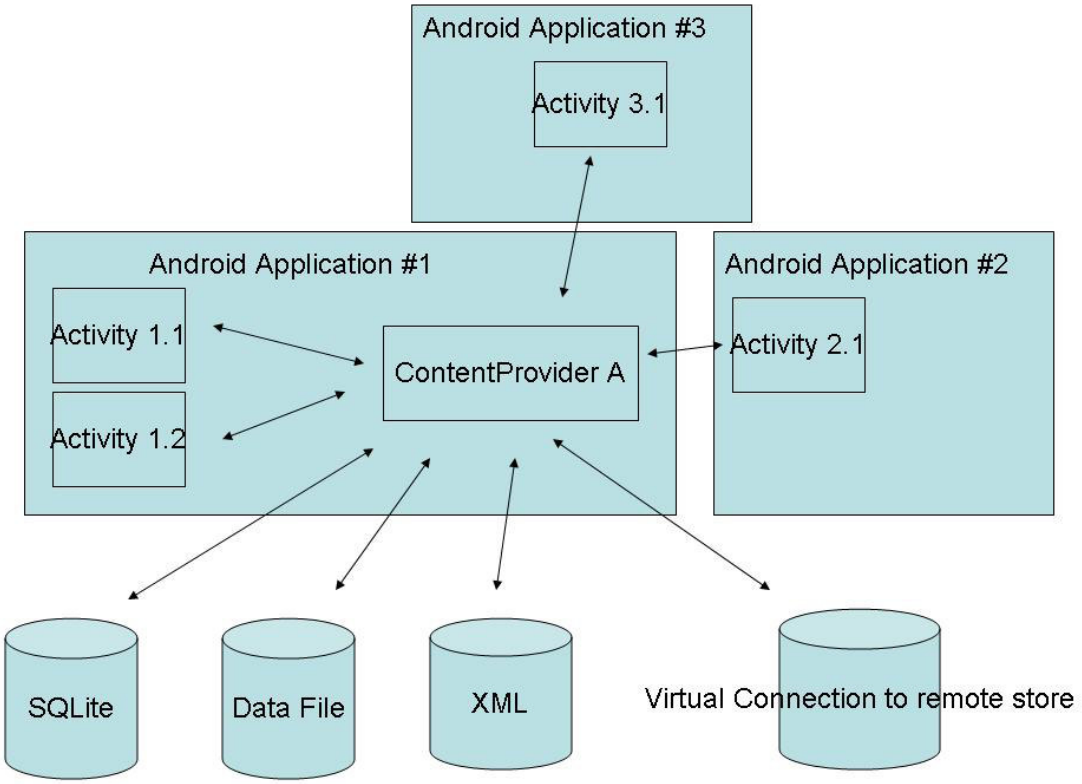


Figure 8: shows how the Android content provider works [Ableson: 2008, 21]

Figure 8 above shows the `ContentProvider` as a centralised component for accessing data from applications, databases and resources.

3.3.4 Manifest Information

One of the important files in Android is the AndroidManifest.xml file which ties all the information together in order for an application to execute. The AndroidManifest.xml file exists in root of the application directory and contains all the design-time relationship of a specific application and Intents. AndroidManifest.xml files act as deployment descriptors for an Android application, meaning that it contains information that describes the application. AndroidManifest.xml describes global variables for the package including the application components like Activities and Services. AndroidManifest.xml file uses the symbol @ to reference information in one of the resources. For example, the XML line below extracted from the route-finding system points to an icon in the *drawable* folder found in the Resources (res) directory

```
<application android: icon="@drawable/icon">
```

Xml files are processed by the Android Asset Packaging Tool (aapt) using the automatically generated R.java file. The R.java class is needed for reference when connecting the code to the User Interface.

In JME, manifest information is included at the packaging stage. Packaging in JME is done because you can't pass classes directly to a MIDP for deployment [Knudsen & Li, 2005: 23]. Packaging is done automatically when using JME Wireless Toolkit. Every Java Archive (JAR) includes a manifest file that describes the contents of the JAR. The information in the manifest is important to the MIDP at runtime because the JAR file manifest contains attributes that fully describe the MIDlet suite. The information includes the class-name and the version of CLDC and MIDP such as shown below

```
MIDlet-Vendor: Jonathan
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.0
MicroEdition-Profile: MIDP-2.0
```

The above shows the name of the vendor, versions of the CLDC which is CLDC-1.0 and MIDP-2.0.

Although both platforms must have a manifest file that serves a similar purpose, the main difference is that, in android you have to code the manifest file, while it's automatically created for you in JME by the wireless toolkit.

3.3.5 How JME Relates To Other Java Platforms

Although its cross-platform nature is the main concept behind the Java philosophy, there are four different editions of the Java platform which each has a significant role in mobility.

Java Standard Edition (JSE) - the basis for the Java platform which defines the Java Virtual Machine (JVM) and libraries that run on the standard personal computers and workstations

Java Enterprise Edition (JEE) **-** the JEE includes the JSE and some APIs, containers and tools. JEE application servers can drive browse-based (e.g. WML and xHTML) mobile applications to be service endpoints for smart mobile clients.

Java Micro Edition (JME) **-** is the ideal mobile client application designed for wireless mobile devices. It contains specially designed, lightweight virtual machines, a bare minimum of

core-class libraries, and lightweight substitutes for standard Java libraries. Figure 9 below shows the architecture of the Java platform.

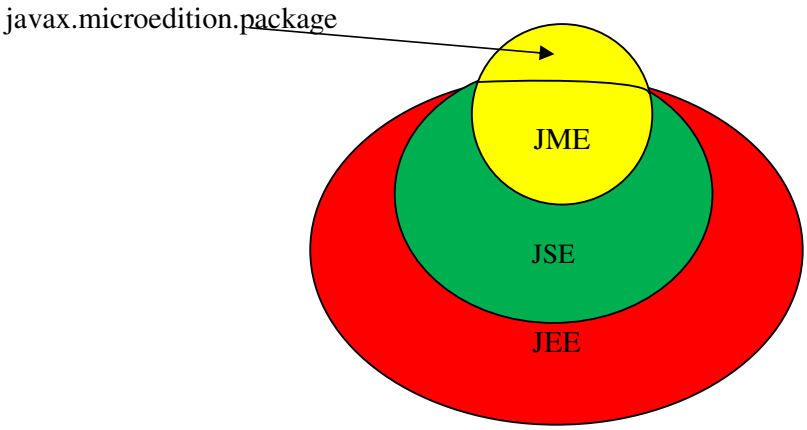


Figure 9: shows the architecture of the Java 2 platform [Yuan, 2004: 19].

JME comprises the lightweight edition of the popular JEE frameworks.

Java Card is for applications that run on smart cards.

3.4 Developing the JME and Android Route-finding Systems

3.4.1 Developing the JME System.

The GPS functionality in JME consists of three main methods.

1) People can query for their current location. This is handled by the `findMe` method. With this method, people are shown their current location on a map.
2) People can also find a certain physical location by entering the physical address of the location and then a route is drawn from their current location to their desired destination. An overlay of their current position is also included. This is all done by the `Find_Address` method.
3) The third feature is the `Find_Business` method, people can query for the nearest business category in a given radius from their current location and then a route to that place is overlaid on top of a map that is shown on the screen of their mobile phones. How all these methods work is illustrated in the flow charts below:

JME flow chart for the `findME` function

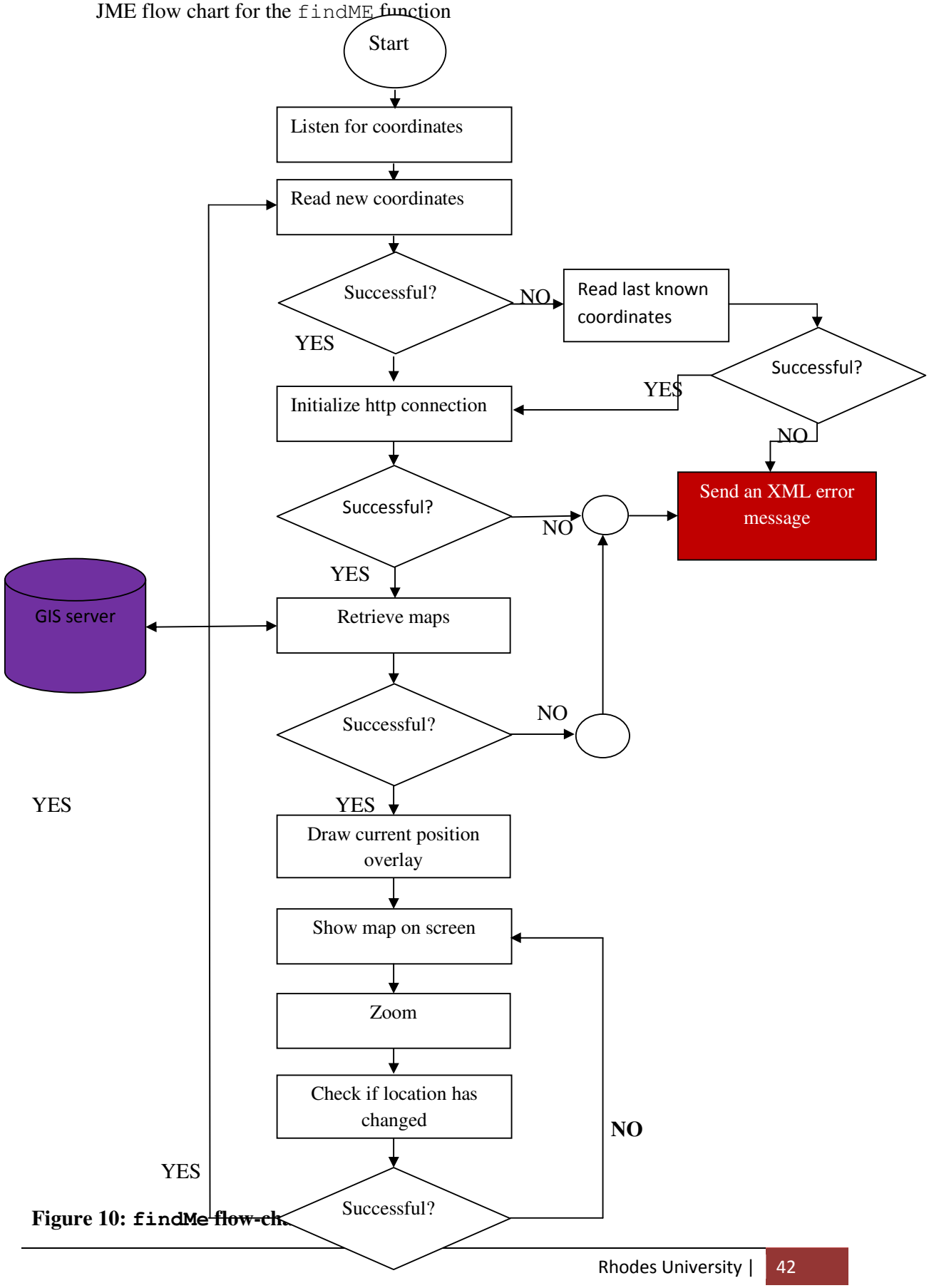


Figure 10: `findMe` flow-ch

When the `findMe` method is running, the built-GPS receiver on the phone listens for GPS coordinates from the satellites. If there are no coordinates, the last known coordinates are used and if they are not available, an error message is displayed on the screen. If there are coordinates, the application uses the coordinates to retrieve a map and draw an overlay of the current position on top of the map. The map is displayed on the screen of the phone with the current location at the centre of the screen. Every time the phone changes location, the phone has a method that reads the new coordinates and updates the current location on the phone. The process is repeated until the user either exits the application or chooses to start another function.

The other two functions are for finding an address and for finding the nearest business and can be illustrated by the same flow chart except that the information that is entered is different. The flow diagram for these two functions is shown in Figure 11 below

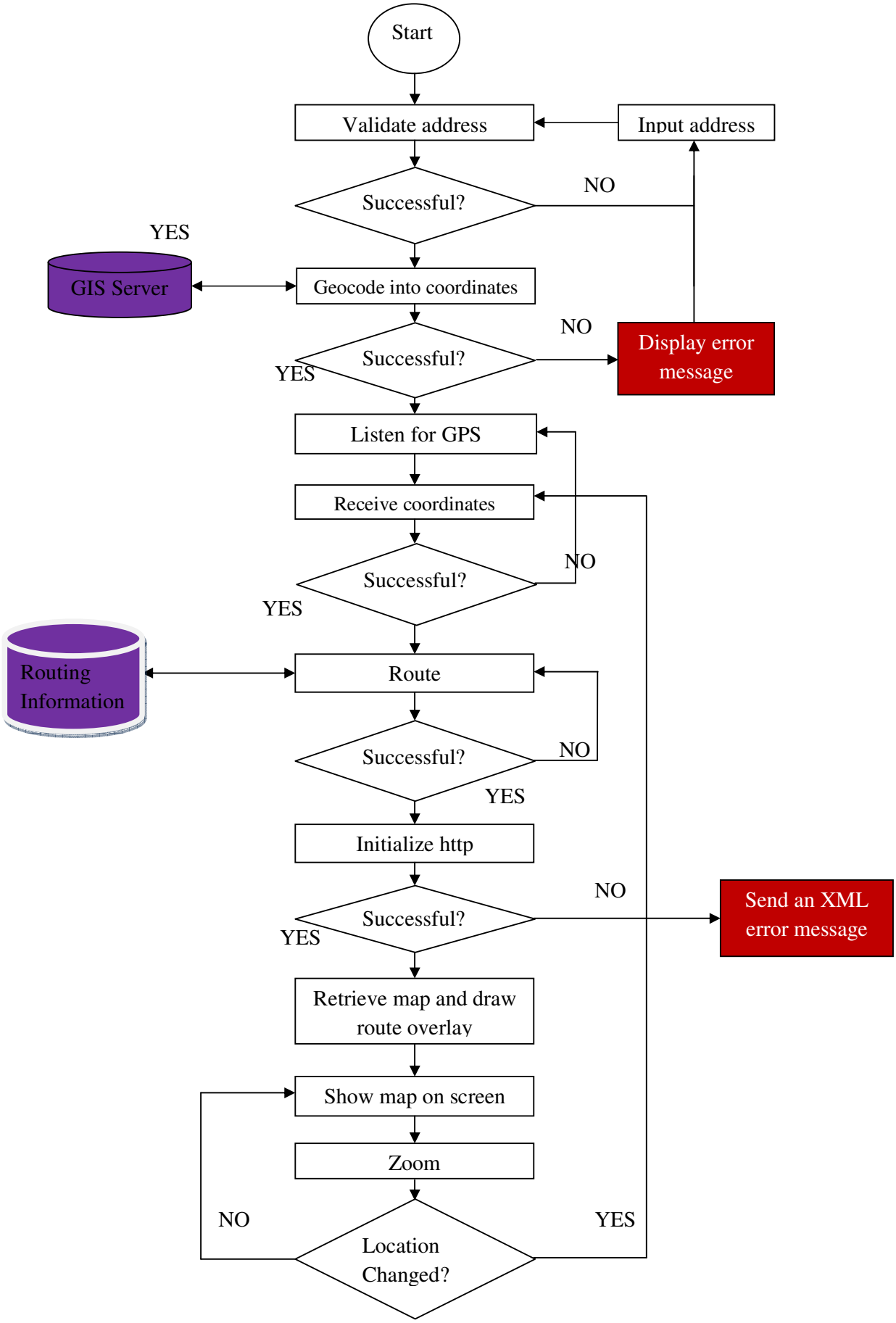


Figure 11: `Find_Address` flow-chart which is also similar to the `Find_Business`

Figure 11 above shows how the `Find_Address` and `Find_Business` classes work. People are prompted for the physical address or the category of business they are looking for. If they are looking for a certain business category, they are also supposed to specify the radius from their current location. This retrieves the nearest business. After entering the address, it is validated to see if it's in the correct format. If the format is not valid, the user is prompted to enter a correct address. The address is then resolved into coordinates by geocoding. To resolve the address into coordinates, we have to send the address through an `HTTP` connection to a GIS server. The server then returns the coordinates that correspond to the address. After retrieving the corresponding coordinates from the server, we then listen and receive coordinates from GPS showing our current location. With the coordinates of our current position, we can now retrieve route information to our destination. We store the coordinates that link us to our destination and then use these coordinates to draw our route on top of a map showing both our current location and our destination. After drawing the route, we then display the map with the overlays on the screen of the phone. When our position changes, we have to redraw our position on the map so that our current position will always correspond to the drawing on the map. The other features of these two classes are that they give alerts every time they come near to a known landmark and give information about that landmark.

Figure 12 below shows the class diagram of the route-finding system and how it works. We wrote all the classes below.

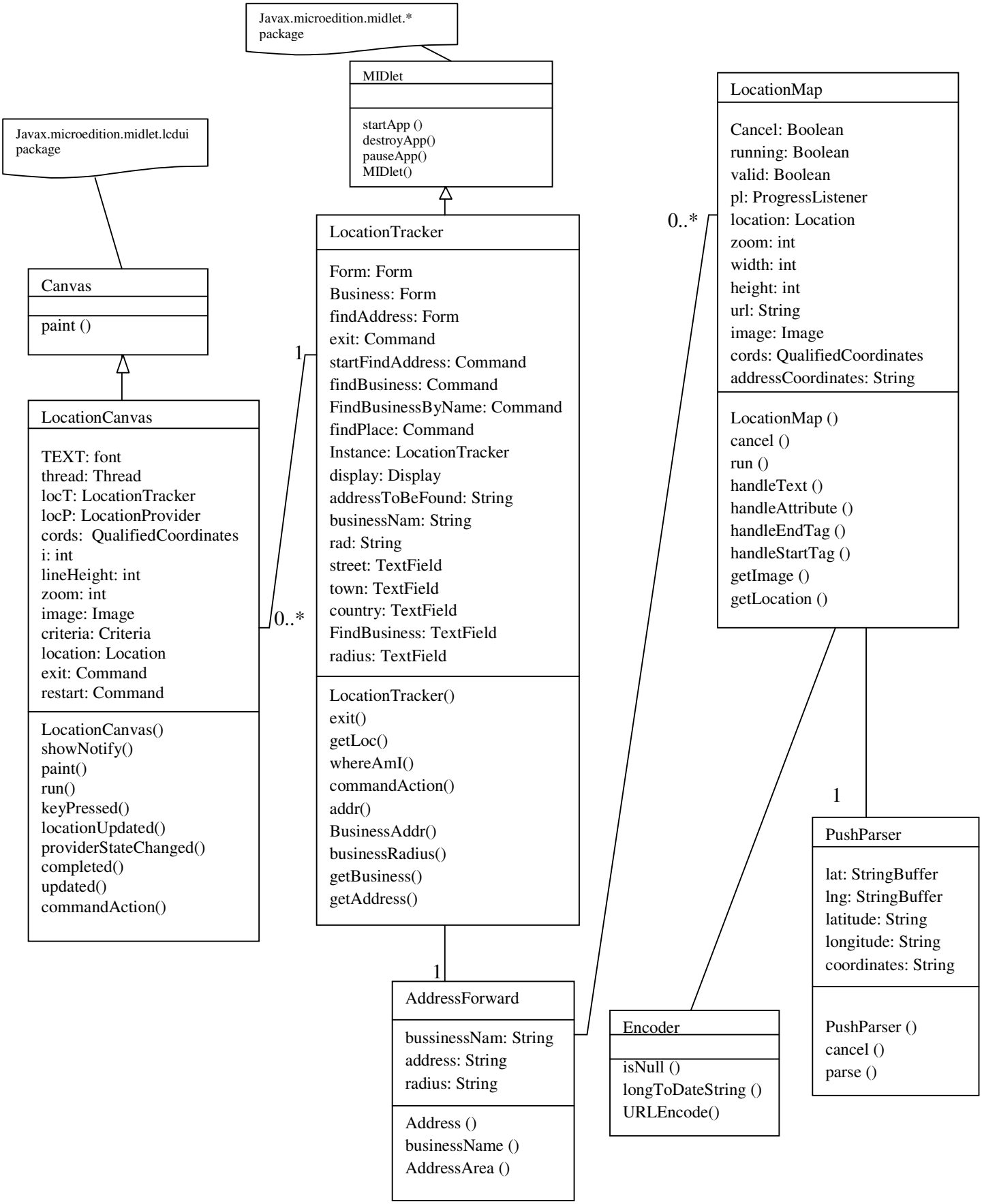


Figure 12: shows the class diagram of the system

Applications developed for the route-finding system in both Android and JME has similar features. These make the analysis of the two platforms easier. In the next section we analyze the development of these features.

3.4.2 Graphical user interfaces

`List` and `Forms` are the main classes that provide interactive displays for small devices in JME. A `List` shows many items that can be displayed on the screen. A `Form` can be used to hold different things including images, textfields, items, gauges and spacers [Kick Butt, 2007:66]. A `Canvas` is another base class for custom screen that was used for displaying the map on the screen. A `Canvas` can be used to create your own screen, draw text, shapes, images and respond to incoming events [Kick Butt, 2007:71]. All the above user interfaces are found in the `javax.microedition.lcdui` package.

In Android, an application with a user interface will have at least one `Activity` [Ableson, 2008:51]. For every user interface screen there is an `Activity`. To present the actual user interface elements to the user, a `View` object is used. `Views` handle what the user will see. These include text elements for labels and feedback buttons and Forms for user inputs and draw graphics to the screen. The AndroidManifest.xml file provides reference to parts of the application to the platform. Without this file, the application will not be able to start. The `setContentView ()` is used to associate an `XML Layout View` file. `Layout` and `View` configurations can be defined in code as Java objects, but it is easy to use both code and XML files in the sense that you have to reference to the XML file using java code and then use the XML file to create the proper view which you want. When using both code and XML files, static views like labels do not need to be referenced in the code. Only views that need runtime manipulation must be referenced in code. When creating the user interfaces in Android, forgetting to declare the view in any of the files will result in an error and one difficulty is that the error message does not point where exactly the error has occurred. Normally a null-pointer exception is thrown because the view is not found in one of the files.

Because of the large number of files needed to show one view in Android, it is more difficult to create user interfaces in Android than in JME. The more complex the process of creating user interfaces, the better the view however, so Android user interfaces look more attractive than the ones in JME.

The relationship between the objects mentioned above is illustrated in Figure 9 below

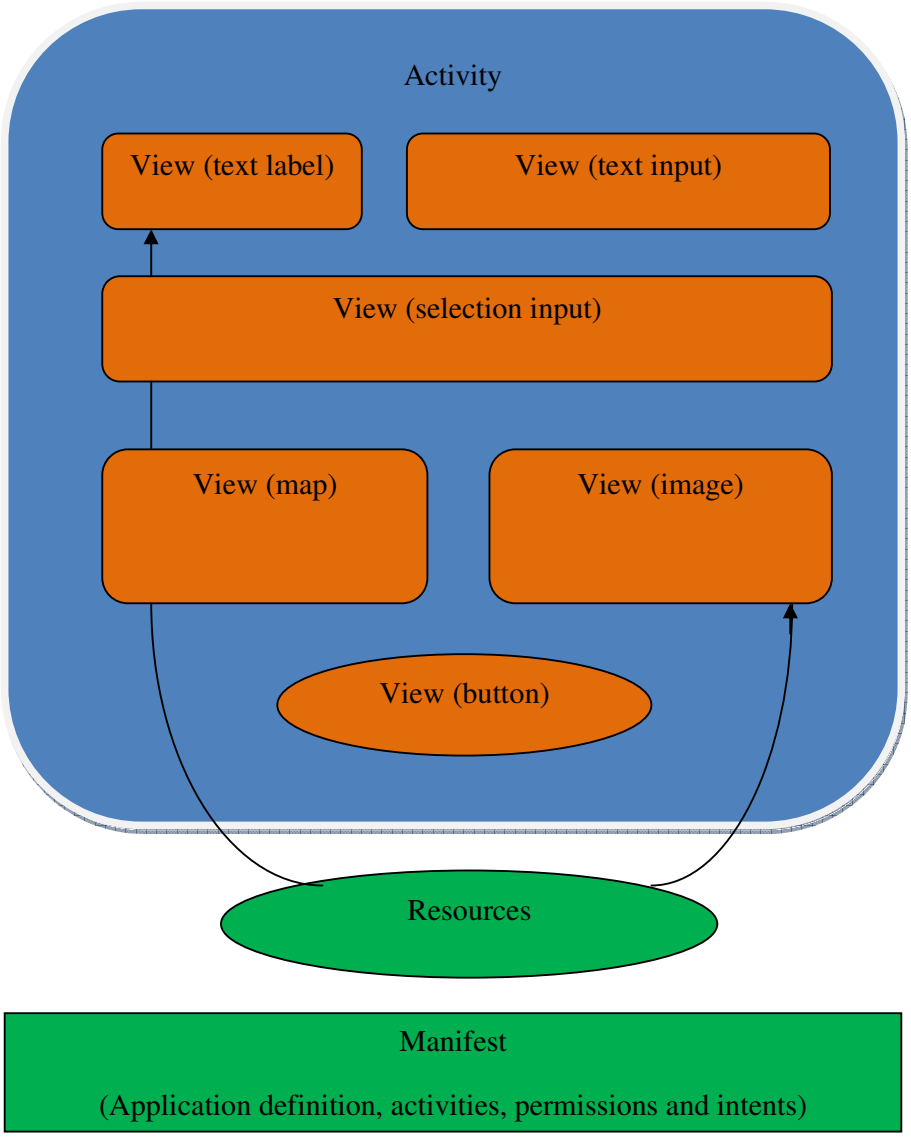


Figure 13: depicts the relationship between Activities, Views, and resources [Ableson, 2008: 51]

Figure 13 above shows how the view class is linked to all the other objects. In the above, if you want to show a view of an image, the image must be retrieved as a resource. The resource can either be a database, server, `HTTP` connection or the resources (res) folder in the project. Also static views like labels are stored in the resource folder as XML files. All the

other views can be referenced by either java code or XML files in the resource folder. The `Manifest` is the base for all the objects, which means that it must be included on everything you do in Android.

The `View` API consists of a set of `View` objects in the `android.view` package. The overview of the `View` API is shown in the class diagram below

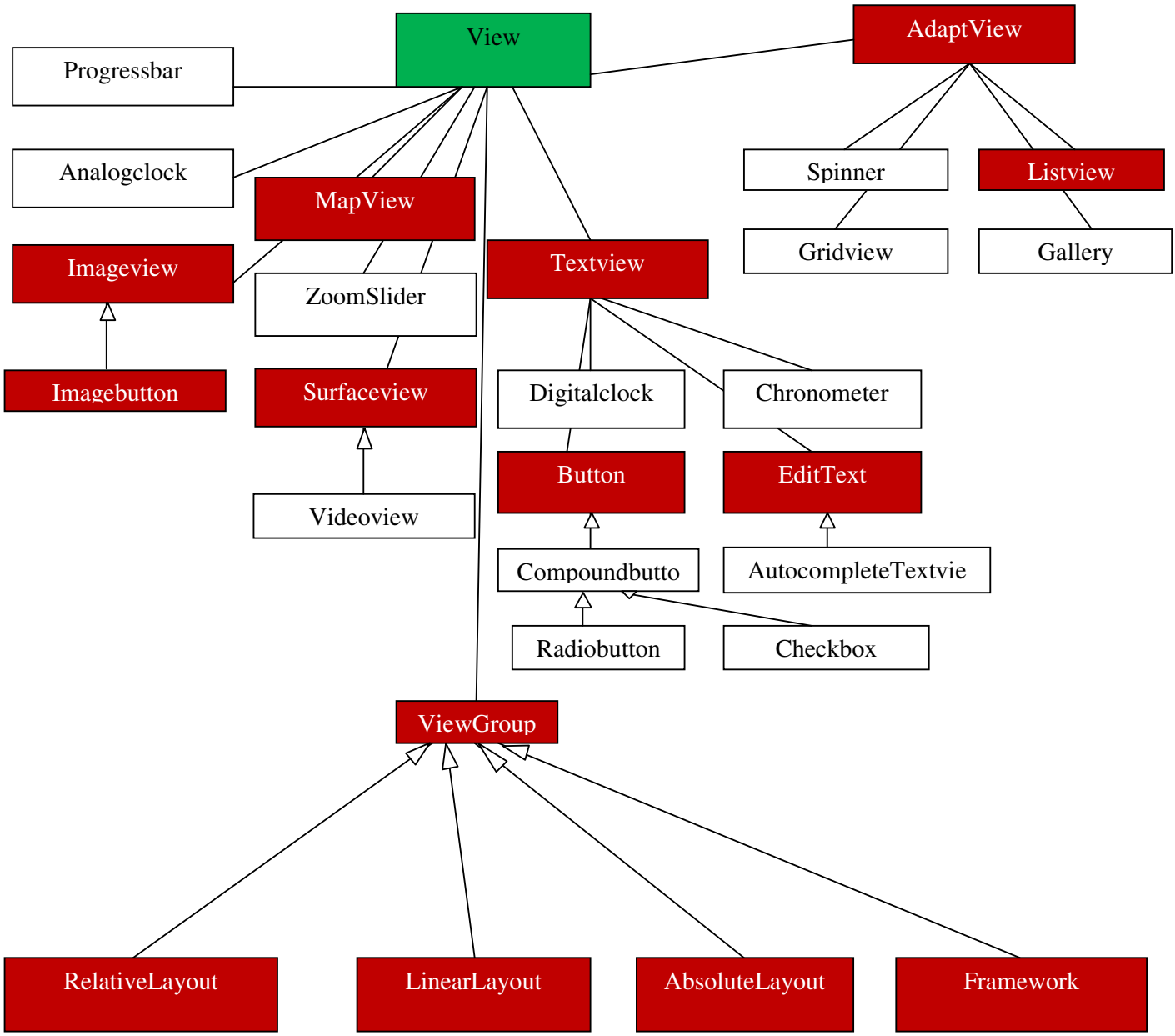
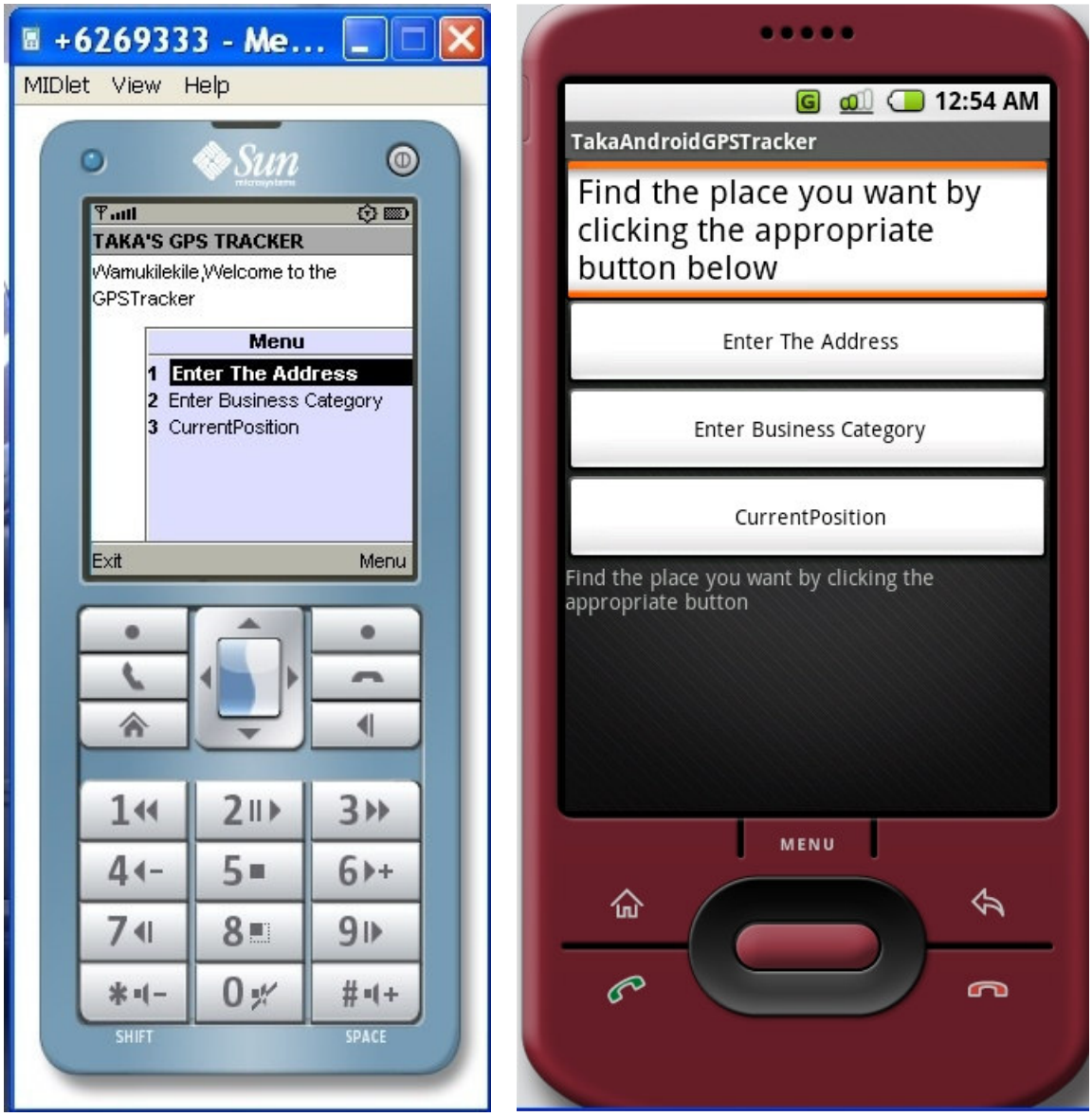


Figure 14: The class diagram showing an overview of most of the View API.

The above diagram shows most of the views in the `android.view` package. The ones with a red background were used in our implementation of the Android system.

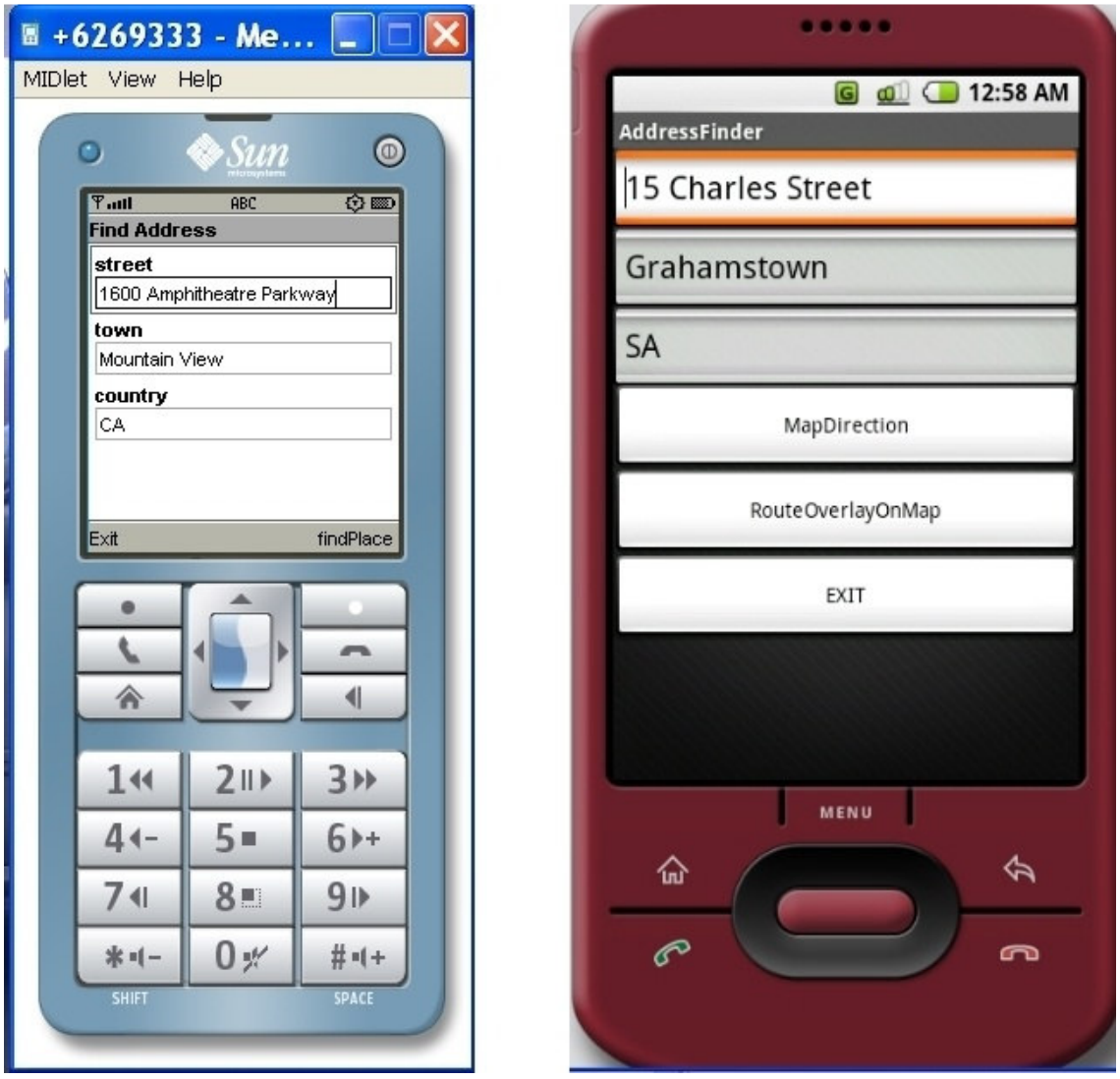
Figures 15-17 show the graphical user interfaces in both systems. Figure 15 shows a `List`. The `List` is implemented in JME by using java coding only and in Android by both java code and XML files. The advantage of using XML files is that you can specify the layout, view and properties of the elements. XML files are there for defining how exactly the view must look like. Although using JME code to show graphical user interfaces is easier because it's only one line that references an already defined view, the view created by the XML files in Android has a better appearance and people can create user interfaces they like. This is because of the use of `XML-layout` files and the JME emulator for the default color phone does not have the individual detail that the Android-specific emulator does have



JME Android

Figure 15: user interfaces in JME and Android (`List`)

Figure 16 below shows the use of Forms to input the physical address to be found. In JME we use `Textfields` for typing in data and `TextEdit` for Android. They are both similar in their use and how information is read from them.



JME Android

Figure 16: shows screenshots of how forms are used in both JME and Android

The last important graphical user interface is the display of the map. In JME we use a `Canvas` to view the map. From the previous sections, we have seen that a `Canvas` can be used to view a lot of things including images. In Android, the `MapView` object is used to view maps. This object is supposed to be referenced in an XML file found in the resources folder. Without being referenced the maps will not be displayed. The map object is found in the `com.google.android.maps` package. Using the `Canvas` allows the map only to be appended and displayed. There are no methods for manipulating maps in JME. We used a `MapActivity` to construct a `MapView` because a `MapView` depends on threads which access the network and file system in the background and these threads are controlled by the life cycle management in the `MapActivity`. The `MapView` gives the map a handle that

can be used to drag the map. Using the `MapView` is easier as compared to the `Canvas` in JME because the `MapView` access Google servers directly without hard-coding. The `MapView` has more functionality as it allows overlays to be drawn on top of the map.

Figure 17 below shows the display of maps

JME `Canvas` **Android's `MapView`**

Figure 17: shows the Map `Canvas` for JME and the `MapView` for Android

3.4.3 Accessing GPS coordinates

Both systems access GPS coordinates. These coordinates are used to locate the current position of the phone. The mobile phones that use these two systems must have an in-built GPS received. Without the receiver, a null pointer exception will be thrown. In our case, we were able to deploy the JME applications on a Nokia N82 as well as in the emulator. As mentioned earlier, there is no phone for Android that has been released yet. For this reason we were developing and testing both applications on emulators. The emulators use mock GPS coordinates. To accesses the GPS coordinates, you have to specify the criteria which you want to use and pass these criteria to the location-provider. The criteria specify features like power consumption, accuracy and location providers; in this case we were dealing with GPS coordinates only. In JME, java code is used to set the criteria, whilst an XML properties file is used for Android.

```
requiresNetwork false
requiresSatellite true
requiresCell false
hasMonetaryCost false
supportsAltitude true
```

Figure 18: shows the Android's properties file

Figure 18 above shows an Android properties file for a GPS provider. The above criteria doesn't require a network connection but there must be a satellite connection since it needs to receive coordinates from the satellites

```
criteria = new Criteria ();
criteria.setHorizontalAccuracy (0);
criteria.setVerticalAccuracy (0);
criteria.setCostAllowed (false);
criteria.isAltitudeRequired ();
```

Figure 19: shows how the criteria is set in JME

One advantage of using an XML file in Android for setting the criteria is that, only one file will be created and every application that uses location information will use those criteria whilst in JME, criteria are specified in every application that is created. Using the XML file in Android is time-conserving and easier than specifying a criterion every time you create a new application.

The javax.microedition.location package is used for accessing the location in JME and the android.location package is used for Android.

3.4.4 Network connections

Both systems use HTTP connections to send the physical addresses entered by the user to GIS servers for geocoding. The javax.microedition.io http-connection is used to set a connection between the map_location class and the server. This connection is used to send the address and receive the geocoded results in either XML or CVS format.

Android uses Intent from the android.content package to start an HTTP connection. For geocoding, Android uses the android.location.Geocoder that accesses the Google servers directly over an HTTP connection and retrieves information about the current location.

3.4.5 Handling Dynamic XML

After geocoding, an XML or CVS document is sent in the JME application. This will contain the information about the destination. To access this information in JME, the document must be parsed and the application may then extract the information that you want to use. The kXML parser was used for JME. We also had to implement some java code for obtaining the latitude and longitude coordinates.

In Android, KML files are sent back with the information from the Geocoder. No user-written code is required for parsing the document. Android has a Geocoder method that takes care of the receiving of the information from the geo-database and the name of the method is Geocoder ().

3.4.6 Route-finding

Finding the appropriate route to use is achieved in Android by going to the com.google.googlenav.DrivingDirection library and this must be declared in the Manifest file for it to function. The DrivingDirections object is used to retrieve the route in the form of coordinates. The coordinates are then stored in an array and then later used to draw the route as an overlay on a map. In JME, routing is done by sending the information to a server and then retrieving the optimum route to destination.

The other function that is implemented in both systems is the ProximityListener. This function alerts the user when he nears a certain landmark.

3.4.7 The Location API

JSR179 API gives access to location data for JME applications. This API provides information about the current location and its surroundings. JSR179 mainly consist of the `Criteria`, `LocationProvider`, `LocationListener`, `ProximityListener`, `Landmark` and a `LandmarkStore`. The `Criteria` gives the specifications for choosing a location provider. The `LocationProvider` provides us the available services that can be used to access the location. The `LocationListener` is used for receiving notifications when the location of the mobile device has changed. The `ProximityListener` is invoked when the devices comes near to a certain landmark.

`Android.location` consists of the classes that define Android location-based and related services. Android has all the classes that are in the JSR179 API. In addition to these classes, `android.location` also consists of the `Address` and `Geocoder` classes. Because of these added functionality, Android is better equipped than JME. The `Address` classes are used to store an array of locations and the `Geocoder` is for handling geocoding and reverse geocoding. Geocoding is the process of transforming human readable physical addresses of a location such as a street address into a (latitude, longitude) coordinate. Reverse geocoding is the process of transforming a (latitude, longitude) coordinate into a physical address.

When comparing the location APIs provided by Android and JME, we can see that Android has more classes than JME. Geocoding and reverse geocoding in JME is done by accessing geocoding information through the network which is slow relative to the current location of the device. Android has got a `LocationManager` that provides access to the system services allowing applications to obtain update of the device's geographical location and firing intents when reaching certain proximity. The use of the `LocationManager` makes it easier for developers because they don't have to code the `ProximityListener` and the `LocationUpdated` methods like what is done in JME

3.4.8 Drawing overlays

The `com.google.android.maps.Overlay` package is used for drawing overlays that are displayed on top of maps in Android. The `draw ()` method used for drawing overlays consist of a `Canvas` that has already applied transformations. The transformations are used to map the overlays on to their accurate position on the map. One advantage of Android over JME is that, Android has got an already defined method called

`MyLocationOverlay` found in the `com.google.android.maps` package. This method automatically draws and updates an overlay of the current location of the phone when it's called. The `Canvas` in JME allows basic drawing and does not have specific methods for drawing map overlays. Considering that most of the overlay drawing is done for you in Android whilst in JME you have to do the coding yourself, it reflects that drawing overlays in Android is much easier than in JME. When it comes to drawing overlays, Android has more functionality than JME.

3.5 Chapter Review

In this chapter we looked at the design and implementation of the route-finding systems in both JME and Android. We analysed the development of the two systems. We started by looking at an overview of both JME and Android. We then went into the logical-flow diagrams and flow charts for the systems. We also looked at the class-diagrams and, lastly how the APIs in both systems were used.

Chapter 4: Analysis and Evaluation

In this Chapter we take a look at how the emerging Android development platform compares to the well-established JME development environment. The comparison will assist developers to choose the right tools for developing data services. We try to identify how best to use each platform's strength and avoid its weaknesses for the development of mobile data services mainly focusing on Location-based services. To achieve this, several key features of the route-finding system were tested. The results of the test are the main focus of this chapter.

4.1 Language

Both Android and JME use java for coding applications. Android has more functionality because it has more classes than JME. Java is very accessible because of its popularity. Many developers use Java and both platforms use Java because of its ability to perform well in devices with limited memory and constrained capabilities. JME use the KVM for compilation, whilst the Dalvik Virtual Machine is used for Android. Android files are converted to Dalvik Executable (dex) files before compilation. Android does not use the standard JVM because of the licensing rules from Sun Microsystems.

4.2 IDEs

Android	JME
Eclipse	Eclipse, NetBeans, SunOne Studio

Table 3: shows supported IDEs

Android can be developed only in one IDE (Eclipse) for the time being. JME has more support when it comes to IDEs. One of the reasons for this is that JME has been around for a longer period when compared to the emerging Android. In this project we used Eclipse to develop both applications because Eclipse supports the development of both Android and JME applications. Eclipse provides:

1) Code auto-completion
2) Syntax highlighting
3) Refactoring
4) Debugging
5) Obfuscating
6) Packaging

4.3 GUI Designing

Android	JME
Programmatically create user-interfaces	Programmatically create user-interfaces
1 line of code	1 line of code
2 XML files (1 for the layout and 1 for referencing)	No XML files required

Table 4: shows a comparison of user-interfaces

Both platforms enable the programmer to code user interfaces and they each use the same number of lines of code to accomplish this. The Android visual designer offers more choices. For every line of code that you create, there is an XML layout file that defines the view of the user interface. This offers flexibility as it allows users to organize the view themselves.

JME has the advantage in that it does not require XML files for the layout but the XML files in Android do offer more flexibility to developers.

4.4 Accessing GPS

Android	JME
In-built GPS receiver	In-built GPS receiver
Required 2 lines of code	Required 11 lines of code
Required 1 XML file for properties	No XML file required
The system is responsible for updating location changed	Location updating has to be coded

Table 5: compares direct access to GPS

Both Android and JME have in-built GPS receivers. Android required 2 lines of code whilst JME required 11 lines of code to access GPS. The `LocationManager` in Android gives access to system location services allowing applications to get updates of their geographical

location and also fire notifications of proximity to particular places, whilst in JME developers have to hard-code the location update method and the `ProximityListener` a method to notify proximity.

Figure 20 below shows the number of times both applications access GPS per second.

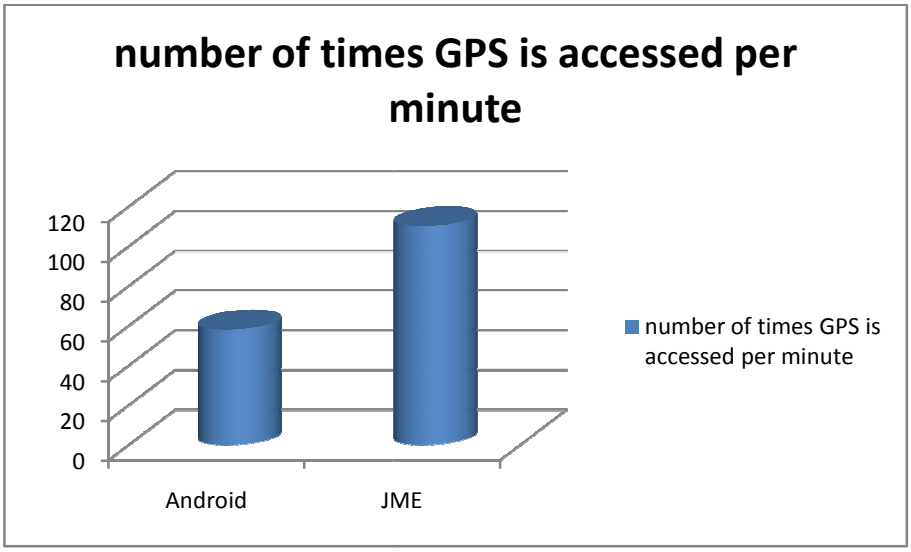


Figure 20: shows number of GPS access by both Android and JME

Figure 20 above shows that JME applications accesses GPS functionality more than twice as often for the same result as Android applications do. A JME application accesses GPS 110 times per minute whilst an Android application access 58 times per minute. This is because the GPS access rate in Android is set to a maximum of 58 times per minute.

Android is recommended for accessing GPS because it uses fewer lines of code, does not require developers to create methods for updating location details and uses the same XML properties file for any GPS application that will be created. Developers in JME, however, have to hard-code the access criteria every time they create a new GPS application. JME is highly recommended when creating applications that will be used in a fast-moving environment since it access GPS more than twice as much compared to Android. This is an advantage because it allows faster response.

4.5 Handling Dynamic XML

Android	JME
Programmatically parses the XML document using either DOM, XMLPullParser or SAX	Programmatically parses the XML document using either DOM, XMLPull or SAX
16 lines of code	34 lines of code

Table 6: shows a comparison of Dynamic XML handling in both platforms

Since both platforms use Java, they have similar approaches to parsing XML files. They both use the same methods for parsing. The only advantage of Android here is that it uses few lines of code to parse and extract any information from the XML file.

4.6 Route finding

Android is recommended for route-finding because it has a `DrivingDirections` class that automatically retrieve routes when given two geographical coordinates. Google utilises Google maps and has an inherent advantage in that it comes out of the same stable as Google Maps.

In JME you need to access GIS servers to retrieve a route, which is a result of an older approach and the lack of a built-in connection to, for example, Google Maps.

4.7 Retrieving maps

In Android, a `MapView` is used to display maps and can be controlled programmatically. The `MapView` allows developers to draw a number of overlays on top of the map. `MapActivity` is used to construct a `MapView`. The `MapActivity` depends on threads which access the network and file-system in the background. Because of these features of the `MapActivity`, developers do not have to create `HTTP` connections for accessing GIS servers, unlike in JME. To display features on top of maps, an `OverlayController` must be used that adds overlays. The `com.google.android.maps.Overlay` package

provides a `PixelCalculator` capable of converting (latitude, longitude) coordinates into an onscreen pixel coordinate. The `PixelCalculator` is used to map the current location of the phone to the centre of the phone screen.

In JME, the map is appended to a `Canvas`. The JME `Canvas` does not have direct access to the network, meaning that developers have to access the network by hard-coding and retrieving appropriate maps. A static map is retrieved and is displayed on the screen of the phone.

Figure 21 below shows the time taken to retrieve maps in both Android and JME

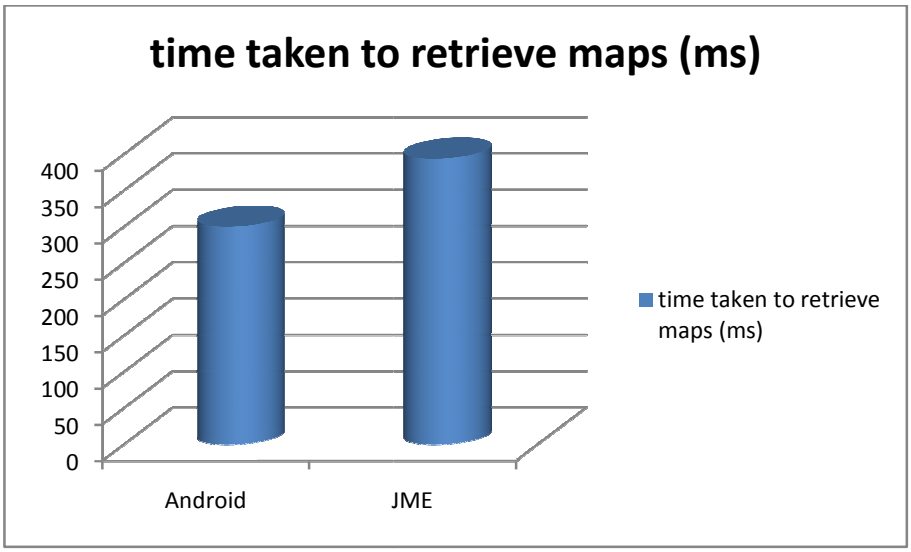


Figure 21: shows the time taken to retrieve maps by both applications

It takes an Android application 300 milliseconds to retrieve a map whilst a JME application takes 394 milliseconds. Android is thus slightly faster in retrieving maps than JME.

Android is recommended when creating applications that include the retrieving of maps because developers don't have to hard-code access to network connections. This decreases the time needed to create and debug an application. Network connections are provided by the `MapActivity` class and they allow applications to automatically retrieve maps, access GIS servers and carries out the geocoding process. Android retrieves maps faster than JME by an average of 0.094 seconds.

4.8 Geocoding

Geocoding in Android is done by the network connections under the `MapActivity`. Geocoding is done automatically in Android by a `Geocoder` () method whilst in JME,

geocoding is done by accessing GIS servers over `HTTP` connections. The `HTTP` connections in JME are manually coded whilst the ones for geocoding in Android are already defined for developers and they connect directly to Google servers.

Figure 22 below shows the time taken for geocoding in both platforms

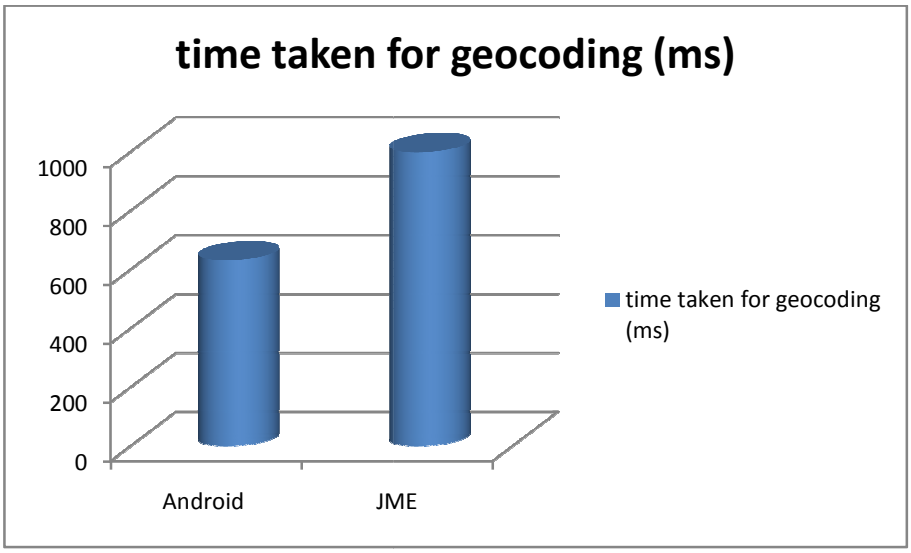


Figure 22: shows the time taken for geocoding in both platforms

Figure 22 above shows that it takes an Android application 631 milliseconds to geocode an address whilst it takes on average about 998 milliseconds for a JME application.

Android is recommended when using geocoding applications because of the fewer lines of coding required. The methods for accessing servers and geocoding are already defined for developers. This makes it easier for developers to create and debug applications.

4.9 Emulator Platforms

The *Qemu* version 0.8.2 emulator platform provides a virtual ARM mobile device on which developers can run their Android applications. When developing JME applications, Sun's Wireless Toolkit and MIDP emulator platforms can be used. The Qemu emulator and the Wireless Toolkit were used for the development of the Android and JME applications respectively.

Figure 23 below shows the Qemu and Wireless Toolkit emulators used in the development of the applications.

The Qemu emulator The JME emulator

Figure 23: shows the emulators that were used for developing applications in the project

4.9.1 Emulator Limitations

Although emulators simulate real phones, they have many limitations. Some of the limitations are listed include

- No support for placing or receiving actual phone calls although developers simulate phone calls (placed and received) through the emulator console in Android and by invoking external invents in JME.
- No support for USB connections
- No support for camera/video capture (input).
- No support for device-attached headphones in Android but developers can use head phones connected to the computer to listen to sounds produced in the JME emulator.

- No support for determining connected state
- No support for determining battery charge level and AC charging state
- No support for determining SD card insert/eject
- No support for Bluetooth

Despite the above limitations, both emulator platforms provide features for memory analysis, network management and monitoring. JME gives a better feedback in the form of tables and graphs as shown below

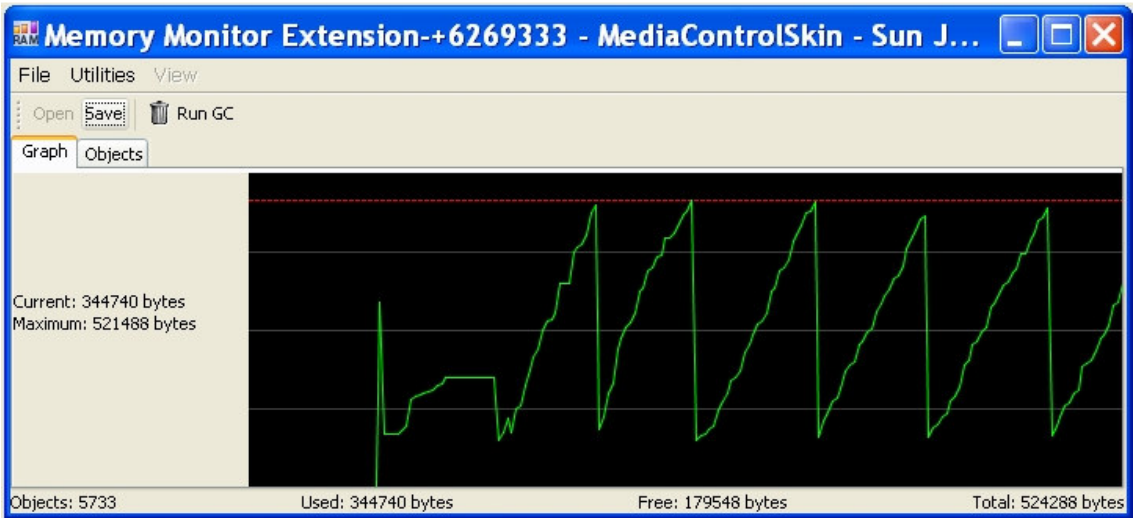


Figure 24: shows the memory monitor in JME

Figure 24 above shows the memory monitor from JME with a current memory use of 344740 bytes

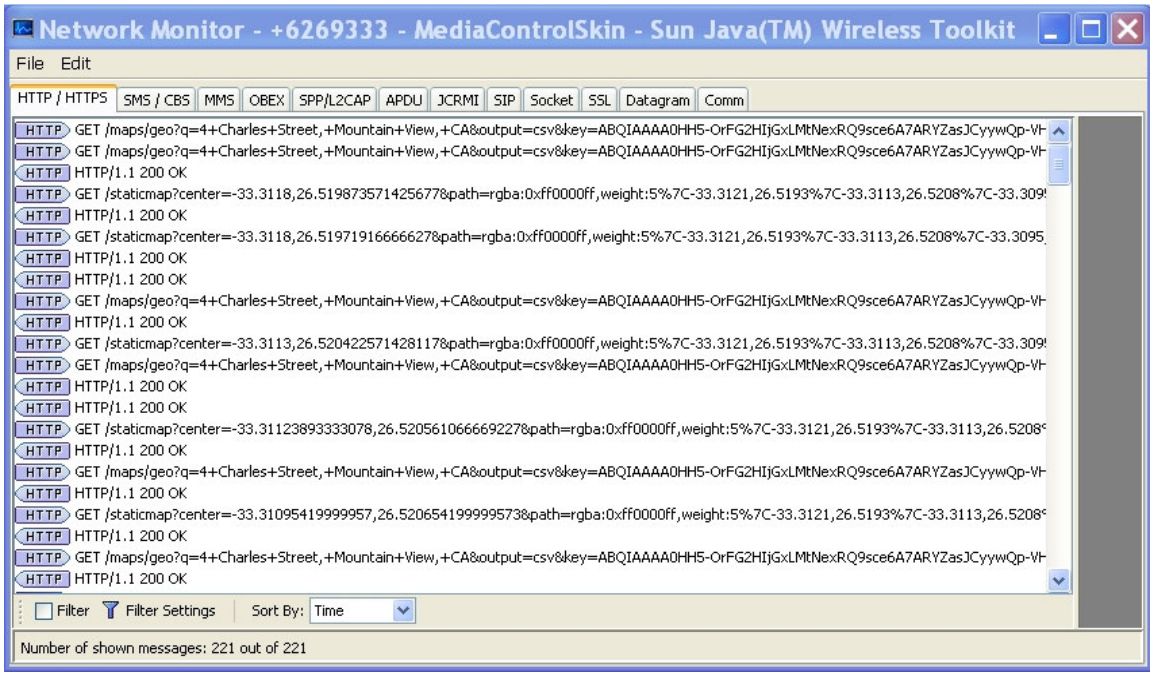


Figure 25: shows the network monitor in JME

4.10 Packaging and Deployment

In JME, a MIDlet is packaged into a jar file. A `jar` file has to be accompanied by a `JAD` (Java Application Descriptor) file that contains attributes that fully describe the MIDlet suite. Applications can be transferred from the computer to a mobile device by either a USB cable or Bluetooth. Over The Air (OTA) connections can be used.

In Android, the Android Asset Packaging Tool (aapt) is used to package an `apk` file that constitutes an Android application. No applications have been transferred to an Android phone yet since there is no Android phone by the time of this writing.

Both platforms produce small packaged size files and the size does not seem to vary with the lines of code but more with the number of files. The JME `Jar` file is 24kb whilst the Android `apk` file is 43kb. The size of the Android `apk` might be bigger than that of the JME `Jar` file because the Android application has extra XML files that are not found in JME.

4.11 Chapter Review

In this chapter we looked at the strength and weaknesses of both JME and Android. This allows developers to make trade-off decisions when developing applications. We did an

extensive analysis between both mobile application development environments. We looked at the lines of code that is needed to carry out a given task. We also looked at the speed of execution of applications and found out that most of the Android applications were a little bit faster than JME applications.

Chapter 5: Conclusion and Possible extensions

This chapter concludes the project by making a summary of the analysis and offering possible extensions of the project

5.1 Conclusion

Location-based services emerged due to the curiosity of humans about their current location, details of their environment and how they might move from one place to another. LBS provide users of mobile devices services according to their current geographical location. Commercially LBS are important because they open a new market for developers, cellular network operators, and service providers to develop and deploy value-added services

LBS answer three main questions

- Where am I?
- What is around me?
- How do I get there?

To answer the above three questions accurately, developers must have the right tools to make it easier and more efficient to develop, deploy and manage location-based applications on mobile devices. As a result of the hunt for better tools, new mobile platforms like Android are now emerging and joining the competition with well established platforms such as JME.

Both Android and JME have key features for the development of location-based services. Although both platforms can be used for developing applications, they have unique features. The route-finding systems created enabled a qualitative and quantitative comparison to be made, resulting in trade-off decisions based on the strength and weaknesses of the platforms. Developers can draw conclusions as to which platforms to best use for the development of particular mobile applications.

In general, JME allows for more rapid creation of applications because it does not require XML configuration files. Android provides more control because of the XML files and has more functionality because of its numerous APIs. Table 7 below shows the classes that are found in both JME and Android

Android	JME
Address	AddressInfo
Criteria	Criteria
Geocoder	*
Location	Location
LocationManager	
LocationProvider	LocationProvider
LocationProviderImpl	*
Geopoint	Coordinates/QualifiedCoordinates
*	Orientation
*	Landmark? LandmarkStore
ItemizedOverlay	*
MapActivity	*
MapController	*
MapView	*
MapView.LayoutParams	*
Overlay	*

OverlayController	*
MyLocationOverlay	*
OverlayItem	*
TrackballgestureDetector	*
TouchgestureDetector	*

Key

Class absent in the platform: *

Table 7: shows the classes that are found in each platform

Table 7 above shows that Android has many more classes for location-based services than JME.

5.2 Project Achievements

The applications developed in this project met the objective of the project. They allowed us to carry out a qualitative and quantitative analysis of both Android and JME. This gave us a chance to figure out the strength and weakness of each platform.

5.3 Project Limitations

Working behind a proxy was one limitation to the project since every machine on the Rhodes network is behind a proxy. The squid proxy does not allow emulators to access the network. This prevented the analysis of how the applications perform behind a proxy and on a real network. To overcome this problem, a direct connection to the network was done through a GSM modem.

The other limitation was that, tools for measuring performance in Android are not available, although they are present in the more mature JME platform.

There is no Android phone yet, so we couldn't test applications on a real phone.

5.4 Possible Extensions

Having an Android phone when it is released will enable comparisons that were not possible without a compatible phone. Sometimes applications behave differently on the actual phone

compared to what they do on emulators. Some exceptions and errors not thrown on emulators normally arise when applications are deployed on a real phone.

5.4.1 A comparative study of Android and the yet to be released JME's JSR293 for Location-based services

JSR 293 will offer more classes for location-based services than the ones present in the current API (JSR179). Table X below shows the classes that are present in the current JME Location API and the ones in the yet to be released API.

JSR179 (Location API v1.0)	JSR293 (Location APIv2.0)
Location	Location*
LocationProvider	LocationProvider*
Criteria	Criteria*
ProximityListener	ProximityListener*
Landmark and LandmarkStore	Landmark and LandmarkStore*
X	LandmarkExchangeFormats
X	Geocoding
X	MapUserInterface
X	Navigation

KEY

Improved class: *

Not present in the API: X

Table 8: compares the classes in the current Location API v1.0 with the yet to be released v2.0

Table 8 above shows that the Location API v2.0 for JME will have better functionality than the current one and that it might also match the functionality currently being offered by the Android API.

Chapter 6: References

NAME	NAME AND DETAILS OF THE SOURCE
	Strategy Analytics. "Location Based Services: Strategic Outlook for Mobile Operators and Solutions Vendors," http://www.strategyanalytics.com, 3 March 2003.
3G	GPS-Enabled Location-Based Services (LBS) Subscribers Will Total 315 Million in Five Years, 28th September, 2006. http://www.3g.co.uk/PR/Sept2006/3701.htm last accessed 21 Oct. 2008
Ableson W.F	Unlocking Android, A Developer's Guide, Copyright 2007 Manning Publications
Ahonen T.T & Barrett J	Services for UTMS: Creating Killer Applications in 3G 2002 edition Published 2002 by John Wiley and Sons
Annex A	Annex A 2002, Background Information on Location Based Services www.sla.gov.sg/doc/new/Aug%2028%20Annex%20A.doc last accessed on 21 Oct. 08
Belic D	ABI Research: Location-based mobile social networking will generate $3.3 billion in global revenues by 2013 Posted by Dusan on Monday, August 4th, 2008 at 2:21 pm http://www.intomobile.com/2008/08/04/abi-research-location-based-mobile-social-networking-will-generate-33-billion-in-global-revenues-by-2013.html
Canalys	Almost 40% of smart phones shipping in EMEA have GPS integrated, Canalys research release 2008/082, Reading (UK) – Friday, 15 August 2008, http://www.canalys.com/pr/2008/r2008082.pdf last accessed 27 Oct. 08
Djuknic G.M & Richton R.E	"Geolocation and Assisted GPS," IEEE Computer (34:2), February 2001, pp. 123-125.
Elsevier B.V.	Computer Communications, Volume 31, Issue 6, 18 April 2008, Pages 1091-1103, Advanced Location-Based Services, Location API 2.0 for J2ME – A new standard in location for Java-enabled mobile phones, http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6TYP-4RR1NT4-5&_user=736737&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_version=1&_urlVersion=0&_userid=736737&md5=d29807cf202d7c8d20c061dca97d248b last accesses 21 Oct. 08
eMarketer	Mobile Location-Based Services on the Move, OCTOBER 6, 2008, http://www.emarketer.com/Article.aspx?id=1006609 last accessed 27 Oct. 08
Fabris N	GPS-Enabled Location-Based Services (LBS) Subscribers Will Total 315 Million in Five Years by Nicole Fabris http://www.abiresearch.com/abiprdisplay.jsp?pressid=731 NEW YORK - September 27, 2006, last accessed 03/11/2008
Ganesan G	Google Android Tutorial http://java.dzone.com/articles/google-android- tutorial?page=0%2C0 ,2008/06/03 - 9:55am
Kin A	LOCATION-BASED SERVICES CALL FOR COLLABORATION 21 Feb. 2003 http://www.ida.gov.sg/doc/Programmes/Programmes_Level3/LBS_CFC_Public_Document-Amended.pdf last accessed 19-10-2008

Knudsen J	Kick Butt with MIDP and MSA: Create Great Mobile Applications ISBN: 9780321463425 Publisher: Addison Wesley, Print Publication Date: 2007/12/21
Li S & Knudsen J	Beginning J2ME From Novice to Professional third edition 2005, Kinetic Publishing Services, LLC
Mitchell K& Whitmore M	"Location Based Services: Locating the Money," in Mobile Commerce: Technology, Theory and Applications, B. E. Mennecke and T. J. Strader (eds.), Idea Group Publishing, Hershey, 2003, pp. 51-66.
Murphy MK	The Busy Coder's Guide to Android Development, Copyright © 2008 CommonsWare, LLC. All Rights Reserved. Printed in the United States of America.
Naraine R	Google Android SDK Hits Security Speed Bump, By Ryan Naraine, 2008-03-04 http://www.eweek.com/c/a/Security/Google-Android-SDK-Hits-Security-Speed-Bump/ last accessed on 26 Oct. 08
Open Geospatial	Open Geospatial Consortium, Inc., Location Service (OpenLS): Core Services, http://www.opengeospatial.org/standards/olscore © 1994-2007 Open GeoSpatial Consortium, Inc. 2008/04/09 last accessed 15/05/2008
Pell A	Test bench: Smart-phones, Phones with a touch of do-anything genius, by Alex Pell October 29, 2006, http://www.timesonline.co.uk/tol/driving/article615307.ece
Rockwell M	"E911: Devil is in the Details," http://www.wirelessweek.com/index.asp?layout=article&articleid= CA298975&spacedesc=Departments&stt=000, 27 May 2003.2003/05/27 last accessed 21/05/2008
Schille & Voisard	Location-based Services by Jochen H. Schiller, Agnès Voisard published by Elsevier's Science & Technology publishing
Smith S & Grubb J	Location and Presence in Mobile Data Services,http://www.boxesandarrows.com/view/location_and_persence_in_mobile_data_services on 2004/08/16 last accessed 09/03/2008
Taves S	Google's Android for phones nearing release, By Scott Taves, msnbc.com contributor, updated 9:26 a.m. ET Sept. 15, 2008, http://www.msnbc.msn.com/id/26674814/ last accessed 21 Oct. 08
TelecomsEurope	Worldwide mobile phone user base hits record 3.25b, TelecomsEurope, June 28, 2007 http://www.telecomseurope.net/article.php?id_article=4208 , last accessed 03/11/2008
The Asian GIS Portal	The Asian GIS Portal, Mobile Location Services to generate US$18.5bn global revenues by 2006, February 2001 http://www.gisdevelopment.net/news/2001/feb/na007.htm last accessed 21 Oct. 08
Tseng Y.C, Wu S.L, Liao W.H & Chao	"Location Awareness in Ad Hoc Wireless Mobile Networks," IEEE Computer (34:6), June 2001, pp. 46-52.

C.M	
Varshne U & Vetter R	"Mobile Commerce: Framework, Applications and Networking Support," ACM/ Kluwer Journal on Mobile Networks and Applications (7:3), 2002, pp. 185-198.
Yuan MJ	Enterprise J2ME DEVELOPING MOBILE JAVA APPLICATIONS (MJY) 2004 Pearson Education, Inc. Publishing as Prentice Hall Professional Technical Reference Upper Saddle River, New Jersey 07458
Zuo X	Trend: Location-based Services in Asia Written by Xuan Zuo on Sunday, August 24, 2008 at 8:24 PM http://www.cscout.com/blog/2008/08/24/location-based-service-converging-to-a-larger-background.html last accessed on 03/11/2008